The Weblate Manual

Versão 4.3.2

Michal Čihař

05 nov., 2020

Documentação de utilizador

1 Documentação de utilizador **1**
- 1.1 Básico do Weblate . . . 1
- 1.2 Registo e perfil de utilizador . . . 1
- 1.3 Traduzir a usar o Weblate . . . 10
- 1.4 Descarregar e enviar traduções . . . 20
- 1.5 Verificações e correções . . . 22
- 1.6 Searching . . . 38
- 1.7 Application developer guide . . . 42
- 1.8 Fluxos de trabalho de tradução . . . 62
- 1.9 Frequently Asked Questions . . . 66
- 1.10 Formatos de ficheiros suportados . . . 73
- 1.11 Integração de controlo de versões . . . 91
- 1.12 Weblate's REST API . . . 98
- 1.13 Cliente Weblate . . . 142
- 1.14 Weblate's Python API . . . 146

2 Documentação de administrador **149**
- 2.1 Instruções de configuração . . . 149
- 2.2 Implantações de Weblate . . . 203
- 2.3 Upgrading Weblate . . . 204
- 2.4 Fazer backup e mover o Weblate . . . 209
- 2.5 Autenticação . . . 215
- 2.6 Controlo de acesso . . . 224
- 2.7 Projetos de tradução . . . 232
- 2.8 Language definitions . . . 248
- 2.9 Tradução contínua . . . 249
- 2.10 Licenciar traduções . . . 259
- 2.11 Processo de tradução . . . 260
- 2.12 Verificações e correções . . . 266
- 2.13 Tradução automática . . . 274
- 2.14 Extensões . . . 280
- 2.15 Memória de Tradução . . . 290
- 2.16 Configuração . . . 292
- 2.17 Sample configuration . . . 318
- 2.18 Management commands . . . 334
- 2.19 Anúncios . . . 344
- 2.20 Lista de componentes . . . 347
- 2.21 Optional Weblate modules . . . 348
- 2.22 Personalizar o Weblate . . . 353
- 2.23 Interface de gestão . . . 355
- 2.24 Obter suporte para o Weblate . . . 362

2.25 Documentos legais . . . 363

3 Documentação de colaborador **365**
3.1 Contribuir ao Weblate . . . 365
3.2 Starting contributing code to Weblate . . . 366
3.3 Código-fonte do Weblate . . . 370
3.4 Debugging Weblate . . . 371
3.5 Weblate internals . . . 372
3.6 Weblate frontend . . . 373
3.7 Reporting issues in Weblate . . . 374
3.8 Weblate testsuite and continuous integration . . . 374
3.9 Data schemas . . . 376
3.10 Releasing Weblate . . . 379
3.11 Sobre o Weblate . . . 380
3.12 Licença . . . 381

4 Histórico de alterações **382**
4.1 Weblate 4.3.2 . . . 382
4.2 Weblate 4.3.1 . . . 382
4.3 Weblate 4.3 . . . 383
4.4 Weblate 4.2.2 . . . 384
4.5 Weblate 4.2.1 . . . 384
4.6 Weblate 4.2 . . . 384
4.7 Weblate 4.1.1 . . . 385
4.8 Weblate 4.1 . . . 385
4.9 Weblate 4.0.4 . . . 386
4.10 Weblate 4.0.3 . . . 387
4.11 Weblate 4.0.2 . . . 387
4.12 Weblate 4.0.1 . . . 387
4.13 Weblate 4.0 . . . 388
4.14 Weblate 3.x series . . . 388
4.15 Weblate 2.x series . . . 400
4.16 Weblate 1.x series . . . 411
4.17 Weblate 0.x series . . . 415

Índice de Módulos do Python **419**

HTTP Routing Table **420**

Índice **423**

CAPÍTULO 1

Documentação de utilizador

1.1 Básico do Weblate

1.1.1 Estrutura de projetos

No Weblate, as traduções são organizadas em projetos e componentes. Cada projeto pode conter vários componentes, os quais contêm traduções para idiomas individuais. O componente corresponde a um ficheiro traduzível (por exemplo, *GNU gettext* ou *Android string resources*). Os projetos existem para ajudá-lo a organizar componentes em conjuntos lógicos (por exemplo, para agrupar todas as traduções usadas dentro de uma aplicação).

Internamente, cada projeto tem traduções para cadeias comuns propagadas em outros componentes dentro dele por predefinição. Isso alivia o fardo da tradução repetitiva e de várias versões. Desative isto conforme *Component configuration*, ainda produzindo erros para traduções aparentemente inconsistentes resultantes.

1.2 Registo e perfil de utilizador

1.2.1 Registo

Todos podem procurar projetos, visualizar traduções ou sugerir traduções por predefinição. Somente utilizadores registados têm permissão para realmente gravar as alterações e são creditados para cada tradução feita.

Pode registar-se seguindo alguns passos simples:

1. Preencha o formulário de registo com as suas credenciais.
2. Ative o registo seguindo a hiperligação no e-mail que receber.
3. Ajuste opcionalmente o seu perfil para escolher quais idiomas conhece.

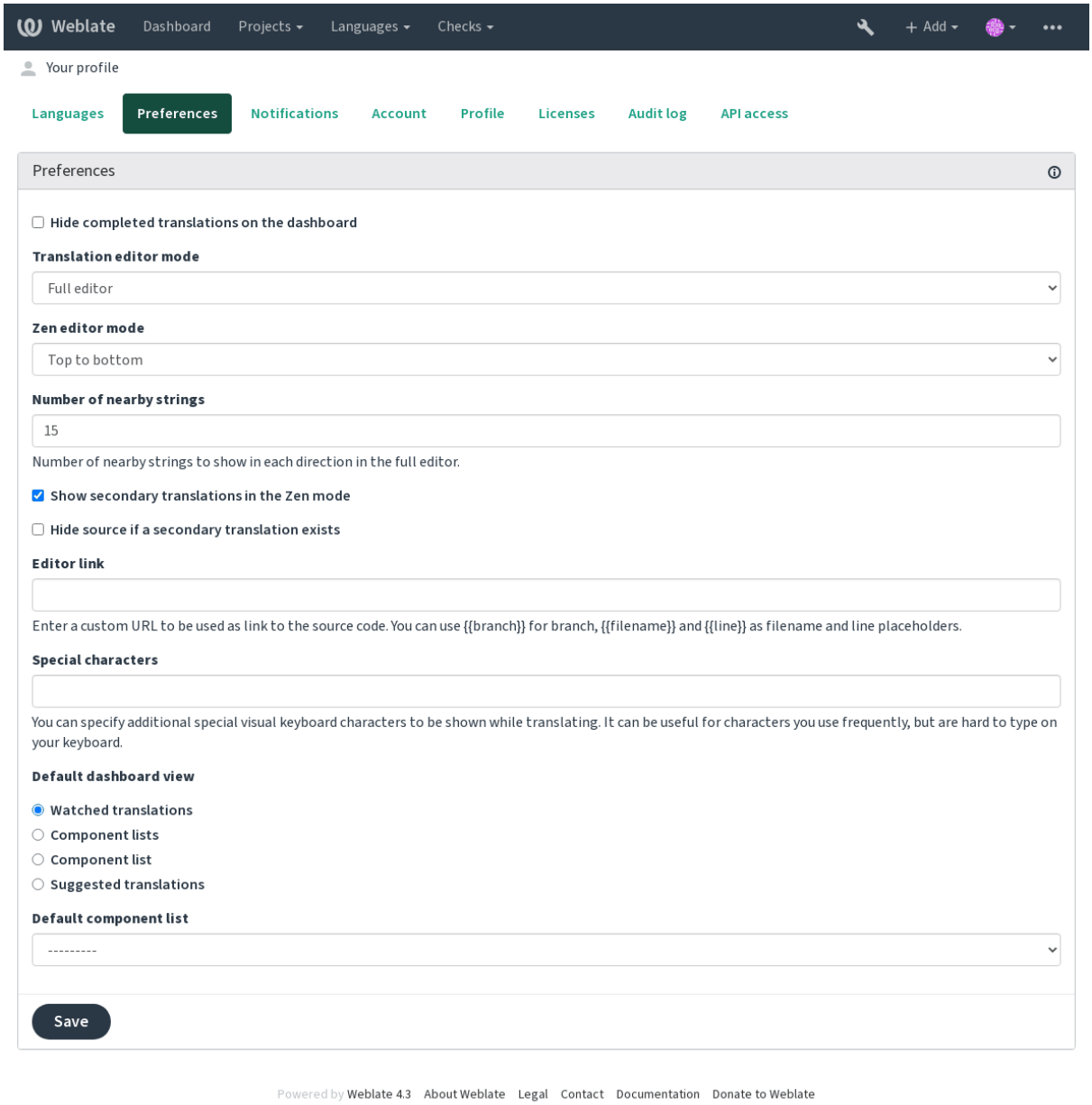
1.2.2 Painel

Ao fazer login verá uma visão geral de projetos e componentes, bem como a respetiva progressão de tradução deles.

Novo na versão 2.5.

Os componentes dos projetos que está a observar são mostrados por predefinição e cruzados com os idiomas da sua preferência.

Dica: Pode mudar para visualizações diferentes a usar as guias de navegação.



O menu tem estas opções:

- *Projetos* > *Visualizar todos os projetos* no menu principal mostra o estado da tradução, para cada projeto, na instância do Weblate.
- Selecionar um idioma no menu principal de *Idiomas* irá mostrar o estado da tradução de todos os projetos, filtrado por um dos seus idiomas primários.

- *Traduções observadas* no Painel vai mostrar o estado da tradução apenas dos projetos que está observando, filtradas por os seus idiomas primários.

Além disso, o menu suspenso também pode mostrar qualquer quantidade de *listas de componentes*, conjuntos de componentes do projeto pré-configurados pelo administrador da Weblate, veja *Lista de componentes*.

Pode configurar a sua exibição de painel predefinido pessoal na secção *Preferências* das configurações do perfil do utilizador.

Nota: Quando o Weblate estiver configurado para um único projeto a usar `SINGLE_PROJECT` no ficheiro `settings.py` (veja *Configuração*), o painel não será mostrado, pois o utilizador será redirecionado para um único projeto ou componente.

1.2.3 Perfil do utilizador

O perfil do utilizador é acessível clicando no ícone do utilizador no topo direito do menu superior e depois no menu *Configurações*.

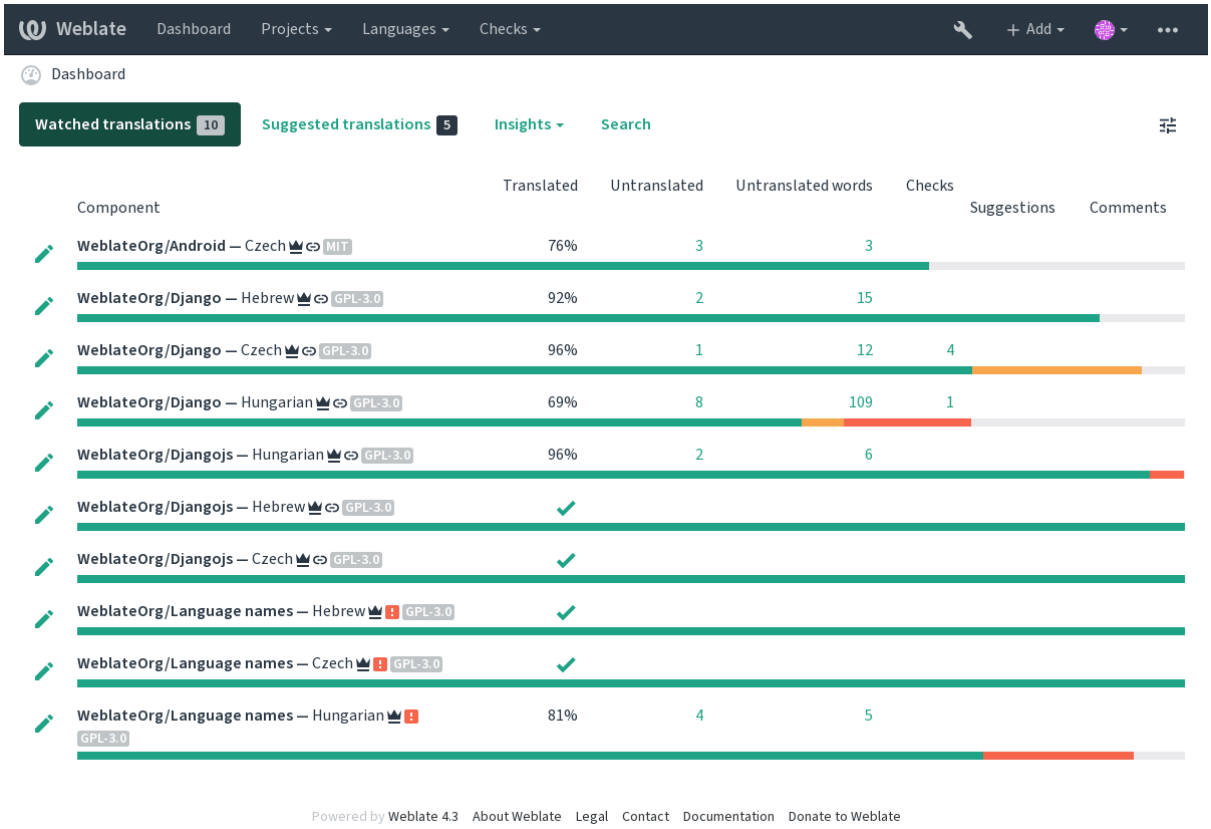
O perfil do utilizador contém as suas preferências. Nome e endereço de e-mail são usados em commits de VCS, por isso mantenha essas informações precisas.

Nota: Todas as seleções de idiomas só oferecem idiomas traduzidos atualmente.

Dica: Solicite ou adicione outros idiomas que deseja traduzir clicando no botão para torná-los também disponíveis.

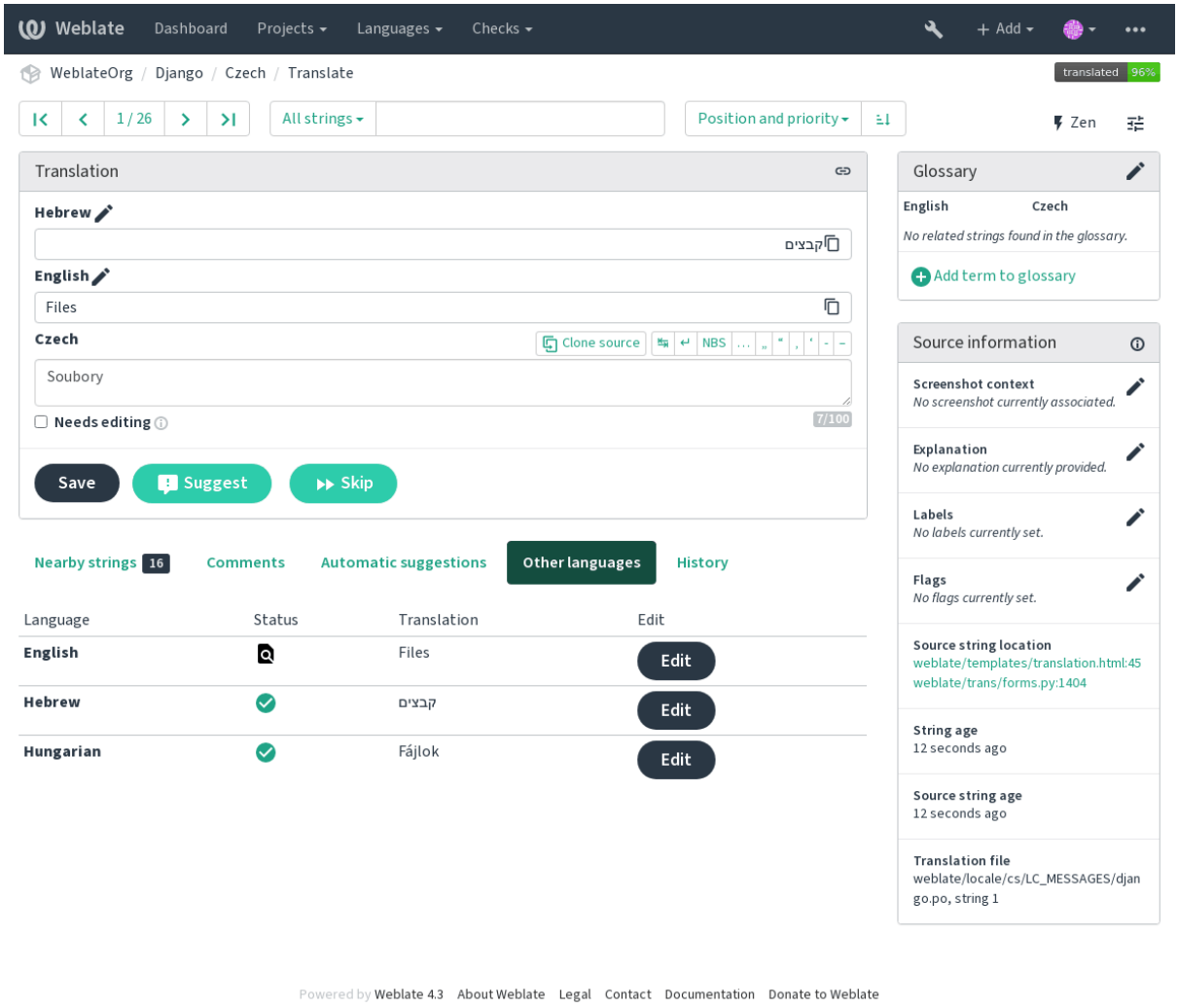
Idiomas traduzidos

Escolha quais idiomas prefere traduzir e eles serão oferecidos na página principal de projetos assistidos, para que tenha acesso mais fácil a todas essas traduções em cada um desses idiomas.



Idiomas secundários

Pode definir quais idiomas secundários são lhe mostrados como um guia durante a tradução. Um exemplo pode ser visto na imagem a seguir, onde o idioma hebreu é mostrado como secundário:



Visualização predefinida do painel

Na guia *Preferências*, pode escolher qual das visualizações disponíveis do painel de instrumentos deve-se apresentar por predefinição. Se escolher a lista de *Lista de componentes*, terá que selecionar qual lista de componentes será exibida a partir da *Lista de componentes predefinida* suspensa.

Veja também:

Lista de componentes

Perfil público

Todos os campos desta página são opcionais e podem ser apagados a qualquer momento, e ao preenchê-los, dá-nos o seu consentimento para compartilhar esses dados onde quer que o seu perfil de utilizador apareça.

Um avatar pode ser mostrado para cada utilizador (dependendo de `ENABLE_AVATARS`). Estas imagens são obtidas utilizando https://gravatar.com/.

Hiperligação do editor

Uma ligação de código-fonte é mostrado no navegador web configurado no *Component configuration* por predefinição.

Dica: Ao definir o *Ligação do editor*, usa o editor local para abrir o ficheiro de código-fonte VCS de cadeias traduzidas. Pode usar *Template markup*.

Geralmente alguma coisa como `editor://open/?file={{filename}}&line={{line}}` é uma boa opção.

Veja também:

Pode encontrar mais informações sobre o registo de protocolos de URL personalizados para o editor na documentação do Nette.

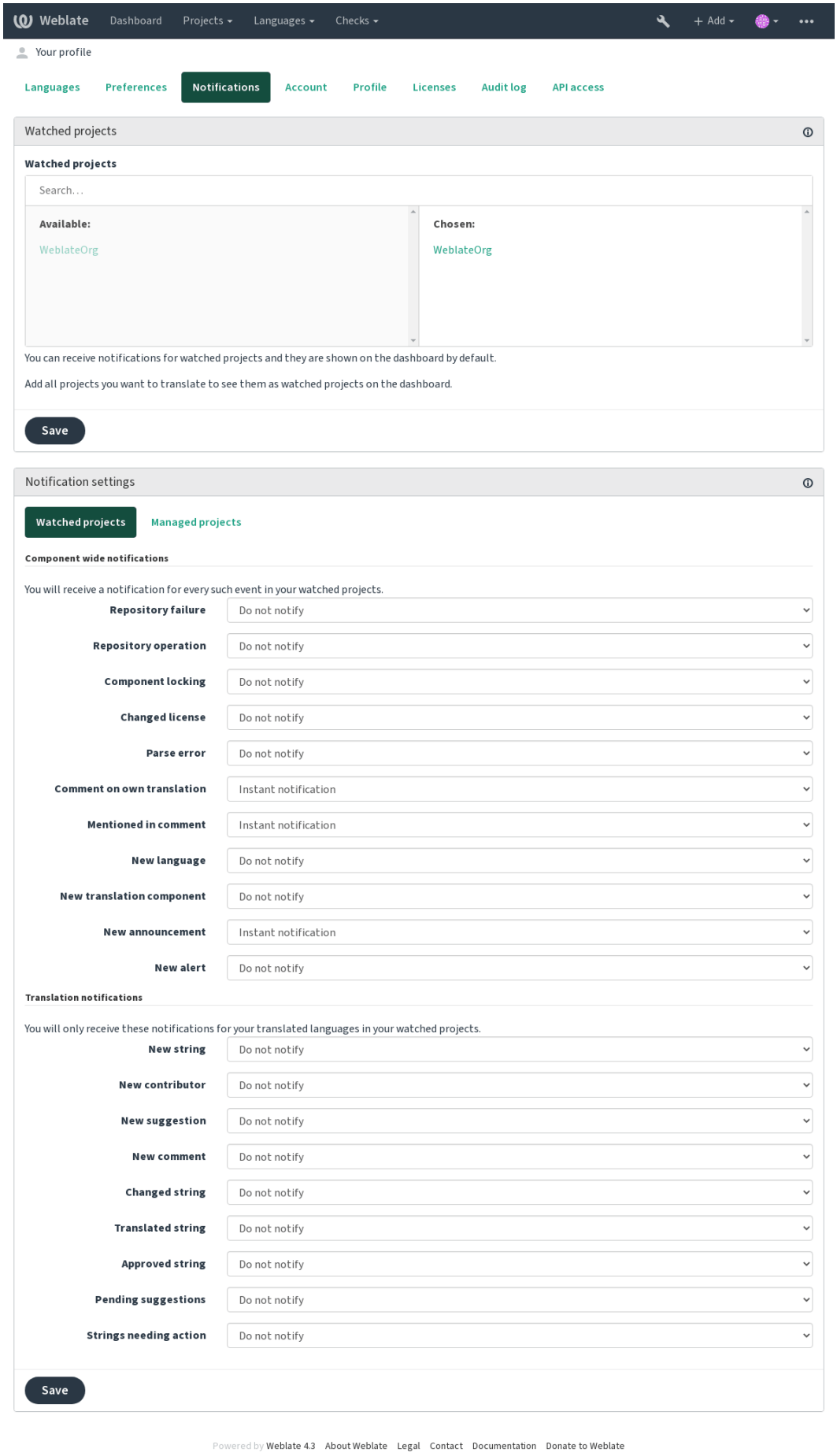
1.2.4 Notificações

Inscreva-se em várias notificações da guia *Notificações*. As notificações para eventos selecionados em projetos assistidos ou administrados serão lhe enviadas por e-mail.

Algumas das notificações são enviadas apenas para eventos nos seus idiomas (por exemplo, sobre novas cadeias para traduzir), enquanto algumas acionam no nível de componente (por exemplo, erros de fusão). Esses dois grupos de notificações são visualmente separados nas configurações.

Pode alternar notificações para projetos assistidos e projetos administrados e pode ser mais ajustado (ou silenciado) por projeto e componente. Visite a página de componentes e selecione a escolha apropriada no menu *Observando*.

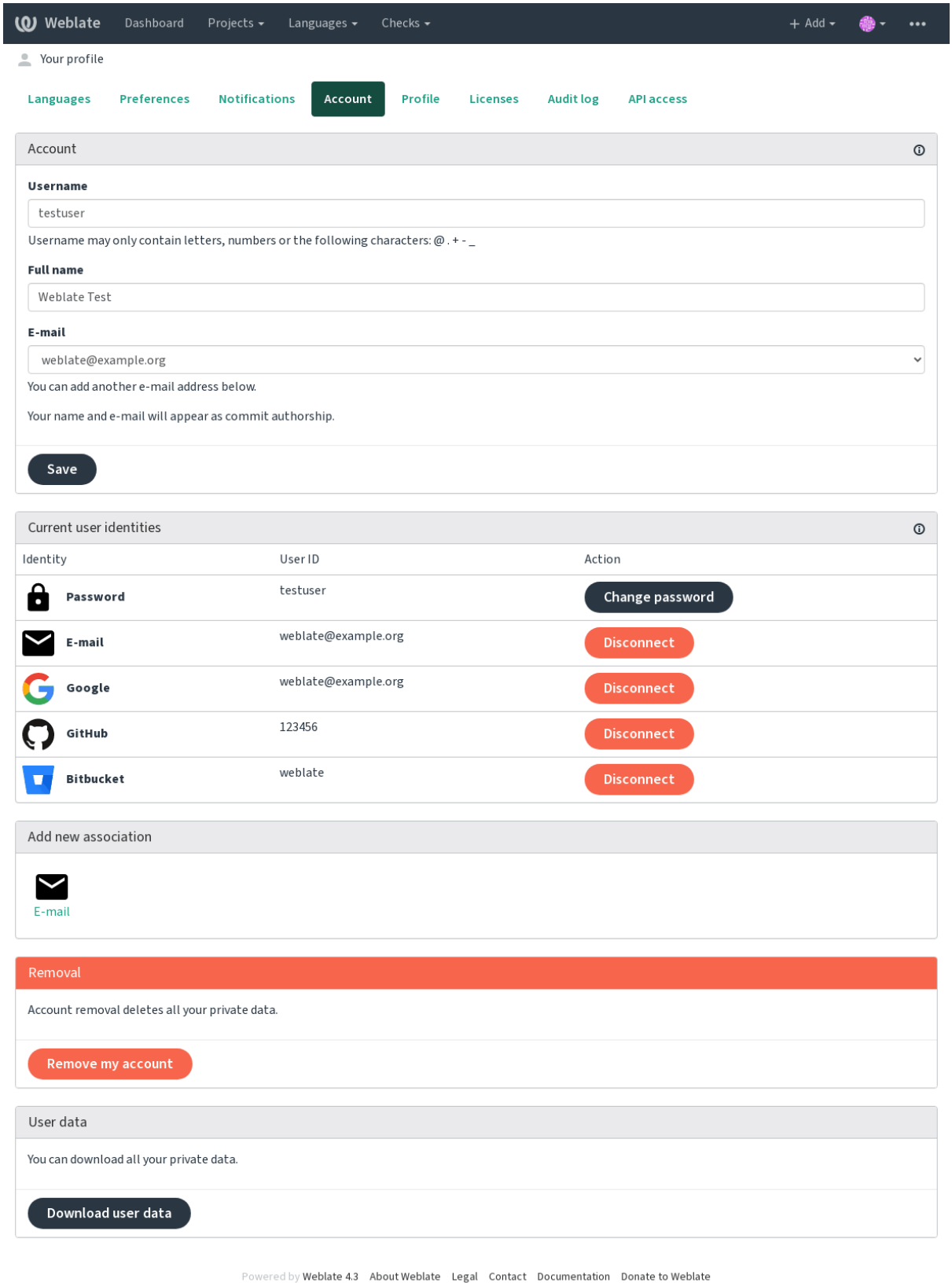
Nota: Não receberá notificações para as suas próprias ações.


Weblate
Dashboard
Projects
Languages
Checks
Add
Your profile
Languages
Preferences
Notifications
Account
Profile
Licenses
Audit log
API access
Watched projects
Watched projects
Search...
Available:
WeblateOrg
Chosen:
WeblateOrg
You can receive notifications for watched projects and they are shown on the dashboard by default.
Add all projects you want to translate to see them as watched projects on the dashboard.
Save
Notification settings
Watched projects
Managed projects
Component wide notifications
You will receive a notification for every such event in your watched projects.
Repository failure
Repository operation
Component locking
Changed license
Parse error
Comment on own translation
Mentioned in comment
New language
New translation component
New announcement
New alert
Do not notify
Instant notification
Translation notifications
You will only receive these notifications for your translated languages in your watched projects.
New string
New contributor
New suggestion
New comment
Changed string
Translated string
Approved string
Pending suggestions
Strings needing action
Save
Powered by Weblate 4.3
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.5 Conta

A guia *Conta* permite configurar detalhes básicos da conta, conectar vários serviços que pode usar para entrar no Weblate, remover a sua conta completamente ou descarregar os seus dados de utilizador (veja *Exportação de dados de utilizadores do Weblate*).

Nota: A lista de serviços depende da configuração do Weblate, mas pode ser feita para incluir sites populares como GitLab, GitHub, Google, Facebook ou Bitbucket ou outros provedores de OAuth 2.0.


Weblate
Dashboard
Projects
Languages
Checks
Add
Your profile
Languages
Preferences
Notifications
Account
Profile
Licenses
Audit log
API access
Account
Username
testuser
Username may only contain letters, numbers or the following characters: @ . + - _
Full name
Weblate Test
E-mail
weblate@example.org
You can add another e-mail address below.
Your name and e-mail will appear as commit authorship.
Save
Current user identities
Identity
User ID
Action
Password
testuser
Change password
E-mail
weblate@example.org
Disconnect
Google
weblate@example.org
Disconnect
GitHub
123456
Disconnect
Bitbucket
weblate
Disconnect
Add new association
E-mail
Removal
Account removal deletes all your private data.
Remove my account
User data
You can download all your private data.
Download user data
Powered by Weblate 4.3
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.6 Registo de auditoria

O registo de auditoria rastreia as ações realizadas com a sua conta. Ele regista o endereço IP e o navegador para cada ação importante com a sua conta. As ações críticas também desencadeiam uma notificação a um endereço de e-mail principal.

Veja também:

Executar por trás de um proxy reverso

1.3 Traduzir a usar o Weblate

Obrigado pelo interesse em traduzir a usar o Weblate. Os projetos podem ser configurados para tradução direta ou por meio de sugestões feitas por utilizadores sem contas.

No geral, há dois modos de tradução:

- O projeto aceita traduções diretas
- O projeto aceita apenas sugestões, que são validadas automaticamente uma vez que uma quantidade definida de votos é alcançada

Por favor, veja *Fluxos de trabalho de tradução* para obter mais informações sobre fluxo de trabalho de tradução.

Opções para a visibilidade do projeto de tradução:

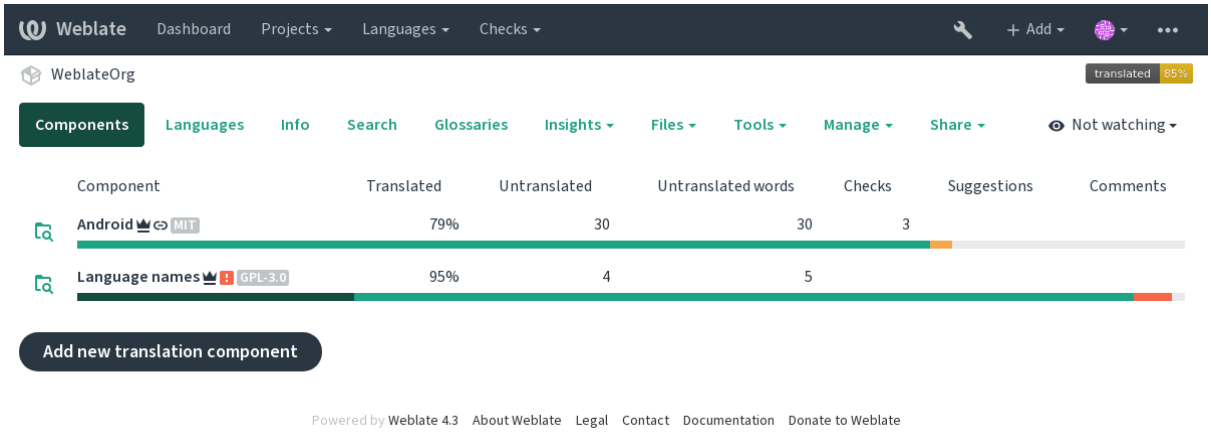
- Publicamente visível e todos podem contribuir
- Visível apenas para um certo grupo de tradutores

Veja também:

Controlo de acesso, *Fluxos de trabalho de tradução*

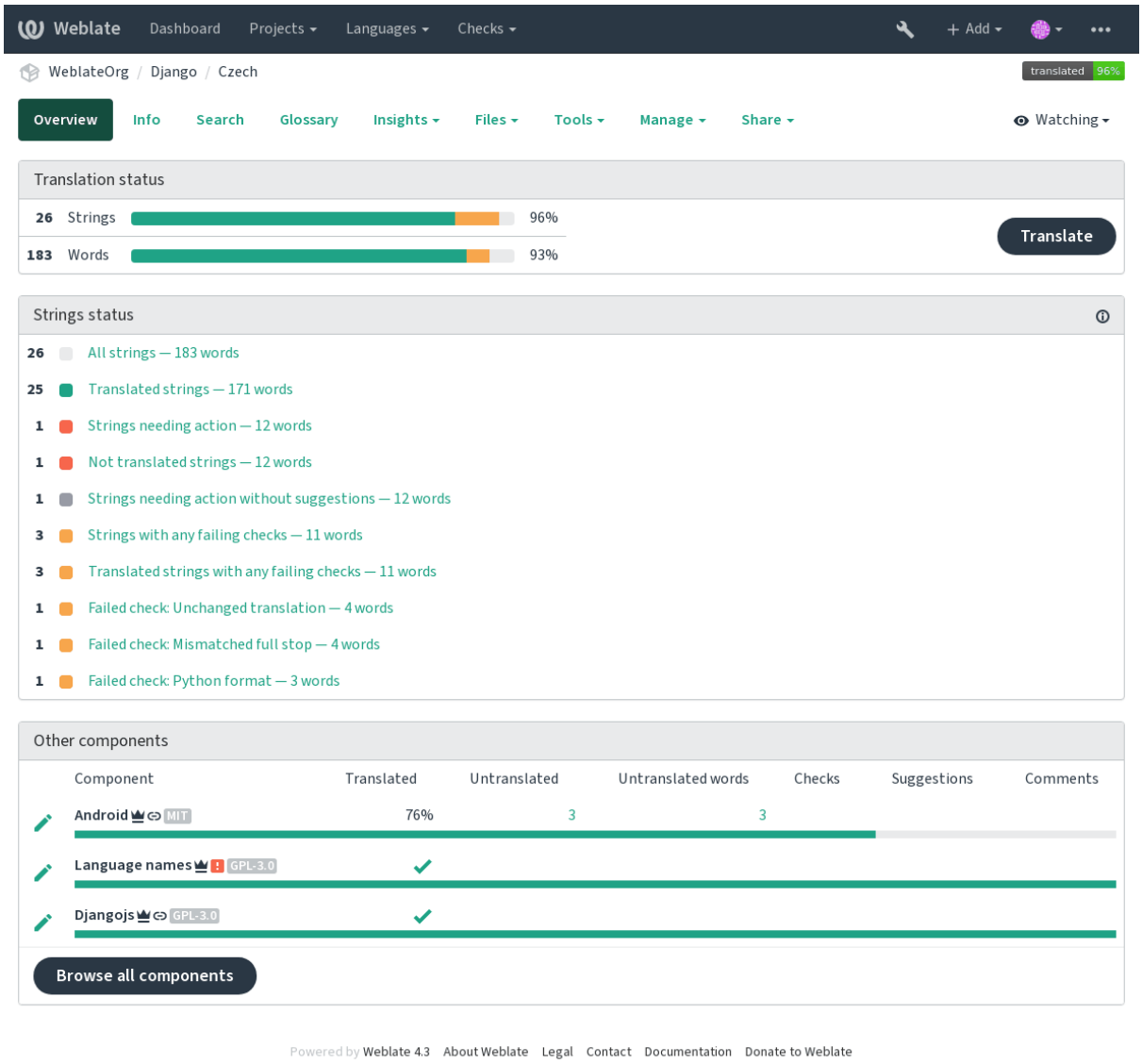
1.3.1 Projetos de tradução

Os projetos de tradução possuem componentes relacionados, relacionados ao mesmo software, livro ou projeto.



1.3.2 Ligações de tradução

Depois de navegar para um componente, um conjunto de ligações leva à tradução real. A tradução é ainda dividida em verificações individuais, como *Cadeias não traduzidas* ou *Cadeias que necessitam edição*. Se todo o projeto for traduzido sem erro, *Todas as cadeias* ainda estão disponíveis. Alternativamente, pode usar o campo de pesquisa para encontrar uma cadeia ou um termo específico.



1.3.3 Sugestões

Nota: As permissões podem variar de acordo com a configuração da sua instância do Weblate.

Utilizadores anônimos só podem (se permitido) encaminhar sugestões. Isso ainda está disponível aos utilizadores autenticados, nos casos em que surge a incerteza sobre a tradução, o que levará outro tradutor a revisá-la.

As sugestões são verificadas diariamente para remover as duplicatas ou sugestões que correspondam à tradução atual.

1.3.4 Comentários

Comentários podem ser enviados em dois escopos - cadeia fonte ou de tradução. Escolha o que corresponda ao tópico que deseja discutir. Os comentários de cadeias fonte são bons para fornecer feedback sobre cadeias originais, por exemplo, de que ele deve ser reformulado ou está confuso.

Pode usar a sintaxe do Markdown nos comentários e mencionar outros utilizadores a usar `@menção`.

1.3.5 Variantes

As variantes são usadas para agrupar variantes do texto em diferentes comprimentos. A superfície pode usar cadeias diferentes dependendo do tamanho do ecrã ou da janela.

Veja também:

String variants

1.3.6 Etiquetas

As etiquetas são usadas para categorizar cadeias dentro de um projeto. Elas podem ser usadas para personalizar ainda mais o fluxo de trabalho de localização, por exemplo, para definir categorias de cadeias.

Veja também:

String labels

1.3.7 Traduzir

Na página de tradução vê-se a cadeia fonte e uma área de edição para tradução. Caso a tradução seja plural, vê-se múltiplas cadeias fonte e áreas de edição, cada um descrito e rotulado em forma plural.

Todos os caracteres especiais de espaço em branco são sublinhados em vermelho e indicados com símbolos cinzentos. Mais de um espaço subsequente também é sublinhado em vermelho para alertar o tradutor para um possível problema de formatação.

Vários pedaços de informações extras podem ser mostrados nesta página, a maioria proveniente do código-fonte do projeto (como contexto, comentários ou onde a mensagem está a ser usada). Quando escolhe idiomas secundários nas suas preferências, a tradução para esses idiomas será mostrada (ver *Idiomas secundários*) acima da cadeia fonte.

Abaixo da tradução, qualquer sugestão feita por outros será mostrada, que pode, por sua vez, aceitar, aceitar com alterações ou apagar.

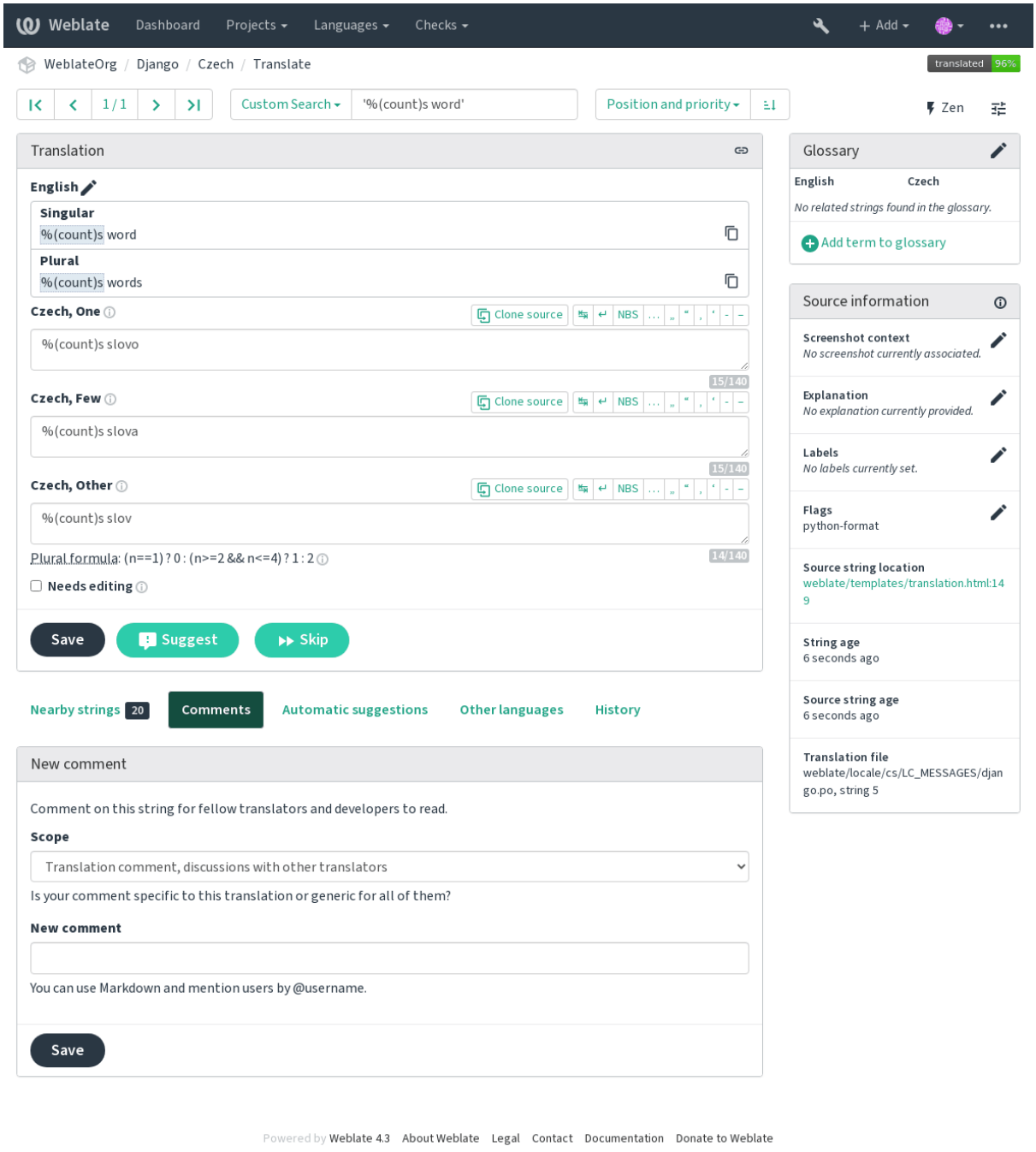
Plurais

Palavras que mudam de forma levando a designação numérica delas em conta são chamadas plurais. Cada idioma tem uma definição de plurais própria . O inglês, por exemplo, possui um plural. Na definição singular de, por exemplo, «car» (carro), implicitamente um carro é referenciado, enquanto na definição plural, «carros» significa dois ou mais carros, ou o conceito de carros como substantivo. Idiomas como, por exemplo, tcheco ou árabe têm mais plurais e as regras deles para plurais também são diferentes.

O Weblate tem suporte total de cada uma dessas formas, em cada respetivo idioma, traduzindo cada plural separadamente. A quantidade de campos e como são usados na aplicação traduzido depende da forma de plural configurada. Weblate mostra as informações básicas, mas encontr uma descrição mais detalhada em Language Plural Rules do Unicode Consortium.

Veja também:

Fórmula de plural



Atalhos de teclado

Alterado na versão 2.18: Os atalhos do teclado foram renovados em 2.18 para reduzir a possibilidade de colidir com o atalhos predefinidos de navegadores ou sistemas.

Os seguintes atalhos de teclado podem ser utilizados durante a tradução:

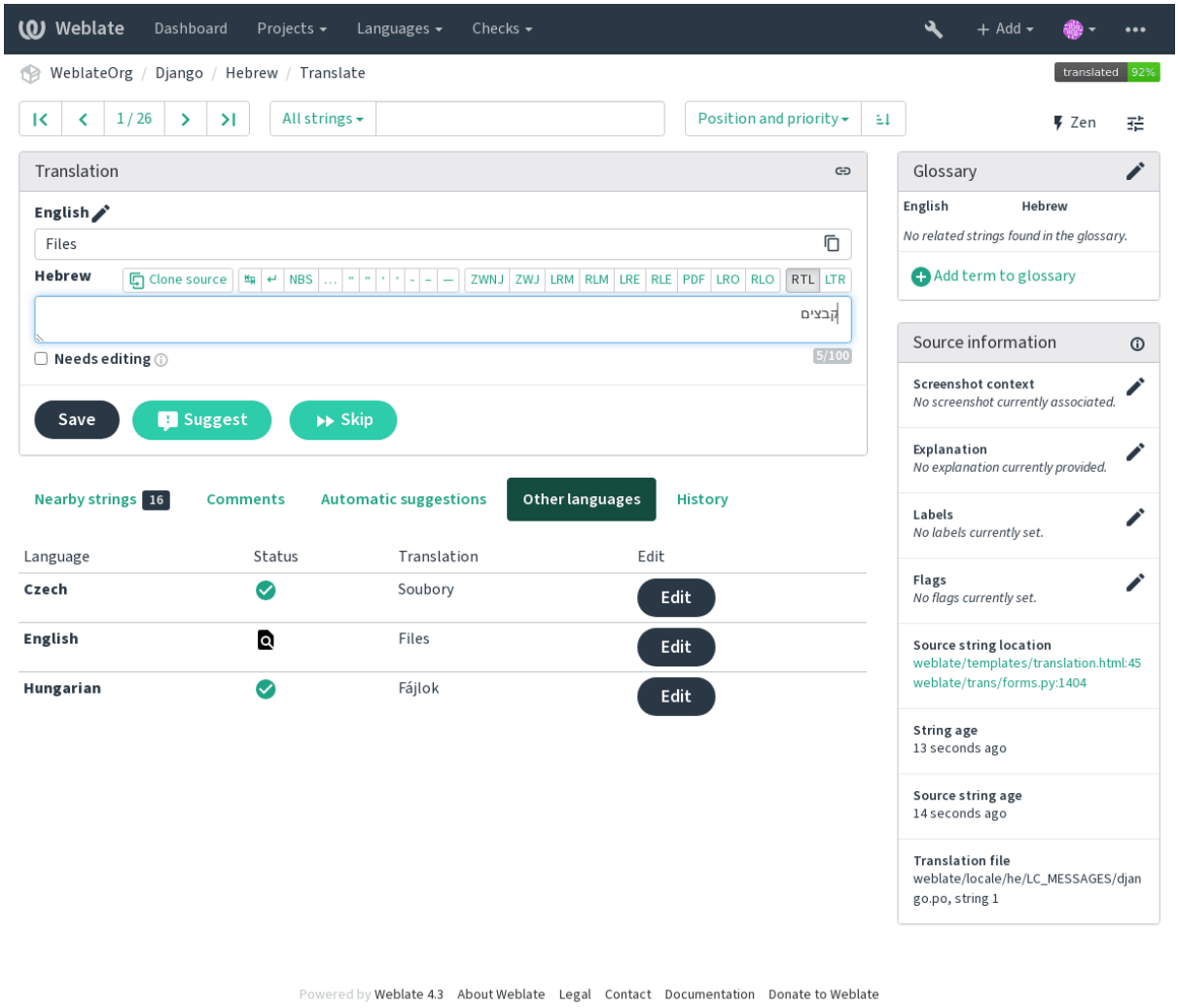
Keyboard shortcut	Descrição
`Alt Home`	De momento, esta tradução está bloqueada.
`Alt Home`	De momento, esta tradução está bloqueada.
`Alt End`	De momento, esta tradução está bloqueada.
`Alt PageUp` or `Ctrl ↑` or `Alt ↑` or `Cmd ↑`	De momento, esta tradução está bloqueada.
`Alt PageDown` or `Ctrl ↓` or `Alt ↓` or `Cmd ↓`	De momento, esta tradução está bloqueada.
`Alt Enter` or `Ctrl Enter` or `Cmd Enter`	Gravar tradução atual.
`Ctrl Shift Enter` or `Cmd Shift Enter`	Desmarca estado de tradução aproximada e envia-a.
`Ctrl E` or `Cmd E`	Muda o foco ao editor de tradução.
`Ctrl U` or `Cmd U`	Muda o foco ao editor de comentários.
`Ctrl M` or `Cmd M`	Shows *Automatic suggestions* tab, see *Sugestões automáticas*.
`Ctrl 1` to `Ctrl 9` or `Cmd 1` to `Cmd 9`	Copia objetos colocáveis de determinada quantidade da cadeia fonte.
`Ctrl M 1` to `9` or `Cmd M 1` to `9`	É passado como um parâmetro único que consiste o nome de uma tradução atual.
`Ctrl I 1` to `9` or `Cmd I 1` to `9`	Ignore um item na lista de verificações falhadas.
`Ctrl J` or `Cmd J`	Mostra a guia de *Cadeias próximas*.
`Ctrl S` or `Cmd S`	Focuses search field.
`Ctrl O` or `Cmd O`	Copia a cadeia fonte.
`Ctrl Y` or `Cmd Y`	Alterna a opção *Necessita edição*.

Teclado visual

Um pequeno teclado visual é mostrado logo acima do campo de tradução. Isto pode ser útil para digitar caracteres normalmente não encontrados ou de digitação difícil.

Os símbolos mostrados são apresentados em três categorias:

- Caracteres configurados pelo utilizador definidos em *Perfil do utilizador*
- Caracteres por idioma fornecidos pelo Weblate (por exemplo, citações ou caracteres específicos RTL)
- Caracteres configurados a usar `SPECIAL_CHARS`



Contexto da tradução

Esta descrição contextual fornece informações relacionadas sobre a cadeia atual.

Atributos da cadeia Coisas como ID da mensagem, contexto (`msgctxt`) ou localização no código-fonte.

Capturas de ecrã Capturas de ecrã podem ser enviadas ao Weblate para melhor informar os tradutores sobre onde e como a cadeia é usada, veja *Visual context for strings*.

Cadeias próximas Exibe mensagens próximas do ficheiro de tradução. Estas também são geralmente usadas num contexto semelhante e se mostram úteis para manter a tradução consistente.

Outras ocorrências No caso de uma mensagem aparecer em vários lugares (por exemplo, vários componentes), esta guia mostra todos eles se forem considerados inconsistentes (veja *Inconsistente*). Pode escolher qual usar.

Memória de tradução Veja cadeias semelhantes traduzidas no passado, veja Memory Management.

Glossário Exibe termos do glossário do projeto usados na mensagem atual.

Alterações recentes Lista de pessoas que modificaram esta mensagem recentemente a usar Weblate.

Projeto Informações do projeto, como instruções para tradutores ou informações sobre o repositório do sistema de controle de versão.

Se o formato de tradução for suportado, também poderá seguir ligações fornecidos para o respetivo código-fonte contendo cada cadeia fonte.

Histórico de tradução

Cada alteração é por predefinição (a menos que desativada nas configurações dos componentes) gravada no banco de dados e pode ser revertida. Opcionalmente, ainda se pode reverter qualquer coisa no sistema de controle de versão subjacente.

Comprimento da cadeia traduzida

Weblate pode limitar o comprimento de tradução em várias formas para garantir a cadeia traduzida não fique muito comprida:

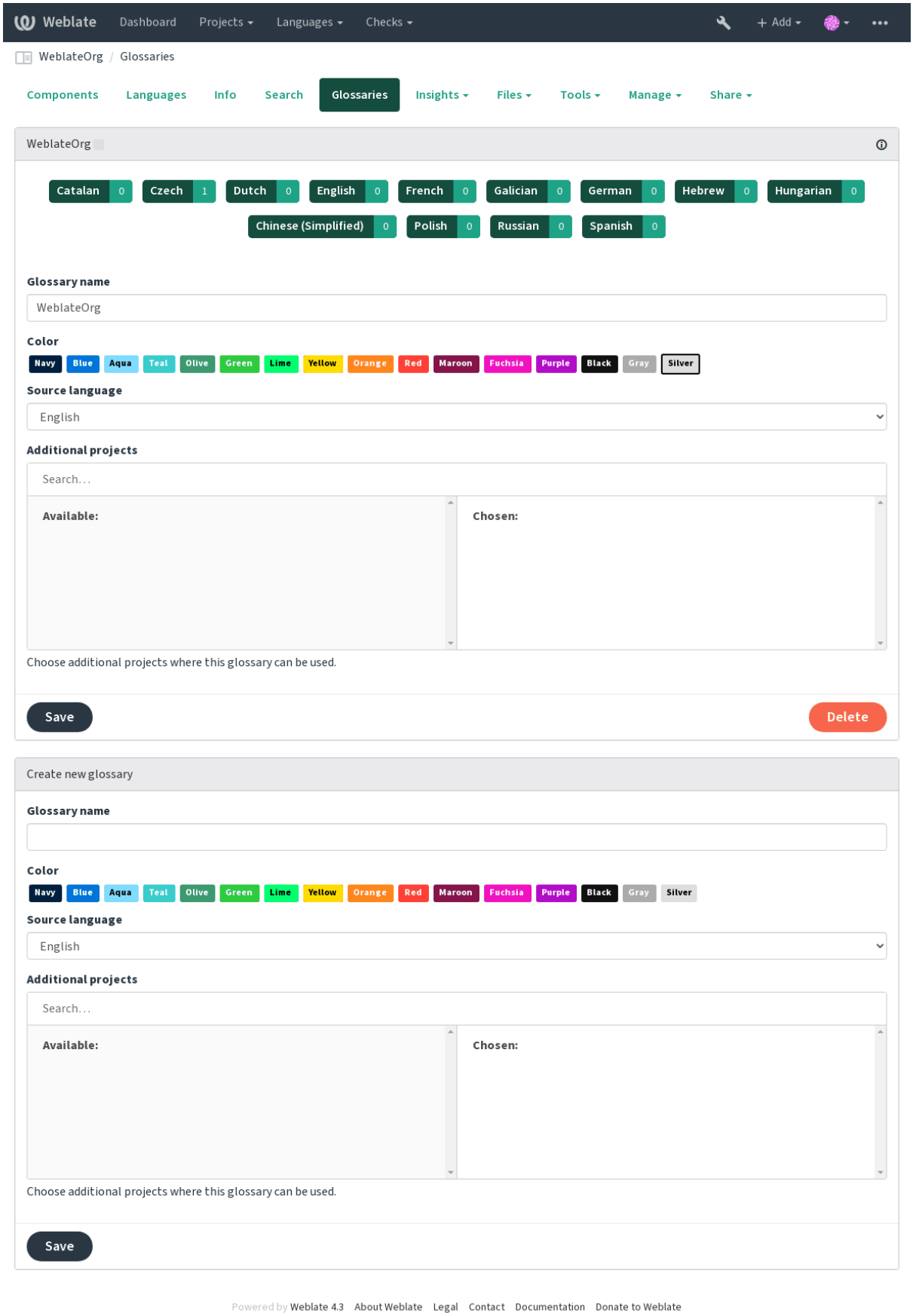
- A limitação predefinida para tradução é dez vezes maior do que a cadeia fonte. Isso pode ser transformado em `LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH`. Caso esteja a acertar isso, ele também pode ser causado pela tradução monolingue sendo configurada como bilíngue, a fazer com que o Weblate veja a chave de tradução como cadeia fonte em vez da cadeia fonte real. Veja *Bilingual and monolingual formats* para obter mais informações.
- Comprimento máximo em caracteres definidos por ficheiro de tradução ou um sinalizador, consulte *Tamanho máximo da tradução*.
- Tamanho máximo renderizado em pixels definido por sinalizadores, veja *Tamanho máximo da tradução*.

1.3.8 Glossário

Cada projeto pode ter um glossário atribuído para qualquer idioma como abreviação para armazenar terminologia. A consistência é mais facilmente mantida desta forma. Os termos de cadeia traduzida atualmente podem ser exibidos nas guias inferiores.

Gestão de glossários

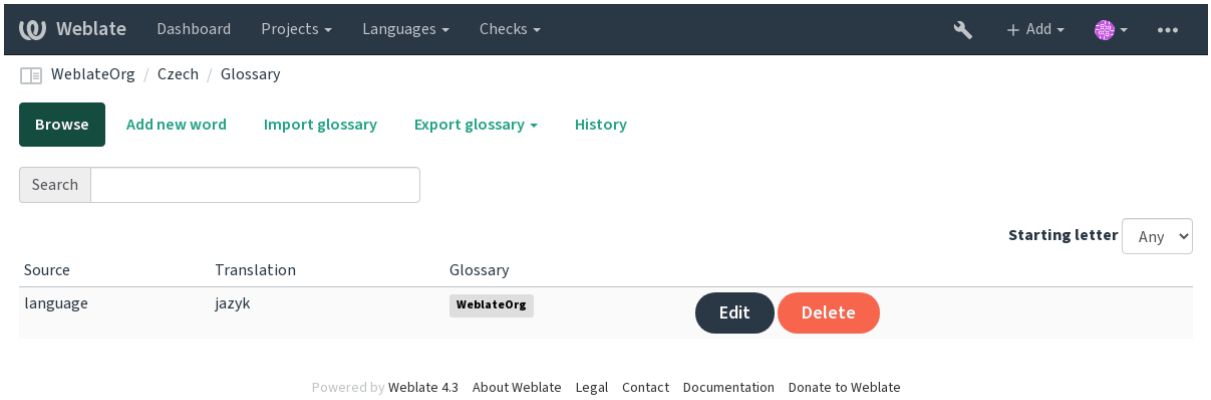
Na guia *Glossários* de cada página do projeto, pode editar glossários existentes.



Um glossário vazio para um determinado projeto é criado automaticamente quando o projeto é criado. Glossários são compartilhados entre todos os componentes do mesmo projeto e também pode optar por compartilhá-los com

outros projetos. Só pode fazer-lo para projetos que pode administrar.

Nesta lista, pode escolher qual glossário gerir (todos os idiomas usados no projeto atual são mostrados). Seguir uma das ligações de idioma o levará a uma página que pode ser usada para editar, importar ou exportar o glossário selecionado ou visualizar o histórico de edição:



1.3.9 Sugestões automáticas

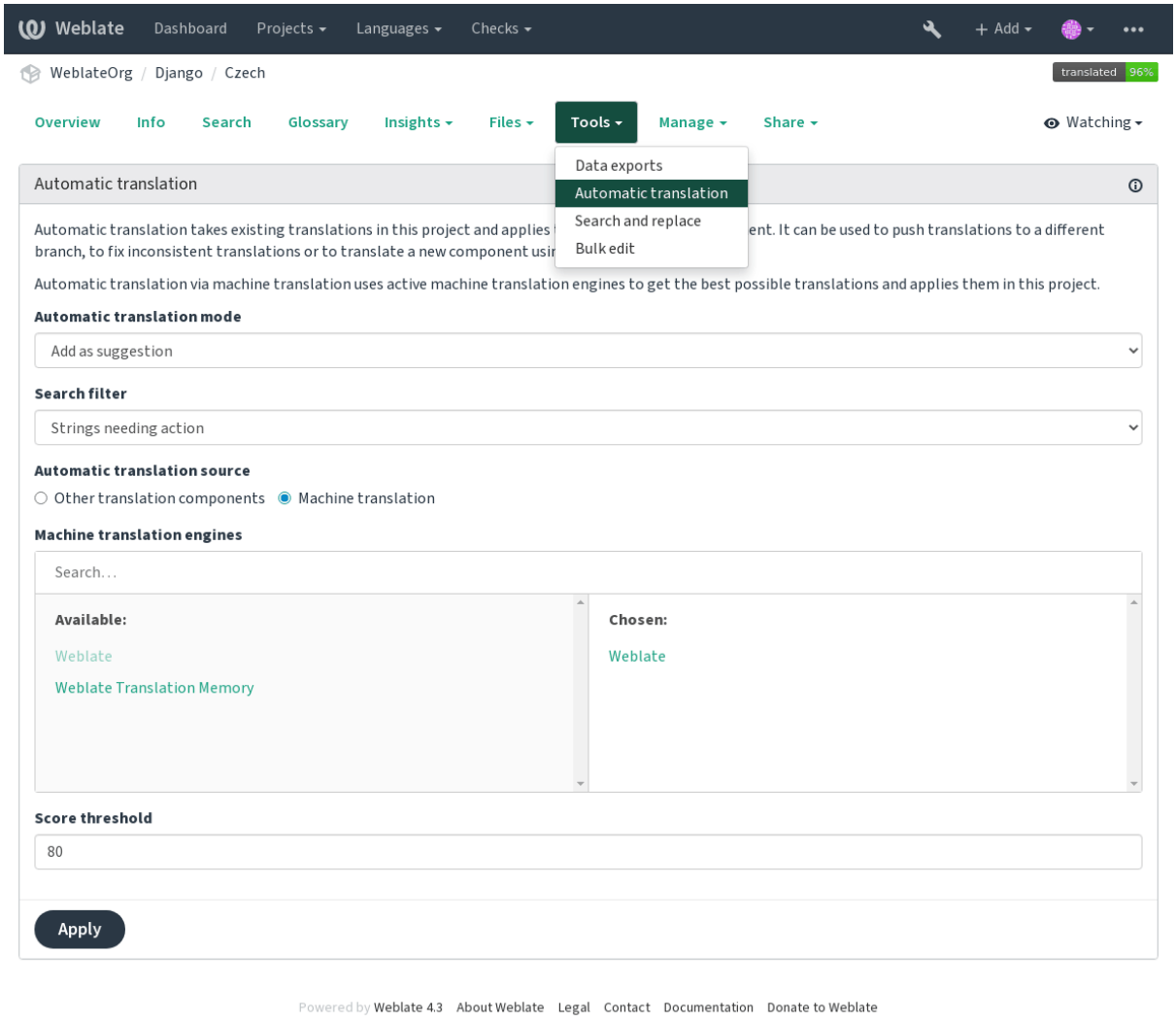
Com base na configuração e no idioma traduzido, o Weblate fornece sugestões de várias ferramentas de tradução automática e *Memória de Tradução*. Todas as traduções automáticas estão disponíveis numa única guia de cada página de tradução.

Veja também:

Encontra a lista de ferramentas suportadas em *Tradução automática*.

1.3.10 Tradução automática

Pode usar a tradução automática para a iniciar a tradução com base em fontes externas. Esta ferramenta chama-se *Tradução automática*, acessível no menu *Ferramentas*, uma vez que tenha selecionado um componente e um idioma:



Dois modos de operação são possíveis:

- Usar outros componentes do Weblate como fonte para traduções.
- Usar serviços selecionados de tradução automática com traduções acima de um certo limite de qualidade.

Também pode escolher quais cadeias devem ser traduzidas automaticamente.

Aviso: Tenha em mente que isso substituirá as traduções existentes se empregadas com filtros amplos, como *Todos as cadeias*.

Útil em várias situações, como a consolidação da tradução entre componentes diferentes (por exemplo, site e aplicação) ou quando estiver a iniciar a tradução para um novo componente a usar traduções existentes (memória de tradução).

Veja também:

Manter traduções iguais entre componentes

1.3.11 Limitação de taxa

Para evitar abusos na interface, há limitações de taxa aplicada a várias operações como pesquisa, envio de formulário de contato ou tradução. Caso seja atingido por isso, fica bloqueado por um certo período até que possa executar a operação novamente.

Os limites predefinidos são descritos no manual administrativo em *Limitação de taxa*, mas podem ser ajustados por configuração.

1.3.12 Edição em massa

A edição em massa permite que execute a operação em quantidades de cadeias. Define os cadeias de pesquisa e operação para executar e todas as cadeias correspondentes são atualizados. As seguintes operações são suportadas:

- Alterar o estado da cadeia (por exemplo, para aprovar todas as cadeias à espera de revisão)
- Ajustar os sinalizadores de tradução (veja *Personalizar o comportamento*)
- Ajustar as etiquetas de cadeias (veja *String labels*)

Dica: Esta ferramenta é chamada *Editor em massa*, acessível no menu *Ferramentas* para cada projeto, componente ou tradução.

Veja também:

Bulk edit addon

1.4 Descarregar e enviar traduções

Pode exportar ficheiros de uma tradução, fazer alterações e importá-los novamente. Isso permite trabalhar off-line e depois mesclar mudanças de volta na tradução existente. Isso funciona mesmo que tenha sido alterado entretanto.

Nota: As opções disponíveis podem ser limitadas por *Controlo de acesso*.

1.4.1 Descarregar traduções

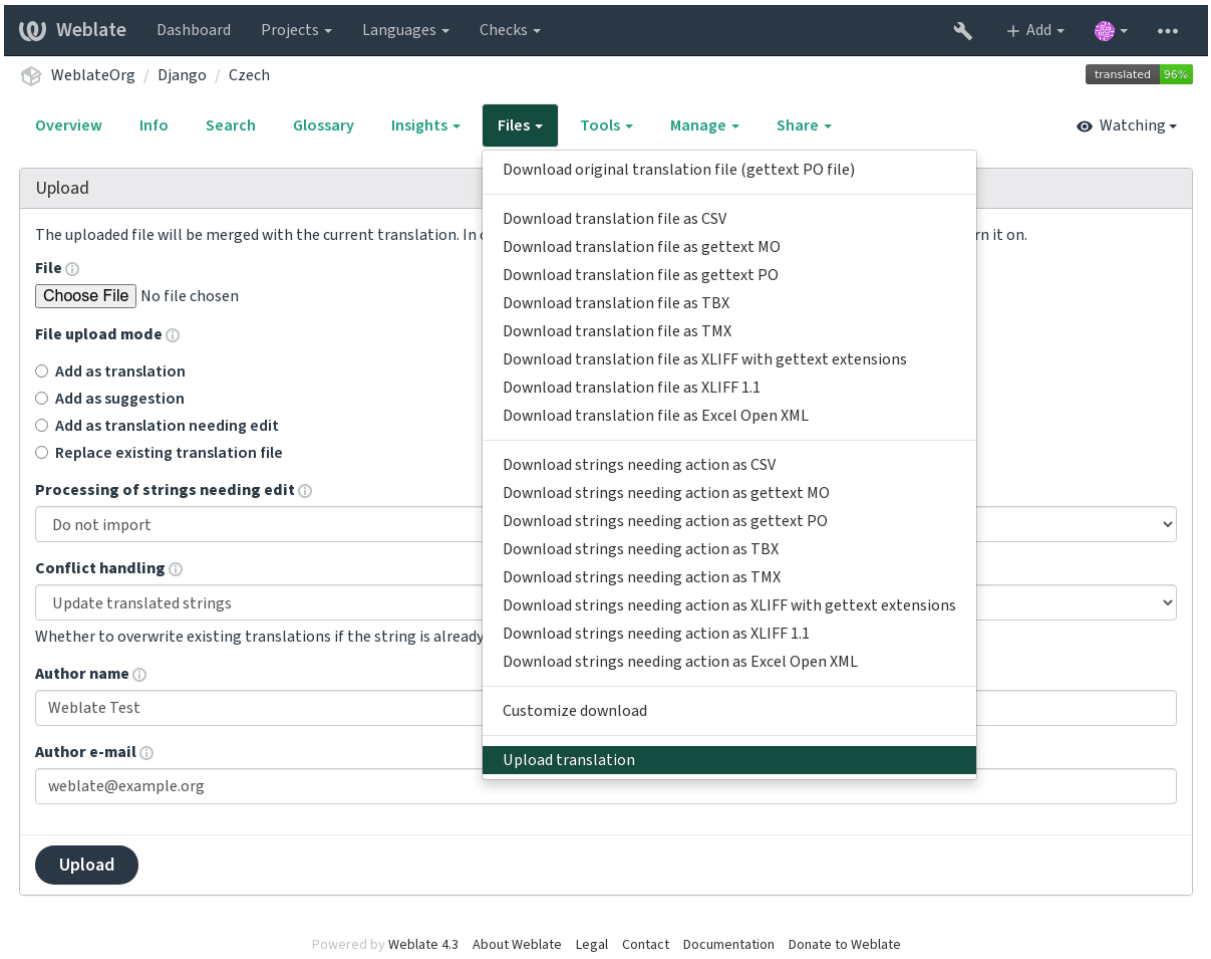
Do painel do projeto ou do componente, os ficheiros traduzíveis podem ser descarregados a usar o *Descarregar ficheiro de tradução original* no menu *Ficheiros*, produzindo uma cópia do ficheiro original à medida que ele é armazenado no sistema de controle de versão upstream.

Pode também descarregar a tradução convertida num dos formatos de localização amplamente utilizados. Os ficheiros convertidos serão enriquecidos com dados fornecidos no Weblate, como contexto adicional, comentários ou sinalizadores.

Vários formatos de ficheiro estão disponíveis, incluindo um ficheiro compilado para usar na sua escolha de aplicação (por exemplo, ficheiros `.mo` para GNU Gettext) a usar o menu *Ficheiros*.

1.4.2 Enviar traduções

Quando tiver feito as suas alterações, use *Enviar tradução* no menu *Ficheiros*.



Formatos de ficheiros suportados

Todos ficheiros num formato de ficheiro suportado pode ser enviado, mas ainda é recomendado usar o mesmo formato de ficheiro como o para a tradução, caso contrário, alguns recursos podem não ser traduzidos corretamente.

Veja também:

Formatos de ficheiros suportados

O ficheiro enviado é mesclado para atualizar a tradução, substituindo as entradas existentes por predefinição (pode ser desativado ou ativado na caixa de diálogo de envio).

Métodos de importação

Estas são as opções apresentadas ao enviar ficheiros de tradução:

Adicionar como tradução (`translate`) Traduções importadas são adicionadas como traduções. Este é o caso de uso mais comum e o comportamento predefinido.

Adicionar como sugestão (`suggest`) As traduções importadas são adicionadas como sugestões, faça isso quando quiser ter as suas cadeias enviadas serem revisadas.

Adicionar como tradução que necessita de edição («fuzzy») As traduções importadas são adicionadas como traduções que necessitam de edição. Isso pode ser útil quando quer que as traduções sejam usadas, mas também revisadas.

Substituir ficheiro de tradução existente («replace») O ficheiro existente é substituído por novo conteúdo. Isso pode levar à perda de traduções existentes, use com cuidado.

Atualizar cadeias fonte («source») Atualiza cadeias fonte em ficheiro de tradução bilíngue. Isso é semelhante ao que *Atualizar ficheiros PO para coincidir com POT (msgmerge)* faz.

Veja também:

POST /api/translations/(string:project)/(string:component)/(string:language)/file/

Gestão de conflitos

Define como lidar com cadeias enviadas que já são traduzidas.

Cadeias necessitando de edição

Há também uma opção de como lidar com cadeias que necessitam de edição no ficheiro importado. Tais cadeias podem ser manuseados de uma das três maneiras seguintes: «Não importar», «Importar como cadeia que necessita edição» ou «Importar como traduzido».

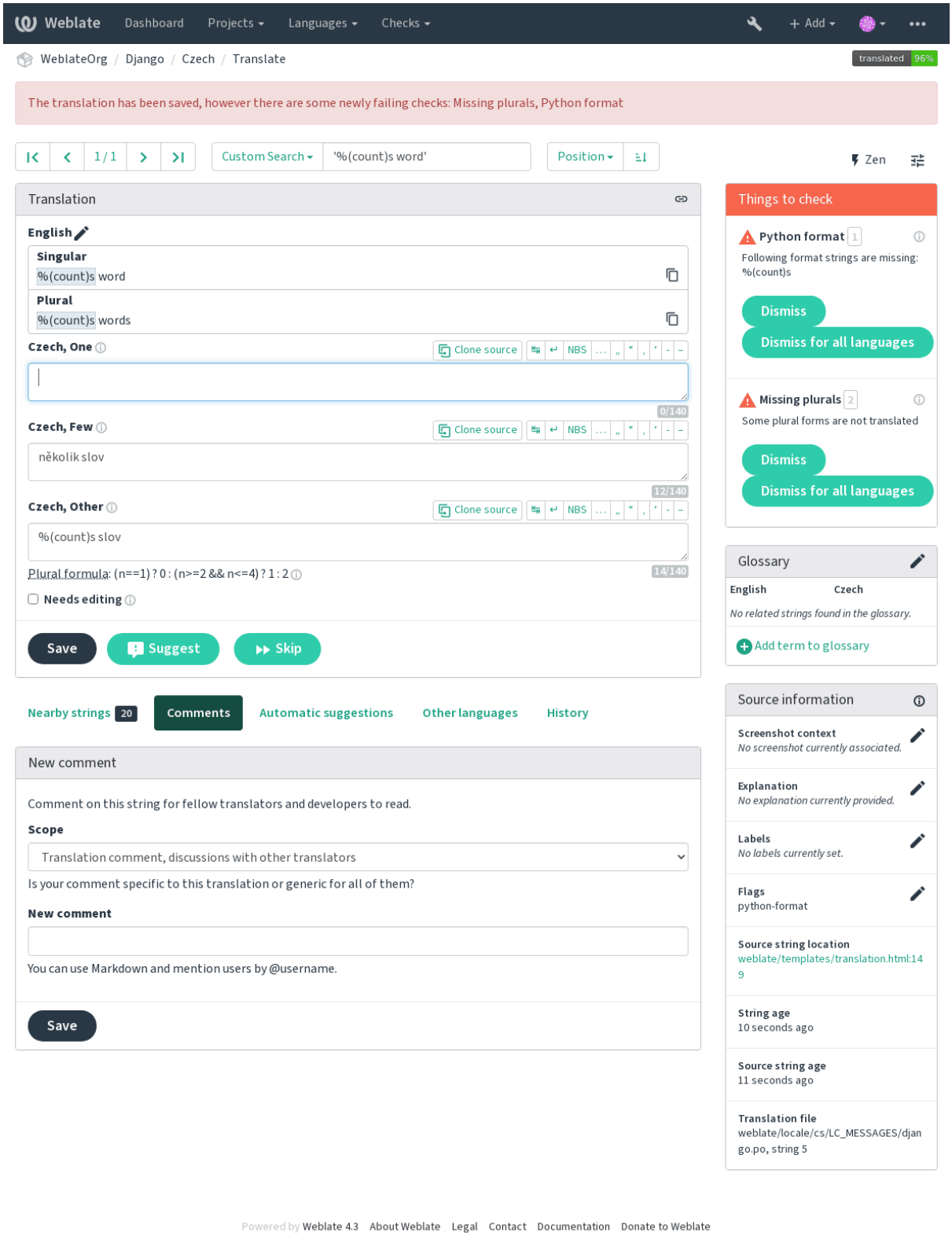
Substituindo autoria

Com permissões administrativas, também pode especificar a autoria do ficheiro enviado. Isso pode ser útil no caso de ter recebido o ficheiro de outra maneira e quiser mesclá-lo em traduções existentes enquanto credita corretamente o autor real.

1.5 Verificações e correções

As verificações de qualidade ajudam a apanhar erros comuns do tradutor, garantindo que a tradução esteja em boa forma. As verificações podem ser ignoradas em caso de falsos positivos.

Quando enviar uma tradução com uma verificação a falhar será imediatamente mostrada ao utilizador:


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Django / Czech / Translate
translated 96%
The translation has been saved, however there are some newly failing checks: Missing plurals, Python format
1 / 1
Custom Search
'%(count)s word'
Position
Zen
Translation
English
Singular
%(count)s word
Plural
%(count)s words
Czech, One
Clone source
NBS
0/140
Czech, Few
Clone source
NBS
několik slov
12/140
Czech, Other
Clone source
NBS
%(count)s slov
Plural formula: (n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2
14/140
Needs editing
Save
Suggest
Skip
Nearby strings 20
Comments
Automatic suggestions
Other languages
History
New comment
Comment on this string for fellow translators and developers to read.
Scope
Translation comment, discussions with other translators
Is your comment specific to this translation or generic for all of them?
New comment
You can use Markdown and mention users by @username.
Save
Things to check
Python format 1
Following format strings are missing: %(count)s
Dismiss
Dismiss for all languages
Missing plurals 2
Some plural forms are not translated
Dismiss
Dismiss for all languages
Glossary
English
Czech
No related strings found in the glossary.
Add term to glossary
Source information
Screenshot context
No screenshot currently associated.
Explanation
No explanation currently provided.
Labels
No labels currently set.
Flags
python-format
Source string location
weblate/templates/translation.html:149
String age
10 seconds ago
Source string age
11 seconds ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 5
Powered by Weblate 4.3
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.5.1 Correções automáticas

Além de *Verificações de qualidade*, o Weblate pode corrigir automaticamente alguns erros comuns em cadeias traduzidas. Use-o com cuidado para não causar erros por meio disto.

Veja também:

`AUTOFIX_LIST`

1.5.2 Verificações de qualidade

O Weblate emprega uma ampla gama de verificações de qualidade em cadeias. A secção a seguir descreve todos eles em mais detalhe. Há também verificações específicas de idiomas. Por favor, preencha um relatório de erro se alguma verificação for relatada por engano.

Veja também:

`CHECK_LIST`, *Personalizar o comportamento*

1.5.3 Verificações de tradução

Executado a cada alteração da tradução, ajuda os tradutores a manter traduções de boa qualidade.

Markup BBcode

BBcode na tradução não corresponde à fonte

BBCode representa marcação simples, como, por exemplo, destacar partes importantes de uma mensagem em fonte em negrito ou itálico.

Esta verificação garante que eles também estejam na tradução.

Nota: O método para detetar BBcode é atualmente bastante simples, então esta verificação pode produzir falsos positivos.

Palavras consecutivas duplicadas

O texto contém a mesma palavra duas vezes de seguida:

Novo na versão 4.1.

Verifica se não há palavras duplicadas consecutivas numa tradução. Isso geralmente indica um erro na tradução.

Dica: Esta verificação inclui regras específicas do idioma para evitar falsos positivos. Caso seja falso no seu caso, avise-nos. Veja *Reporting issues in Weblate*.

Espaço duplo

A tradução contém espaços duplos

Verifica se espaços duplos estão presentes na tradução para evitar falsos positivos em outras verificações relacionadas ao espaço.

A verificação é falsa quando espaços duplos são encontrados na fonte, o que significa que os espaços duplos são intencionais.

Cadeias formatadas

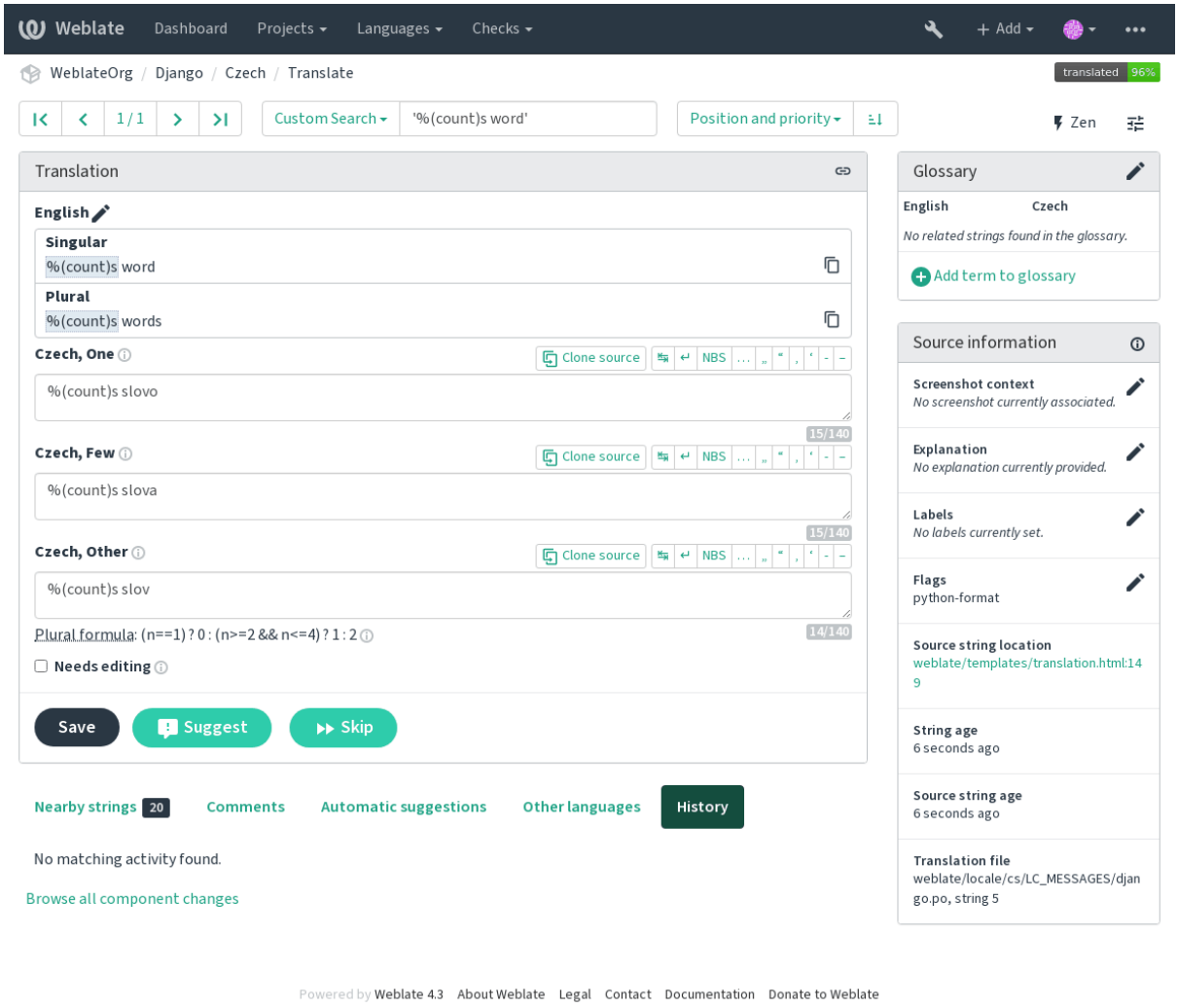
Verifica se a formatação em cadeias é replicada entre a fonte e a tradução. Omitir cadeias de formato na tradução geralmente causa problemas graves, de modo que a formatação em cadeias geralmente deve coincidir com a fonte.

O Weblate tem suporte a verificar cadeias de formato em vários idiomas. A verificação não é ativada automaticamente, somente se uma cadeia for sinalizada adequadamente (por exemplo, «c-format" para formato C). O Gettext adiciona-o automaticamente, mas provavelmente terá que adicioná-lo manualmente para outros formatos de ficheiro ou se os seus ficheiros PO não forem gerados por **`xgettext`**.

Isso pode ser feito por unidade (ver *Additional info on source strings*) na *Component configuration*. Tê-lo definido por componente é mais simples, mas pode levar a falsos positivos no caso de a cadeia não ser interpretada como uma cadeia de formatação, mas a sintaxe de textos de formato passa a ser usada.

Dica: Caso a verificação de formato específico não esteja disponível no Weblate, pode usar *Espaços reservados* genéricos.

Além de verificar, isso também destacará as cadeias de formatação para inseri-los facilmente em cadeias traduzidas:



Cadeia de interpolação AngularJS

A cadeia de interpolação AngularJS não corresponde à fonte

Cadeia de formato nomeado	`O seu saldo é {{amount}} {{ currency }}`
Sinalize para ativar	*angularjs-format*

Veja também:

AngularJS: API: $interpolate

Formato C

A cadeia de formato C não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Cadeia de formato de posição	`O seu saldo é %1$d %2$s`
Sinalize para ativar	*c-format*

Veja também:

Cadeias de formatação C, formatação de printf C

Formato C#

A cadeia de formato C# não corresponde à fonte

Cadeia de formato de posição	`Há {0} maçãs`
Sinalize para ativar	*c-sharp-format*

Veja também:

C# String Format

Literais de modelo de ECMAScript

Os literais de modelo de ECMAScript não correspondem à fonte

Interpolação	`Há ${number} maçãs`
Sinalize para ativar	*es-format*

Veja também:

Literais de modelo

Interpolação de i18next

A interpolação de i18next não corresponde à fonte

Novo na versão 4.0.

Interpolação	`Há {{number}} maçãs`
Aninhamento	`Há $t(number) maçãs`
Sinalize para ativar	*i18next-interpolation*

Veja também:

Interpolação i18next

Formato Java

A cadeia de formato Java não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Cadeia de formato de posição	`O seu saldo é %1$d %2$s`
Sinalize para ativar	*java-format*

Veja também:

Java Format Strings

Formato de Mensagem Java

A cadeia de MessageFormat de Java não corresponde à fonte

Cadeia de formato de posição	`Há {0} maçãs`
Sinalize para ativar	*java-messageformat* ativa a verificação incondicionalmente
	auto-java-messageformat ativa a verificação somente se houver uma cadeia de formato na fonte

Veja também:

MessageFormat de Java

Formato JavaScript

A cadeia de formato JavaScript não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Sinalize para ativar	*javascript-format*

Veja também:

Cadeias de formatação JavaScript

Espaços reservados de percentagem

Os espaços reservados de percentagem não correspondem à fonte

Novo na versão 4.0.

Cadeia de formato simples	`Há %number% maçãs`
Sinalize para ativar	*percent-placeholders*

Formato Perl

A cadeia de formato Perl não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Cadeia de formato de posição	`O seu saldo é %1$d %2$s`
Sinalize para ativar	*perl-format*

Veja também:

sprintf de Perl, Cadeias de Formatação Perl

Formato PHP

A cadeia de formato PHP não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Cadeia de formato de posição	`O seu saldo é %1$d %2$s`
Sinalize para ativar	*php-format*

Veja também:

Documentação de PHP sprintf, Cadeias de formato PHP

Formato de chaveta Python

A cadeia de formato de chaves Python não corresponde à fonte

Cadeia de formato simples	`Há {} maçãs`
Cadeia de formato nomeado	`O seu saldo é {amount} {currency}`
Sinalize para ativar	*python-brace-format*

Veja também:

Formato de chaves Python, Cadeias de formato Python

Formato Python

A cadeia de formato Python não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Cadeia de formato nomeado	`O seu saldo é %(amount) %(currency)`
Sinalize para ativar	*python-format*

Veja também:

Formatação de cadeias Python, Python Format Strings

Formato Qt

A cadeia de formato Qt não corresponde à fonte

Cadeia de formato de posição	`Há %1 maçãs`
Sinalize para ativar	*qt-format*

Veja também:

Qt QString::arg()

Forma plural Qt

A cadeia de formato de plural do Qt não corresponde à fonte

Cadeia de formato de plural	`Há %Ln maçã(s)`
Sinalize para ativar	*qt-plural-format*

Veja também:

Guia de i18n do Qt

Formato Ruby

A cadeia de formato Ruby não corresponde à fonte

Cadeia de formato simples	`Há %d maçãs`
Cadeia de formato de posição	`O seu saldo é %1$f %2$s`
Cadeia de formato nomeado	`O seu saldo é %+.2<amount>f %<currency>s`
Cadeia de modelo nomeado	`O seu saldo é %{amount} %{currency}`
Sinalize para ativar	*ruby-format*

Veja também:

Ruby Kernel#sprintf

Formatação vue I18n

A formatação Vue I18n não corresponde com a fonte

Formatação nomeada	`Há {count} maçãs`
Formatação i18n de Rails	`Há %{count} maçãs`
Mensagens de localidade vinculadas	`@:message.dio @:message.the_world!`
Sinalize para ativar	*vue-format*

Veja também:

Vue I18n Formatting, Vue I18n Linked locale messages

Foi traduzido

Esta cadeia foi traduzida no passado

Significa que uma cadeia já foi traduzida. Isso pode acontecer quando as traduções foram revertidas no VCS ou perdidas de outra forma.

Inconsistente

Esta cadeia tem mais que uma tradução neste projeto ou não é traduzida em alguns componentes.

O Weblate verifica traduções da mesma cadeia em todas as traduções de um projeto para ajudar a manter traduções consistentes.

A verificação falha em traduções diferentes de uma cadeia dentro de um projeto. Isso também pode levar a inconsistências nas verificações exibidas. Pode encontrar outras traduções desta cadeia na guia *Outras ocorrências*.

Nota: Esta verificação também é disparada no caso de a cadeia estar traduzida num componente e não em outro. Ela pode ser usada como uma maneira rápida de manusear manualmente cadeias que não estão traduzidas em alguns componentes apenas clicando no botão *Usar esta tradução* exibido em cada linha na guia *Outras ocorrências*.

Pode usar *Tradução automática* para automatizar a tradução de cadeias recém-adicionadas que já são traduzidas em outro componente.

Veja também:

Manter traduções iguais entre componentes

Letra Kashida utilizada

As letras kashida decorativas não devem ser usadas

Novo na versão 3.5.

As letras Kashida decorativas não devem ser usadas na tradução. Estas também são conhecidas como Tatweel.

Veja também:

Kashida na Wikipédia

Hiperligações de marcação

Markdown links do not match source

Novo na versão 3.5.

Markdown links do not match source.

Veja também:

Markdown links

Referências de Markdown

Markdown link references do not match source

Novo na versão 3.5.

Markdown link references do not match source.

Veja também:

Markdown links

Sintaxe de Markdown

Markdown syntax does not match source

Novo na versão 3.5.

A sintaxe de Markdown não coincide com a fonte

Veja também:

Markdown span elements

Tamanho máximo da tradução

Translation should not exceed given length

Checks that translations are of acceptable length to fit available space. This only checks for the length of translation characters.

Unlike the other checks, the flag should be set as a `key:value` pair like `max-length:100`.

Dica: This check looks at number of chars, what might not be the best metric when using proportional fonts to render the text. The *Tamanho máximo da tradução* check does check actual rendering of the text.

The `replacements:` flag might be also useful to expand placeables before checking the string.

Tamanho máximo da tradução

Translation rendered text should not exceed given size

Novo na versão 3.7.

Translation rendered text should not exceed given size. It renders the text with line wrapping and checks if it fits into given boundaries.

This check needs one or two parameters - maximal width and maximal number of lines. In case the number of lines is not provided, one line text is considered.

You can also configure used font by `font-*` directives (see *Personalizar o comportamento*), for example following translation flags say that the text rendered with ubuntu font size 22 should fit into two lines and 500 pixels:

```
max-size:500:2, font-family:ubuntu, font-size:22
```

Dica: You might want to set `font-*` directives in *Component configuration* to have the same font configured for all strings within a component. You can override those values per string in case you need to customize it per string.

The `replacements:` flag might be also useful to expand placeables before checking the string.

Veja também:

Gerir letras, *Personalizar o comportamento*, *Tamanho máximo da tradução*

\n não correspondente

Quantidade de \n na tradução não corresponde à da fonte

Usually escaped newlines are important for formatting program output. Check fails if the number of `\n` literals in translation do not match the source.

Dois pontos não correspondentes

Source and translation do not both end with a colon

Checks that colons are replicated between both source and translation. The presence of colons is also checked for various languages where they do not belong (Chinese or Japanese).

Veja também:

Colon on Wikipedia

Reticências não correspondentes

Source and translation do not both end with an ellipsis

Checks that trailing ellipses are replicated between both source and translation. This only checks for real ellipsis (…) not for three dots (`...`).

An ellipsis is usually rendered nicer than three dots in print, and sounds better with text-to-speech.

Veja também:

Ellipsis on Wikipedia

Ponto de exclamação não correspondente

Source and translation do not both end with an exclamation mark

Checks that exclamations are replicated between both source and translation. The presence of exclamation marks is also checked for various languages where they do not belong (Chinese, Japanese, Korean, Armenian, Limbu, Myanmar or Nko).

Veja também:

Exclamation mark on Wikipedia

Ponto final não correspondente

Source and translation do not both end with a full stop

Checks that full stops are replicated between both source and translation. The presence of full stops is checked for various languages where they do not belong (Chinese, Japanese, Devanagari or Urdu).

Veja também:

Full stop on Wikipedia

Ponto de interrogação não correspondente

A fonte e a tradução não terminam ambas com um ponto de interrogação

Checks that question marks are replicated between both source and translation. The presence of question marks is also checked for various languages where they do not belong (Armenian, Arabic, Chinese, Korean, Japanese, Ethiopic, Vai or Coptic).

Veja também:

Question mark on Wikipedia

Ponto e vírgula não correspondente

Source and translation do not both end with a semicolon

Checks that semicolons at the end of sentences are replicated between both source and translation. This can be useful to keep formatting of entries such as desktop files.

Veja também:

Semicolon on Wikipedia

Quebras de linha não coincidentes

Number of new lines in translation does not match source

Usually newlines are important for formatting program output. Check fails if the number of `\n` literals in translation do not match the source.

Faltam plurais

Some plural forms are not translated

Checks that all plural forms of a source string have been translated. Specifics on how each plural form is used can be found in the string definition.

Failing to fill in plural forms will in some cases lead to displaying nothing when the plural form is in use.

Espaços reservados

Translation is missing some placeholders:

Novo na versão 3.9.

Alterado na versão 4.3: Pode usar expressões regulares como espaço reservado.

Translation is missing some placeholders. These are either extracted from the translation file or defined manually using `placeholders` flag, more can be separated with colon, strings with space can be quoted:

```
placeholders:$URL$:$TARGET$:"some long text"
```

In case you have some syntax for placeholders, you can use an regular expression:

```
placeholders:r"%[^% ]%"
```

Veja também:

Personalizar o comportamento

Espaçamento da pontuação

Missing non breakable space before double punctuation sign

Novo na versão 3.9.

Checks that there is non breakable space before double punctuation sign (exclamation mark, question mark, semicolon and colon). This rule is used only in a few selected languages like French or Breton, where space before double punctuation sign is a typographic rule.

Veja também:

French and English spacing on Wikipedia

Expressão regular

Translation does not match regular expression:

Novo na versão 3.9.

Translation does not match regular expression. The expression is either extracted from the translation file or defined manually using `regex` flag:

```
regex:^foo|bar$
```

Mesmos plurais

Some plural forms are translated in the same way

Check that fails if some plural forms are duplicated in the translation. In most languages they have to be different.

Nova linha no início

Source and translation do not both start with a newline

Newlines usually appear in source strings for good reason, omissions or additions can lead to formatting problems when the translated text is put to use.

Veja também:

Nova linha no final

Espaços no início

Source and translation do not both start with same number of spaces

A space in the beginning of a string is usually used for indentation in the interface and thus important to keep.

Nova linha no final

Source and translation do not both end with a newline

Newlines usually appear in source strings for good reason, omissions or additions can lead to formatting problems when the translated text is put to use.

Veja também:

Nova linha no início

Espaço no final

Source and translation do not both end with a space

Checks that trailing spaces are replicated between both source and translation.

Trailing space is usually utilized to space out neighbouring elements, so removing it might break layout.

Tradução inalterada

Source and translation are identical

Happens if the source and corresponding translation strings is identical, down to at least one of the plural forms. Some strings commonly found across all languages are ignored, and various markup is stripped. This reduces the number of false positives.

This check can help find strings mistakenly untranslated.

The default behavior of this check is to exclude words from the built-in blacklist from the checking. These are words which are frequently not being translated. This is useful to avoid false positives on short strings, which consist only of single word which is same in several languages. This blacklist can be disabled by adding `strict-same` flag to string or component.

Veja também:

Component configuration, *Personalizar o comportamento*

HTML inseguro

The translation uses unsafe HTML markup

Novo na versão 3.9.

The translation uses unsafe HTML markup. This check has to be enabled using `safe-html` flag (see *Personalizar o comportamento*). There is also accompanied autofixer which can automatically sanitize the markup.

Veja também:

The HTML check is performed by the Bleach library developed by Mozilla.

URL

The translation does not contain an URL

Novo na versão 3.5.

The translation does not contain an URL. This is triggered only in case the unit is marked as containing URL. In that case the translation has to be a valid URL.

Markup XML

XML tags in translation do not match source

This usually means the resulting output will look different. In most cases this is not a desired result from changing the translation, but occasionally it is.

Checks that XML tags are replicated between both source and translation.

Sintaxe XML

The translation is not valid XML

Novo na versão 2.8.

The XML markup is not valid.

Espaçamento nulo

Translation contains extra zero-width space character

Zero-width space (<U+200B>) characters are used to break messages within words (word wrapping).

As they are usually inserted by mistake, this check is triggered once they are present in translation. Some programs might have problems when this character is used.

Veja também:

Zero width space on Wikipedia

1.5.4 Source checks

Source checks can help developers improve the quality of source strings.

Reticências

The string uses three dots (…) instead of an ellipsis character (…)

This fails when the string uses three dots (`...`) when it should use an ellipsis character (…).

Using the Unicode character is in most cases the better approach and looks better rendered, and may sound better with text-to-speech.

Veja também:

Ellipsis on Wikipedia

Não traduzido há muito tempo

The string has not been translated for a long time

Novo na versão 4.1.

When the string has not been translated for a long time, it is can indicate problem in a source string making it hard to translate.

Várias verificações falhadas

The translations in several languages have failing checks

Numerous translations of this string have failing quality checks. This is usually an indication that something could be done to improve the source string.

This check failing can quite often be caused by a missing full stop at the end of a sentence, or similar minor issues which translators tend to fix in translation, while it would be better to fix it in the source string.

Várias variáveis sem nome

There are multiple unnamed variables in the string, making it impossible for translators to reorder them

Novo na versão 4.1.

There are multiple unnamed variables in the string, making it impossible for translators to reorder them.

Consider using named variables instead to allow translators to reorder them.

Não pluralizado

The string is used as plural, but not using plural forms

The string is used as a plural, but does not use plural forms. In case your translation system supports this, you should use the plural aware variant of it.

For example with Gettext in Python it could be:

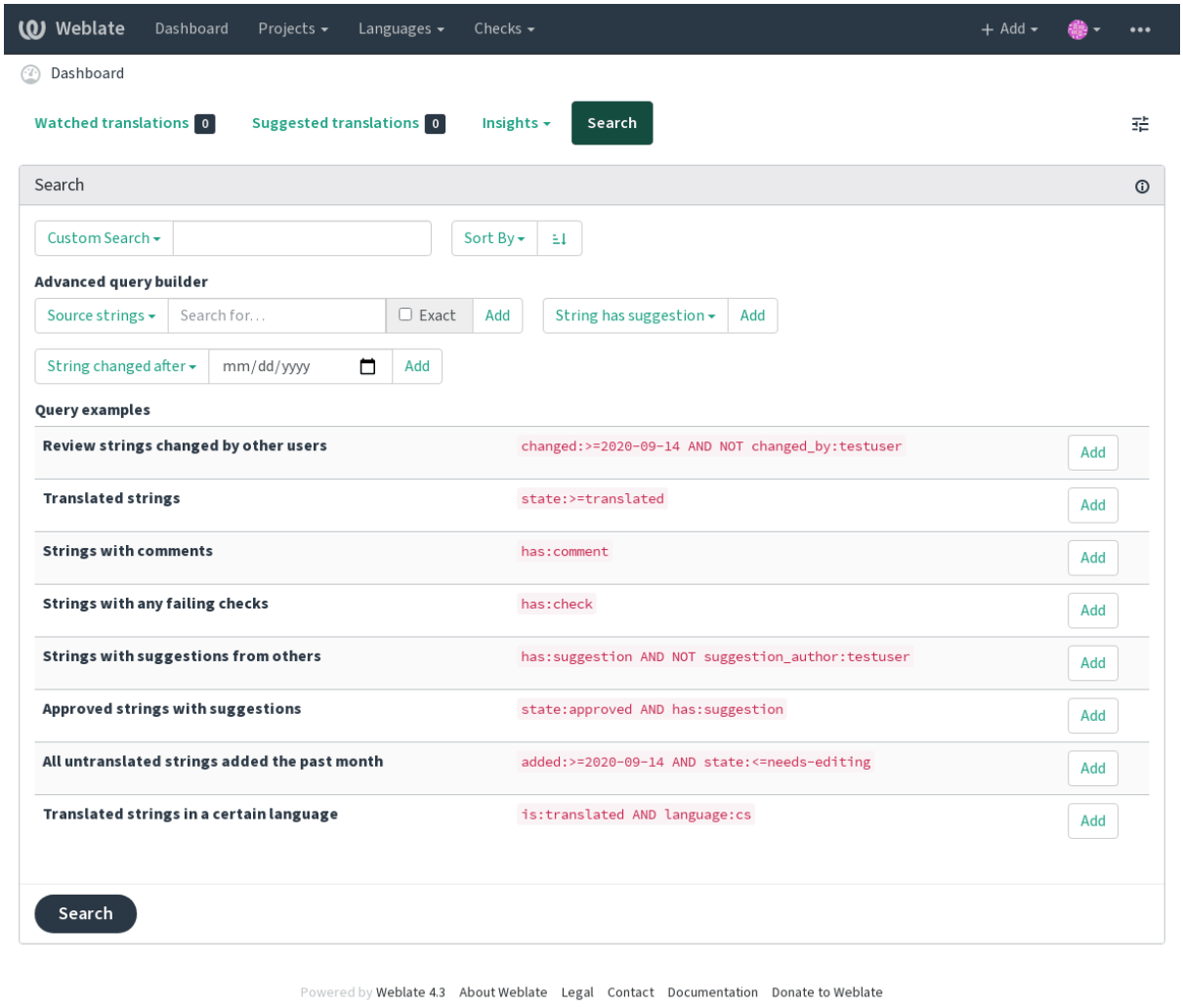
```
from gettext import ngettext

print ngettext('Selected %d file', 'Selected %d files', files) % files
```

1.6 Searching

Novo na versão 3.9.

Advanced queries using boolean operations, parentheses, or field specific lookup can be used to find the strings you want.

When no field is defined, the lookup happens on *Source*, *Translate* and *Context* fields.



1.6.1 Simple search

Any phrase typed into the search box is split into words. Strings containing any of them are shown. To look for an exact phrase, put «the searchphrase» into quotes (both single (“) and double («) quotes will work): `"this is a quoted string"` or `'another quoted string'`.

1.6.2 Fields

`source:TEXT` Source string case insensitive search.

`target:TEXT` Target string case insensitive search.

`context:TEXT` Context string case insensitive search.

`key:TEXT` Key string case insensitive search.

`note:TEXT` Comment string case insensitive search.

`location:TEXT` Location string case insensitive search.

`priority:NUMBER` String priority.

`added:DATETIME` Timestamp for when the string was added to Weblate.

`state:TEXT` State search (`approved`, `translated`, `needs-editing`, `empty`, `read-only`), supports *Field operators*.

pending:BOOLEAN String pending for flushing to VCS.

has:TEXT Search for string having attributes - `plural`, `context`, `suggestion`, `comment`, `check`, `dismissed-check`, `translation`, `variant`, `screenshot` (works only on source strings).

is:TEXT Search for string states (`pending`, `translated`, `untranslated`).

language:TEXT String target language.

component:TEXTO Slug de componente, veja: ref: *componente-slug*.

project:TEXTO Project slug, see *Project slug*.

changed_by:TEXT String was changed by author with given username.

changed:DATETIME String was changed on date, supports *Field operators*.

check:TEXT String has failing check.

dismissed_check:TEXT String has dismissed check.

comment:TEXT Search in user comments.

comment_author:TEXT Filter by comment author.

suggestion:TEXT Search in suggestions.

suggestion_author:TEXT Filter by suggestion author.

1.6.3 Boolean operators

You can combine lookups using `AND`, `OR`, `NOT` and parentheses to form complex queries. For example: `state:translated AND (source:hello OR source:bar)`

1.6.4 Field operators

You can specify operators, ranges or partial lookups for date or numeric searches:

state:>=translated State is `translated` or better (`approved`).

changed:2019 Changed in year 2019.

changed:[2019-03-01 to 2019-04-01] Changed between two given dates.

1.6.5 Exact operators

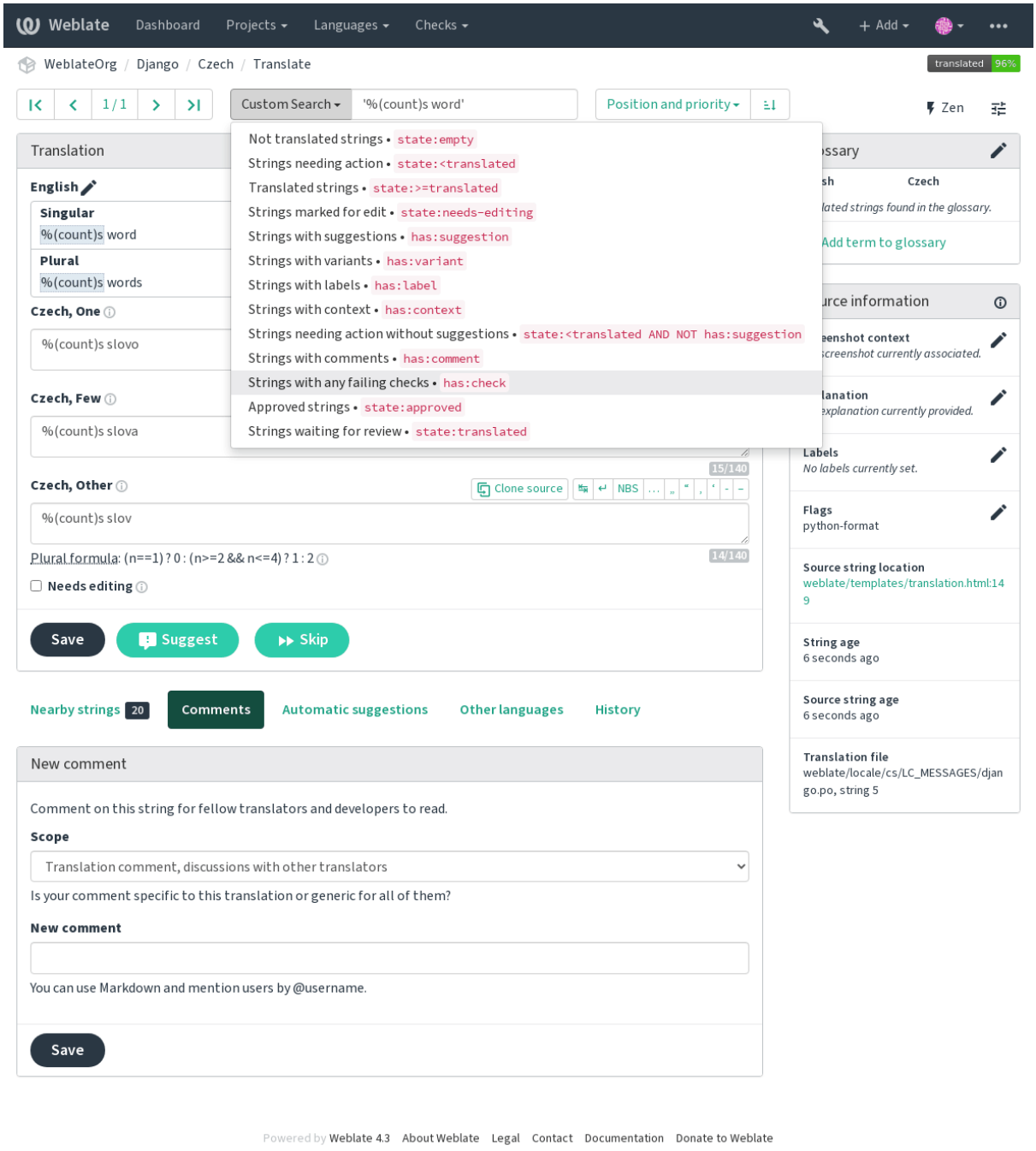
You can do an exact match query on different string fields using = operator. For example, to search for all source strings exactly matching `hello world`, use: `source:="hello world"`. For searching single word expressions, you can skip quotes. For example, to search for all source strings matching `hello`, you can use: `source:=hello`.

1.6.6 Regular expressions

Anywhere text is accepted you can also specify a regular expression as `r"regexp"`. For instance, to search for all source strings which contain any digit between 2 and 5, use: `source:r"[2-5]"`

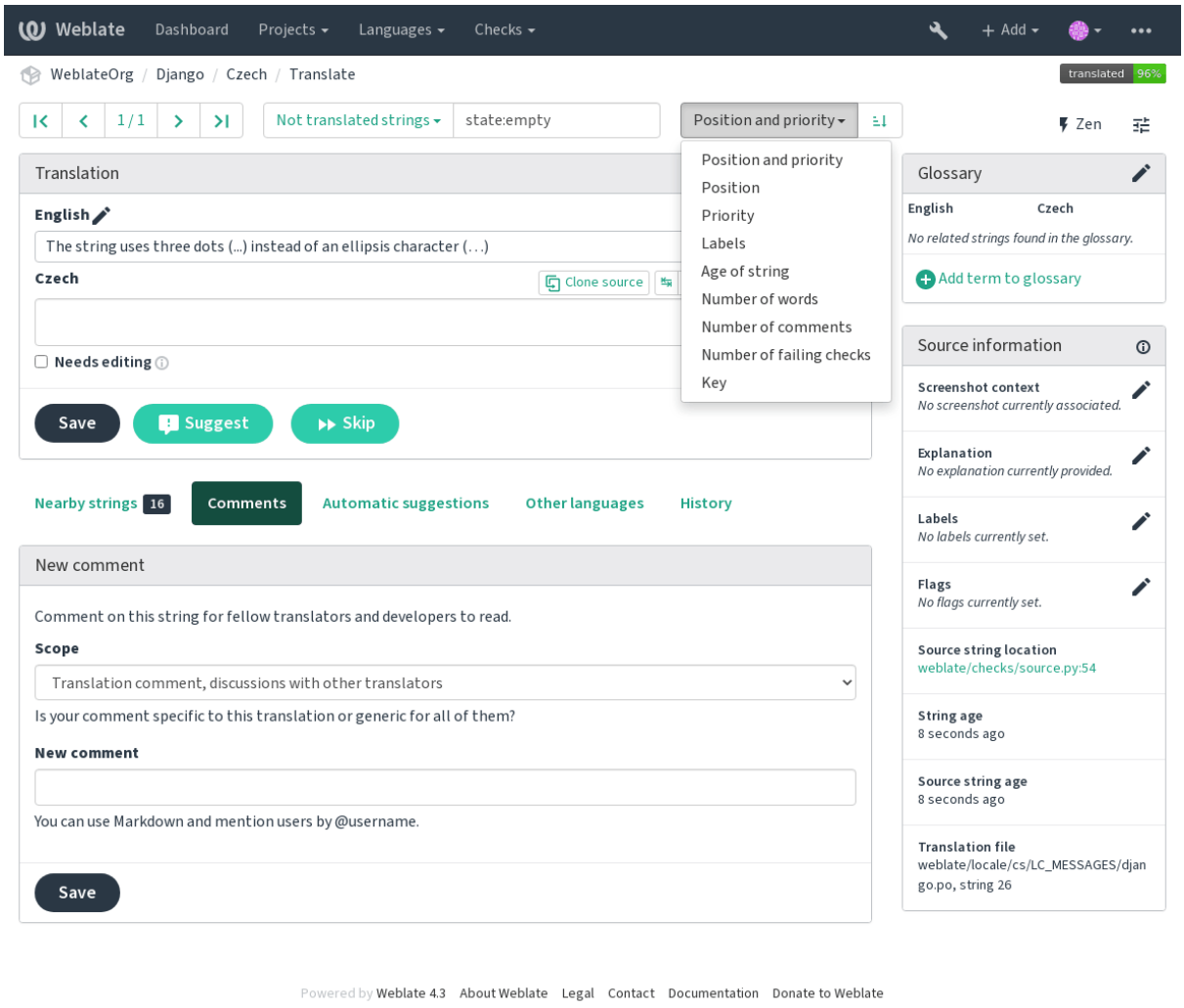
1.6.7 Predefined queries

You can select out of predefined queries on the search page, this allows you to quickly access the most frequent searches:



1.6.8 Ordering the results

There are many options to order the strings according to your needs:



1.7 Application developer guide

Using Weblate is a process that brings your users closer to you, by bringing you closer to your translators. It is up to you to decide how many of its features you want to make use of.

1.7.1 Starting with internationalization

Have a project and want to translate it into several languages? This guide will help you do so. Several typical situations are showcased, but most of the examples are generic and can be applied to other scenarios as well.

Before translating any software, you should realize that languages around the world are really different and you should not make any assumption based on your experience. For most of languages it will look weird if you try to concatenate a sentence out of translated segments. You also should properly handle plural forms because many languages have complex rules for that and the internationalization framework you end up using should support this.

Last but not least, sometimes it might be necessary to add some context to the translated string. Imagine a translator would get string `Sun` to translate. Without context most people would translate that as our closest star, but it might be actually used as an abbreviation for Sunday.

Choosing internationalization framework

Choose whatever is standard on your platform, try to avoid reinventing the wheel by creating your own framework to handle localizations. Weblate supports most of the widely used frameworks, see *Formatos de ficheiros suportados* for more information (especially *Translation types capabilities*).

Our personal recommendation for some platforms is in the following table. This is based on our experience, but that can not cover all use cases, so always consider your environment when doing the choice.

Platform	Recommended format
Android	*Android string resources*
iOS	*Apple iOS strings*
Qt	*Qt Linguist .ts*
Python	*GNU gettext*
PHP	*GNU gettext*[1]
C/C++	*GNU gettext*
C#	*.XML resource files*
Perl	*GNU gettext*
Ruby	*Ruby YAML files*
Web extensions	*WebExtension JSON*
Java	*XLIFF*[2]
JavaScript	*JSON i18next files*[3]

The more detailed workflow for some formats is described in following chapters:

- *Translating software using GNU Gettext*
- *Translating documentation using Sphinx*
- *Traduzir HTML e JavaScript a usar CDN Weblate*

Integrating with Weblate

Getting translations updates from Weblate

To fetch updated strings from Weblate you can simply fetch the underlying repository (either from filesystem or it can be made available through *Git exporter*). Prior to this, you might want to commit any pending changes (see *Commits adiados*). This can be achieved in the user interface (in the *Repository maintenance*) or from command line using *Cliente Weblate*.

This can be automated if you grant Weblate push access to your repository and configure *Push URL* in the *Component configuration*.

Veja também:

Tradução contínua

[1] The native Gettext support in PHP is buggy and often missing on Windows builds, it is recommended to use third party library motranslator instead.

[2] You can also use *Java properties* if plurals are not needed.

[3] You can also use plain *JSON files* if plurals are not needed.

Pushing string changes to Weblate

To push newly updated strings to Weblate, just let it pull from the upstream repository. This can be achieved in the user interface (in the *Repository maintenance*) or from command line using *Cliente Weblate*.

This can be automated by installing a webhook on your repository to trigger Weblate whenever there is a new commit, see *Atualizar repositórios* for more details.

Veja também:

Tradução contínua

1.7.2 Translating software using GNU Gettext

GNU Gettext is one of the most widely used tool for internationalization of free software. It provides a simple yet flexible way to localize the software. It has great support for plurals, it can add further context to the translated string and there are quite a lot of tools built around it. Of course it has great support in Weblate (see *GNU gettext* file format description).

Nota: If you are about to use it in proprietary software, please consult licensing first, it might not be suitable for you.

GNU Gettext can be used from a variety of languages (C, Python, PHP, Ruby, JavaScript and many more) and usually the UI frameworks already come with some support for it. The standard usage is through the *gettext()* function call, which is often aliased to *_()* to make the code simpler and easier to read.

Additionally it provides *pgettext()* call to provide additional context to translators and *ngettext()* which can handle plural types as defined for target language.

As a widely spread tool, it has many wrappers which make its usage really simple, instead of manual invoking of Gettext described below, you might want to try one of them, for example intltool.

Sample program

The simple program in C using Gettext might look like following:

```
#include <libintl.h>
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int count = 1;
    setlocale(LC_ALL, "");
    bindtextdomain("hello", "/usr/share/locale");
    textdomain("hello");
    printf(
        ngettext(
            "Orangutan has %d banana.\n",
            "Orangutan has %d bananas.\n",
            count
        ),
        count
    );
    printf("%s\n", gettext("Thank you for using Weblate."));
    exit(0);
}
```

Extracting translatable strings

Once you have code using the gettext calls, you can use **xgettext** to extract messages from it and store them into a .pot:

```
$ xgettext main.c -o po/hello.pot
```

Nota: There are alternative programs to extract strings from the code, for example pybabel.

This creates a template file, which you can use for starting new translations (using **msginit**) or updating existing ones after code change (you would use **msgmerge** for that). The resulting file is simply a structured text file:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-10-23 11:02+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=INTEGER; plural=EXPRESSION;\n"

#: main.c:14
#, c-format
msgid "Orangutan has %d banana.\n"
msgid_plural "Orangutan has %d bananas.\n"
msgstr[0] ""
msgstr[1] ""

#: main.c:20
msgid "Thank you for using Weblate."
msgstr ""
```

Each `msgid` line defines a string to translate, the special empty string in the beginning is the file header containing metadata about the translation.

Starting new translation

With the template in place, we can start our first translation:

```
$ msginit -i po/hello.pot -l cs --no-translator -o po/cs.po
Created cs.po.
```

The just created `cs.po` already has some information filled in. Most importantly it got the proper plural forms definition for chosen language and you can see number of plurals have changed according to that:

```
# Czech translations for PACKAGE package.
# Copyright (C) 2015 THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
```

(continues on next page)

(continuação da página anterior)

```
# Automatically generated, 2015.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-10-23 11:02+0200\n"
"PO-Revision-Date: 2015-10-23 11:02+0200\n"
"Last-Translator: Automatically generated\n"
"Language-Team: none\n"
"Language: cs\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=ASCII\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;\n"

#: main.c:14
#, c-format
msgid "Orangutan has %d banana.\n"
msgid_plural "Orangutan has %d bananas.\n"
msgstr[0] ""
msgstr[1] ""
msgstr[2] ""

#: main.c:20
msgid "Thank you for using Weblate."
msgstr ""
```

This file is compiled into an optimized binary form, the .mo file used by the GNU Gettext functions at runtime.

Updating strings

Once you add more strings or change some strings in your program, you execute again **`xgettext`** which regenerates the template file:

```
$ xgettext main.c -o po/hello.pot
```

Then you can update individual translation files to match newly created templates (this includes reordering the strings to match new template):

```
$ msgmerge --previous --update po/cs.po po/hello.pot
```

Importing to Weblate

To import such translation into Weblate, all you need to define are the following fields when creating component (see *Component configuration* for detailed description of the fields):

Field	Value
Repositório do código-fonte	URL of the VCS repository with your project
File mask	`po/*.po`
Modelo para novas traduções	`po/hello.pot`
Formato de ficheiro	Choose *Gettext PO file*
Novo idioma	Choose *Create new language file*

And that's it, you're now ready to start translating your software!

Veja também:

You can find a Gettext example with many languages in the Weblate Hello project on GitHub: <https://github.com/WeblateOrg/hello>.

1.7.3 Translating documentation using Sphinx

Sphinx is a tool for creating beautiful documentation. It uses simple reStructuredText syntax and can generate output in many formats. If you're looking for an example, this documentation is also built using it. The very useful companion for using Sphinx is the Read the Docs service, which will build and publish your documentation for free.

I will not focus on writing documentation itself, if you need guidance with that, just follow instructions on the Sphinx website. Once you have documentation ready, translating it is quite easy as Sphinx comes with support for this and it is quite nicely covered in their Internationalization. It's matter of few configuration directives and invoking of the `sphinx-intl` tool.

If you are using Read the Docs service, you can start building translated documentation on the Read the Docs. Their Localization of Documentation covers pretty much everything you need - creating another project, set its language and link it from main project as a translation.

Now all you need is translating the documentation content. Sphinx generates PO file for each directory or top level file, what can lead to quite a lot of files to translate (depending on `gettext_compact` settings). You can import the `index.po` into Weblate as an initial component and then configure *Descoberta de componentes* addon to automatically discover all others.

Table 1: Component configuration

Nome do componente	`Documentação`
File mask	`docs/locales/*/LC_MESSAGES/index.po`
Modelo para novas traduções	`docs/locales/index.pot`
Formato de ficheiro	*ficheiro gettext PO*
Marcadores de tradução	`rst-text`

Table 2: Configuração da descoberta de componentes

Expressão regular para corresponder ficheiros de tradução	`docs/locales/(?P<language>[^/.]*)/LC_MESSAGES/(?P<component>[^/]*)\.po`
Personalizar nome do componente	`Documentation: {{ component|title }}`
Defina o ficheiro base para as novas traduções	`docs/locales/{{ component }}.pot`

Dica: Would you prefer Sphinx to generate just single PO file? There is a hacky way to achieve this (used by Weblate documentation) by overriding Sphinx way to get a Gettext domain of a document. Place following snippet to your Sphinx configuration in `conf.py`:

```
import sphinx.transforms.i18n
import sphinx.util.i18n

# Hacky way to have all localized content in single domain
sphinx.transforms.i18n.docname_to_domain = (
    sphinx.util.i18n.docname_to_domain
) = lambda docname, compact: "docs"
```

This might be directly supported by Sphinx in future releases, see <https://github.com/sphinx-doc/sphinx/issues/784>.

Veja também:

The Odorik python module documentation is built using Sphinx, Read the Docs and translated using Weblate.

1.7.4 Traduzir HTML e JavaScript a usar CDN Weblate

Starting with Weblate 4.2 it is possible to export localization to a CDN using *CDN de localização JavaScript* addon.

Nota: Este recurso é configurado no Hosted Weblate. Requer configuração adicional na sua instalação, veja `LOCALIZE_CDN_URL` e `LOCALIZE_CDN_PATH`.

Upon installation into your component it will push committed translations (see *Commits adiados*) to the CDN and these can be used in your web pages to localize them.

Criar componente

First, you need to create a monolingual component which will hold your strings, see *Adding translation projects and components* for generic instructions on that.

In case you have existing repository to start with (for example the one containing HTML files), create an empty JSON file in the repository for the source language (see *Idioma fonte*), for example `locales/en.json`. The content should be `{}` to indicate an empty object. Once you have that, the repository can be imported into Weblate and you can start with an addon configuration.

Dica: In case you have existing translations, you can place them into the language JSON files and those will be used in Weblate.

For those who do not want to use existing repository (or do not have one), choose *Start from scratch* when creating component and choose *JSON file* as a file format (it is okay to choose any monolingual format at this point).

Configurar extensão de Weblate CDN

The *CDN de localização JavaScript* addon provides few configuration options.

Limiar de tradução Translations translated above this threshold will be included in the CDN.

Seletor de CSS Configures which strings from the HTML documents are translatable, see *Extração de cadeias para CDN Weblate* and *Localização de HTML a usar CDN Weblate*.

Nome do cookie do idioma Name of cookie which contains user selected language. Used in the JavaScript snippet for *Localização de HTML a usar CDN Weblate*.

Extrair cadeias de ficheiros de HTML List of files in the repository or URLs where Weblate will look for translatable strings and offer them for a translation, see *Extração de cadeias para CDN Weblate*.

Extração de cadeias para CDN Weblate

As cadeias de tradução devem estar presentes no Weblate. Pode gerí-los manualmente, usar API para criá-los ou listar ficheiros ou URLs a usar *Extrair cadeias de ficheiros HTML* e o Weblate irá extraí-los automaticamente. Os ficheiros devem apresentar no repositório ou conter URLs remotas que serão descarregadas e analisadas regularmente pelo Weblate.

A configuração predefinida para *Seletor CSS* extrai elementos com classe CSS `l10n`. Por exemplo, extrairia duas cadeias dos seguintes trechos:

```
<section class="content">
    <div class="row">
        <div class="wrap">
            <h1 class="section-title min-m l10n">Maintenance in progress</h1>
            <div class="page-desc">
                <p class="l10n">We're sorry, but this site is currently down for␣
↪maintenance.</p>
```

(continues on next page)

(continuação da página anterior)

```
            </div>
        </div>
    </div>
</section>
```

Caso não deseje modificar o código existente, também pode usar * como um seletor para processar todos os elementos.

Nota: No momento, apenas o texto dos elementos é extraído. Este complemento não suporta a localização de atributos de elementos ou elementos com filhos.

Localização de HTML a usar CDN Weblate

Para localizar um documento HTML, precisa carregar o script `weblate.js`:

```
<script src="https://weblate-cdn.com/a5ba5dc29f39498aa734528a54b50d0a/weblate.js"␣
↪async></script>
```

Ao carregar, isto encontrará todos os elementos traduzíveis correspondentes automaticamente (com base na configuração *Seletor CSS*) e substituirá o texto deles por uma tradução.

O idioma do utilizador é detetado do cookie configurado e volta aos idiomas preferidos do utilizador configurados no navegador.

The *Language cookie name* can be useful for integration with other applications (for example choose `django_language` when using Django).

Localização de JavaScript

As traduções individuais são expostas como ficheiros JSON bilingues sob o CDN. Para buscar um, pode usar o seguinte código:

```
fetch(("https://weblate-cdn.com/a5ba5dc29f39498aa734528a54b50d0a/cs.json")
  .then(response => response.json())
  .then(data => console.log(data));
```

A lógica de localização real precisa ser implementada neste caso.

1.7.5 Translation component alerts

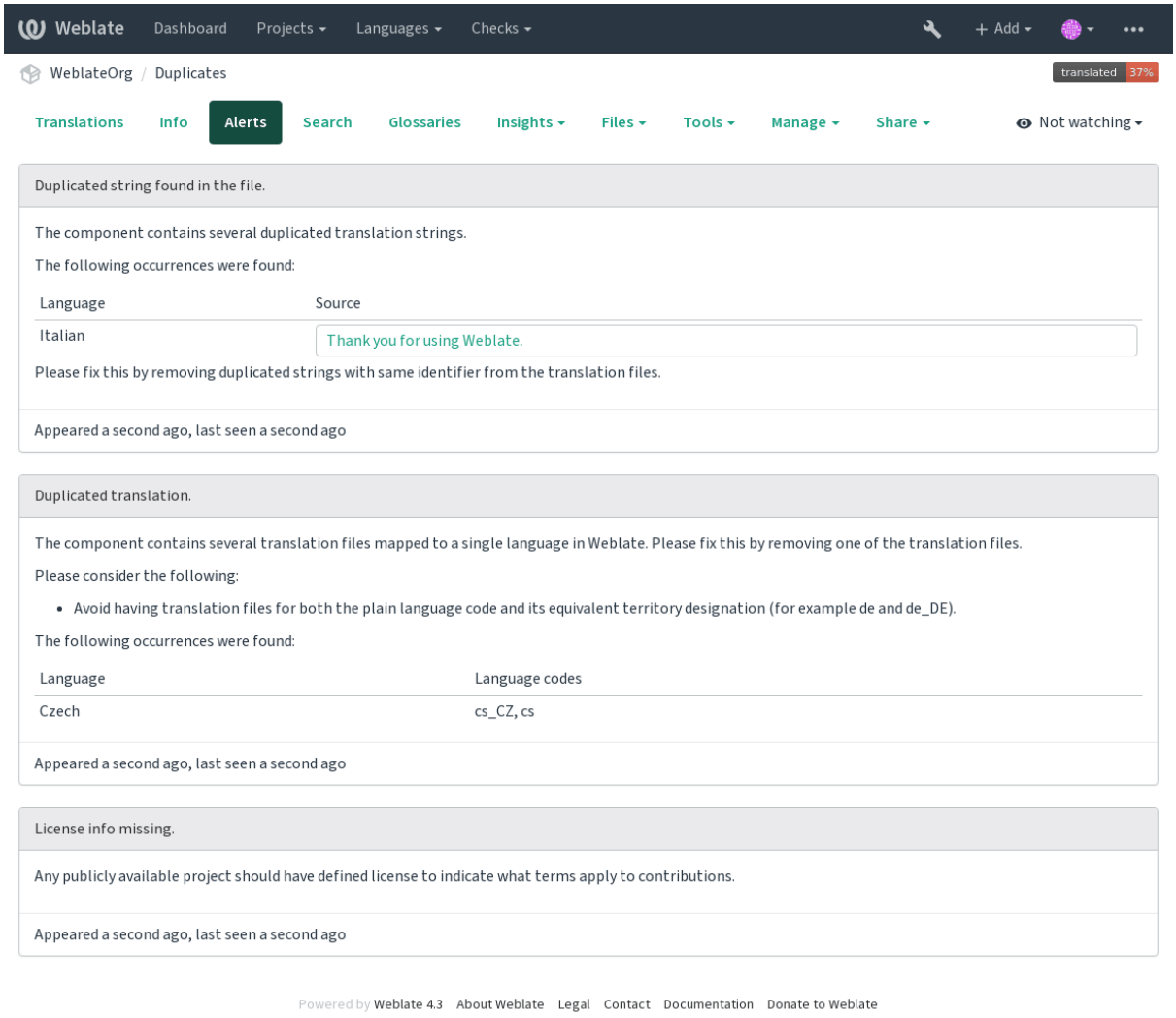
Shows errors in the Weblate configuration or the translation project for any given translation component. Guidance on how to address found issues is also offered.

Currently the following is covered:

- Duplicated source strings in translation files
- Duplicated languages within translations
- Merge or update failures in the source repository
- Unused new base in component settings
- Parse errors in the translation files
- Duplicate filemask used for linked components
- Broken URLs
- Licenças ausentes

Alerts are listed on each respective component page as *Alerts*. If it is missing, the component clears all current checks. Alerts can not be ignored, but will disappear once the underlying problem has been fixed.

A component with both duplicated strings and languages looks like this:



Veja também:

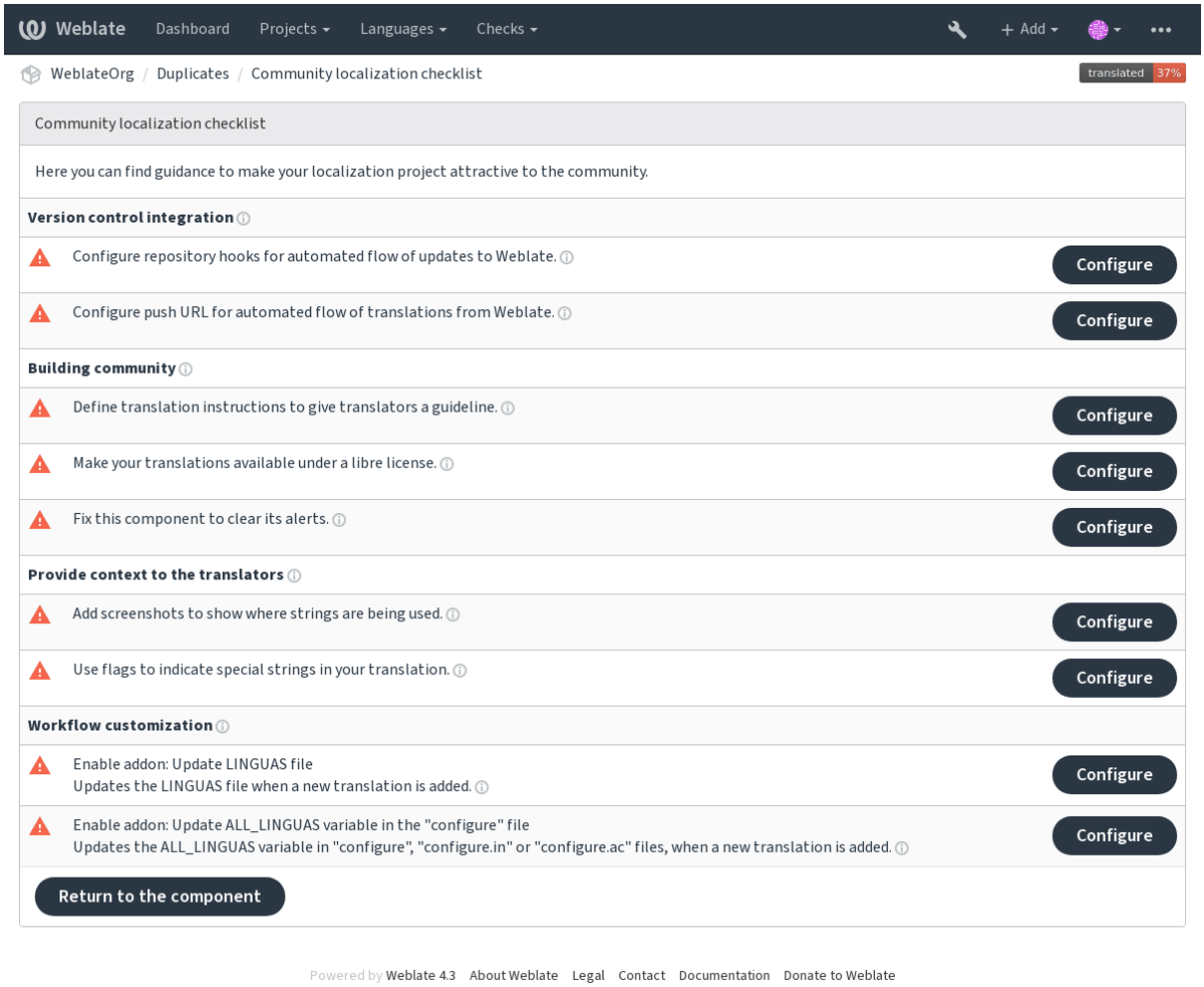
Usar uma autoridade certificadora personalizada

1.7.6 Building translators community

Lista de verificação de localização da comunidade

Novo na versão 3.9.

The *Community localization checklist* which can be found in the menu of each component can give you guidance to make your localization process easy for community translators.



1.7.7 Managing translations

Adding new translations

New strings can be made available for translation when they appear in the base file, called *Template for new translations* (see *Component configuration*). If your file format doesn't require such a file, as is the case with most monolingual translation flows, you can start with blank files).

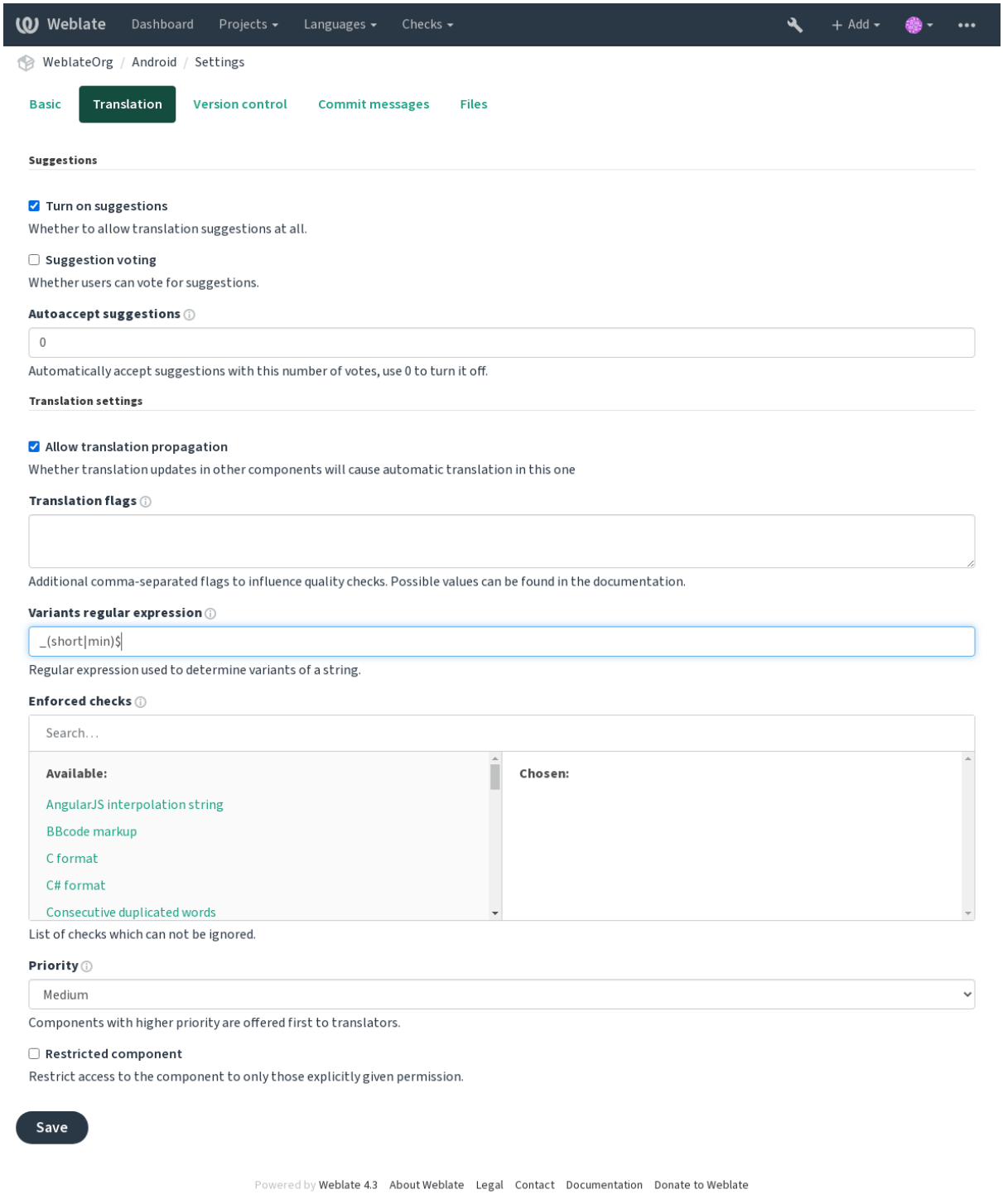
New languages can be added right away when requested by a user in Weblate, or a notification will be sent to project admins for approval and manual addition. This can be done using *Start new translation* in *Component configuration*.

Nota: Project admins can always start translation within Weblate directly.

Language files added manually to the VCS are added to the component when Weblate updates the repository. About repository update settings, see *Atualizar repositórios*.

String variants

Variants are useful to group several strings together so that translators can see all variants of the string at one place. You can define regular expression to group the strings in the *Component configuration*:

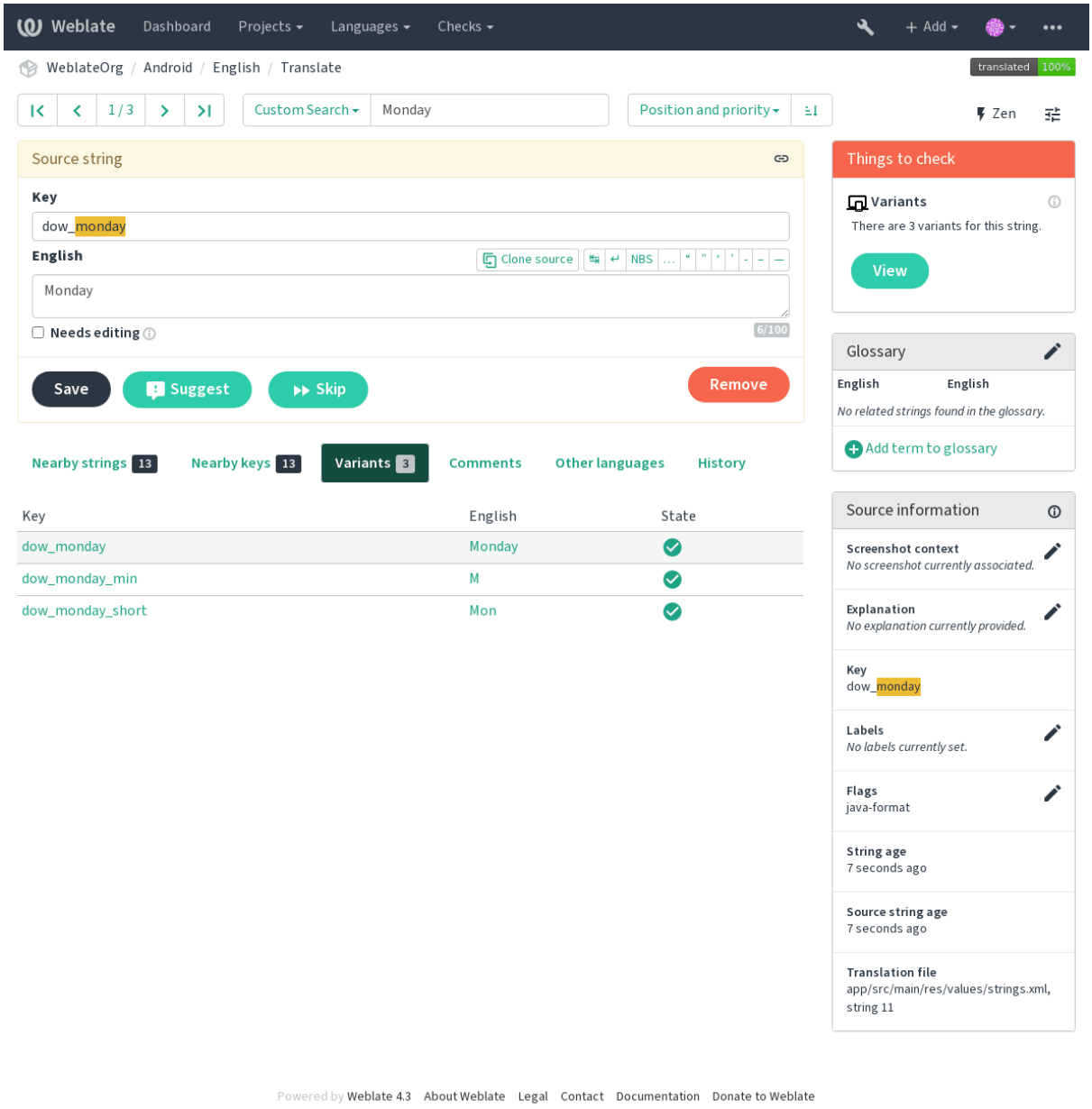


The expression is matched against *Key* to generate root key of the variant. All matching strings are then part of single variants group, including the translation exactly matching the root key, even if that is not matched by the regular expression.

The following table lists some usage examples:

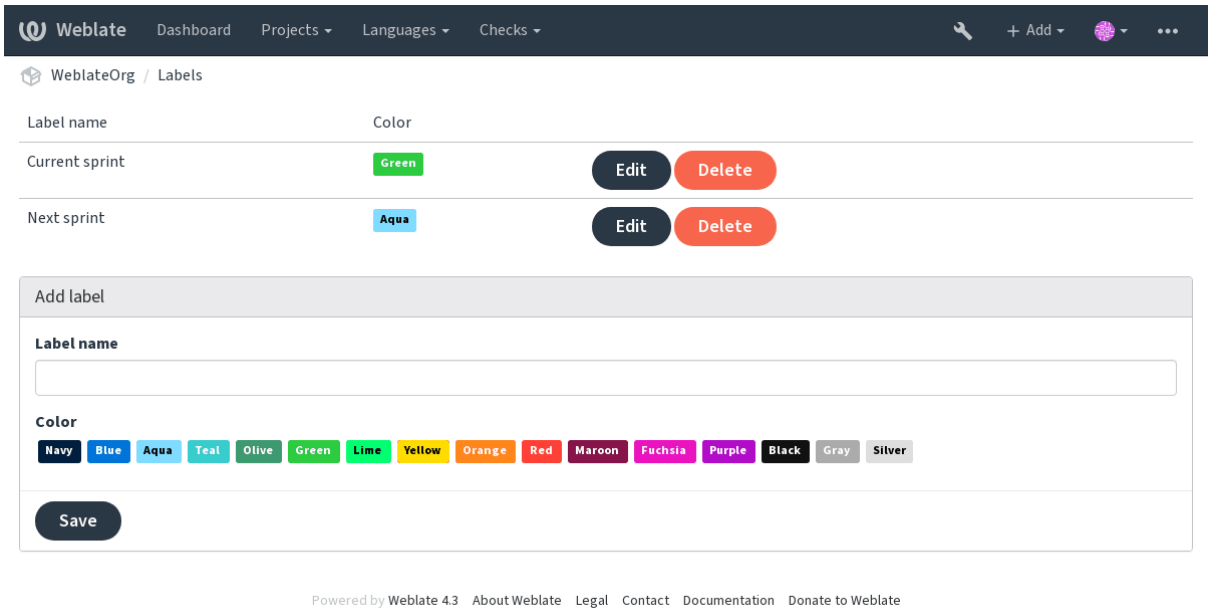
Use case	Regular expression variant	Matched translation keys
Suffix identification	`(Short|Min)$`	`monthShort`, `monthMin`, `month`
Inline identification	`#[SML]`	`dial#S.key`, `dial#M.key`, `dial.key`

The variant is later grouped when translating:



String labels

Split component translation strings into categories by text and colour in the project configuration.



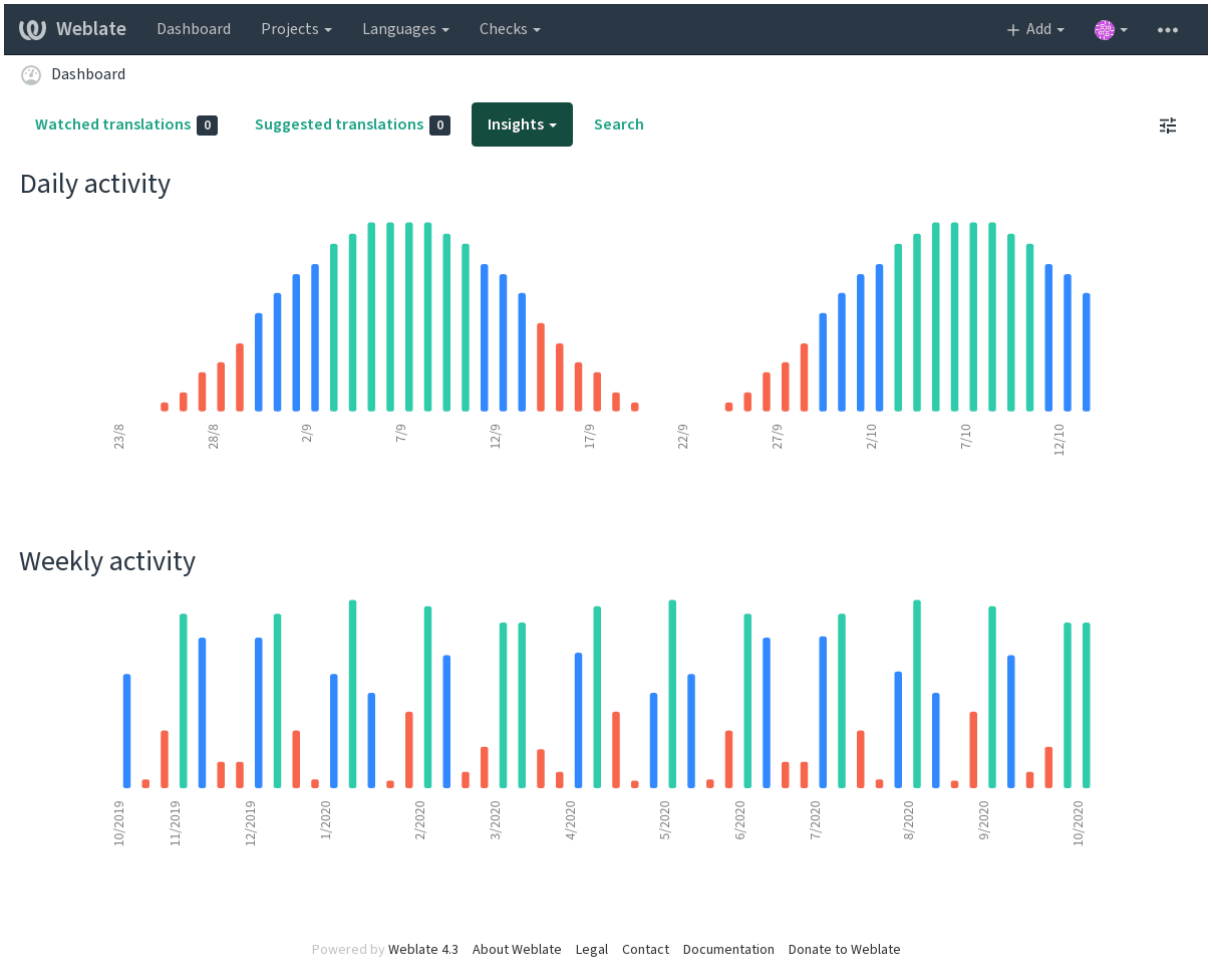
Dica: Labels can be assigned to units in *Additional info on source strings* by bulk editing, or using the *Edição em massa* addon.

1.7.8 Reviewing strings

Activity reports

Activity reports check changes of translations, for projects, components or individual users.

The activity reports for a project or component is accessible from its dashboard, on the *Insights* tab, selecting *Activity*.



More reports are accessible on the *Insights* tab, selecting *Translation reports*.

The activity of the currently signed in user can be seen by clicking on *Profile* from the user menu on the top right.

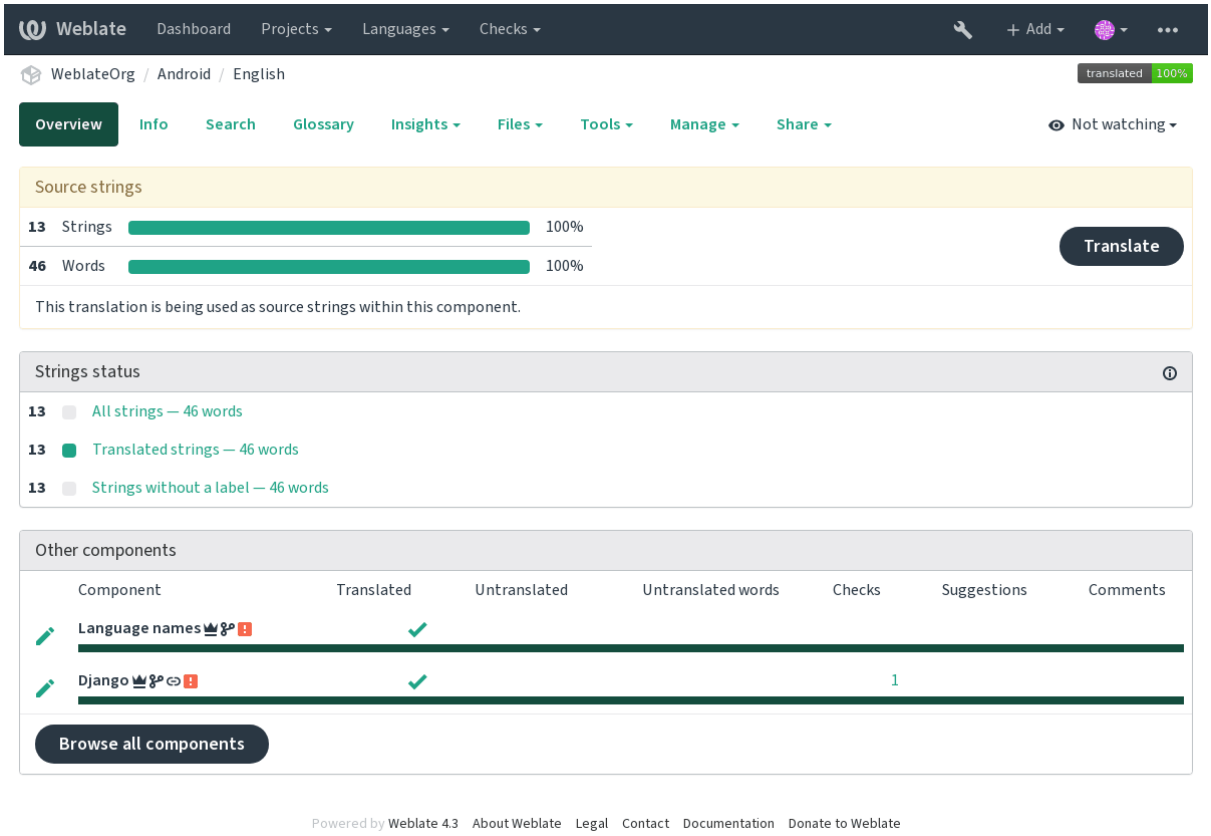
Source strings checks

There are many *Verificações de qualidade*, some of them focus on improving the quality of source strings. Many failing checks suggest a hint to make source strings easier to translate. All types of failing source checks are displayed on the *Source* tab of every component.

Translation string checks

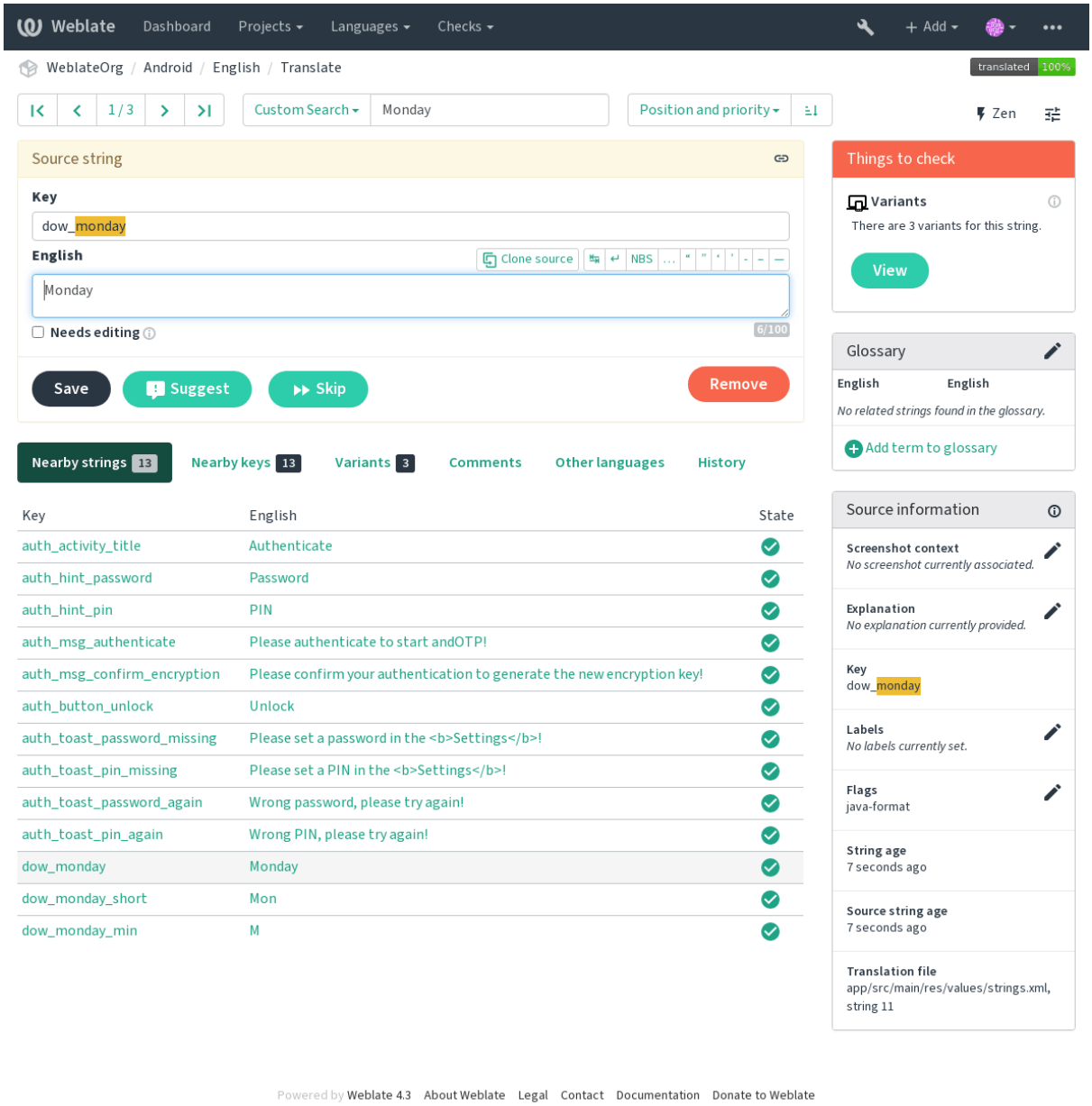
Erroneous failing translation string checks indicate the problem is with the source string. Translators sometimes fix mistakes in the translation instead of reporting it - a typical example is a missing full stop at the end of a sentence.

Reviewing all failing checks can provide valuable feedback to improve its source strings. To make source strings review easier, Weblate automatically creates a translation for the source language and shows you source level checks there:



One of the most interesting checks here is the *Várias verificações falhadas* - it is triggered whenever there is failure on multiple translations of a given string. Usually this is something to look for, as this is a string which translators have problems translating properly.

The detailed listing is a per language overview:

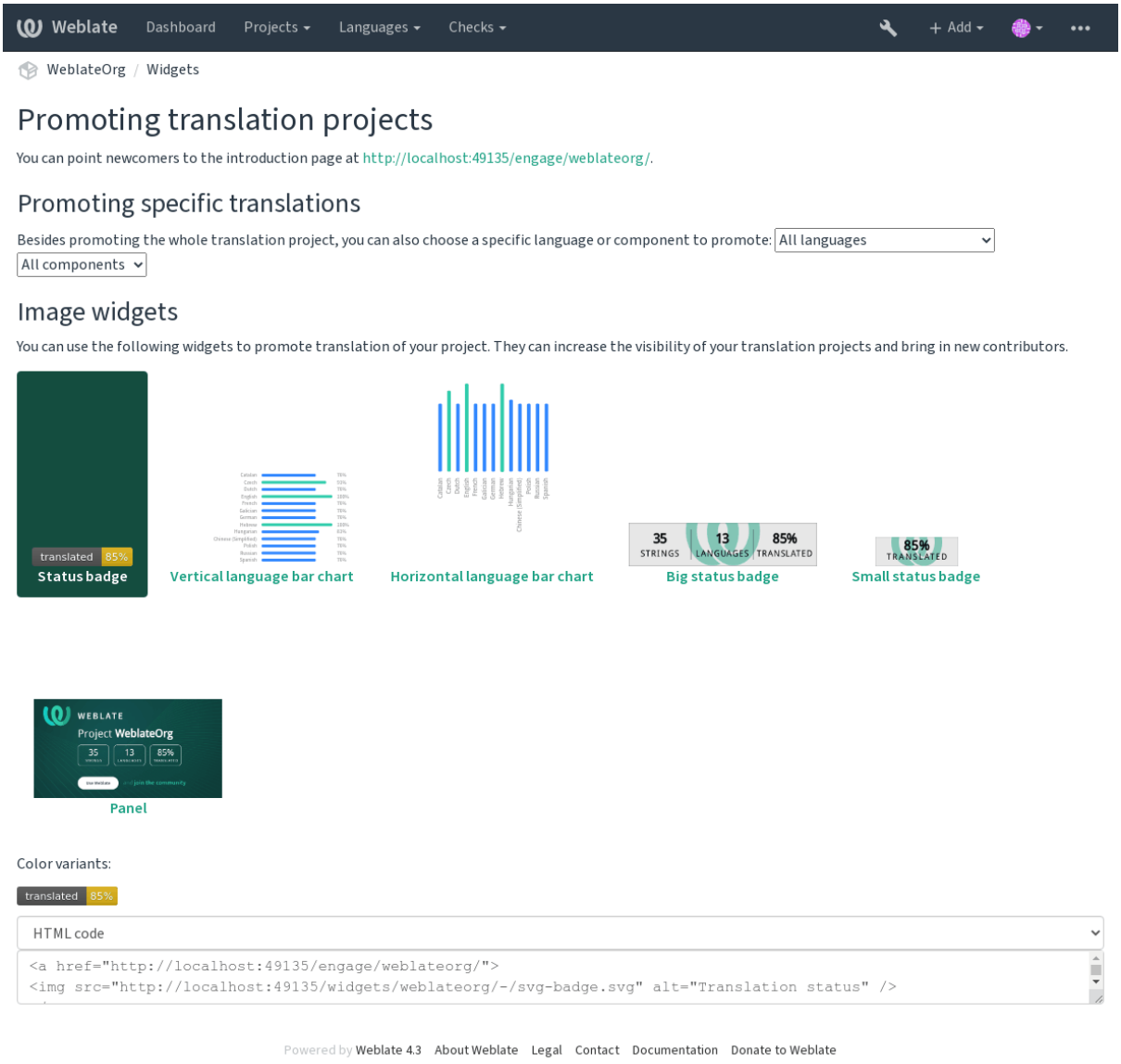


String comments

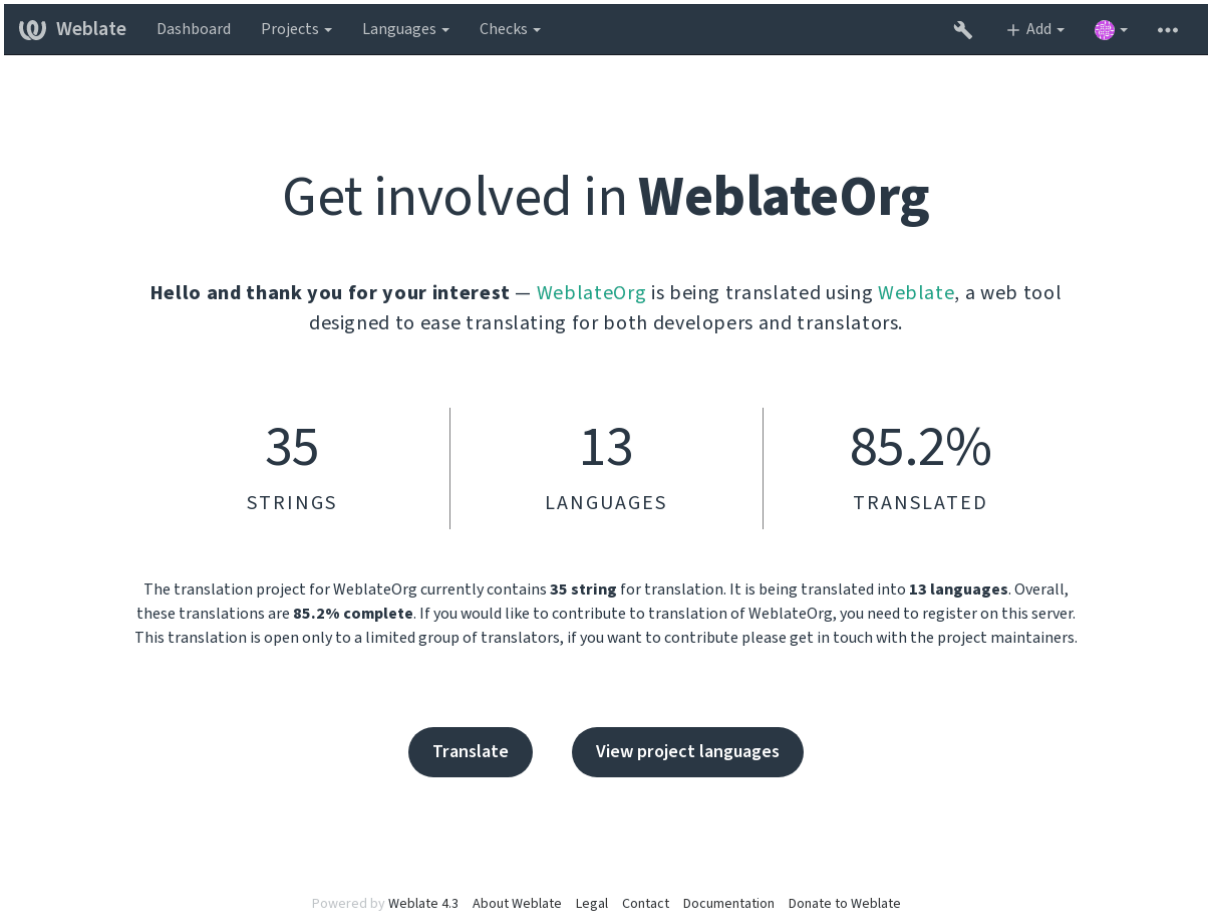
Translators can comment on both translation and source strings. Each *Component configuration* can be configured to receive such comments to an e-mail address, and using the developers mailing list is usually the best approach. This way you can keep an eye on when problems arise in translation, take care of them, and fix them quickly.

1.7.9 Promoting the translation

Weblate provides you widgets to share on your website or other sources to promote the translation project. It also has a nice welcome page for new contributors to give them basic information about the translation. Additionally you can share information about translation using Facebook or Twitter. All these possibilities can be found on the *Share* tab:

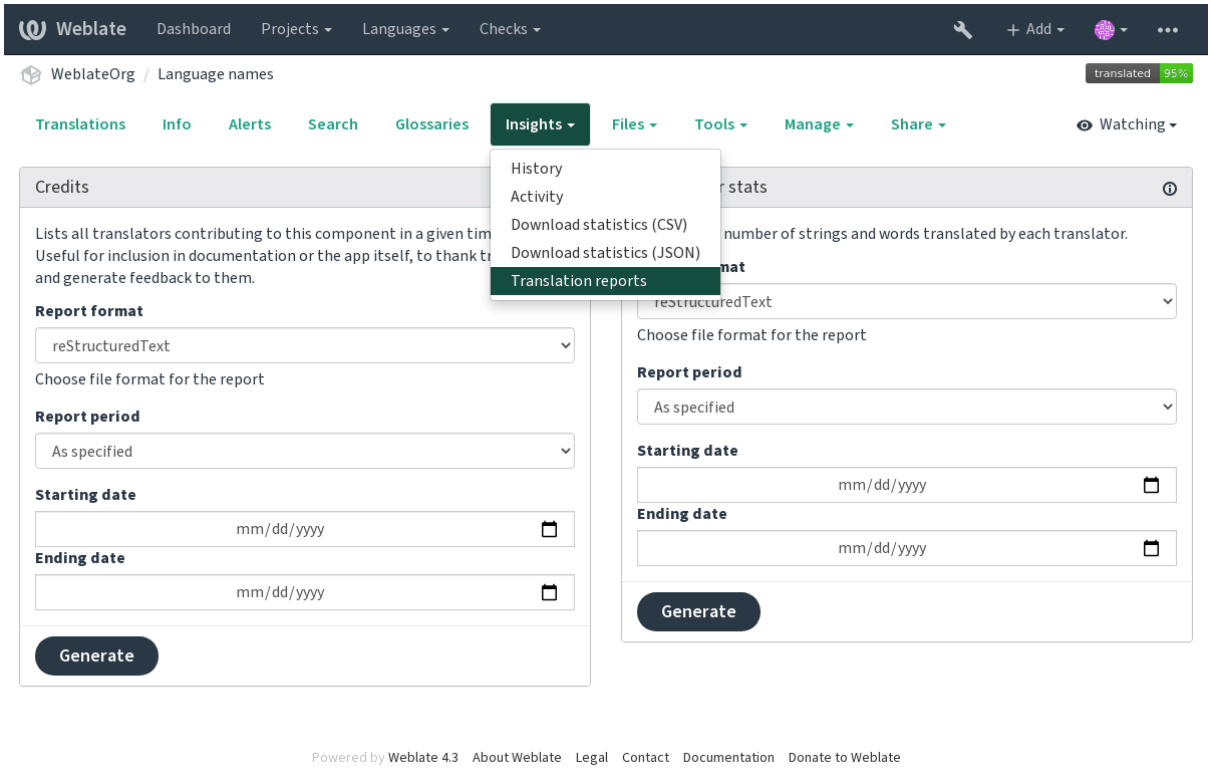


All these badges are provided with the link to simple page which explains users how to translate using Weblate:



1.7.10 Translation progress reporting

Reporting features give insight into how a translation progresses over a given period. A summary of contributions to any given component over time is provided. The reporting tool is found in the *Insights* menu of any translation component, project or on the dashboard:



Several reporting tools are available on this page and all can produce output in HTML, reStructuredText or JSON. The first two formats are suitable for embedding statistics into existing documentation, while JSON is useful for further processing of the data.

Translator credits

Generates a document usable for crediting translators - sorted by language and lists all contributors to a given language:

```
* Czech

    * Michal Čihař <michal@cihar.com> (10)
    * John Doe <john@example.com> (5)

* Dutch

    * Jane Doe <jane@example.com> (42)
```

It will render as:

- Checo
  - Michal Čihař <michal@cihar.com> (10)
  - John Doe <john@example.com> (5)
- Holandês
  - Jae Doe <jane@example.com> (42)

Dica: The number in parenthesis indicates number of contributions in given period.

Estatísticas do colaborador

Generates the number of translated words and strings by translator name:

```
======================================== ========================================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ========================
Name                                    Email
 ↪Count total              Source words total       Source chars total
 ↪Target words total       Target chars total       Count new
 ↪Source words new         Source chars new         Target words new
 ↪Target chars new         Count approved           Source words approved
 ↪Source chars approved    Target words approved    Target chars approved    Count
 ↪edited             Source words edited      Source chars edited      Target
 ↪words edited      Target chars edited
======================================== ========================================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ========================
Michal Čihař                            michal@cihar.com
 ↪                      1                        3                       24
 ↪                  3                       21                       1
 ↪          3                       24                       3
 ↪    21                      0                        0                      0
 ↪                        0                        0                       0
 ↪                    0                       0                       0
 ↪            0
Allan Nordhøy                           allan@example.com
 ↪                      2                        5                       25
 ↪                  4                       28                       2
 ↪          3                       24                       3
 ↪    21                      0                        0                      0
 ↪                        0                        0                       0
 ↪                    0                       0                       0
 ↪            0
======================================== ========================================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ========================
```

And it will get rendered as:

Nome	Email	Count total	Source words total	Source characters total	Target words total	Target characters total	Count new	Source words new	Source characters new	Target words new	Target characters new	Count approved	Source words approved	Source characters approved	Target words approved	Target characters approved	Count edited	Source words edited	Source characters edited	Target words edited	Target characters edited
Michal Čihař	michal@cihar.com	1	3	24	3	21	1	3	24	3	21	0	0	0	0	0	0	0	0	0	0
Allan Nordhøy	allan@example.com	2	5	25	4	28	2	3	24	3	21	0	0	0	0	0	0	0	0	0	0

It can be useful if you pay your translators based on amount of work, it gives you various stats on translators work.

All stats are available in three variants:

Total Overall number of edited strings.

New Newly translated strings which didn't have translation before.

Approved Count for string approvals in review workflow (see *Revisores dedicados*).

Edited Edited strings which had translation before.

The following metrics are available for each:

Count Number of strings.

Edits Number of edits in the string, measured in Damerau–Levenshtein distance.

Source words Number of words in the source string.

Source characters Number of characters in the source string.

Target words Number of words in the translated string.

Target characters Number of characters in the translated string.

1.8 Fluxos de trabalho de tradução

Vários fluxos de trabalho de tradução são suportados.

A lista a seguir não é uma lista completa de maneiras de configurar o Weblate. Pode basear outros fluxos de trabalho nos exemplos mais usuais listados aqui.

1.8.1 Acesso à tradução

Os *Controlo de acesso* não são muito discutidos nos fluxos de trabalho, pois cada opção de controle de acesso pode ser aplicada a qualquer fluxo de trabalho. Consulte essa documentação para obter informações sobre como gerir o acesso às traduções.

Nos capítulos a seguir, *qualquer utilizador* significa um utilizador que tenha acesso à tradução. Pode ser qualquer utilizador autenticado se o projeto for público ou um utilizador que tenha uma permissão *Traduzir* para o projeto.

1.8.2 Translation states

Cada cadeia traduzida pode estar num dos seguintes estados:

Não traduzido A tradução está vazia, pode ou não estar armazenada no ficheiro, dependendo do formato do ficheiro.

Precisa de edição A tradução precisa ser editada, isso geralmente é o resultado de uma mudança de cadeias fonte. A tradução está armazenada no ficheiro, dependendo do formato do ficheiro que pode ser marcado como necessidade de edição (por exemplo, à medida que recebe uma sinalização de «fuzzy»).

A aguardar por revisão A tradução está feita, mas não revisada. É armazenada no ficheiro como uma tradução válida.

Aprovadas A tradução foi aprovada na revisão. Já não pode ser alterada por tradutores, mas apenas por revisores. Tradutores só podem adicioná-las sugestões.

Sugestões As sugestões estão armazenadas apenas no Weblate e não no ficheiro de tradução.

1.8.3 Tradução direta

Esta é a configuração mais usual para equipas menores, qualquer um pode traduzir diretamente. Esta também é a configuração predefinida no Weblate.

- *Qualquer utilizador* pode editar traduções.
- Sugestões são formas opcionais de sugerir alterações, quando os tradutores não têm certeza sobre a alteração.

Configuração	Value	Nota
Activar revisões	inativo	Configurado a nível de projeto.
Ativar sugestões	ativo	É útil para os utilizadores serem capazes de sugerir quando não têm certeza.
Votação de sugestão	inativo	
Aceitar sugestões automaticamente	0	
Grupo de tradutores	*Utilizadores*	Ou *Traduzir* com *Controlo de acesso*.
Grupo de revisores	N/D	Não usado.

1.8.4 Revisão por pares

Com este fluxo de trabalho, qualquer pessoa pode adicionar sugestões e precisa da aprovação de um ou mais membros adicionais antes de ser aceite como tradução.

- *Qualquer utilizador* pode adicionar sugestões.
- *Qualquer utilizador* pode votar em sugestões.
- Sugestões tornam-se traduções quando dado uma quantidade predeterminada de votos.

Configuração	Value	Nota
Activar revisões	inativo	Configurado a nível de projeto.
Ativar sugestões	ativo	
Votação de sugestão	inativo	
Aceitar sugestões automaticamente	1	Pode definir um valor mais alto para exibir mais revisões por pares.
Grupo de tradutores	*Utilizadores*	Ou *Traduzir* com *Controlo de acesso*.
Grupo de revisores	N/D	Não usado, todos os tradutores revisam.

1.8.5 Revisores dedicados

Novo na versão 2.18: O fluxo de trabalho adequado de revisão é suportado desde o Weblate 2.18.

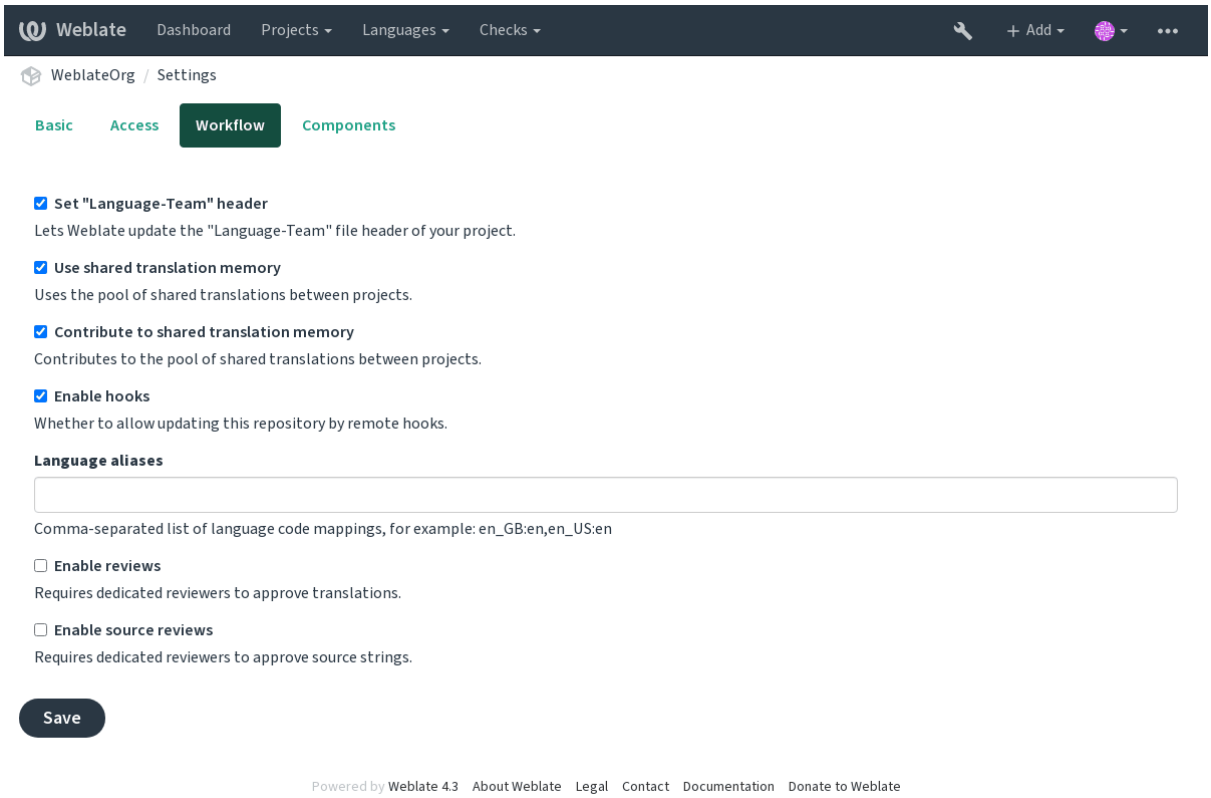
Com revisores dedicados tem dois grupos de utilizadores, um capaz de enviar traduções e outro capaz de revisá-los para garantir que as traduções sejam consistentes e que a qualidade seja boa.

- *Qualquer utilizador* pode editar traduções não aprovadas.
- *Revisor* pode aprovar / retirar a aprovação de cadeias.
- *Revisor* pode editar todas as traduções (incluindo as aprovadas).
- Sugestões também podem ser usadas para sugerir alterações para cadeias aprovadas.

Configuração	Value	Nota
Activar revisões	ativo	Configurado a nível de projeto.
Ativar sugestões	inativo	É útil para os utilizadores serem capazes de sugerir quando não têm certeza.
Votação de sugestão	inativo	
Aceitar sugestões automaticamente	0	
Grupo de tradutores	*Utilizadores*	Ou *Traduzir* com *Controlo de acesso*.
Grupo de revisores	*Revisores*	Ou “Revisão” com :ref:”privileges”.

1.8.6 Ativar revisões

As revisões podem ser ativadas na configuração do projeto, a partir da subpágina *Fluxo de trabalho* das configurações do projeto (encontra-se no menu *Gerir* → *Configurações*):

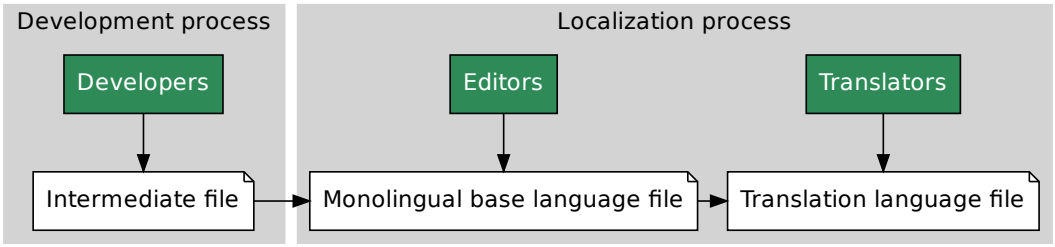


Nota: Dependendo da configuração do Weblate, a configuração pode não estar-lhe disponível. Por exemplo, no Hosted Weblate, isso não está disponível para projetos hospedados gratuitamente.

1.8.7 Portal de qualidade para cadeias fonte

Em muitos casos, as cadeias fonte do idioma de origem vêm de programadores, porque eles escrevem o código e fornecem cadeias iniciais. No entanto, os programadores muitas vezes não são falantes nativos do idioma de origem e não fornecem a qualidade desejada das cadeias fonte. A tradução intermediária pode ajudá-los a lidar com isso - há uma rota de qualidade adicional para as cadeias entre programadores e tradutores e utilizadores.

Ao definir um *Ficheiro de idioma intermédio*, este ficheiro será usado como fonte para as cadeias, mas será editado para o idioma de origem para poli-lo. Uma vez que o texto esteja pronto no idioma de origem, também estará disponível para os tradutores traduzirem em idiomas adicionais.



Veja também:

Ficheiro de idioma intermédio, *Ficheiro de idioma base monolingue*, *Bilingual and monolingual formats*

1.8.8 Revisões de cadeias fonte

Com o *Ativar revisões de fontes* ativado, o processo de revisão pode ser aplicado em cadeias fonte. Uma vez ativado, os utilizadores podem relatar problemas nas cadeias fonte. O processo real depende se usa formatos bilíngues ou monolíngues.

Para formatos monolíngues, a revisão de cadeias fonte se comporta da mesma forma que com *Revisores dedicados* - uma vez que o problema é relatado na cadeia fonte, é marcado como *Necessita edição*.

Os formatos bilíngues não permitem a edição direta de cadeias fonte (estes são normalmente extraídos diretamente do código-fonte). Neste caso, o rótulo *Fonte precisa de revisão* é anexado às cadeias relatadas por tradutores. Deve revisar esses textos e editá-los na fonte ou remover o rótulo.

Veja também:

Bilingual and monolingual formats, *Revisores dedicados*, *String labels*

1.9 Frequently Asked Questions

1.9.1 Configuração

How to create an automated workflow?

Weblate can handle all the translation things semi-automatically for you. If you give it push access to your repository, the translations can happen without interaction, unless some merge conflict occurs.

1. Set up your Git repository to tell Weblate when there is any change, see *Hooks de notificação* for info on how to do it.
2. Set a push URL at your *Component configuration* in Weblate, this allows Weblate to push changes to your repository.
3. Turn on push-on-commit on your *Project configuration* in Weblate, this will make Weblate push changes to your repository whenever they happen at Weblate.

Veja também:

Tradução contínua, *Evitar conflitos de mesclagem*

How to access repositories over SSH?

Please see *Accessing repositories* for info on setting up SSH keys.

How to fix merge conflicts in translations?

Merge conflicts happen from time to time when the translation file is changed in both Weblate and the upstream repository concurrently. You can usually avoid this by merging Weblate translations prior to making changes in the translation files (e.g. before running msgmerge). Just tell Weblate to commit all pending translations (you can do it in *Repository maintenance* in the *Manage* menu) and merge the repository (if automatic push is not on).

If you've already ran into a merge conflict, the easiest way is to solve all conflicts locally at your workstation - is to simply add Weblate as a remote repository, merge it into upstream and fix any conflicts. Once you push changes back, Weblate will be able to use the merged version without any other special actions.

Nota: Depending on your setup, access to the Weblate repository might require authentication. When using the built in *Git exporter* in Weblate, you authenticate with your username and the API key.

```
# Commit all pending changes in Weblate, you can do this in the UI as well:
wlc commit
# Lock the translation in Weblate, again this can be done in the UI as well:
wlc lock
# Add Weblate as remote:
git remote add weblate https://hosted.weblate.org/git/project/component/
# You might need to include credentials in some cases:
git remote add weblate https://username:APIKEY@hosted.weblate.org/git/project/
↪component/

# Update weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master

# Resolve conflicts:
edit …
```

(continues on next page)

(continuação da página anterior)

```
git add …
…
git commit

# Push changes to upstream repository, Weblate will fetch merge from there:
git push

# Open Weblate for translation:
wlc unlock
```

If you're using multiple branches in Weblate, you can do the same to all of them:

```
# Add and update Weblate remotes
git remote add weblate-one https://hosted.weblate.org/git/project/one/
git remote add weblate-second https://hosted.weblate.org/git/project/second/
git remote update weblate-one weblate-second

# Merge QA_4_7 branch:
git checkout QA_4_7
git merge weblate-one/QA_4_7
... # Resolve conflicts
git commit

# Merge master branch:
git checkout master
git merge weblates-second/master
... # Resolve conflicts
git commit

# Push changes to the upstream repository, Weblate will fetch the merge from there:
git push
```

In case of gettext PO files, there is a way to merge conflicts in a semi-automatic way:

Fetch and keep a local clone of the Weblate Git repository. Also get a second fresh local clone of the upstream Git repository (i. e. you need two copies of the upstream Git repository: An intact and a working copy):

```
# Add remote:
git remote add weblate /path/to/weblate/snapshot/

# Update Weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master

# Resolve conflicts in the PO files:
for PO in `find . -name '*.po'` ; do
    msgcat --use-first /path/to/weblate/snapshot/$PO\
               /path/to/upstream/snapshot/$PO -o $PO.merge
    msgmerge --previous --lang=${PO%.po} $PO.merge domain.pot -o $PO
    rm $PO.merge
    git add $PO
done
git commit

# Push changes to the upstream repository, Weblate will fetch merge from there:
git push
```

Veja também:

How to export the Git repository that Weblate uses?, *Tradução contínua*, *Evitar conflitos de mesclagem*

How do I translate several branches at once?

Weblate supports pushing translation changes within one *Project configuration*. For every *Component configuration* which has it turned on (the default behavior), the change made is automatically propagated to others. This way translations are kept synchronized even if the branches themselves have already diverged quite a lot, and it is not possible to simply merge translation changes between them.

Once you merge changes from Weblate, you might have to merge these branches (depending on your development workflow) discarding differences:

```
git merge -s ours origin/maintenance
```

Veja também:

Manter traduções iguais entre componentes

How to translate multi-platform projects?

Weblate supports a wide range of file formats (see *Formatos de ficheiros suportados*) and the easiest approach is to use the native format for each platform.

Once you have added all platform translation files as components in one project (see *Adding translation projects and components*), you can utilize the translation propagation feature (turned on by default, and can be turned off in the *Component configuration*) to translate strings for all platforms at once.

Veja também:

Manter traduções iguais entre componentes

How to export the Git repository that Weblate uses?

There is nothing special about the repository, it lives under the `DATA_DIR` directory and is named `vcs/<project>/<component>/`. If you have SSH access to this machine, you can use the repository directly.

For anonymous access, you might want to run a Git server and let it serve the repository to the outside world.

Alternatively, you can use *Git exporter* inside Weblate to automate this.

What are the options for pushing changes back upstream?

This heavily depends on your setup, Weblate is quite flexible in this area. Here are examples of some workflows used with Weblate:

- Weblate automatically pushes and merges changes (see *How to create an automated workflow?*).
- You manually tell Weblate to push (it needs push access to the upstream repository).
- Somebody manually merges changes from the Weblate git repository into the upstream repository.
- Somebody rewrites history produced by Weblate (e.g. by eliminating merge commits), merges changes, and tells Weblate to reset the content in the upstream repository.

Of course you are free to mix all of these as you wish.

How can I limit Weblate access to only translations, without exposing source code to it?

You can use git submodule for separating translations from source code while still having them under version control.

1. Create a repository with your translation files.
2. Add this as a submodule to your code:

```
git submodule add git@example.com:project-translations.git path/to/translations
```

3. Link Weblate to this repository, it no longer needs access to the repository containing your source code.
4. You can update the main repository with translations from Weblate by:

```
git submodule update --remote path/to/translations
```

Please consult the git submodule documentation for more details.

How can I check whether my Weblate is set up properly?

Weblate includes a set of configuration checks which you can see in the admin interface, just follow the *Performance report* link in the admin interface, or open the `/manage/performance/` URL directly.

Why are all commits committed by Weblate <noreply@weblate.org>?

This is the default committer name, configured when you create a translation component. You can change it in the administration at any time.

The author of every commit (if the underlying VCS supports it) is still recorded correctly as the user that made the translation.

Veja também:

Component configuration

1.9.2 Usage

How do I review the translations of others?

- You can subscribe to any changes made in *Notificações* and then check others contributions as they come in by e-mail.
- There is a review tool available at the bottom of the translation view, where you can choose to browse translations made by others since a given date.

How do I provide feedback on a source string?

On context tabs below translation, you can use the *Comments* tab to provide feedback on a source string, or discuss it with other translators.

How can I use existing translations while translating?

- Use the import functionality to load compendium as translations, suggestions or translations needing review. This is the best approach for a one-time translation using a compendium or a similar translation database.
- You can set up *tmserver* with all databases you have and let Weblate use it. This is good when you want to use it several times during translation.
- Another option is to translate all related projects in a single Weblate instance, which will make it automatically pick up translations from other projects as well.

Veja também:

Tradução automática, *Sugestões automáticas*

Does Weblate update translation files besides translations?

Weblate tries to limit changes in translation files to a minimum. For some file formats it might unfortunately lead to reformatting the file. If you want to keep the file formatted your way, please use a pre-commit hook for that.

For monolingual files (see *Formatos de ficheiros suportados*) Weblate might add new translation strings not present in the *template*, and not in actual translations. It does not however perform any automatic cleanup of stale strings as that might have unexpected outcomes. If you want to do this, please install a pre-commit hook which will handle the cleanup according to your requirements.

Weblate also will not try to update bilingual files in any way, so if you need `po` files being updated from `pot`, you need to do it yourself.

Veja também:

Processar repositório com scripts

Where do language definitions come from and how can I add my own?

The basic set of language definitions is included within Weblate and Translate-toolkit. This covers more than 150 languages and includes info about plural forms or text direction.

You are free to define your own languages in the administrative interface, you just need to provide info about it.

Can Weblate highlight changes in a fuzzy string?

Weblate supports this, however it needs the data to show the difference.

For Gettext PO files, you have to pass the parameter `--previous` to **`msgmerge`** when updating PO files, for example:

```
msgmerge --previous -U po/cs.po po/phpmyadmin.pot
```

For monolingual translations, Weblate can find the previous string by ID, so it shows the differences automatically.

Why does Weblate still show old translation strings when I've updated the template?

Weblate does not try to manipulate the translation files in any way other than allowing translators to translate. So it also does not update the translatable files when the template or source code have been changed. You simply have to do this manually and push changes to the repository, Weblate will then pick up the changes automatically.

Nota: It is usually a good idea to merge changes done in Weblate before updating translation files, as otherwise you will usually end up with some conflicts to merge.

For example with gettext PO files, you can update the translation files using the **msgmerge** tool:

```
msgmerge -U locale/cs/LC_MESSAGES/django.mo locale/django.pot
```

In case you want to do the update automatically, you can install addon *Atualizar ficheiros PO para coincidir com POT (msgmerge)*.

1.9.3 Troubleshooting

Requests sometimes fail with «too many open files» error

This happens sometimes when your Git repository grows too much and you have many of them. Compressing the Git repositories will improve this situation.

The easiest way to do this is to run:

```
# Go to DATA_DIR directory
cd data/vcs
# Compress all Git repositories
for d in */* ; do
    pushd $d
    git gc
    popd
done
```

Veja também:

DATA_DIR

When accessing the site I get a «Bad Request (400)» error

This is most likely caused by an improperly configured *ALLOWED_HOSTS*. It needs to contain all hostnames you want to access on your Weblate. For example:

```
ALLOWED_HOSTS = ['weblate.example.com', 'weblate', 'localhost']
```

Veja também:

Configuração de hosts permitidos

What does mean «There are more files for the single language (en)»?

This typically happens when you have translation file for source language. Weblate keeps track of source strings and reserves source language for this. The additional file for same language is not processed.

- Se a tradução para o idioma de origem for desejada, por favor altere o *Idioma fonte* nas configurações dos componentes.
- Caso o ficheiro de tradução para o idioma de origem não seja necessário, por favor, remova-o do repositório.
- Caso o ficheiro de tradução para o idioma de origem seja necessário, mas deveria ser ignorado pelo Weblate, por favor, ajuste o filtro do idioma para excluí-lo.

1.9.4 Funcionalidades

Does Weblate support other VCSes than Git and Mercurial?

Weblate currently does not have native support for anything other than *Git* (with extended support for *GitHub*, *Gerrit* and *Subversion*) and *Mercurial*, but it is possible to write backends for other VCSes.

You can also use *Git remote helpers* in Git to access other VCSes.

Weblate also supports VCS less operation, see *Local files*.

Nota: For native support of other VCSes, Weblate requires using distributed VCS, and could probably be adjusted to work with anything other than Git and Mercurial, but somebody has to implement this support.

Veja também:

Integração de controlo de versões

How does Weblate credit translators?

Every change made in Weblate is committed into VCS under the translators name. This way every single change has proper authorship, and you can track it down using the standard VCS tools you use for code.

Additionally, when the translation file format supports it, the file headers are updated to include the translator's name.

Veja também:

`list_translators`, *Translation progress reporting*

Why does Weblate force showing all PO files in a single tree?

Weblate was designed in a way that every PO file is represented as a single component. This is beneficial for translators, so they know what they are actually translating. If you feel your project should be translated as one, consider merging these po files. It will make life easier even for translators not using Weblate.

Nota: In case there is great demand for this feature, it might be implemented in future versions.

Why does Weblate use language codes such sr_Latn or zh_Hant?

These are language codes defined by **RFC 4646** to better indicate that they are really different languages instead previously wrongly used modifiers (for `@latin` variants) or country codes (for Chinese).

Weblate still understands legacy language codes and will map them to current one - for example `sr@latin` will be handled as `sr_Latn` or `zh@CN` as `zh_Hans`.

1.10 Formatos de ficheiros suportados

Weblate supports most translation format understood by translate-toolkit, however each format being slightly different, some issues with formats that are not well tested can arise.

Veja também:

Translation Related File Formats

Nota: When choosing a file format for your application, it's better to stick some well established format in the toolkit/platform you use. This way your translators can additionally use whatever tools they are used to, and will more likely contribute to your project.

1.10.1 Bilingual and monolingual formats

Both monolingual and bilingual formats are supported. Bilingual formats store two languages in single file—source and translation (typical examples are *GNU gettext*, *XLIFF* or *Apple iOS strings*). On the other side, monolingual formats identify the string by ID, and each language file contains only the mapping of those to any given language (typically *Android string resources*). Some file formats are used in both variants, see the detailed description below.

For correct use of monolingual files, Weblate requires access to a file containing complete list of strings to translate with their source—this file is called *Ficheiro de idioma base monolingue* within Weblate, though the naming might vary in your paradigm.

Additionally this workflow can be extended by utilizing *Ficheiro de idioma intermédio* to include strings provided by developers, but not to be used as is in the final strings.

1.10.2 Deteção automática

Weblate can automatically detect several widespread file formats, but this detection can harm your performance and will limit features specific to given file format (for example automatic addition of new translations).

1.10.3 Translation types capabilities

Capabilities of all supported formats:

Format	Linguality[1]	Plurals[2]	Comments[3]	Context[4]	Location[5]	Flags[8]	Additional states[6]
GNU gettext	bilingual	yes	yes	yes	yes	yes[9]	needs editing
Monolingual gettext	mono	yes	yes	yes	yes	yes[9]	needs editing
XLIFF	both	yes	yes	yes	yes	yes[10]	needs editing, approved

continues on next page

Table 3 – continuação da página anterior

Format	Linguality[1]	Plurals[2]	Comments[3]	Context[4]	Location[5]	Flags[8]	Additional states[6]
Java properties	both	no	yes	no	no	no	
GWT properties	mono	yes	yes	no	no	no	
Joomla translations	mono	no	yes	no	yes	no	
Qt Linguist .ts	both	yes	yes	no	yes	yes[10]	needs editing
Android string resources	mono	yes	yes[7]	no	no	yes[10]	
Apple iOS strings	bilingual	no	yes	no	no	no	
Cadeias de PHP	mono	no[11]	yes	no	no	no	
JSON files	mono	no	no	no	no	no	
JSON i18next files	mono	yes	no	no	no	no	
go-i18n JSON files	mono	yes	no	no	no	no	
ARB File	mono	yes	yes	no	no	no	
WebExtension JSON	mono	yes	yes	no	no	no	
.XML resource files	mono	no	yes	no	no	yes[10]	
CSV files	mono	no	yes	yes	yes	no	needs editing
YAML files	mono	no	yes	no	no	no	
Ruby YAML files	mono	yes	yes	no	no	no	
DTD files	mono	no	no	no	no	no	
Flat XML	mono	no	no	no	no	yes[10]	
Windows RC files	mono	no	yes	no	no	no	
Excel Open XML	mono	no	yes	yes	yes	no	needs editing
Ficheiros de metadados da App Store	mono	no	no	no	no	no	
Subtitle files	mono	no	no	no	yes	no	
HTML files	mono	no	no	no	no	no	
OpenDocument Format	mono	no	no	no	no	no	
IDML Format	mono	no	no	no	no	no	
INI translations	mono	no	no	no	no	no	

continues on next page

Table 3 – continuação da página anterior

Format	Linguality[1]	Plurals[2]	Comments[3]	Context[4]	Location[5]	Flags[8]	Additional states[6]
Traduções Inno Setup INI	mono	no	no	no	no	no	

1.10.4 GNU gettext

Most widely used format for translating libre software. This was first format supported by Weblate and still has the best support.

Contextual info stored in the file is supported by adjusting its headers or linking to corresponding source files.

The bilingual gettext PO file typically looks like this:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "Monday"
msgstr "Pondělí"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "Tuesday"
msgstr "Úterý"

#: weblate/accounts/avatar.py:163
msgctxt "No known user"
msgid "None"
msgstr "Žádný"
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`po/*.po`
Ficheiro de idioma base monolingue	*Empty*
Modelo para novas traduções	`po/messages.pot`
Formato de ficheiro	*Gettext PO file*

Veja também:

Translating software using GNU Gettext, *Translating documentation using Sphinx*, Gettext on Wikipedia, PO Files, *Atualizar a variável ALL_LINGUAS no ficheiro «configure»*, *Personalizar a saída gettext*, *Atualizar ficheiro LINGUAS*, *Gerar ficheiros MO*, *Atualizar ficheiros PO para coincidir com POT (msgmerge)*

[1] See *Bilingual and monolingual formats*

[2] Plurals are necessary to properly localize strings with variable count.

[3] Comments can be used to pass additional info about the string to translate.

[4] Context is used to differentiate identical strings used in different scopes (for example *Sun* can be used as an abbreviated name of the day «Sunday» or as the name of our closest star).

[5] Location of a string in source code might help proficient translators figure out how the string is used.

[8] See *Personalizar o comportamento*

[6] Additional states supported by the file format in addition to «Not translated» and «Translated».

[9] The gettext type comments are used as flags.

[10] The flags are extracted from the non-standard attribute `weblate-flags` for all XML based formats. Additionally `max-length:N` is supported through the `maxwidth` attribute as defined in the XLIFF standard, see *Specifying translation flags*.

[7] XML comment placed before the `<string>` element, parsed as a developer comment.

[11] The plurals are supported only for Laravel which uses in string syntax to define them, see Localization in Laravel.

Monolingual gettext

Some projects decide to use gettext as monolingual formats—they code just the IDs in their source code and the string then needs to be translated to all languages, including English. This is supported, though you have to choose this file format explicitly when importing components into Weblate.

The monolingual gettext PO file typically looks like this:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-monday"
msgstr "Pondělí"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-tuesday"
msgstr "Úterý"

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "Žádný"
```

While the base language file will be:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-monday"
msgstr "Monday"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-tuesday"
msgstr "Tuesday"

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "None"
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`po/*.po`
Ficheiro de idioma base monolingue	`po/en.po`
Modelo para novas traduções	`po/messages.pot`
Formato de ficheiro	*Gettext PO file (monolingual)*

1.10.5 XLIFF

XML-based format created to standardize translation files, but in the end it is one of many standards, in this area.

XML Localization Interchange File Format (XLIFF) is usually used as bilingual, but Weblate supports it as monolingual as well.

Veja também:

XML Localization Interchange File Format (XLIFF) specification

Translation states

Alterado na versão 3.3: Weblate ignored the state attribute prior to the 3.3 release.

The `state` attribute in the file is partially processed and mapped to the «Needs edit» state in Weblate (the following states are used to flag the string as needing edit if there is a target present: `new`, `needs-translation`, `needs-adaptation`, `needs-l10n`). Should the `state` attribute be missing, a string is considered translated as soon as a `<target>` element exists.

If the translation string has `approved="yes"`, it will also be imported into Weblate as «Approved», anything else will be imported as «Waiting for review» (which matches the XLIFF specification).

While saving, Weblate doesn't add those attributes unless necessary:

- The `state` attribute is only added in case string is marked as needing edit.
- The `approved` attribute is only added in case string has been reviewed.
- In other cases the attributes are not added, but they are updated in case they are present.

That means that when using the XLIFF format, it is strongly recommended to turn on the Weblate review process, in order to see and change the approved state of strings.

See *Revisores dedicados*.

Similarly upon importing such files (in the upload form), you should choose *Import as translated* under *Processing of strings needing edit*.

Whitespace and newlines in XLIFF

Generally types or amounts of whitespace is not differentiated between in XML formats. If you want to keep it, you have to add the `xml:space="preserve"` flag to the string.

Por exemplo:

```
    <trans-unit id="10" approved="yes">
        <source xml:space="preserve">hello</source>
        <target xml:space="preserve">Hello, world!
</target>
    </trans-unit>
```

Specifying translation flags

You can specify additional translation flags (see *Personalizar o comportamento*) by using the `weblate-flags` attribute. Weblate also understands `maxwidth` and `font` attributes from the XLIFF specification:

```
<trans-unit id="10" maxwidth="100" size-unit="pixel" font="ubuntu;22;bold">
   <source>Hello %s</source>
</trans-unit>
<trans-unit id="20" maxwidth="100" size-unit="char" weblate-flags="c-format">
   <source>Hello %s</source>
</trans-unit>
```

The `font` attribute is parsed for font family, size and weight, the above example shows all of that, though only font family is required. Any whitespace in the font family is converted to underscore, so `Source Sans Pro` becomes `Source_Sans_Pro`, please keep that in mind when naming the font group (see *Gerir letras*).

Typical Weblate *Component configuration* for bilingual XLIFF	
Máscara de ficheiro	`localizations/*.xliff`
Ficheiro de idioma base monolingue	*Empty*
Modelo para novas traduções	`localizations/en-US.xliff`
Formato de ficheiro	*XLIFF Translation File*

Typical Weblate *Component configuration* for monolingual XLIFF	
File mask	`localizations/*.xliff`
Ficheiro de idioma base monolingue	`localizations/en-US.xliff`
Modelo para novas traduções	`localizations/en-US.xliff`
Formato de ficheiro	*XLIFF Translation File*

Veja também:

XLIFF on Wikipedia, XLIFF, font attribute in XLIFF 1.2, maxwidth attribute in XLIFF 1.2

1.10.6 Java properties

Native Java format for translations.

Java properties are usually used as monolingual translations.

Weblate supports ISO-8859-1, UTF-8 and UTF-16 variants of this format. All of them support storing all Unicode characters, it is just differently encoded. In the ISO-8859-1, the Unicode escape sequences are used (for example `zkou\u0161ka`), all others encode characters directly either in UTF-8 or UTF-16.

Nota: Loading escape sequences works in UTF-8 mode as well, so please be careful choosing the correct encoding set to match your application needs.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`src/app/Bundle_*.properties`
Ficheiro de idioma base monolingue	`src/app/Bundle.properties`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Java Properties (ISO-8859-1)*

Veja também:

Java properties on Wikipedia, Mozilla and Java properties files, *Formata as propriedades do ficheiro Java*, *Limpeza de ficheiros de tradução*

1.10.7 GWT properties

Native GWT format for translations.

GWT properties are usually used as monolingual translations.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`src/app/Bundle_*.properties`
Ficheiro de idioma base monolingue	`src/app/Bundle.properties`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*GWT Properties*

Veja também:

GWT localization guide Mozilla and Java properties files, *Formata as propriedades do ficheiro Java*, *Limpeza de ficheiros de tradução*

1.10.8 INI translations

Novo na versão 4.1.

INI file format for translations.

INI translations are usually used as monolingual translations.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`language/*.ini`
Ficheiro de idioma base monolingue	`language/en.ini`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*INI File*

Veja também:

INI Files, *Joomla translations*, *Traduções Inno Setup INI*

1.10.9 Traduções Inno Setup INI

Novo na versão 4.1.

Formato de ficheirio Inno Setup INI para traduções.

As traduções Inno Setup INI são normalmente usadas como traduções monolingues.

Nota: The only notable difference to *INI translations* is in supporting `%n` and `%t` placeholders for line break and tab.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`language/*.islu`
Ficheiro de idioma base monolingue	`language/en.islu`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Ficheiro Inno Setup INI*

Nota: Only Unicode files (`.islu`) are currently supported, ANSI variant (`.isl`) is currently not supported.

Veja também:

INI Files, *Joomla translations*, *INI translations*

1.10.10 Joomla translations

Novo na versão 2.12.

Native Joomla format for translations.

Joomla translations are usually used as monolingual translations.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`language/*/com_foobar.ini`
Ficheiro de idioma base monolingue	`language/en-GB/com_foobar.ini`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Joomla Language File*

Veja também:

Specification of Joomla language files, Mozilla and Java properties files, *INI translations*, *Traduções Inno Setup INI*

1.10.11 Qt Linguist .ts

Translation format used in Qt based applications.

Qt Linguist files are used as both bilingual and monolingual translations.

Typical Weblate *Component configuration* when using as bilingual	
Máscara de ficheiro	`i18n/app.*.ts`
Ficheiro de idioma base monolingue	*Empty*
Modelo para novas traduções	`i18n/app.de.ts`
Formato de ficheiro	*Qt Linguist Translation File*

Typical Weblate *Component configuration* when using as monolingual	
Máscara de ficheiro	`i18n/app.*.ts`
Ficheiro de idioma base monolingue	`i18n/app.en.ts`
Modelo para novas traduções	`i18n/app.en.ts`
Formato de ficheiro	*Qt Linguist Translation File*

Veja também:

Qt Linguist manual, Qt .ts, *Bilingual and monolingual formats*

1.10.12 Android string resources

Android specific file format for translating applications.

Android string resources are monolingual, the *Ficheiro de idioma base monolingue* file is stored in a different location from the others `res/values/strings.xml`.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`res/values-*/strings.xml`
Ficheiro de idioma base monolingue	`res/values/strings.xml`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Android String Resource*

Veja também:

Android string resources documentation, Android string resources

---

Nota: Android *string-array* structures are not currently supported. To work around this, you can break your string arrays apart:

```
<string-array name="several_strings">
    <item>First string</item>
    <item>Second string</item>
</string-array>
```

become:

```
<string-array name="several_strings">
    <item>@string/several_strings_0</item>
    <item>@string/several_strings_1</item>
</string-array>
```

(continues on next page)

(continuação da página anterior)

```
<string name="several_strings_0">First string</string>
<string name="several_strings_1">Second string</string>
```

The *string-array* that points to the *string* elements should be stored in a different file, and not be made available for translation.

This script may help pre-process your existing strings.xml files and translations: https://gist.github.com/paour/11291062

1.10.13 Apple iOS strings

Apple specific file format for translating applications, used for both iOS and iPhone/iPad application translations.

Apple iOS strings are usually used as bilingual translations.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`Resources/*.lproj/Localizable.strings`
Ficheiro de idioma base monolingue	`Resources/en.lproj/Localizable.strings` or `Resources/Base.lproj/Localizable.strings`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*iOS Strings (UTF-8)*

Veja também:

Apple «strings files» documentation, Mac OSX strings

1.10.14 Cadeias de PHP

PHP translations are usually monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Example file:

```
<?php
$LANG['foo'] = 'bar';
$LANG['foo1'] = 'foo bar';
$LANG['foo2'] = 'foo bar baz';
$LANG['foo3'] = 'foo bar baz bag';
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`lang/*/texts.php`
Ficheiro de idioma base monolingue	`lang/en/texts.php`
Modelo para novas traduções	`lang/en/texts.php`
Formato de ficheiro	*PHP strings*

Cadeias de PHP Laravel

Alterado na versão 4.1.

The Laravel PHP localization files are supported as well with plurals:

```
<?php
return [
    'apples' => 'There is one apple|There are many apples',
];
```

Veja também:

PHP, Localization in Laravel

1.10.15 JSON files

Novo na versão 2.0.

Alterado na versão 2.16: Since Weblate 2.16 and with translate-toolkit at-least 2.2.4, nested structure JSON files are supported as well.

Alterado na versão 4.3: The structure of JSON file is properly preserved even for complex situations which were broken in prior releases.

JSON format is used mostly for translating applications implemented in JavaScript.

Weblate currently supports several variants of JSON translations:

- Simple key / value files, used for example by *vue-i18n* or *react-intl*.
- Files with nested keys.
- *JSON i18next files*
- *go-i18n JSON files*
- *WebExtension JSON*
- *ARB File*

JSON translations are usually monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Example file:

```
{
  "Hello, world!\n": "Ahoj světe!\n",
  "Orangutan has %d banana.\n": "",
  "Try Weblate at https://demo.weblate.org/!\n": "",
  "Thank you for using Weblate.": ""
}
```

Nested files are supported as well (see above for requirements), such a file can look like:

```
{
  "weblate": {
    "hello": "Ahoj světe!\n",
    "orangutan": "",
    "try": "",
    "thanks": ""
  }
}
```

Dica: The *JSON file* and *JSON nested structure file* can both handle same type of files. The only difference between them is when adding new strings. The nested variant tries to parse the key and insert the new string into the matching structure.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`langs/translation-*.json`
Ficheiro de idioma base monolingue	`langs/translation-en.json`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*JSON nested structure file*

Veja também:

JSON, *Personalizar a saída JSON*, *Limpeza de ficheiros de tradução*,

1.10.16 JSON i18next files

Alterado na versão 2.17: Since Weblate 2.17 and with translate-toolkit at-least 2.2.5, i18next JSON files with plurals are supported as well.

i18next is an internationalization framework written in and for JavaScript. Weblate supports its localization files with features such as plurals.

i18next translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Nota: Weblate supports the i18next JSON v3 format. The v2 and v1 variants are mostly compatible, with exception of how plurals are handled.

Example file:

```
{
  "hello": "Hello",
  "apple": "I have an apple",
  "apple_plural": "I have {{count}} apples",
  "apple_negative": "I have no apples"
}
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`langs/*.json`
Ficheiro de idioma base monolingue	`langs/en.json`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*i18next JSON file*

Veja também:

JSON, i18next JSON Format, *Personalizar a saída JSON*, *Limpeza de ficheiros de tradução*

1.10.17 go-i18n JSON files

Novo na versão 4.1.

go-i18n translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Nota: Weblate supports the go-i18n JSON v1 format, for flat JSON formats please use *JSON files*. The v2 format with hash is currently not supported.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`langs/*.json`
Ficheiro de idioma base monolingue	`langs/en.json`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*go-i18n JSON file*

Veja também:

JSON, go-i18n, *Personalizar a saída JSON*, *Limpeza de ficheiros de tradução*,

1.10.18 ARB File

Novo na versão 4.1.

ARB translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`lib/l10n/intl_*.arb`
Ficheiro de idioma base monolingue	`lib/l10n/intl_en.arb`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*ARB file*

Veja também:

JSON, Application Resource Bundle Specification, Internationalizing Flutter apps, *Personalizar a saída JSON*, *Limpeza de ficheiros de tradução*

1.10.19 WebExtension JSON

Novo na versão 2.16: This is supported since Weblate 2.16 and with translate-toolkit at-least 2.2.4.

File format used when translating extensions for Mozilla Firefox or Google Chromium.

Nota: While this format is called JSON, its specification allows to include comments, which are not part of JSON specification. Weblate currently does not support file with comments.

Example file:

```
{
  "hello": {
    "message": "Ahoj světe!\n",
    "description": "Description",
    "placeholders": {
      "url": {
```

(continues on next page)

(continuação da página anterior)

```
        "content": "$1",
        "example": "https://developer.mozilla.org"
      }
    }
  },
  "orangutan": {
    "message": "",
    "description": "Description"
  },
  "try": {
    "message": "",
    "description": "Description"
  },
  "thanks": {
    "message": "",
    "description": "Description"
  }
}
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`_locales/*/messages.json`
Ficheiro de idioma base monolingue	`_locales/en/messages.json`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*WebExtension JSON file*

Veja também:

JSON, Google chrome.i18n, Mozilla Extensions Internationalization

1.10.20 .XML resource files

Novo na versão 2.3.

A .XML resource (.resx) file employs a monolingual XML file format used in Microsoft .NET applications. It is interchangeable with .resw, when using identical syntax to .resx.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`Resources/Language.*.resx`
Ficheiro de idioma base monolingue	`Resources/Language.resx`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Ficheiro de recursos .NET*

Veja também:

.NET Resource files (.resx), *Limpeza de ficheiros de tradução*,

1.10.21 CSV files

Novo na versão 2.4.

CSV files can contain a simple list of source and translation. Weblate supports the following files:

- Files with header defining fields (source, translation, location, …). This is the recommended approach, as it is the least error prone.
- Files with two fields—source and translation (in this order), choose *Simple CSV file* as file format
- Files with fields as defined by translate-toolkit: location, source, target, ID, fuzzy, context, translator_comments, developer_comments

Aviso: The CSV format currently automatically detects the dialect of the CSV file. In some cases the automatic detection might fail and you will get mixed results. This is especially true for CSV files with newlines in the values. As a workaround it is recommended to omit quoting characters.

Example file:

```
Thank you for using Weblate.,Děkujeme za použití Weblate.
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`locale/*.csv`
Ficheiro de idioma base monolingue	*Empty*
Modelo para novas traduções	`locale/en.csv`
Formato de ficheiro	*CSV file*

Veja também:

CSV

1.10.22 YAML files

Novo na versão 2.9.

The plain YAML files with string keys and values. Weblate also extract strings from lists or dictionaries.

Example of a YAML file:

```
weblate:
  hello: ""
  orangutan": ""
  try": ""
  thanks": ""
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`translations/messages.*.yml`
Ficheiro de idioma base monolingue	`translations/messages.en.yml`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*YAML file*

Veja também:

YAML, *Ruby YAML files*

1.10.23 Ruby YAML files

Novo na versão 2.9.

Ruby i18n YAML files with language as root node.

Example Ruby i18n YAML file:

```
cs:
  weblate:
    hello: ""
    orangutan: ""
    try: ""
    thanks: ""
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`translations/messages.*.yml`
Ficheiro de idioma base monolingue	`translations/messages.en.yml`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Ruby YAML file*

Veja também:

YAML, *YAML files*

1.10.24 DTD files

Novo na versão 2.18.

Example DTD file:

```
<!ENTITY hello "">
<!ENTITY orangutan "">
<!ENTITY try "">
<!ENTITY thanks "">
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`locale/*.dtd`
Ficheiro de idioma base monolingue	`locale/en.dtd`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*DTD file*

Veja também:

Mozilla DTD format

1.10.25 Flat XML files

Novo na versão 3.9.

Example of a flat XML file:

```
<?xml version='1.0' encoding='UTF-8'?>
<root>
  <str key="hello_world">Hello World!</str>
  <str key="resource_key">Translated value.</str>
</root>
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`locale/*.xml`
Ficheiro de idioma base monolingue	`locale/en.xml`
Modelo para novas traduções	*Empty*
Formato de ficheiro	*Flat XML file*

Veja também:

Flat XML

1.10.26 Windows RC files

Alterado na versão 4.1: Support for Windows RC files has been rewritten.

Nota: Support for this format is currently in beta, feedback from testing is welcome.

Example Windows RC file:

```
LANGUAGE LANG_CZECH, SUBLANG_DEFAULT

STRINGTABLE
BEGIN
    IDS_MSG1                "Hello, world!\n"
    IDS_MSG2                "Orangutan has %d banana.\n"
    IDS_MSG3                "Try Weblate at http://demo.weblate.org/!\n"
    IDS_MSG4                "Thank you for using Weblate."
END
```

Typical Weblate *Component configuration*	
Máscara de ficheiro	`lang/*.rc`
Ficheiro de idioma base monolingue	`lang/en-US.rc`
Modelo para novas traduções	`lang/en-US.rc`
Formato de ficheiro	*RC file*

Veja também:

Windows RC files

1.10.27 Ficheiros de metadados da App Store

Novo na versão 3.5.

Metadata used for publishing apps in various app stores can be translated. Currently the following tools are compatible:

- Triple-T gradle-play-publisher
- Fastlane
- F-Droid

The metadata consists of several textfiles, which Weblate will present as separate strings to translate.

Typical Weblate *Component configuration*	
Máscara de ficheiro	`fastlane/android/metadata/*`
Ficheiro de idioma base monolingue	`fastlane/android/metadata/en-US`
Modelo para novas traduções	`fastlane/android/metadata/en-US`
Formato de ficheiro	*App store metadata files*

1.10.28 Subtitle files

Novo na versão 3.7.

Weblate pode traduzir vários ficheiros de legenda:

- SubRip subtitle file (`*.srt`)
- MicroDVD subtitle file (`*.sub`)
- Advanced Substation Alpha subtitles file (`*.ass`)
- Substation Alpha subtitle file (`*.ssa`)

Typical Weblate *Component configuration*	
Máscara de ficheiro	`path/*.srt`
Ficheiro de idioma base monolingue	`path/en.srt`
Modelo para novas traduções	`path/en.srt`
Formato de ficheiro	*SubRip subtitle file*

Veja também:

Subtitles

1.10.29 Excel Open XML

Novo na versão 3.2.

Excel Open XML (.xlsx) files can be imported and exported.

When uploading XLSX files for translation, be aware that only the active worksheet is considered, and there must be at least a column called `source` (which contains the source string) and a column called `target` (which contains the translation). Additionally there should be the column called `context` (which contains the context path of the translation string). If you use the XLSX download for exporting the translations into an Excel workbook, you already get a file with the correct file format.

1.10.30 HTML files

Novo na versão 4.1.

Nota: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the HTML files and offered for the translation.

Veja também:

HTML

1.10.31 OpenDocument Format

Novo na versão 4.1.

Nota: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the OpenDocument files and offered for the translation.

Veja também:

OpenDocument Format

1.10.32 IDML Format

Novo na versão 4.1.

Nota: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the Adobe InDesign Markup Language files and offered for the translation.

1.10.33 Outros

Most formats supported by translate-toolkit which support serializing can be easily supported, but they did not (yet) receive any testing. In most cases some thin layer is needed in Weblate to hide differences in behavior of different translate-toolkit storages.

Veja também:

Translation Related File Formats

1.10.34 Adding new translations

Alterado na versão 2.18: In versions prior to 2.18 the behaviour of adding new translations was file format specific.

Weblate can automatically start new translation for all of the file formats.

Some formats expect to start with an empty file and only translated strings to be included (for example *Android string resources*), while others expect to have all keys present (for example *GNU gettext*). In some situations this really doesn't depend on the format, but rather on the framework you use to handle the translation (for example with *JSON files*).

When you specify *Modelo para novas traduções* in *Component configuration*, Weblate will use this file to start new translations. Any exiting translations will be removed from the file when doing so.

When *Modelo para novas traduções* is empty and the file format supports it, an empty file is created where new strings will be added once they are translated.

The *Estilo de código de idioma* allows you to customize language code used in generated filenames:

Predefinição baseada no formato do ficheiro Dependent on file format, for most of them POSIX is used.

Estilo POSIX utilizando o sublinhado como um separador Typically used by gettext and related tools, produces language codes like `pt_BR`.

Estilo de POSIX utilizando o sublinhado como um separador, incluindo o código do país POSIX style language code including the country code even when not necessary (for example `cs_CZ`).

Estilo BCP utilizando o hífen como um separador Typically used on web platforms, produces language codes like `pt-BR`.

Estilo de BCP utilizando o hífen como um separador, incluindo o código do país BCP style language code including the country code even when not necessary (for example `cs-CZ`).

Estilo Android Only used in Android apps, produces language codes like `pt-rBR`.

Estilo Java Used by Java—mostly BCP with legacy codes for Chinese.

Nota: Weblate recognizes any of these when parsing translation files, the above settings only influences how new files are created.

1.10.35 Cadeias somente leitura

Novo na versão 3.10.

Read-only strings from translation files will be included, but can not be edited in Weblate. This feature is natively supported by few formats (*XLIFF* and *Android string resources*), but can be emulated in others by adding a `read-only` flag, see *Personalizar o comportamento*.

1.11 Integração de controlo de versões

Weblate currently supports *Git* (with extended support for *GitHub*, *Gerrit* and *Subversion*) and *Mercurial* as version control backends.

1.11.1 Accessing repositories

The VCS repository you want to use has to be accessible to Weblate. With a publicly available repository you just need to enter the correct URL (for example `https://github.com/WeblateOrg/weblate.git`), but for private repositories or for push URLs the setup is more complex and requires authentication.

Accessing repositories from Hosted Weblate

For Hosted Weblate there is a dedicated push user registered on GitHub, Bitbucket, Codeberg and GitLab (with username *weblate* named *Weblate push user*). You need to add this user as a collaborator and give it appropriate permission to your repository (read only is okay for cloning, write is required for pushing). Depending on service and your organization settings, this happens immediately or requires confirmation from Weblate side.

The invitations on GitHub are accepted automatically within five minutes, on other services manual processing might be needed, so please be patient.

Once the *weblate* user is added, you can configure *Repositório do código-fonte* and *URL de submissão do repositório* using SSH protocol (for example `git@github.com:WeblateOrg/weblate.git`).

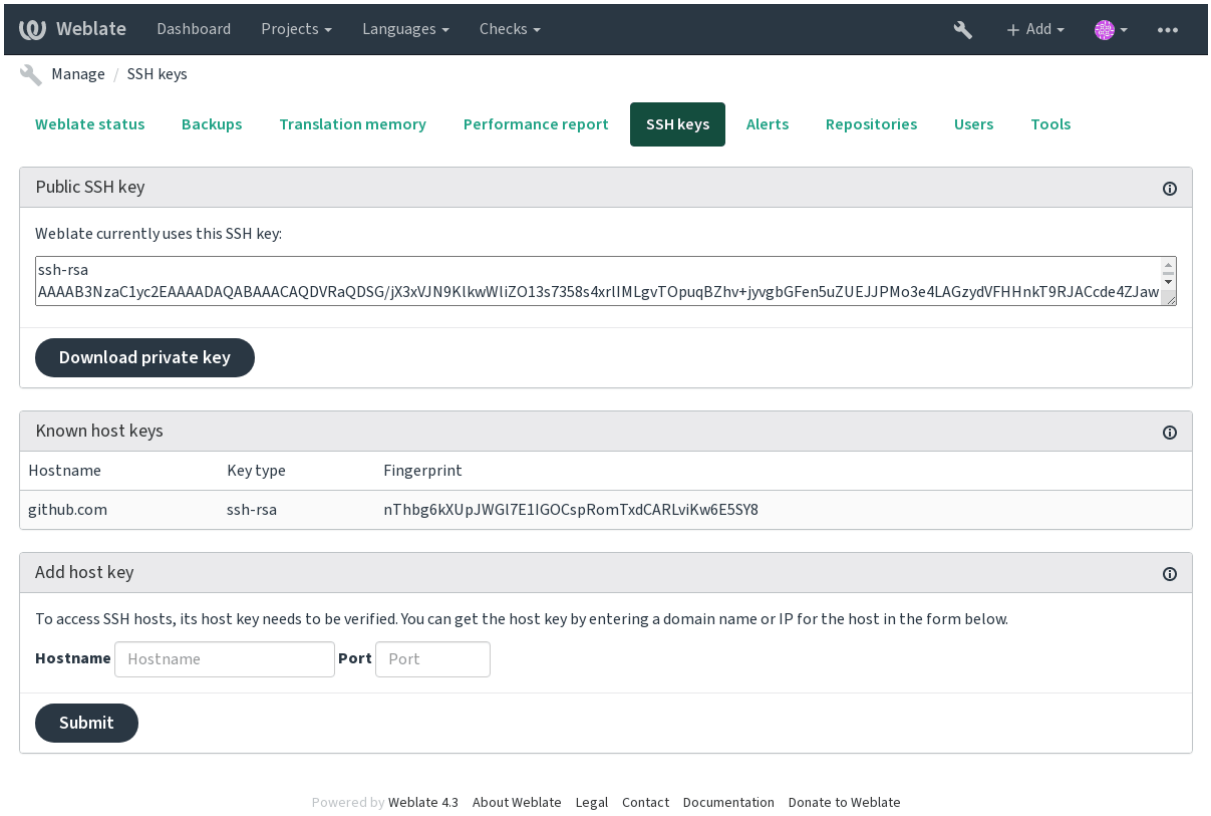
SSH repositories

The most frequently used method to access private repositories is based on SSH. Authorize the public Weblate SSH key (see *Weblate SSH key*) to access the upstream repository this way.

> **Aviso:** On GitHub, each key can be added to only one repository, see *GitHub repositories* and *Accessing repositories from Hosted Weblate*.

Weblate also stores the host key fingerprint upon first connection, and fails to connect to the host should it be changed later (see *Verifying SSH host keys*).

In case adjustment is needed, do so from the Weblate admin interface:



Weblate SSH key

The Weblate public key is visible to all users browsing the *About* page.

Admins can generate or display the public key currently used by Weblate in the connection (from *SSH keys*) on the admin interface landing page.

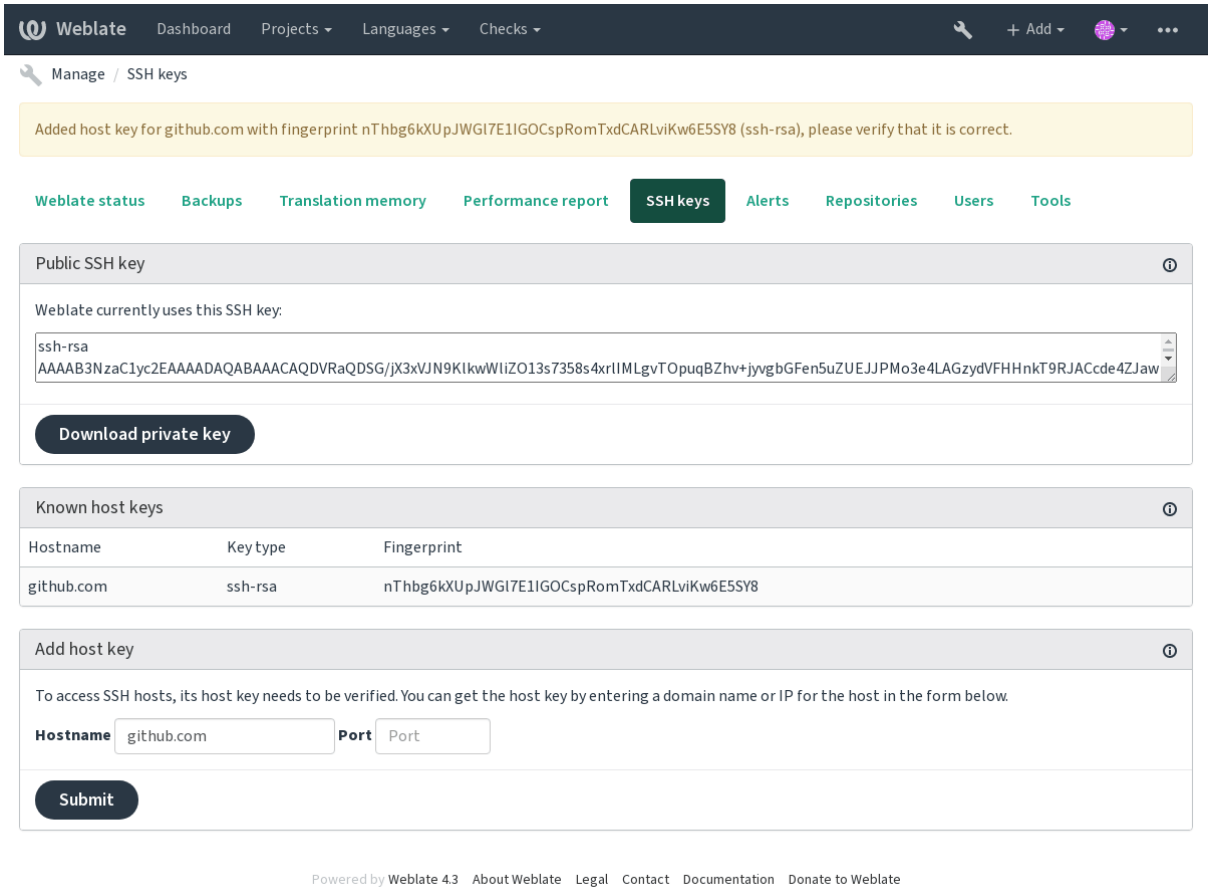
Nota: The corresponding private SSH key can not currently have a password, so make sure it is well protected.

Dica: Make a backup of the generated private Weblate SSH key.

Verifying SSH host keys

Weblate automatically remembers the SSH host keys on first access and remembers them for further use.

In case you want to verify them before connecting to the repository, verify the SSH host keys of the servers you are going to access in *Add host key*, from the same section of the admin interface. Enter the hostname you are going to access (e.g. `gitlab.com`), and press *Submit*. Verify its fingerprint matches the server you added. They are shown in the confirmation message:



GitHub repositories

Access via SSH is possible (see *SSH repositories*), but in case you need to access more than one repository, you will hit a GitHub limitation on allowed SSH key usage (since one key can be used only for one repository).

In case the *Ramo do push* is not set, the project is forked and changes pushed through a fork. In case it is set, changes are pushed to the upstream repository and chosen branch.

For smaller deployments, use HTTPS authentication with a personal access token and your GitHub account, see Creating an access token for command-line use.

For bigger setups, it is usually better to create a dedicated user for Weblate, assign it the public SSH key generated in Weblate (see *Weblate SSH key*) and grant it access to all the repositories you want to translate. This approach is also used for Hosted Weblate, there is dedicated *weblate* user for that.

Veja também:

Accessing repositories from Hosted Weblate

Weblate internal URLs

To share one repository between different components you can use a special URL like `weblate://project/component`. This way, the component will share the VCS repository configuration with the referenced component (`project/component` in the example).

Weblate automatically adjusts repository URL when creating component when it finds component with matching repository setup. You can override this in last step of component configuration.

Reasons to use this:

- Saves disk space on the server, the repository is stored just once.
- Makes the updates faster, only one repository is updated.
- There is just single exported repository with Weblate translations (see *Git exporter*).
- Some addons can operate on more components sharing single repository, for example *Squash de commits git*.

HTTPS repositories

To access protected HTTPS repositories, include the username and password in the URL. Don't worry, Weblate will strip this info when the URL is shown to users (if even allowed to see the repository URL at all).

For example the GitHub URL with authentication added might look like: `https://user:your_access_token@github.com/WeblateOrg/weblate.git`.

Nota: If your username or password contains special characters, those have to be URL encoded, for example `https://user%40example.com:%24password%23@bitbucket.org/...`.

Using proxy

If you need to access HTTP/HTTPS VCS repositories using a proxy server, configure the VCS to use it.

This can be done using the `http_proxy`, `https_proxy`, and `all_proxy` environment variables, (as described in the cURL documentation) or by enforcing it in the VCS configuration, for example:

```
git config --global http.proxy http://user:password@proxy.example.com:80
```

Nota: The proxy configuration needs to be done under user running Weblate (see also *Permissões do sistema de ficheiros*) and with `HOME=$DATA_DIR/home` (see *DATA_DIR*), otherwise Git executed by Weblate will not use it.

Veja também:

The cURL manpage, Git config documentation

1.11.2 Git

Veja também:

See *Accessing repositories* for info on how to access different kinds of repositories.

Git com push forçado

This behaves exactly like Git itself, the only difference being that it always force pushes. This is intended only in the case of using a separate repository for translations.

Aviso: Use with caution, as this easily leads to lost commits in your upstream repository.

Customizing Git configuration

Weblate invokes all VCS commands with `HOME=$DATA_DIR/home` (see *DATA_DIR*), therefore editing the user configuration needs to be done in `DATA_DIR/home/.git`.

Git remote helpers

You can also use Git remote helpers for additionally supporting other version control systems, but be prepared to debug problems this may lead to.

At this time, helpers for Bazaar and Mercurial are available within separate repositories on GitHub: git-remote-hg and git-remote-bzr. Download them manually and put somewhere in your search path (for example `~/bin`). Make sure you have the corresponding version control systems installed.

Once you have these installed, such remotes can be used to specify a repository in Weblate.

To clone the `gnuhello` project from Launchpad using Bazaar:

```
bzr::lp:gnuhello
```

For the `hello` repository from selenic.com using Mercurial:

```
hg::http://selenic.com/repo/hello
```

Aviso: The inconvenience of using Git remote helpers is for example with Mercurial, the remote helper sometimes creates a new tip when pushing changes back.

1.11.3 GitHub

Novo na versão 2.3.

This adds a thin layer atop *Git* using the Github API to allow pushing translation changes as pull requests, instead of pushing directly to the repository.

Git pushes changes directly to a repository, while *GitHub* creates pull requests. The latter is not needed for merely accessing Git repositories.

Veja também:

Fazendo push das alterações do Weblate

Pushing changes to GitHub as pull requests

If not wanting to push translations to a GitHub repository, they can be sent as either one or many pull requests instead.

You need to configure API credentials to make this work.

Veja também:

GITHUB_USERNAME, *GITHUB_TOKEN*, *GITHUB_CREDENTIALS*

1.11.4 GitLab

Novo na versão 3.9.

This just adds a thin layer atop *Git* using the GitLab API to allow pushing translation changes as merge requests instead of pushing directly to the repository.

There is no need to use this to access Git repositories, ordinary *Git* works the same, the only difference is how pushing to a repository is handled. With *Git* changes are pushed directly to the repository, while *GitLab* creates merge request.

Veja também:

Fazendo push das alterações do Weblate

Pushing changes to GitLab as merge requests

If not wanting to push translations to a GitLab repository, they can be sent as either one or many merge requests instead.

You need to configure API credentials to make this work.

Veja também:

GITLAB_USERNAME, *GITLAB_TOKEN*, *GITLAB_CREDENTIALS*

1.11.5 Pagure

Novo na versão 4.3.2.

This just adds a thin layer atop *Git* using the Pagure API to allow pushing translation changes as merge requests instead of pushing directly to the repository.

There is no need to use this to access Git repositories, ordinary *Git* works the same, the only difference is how pushing to a repository is handled. With *Git* changes are pushed directly to the repository, while *Pagure* creates merge request.

Veja também:

Fazendo push das alterações do Weblate

Enviar alterações para o Pagure como solicitações de mesclagem

If not wanting to push translations to a Pagure repository, they can be sent as either one or many merge requests instead.

You need to configure API credentials to make this work.

Veja também:

PAGURE_USERNAME, *PAGURE_TOKEN*, *PAGURE_CREDENTIALS*

1.11.6 Gerrit

Novo na versão 2.2.

Adds a thin layer atop *Git* using the git-review tool to allow pushing translation changes as Gerrit review requests, instead of pushing them directly to the repository.

The Gerrit documentation has the details on the configuration necessary to set up such repositories.

1.11.7 Mercurial

Novo na versão 2.1.

Mercurial is another VCS you can use directly in Weblate.

Nota: It should work with any Mercurial version, but there are sometimes incompatible changes to the command-line interface which breaks Weblate integration.

Veja também:

See *Accessing repositories* for info on how to access different kinds of repositories.

1.11.8 Subversion

Novo na versão 2.8.

Weblate uses git-svn to interact with subversion repositories. It is a Perl script that lets subversion be used by a Git client, enabling users to maintain a full clone of the internal repository and commit locally.

Nota: Weblate tries to detect Subversion repository layout automatically - it supports both direct URLs for branch or repositories with standard layout (branches/, tags/ and trunk/). More info about this is to be found in the git-svn documentation. If your repository does not have a standard layout and you encounter errors, try including the branch name in the repository URL and leaving branch empty.

Alterado na versão 2.19: Before this, there was only support for standard layout repositories.

Subversion credentials

Weblate expects you to have accepted the certificate up-front and if needed, your credentials. It will look to insert them into the DATA_DIR directory. Accept the certificate by using *svn* once with the *$HOME* environment variable set to the DATA_DIR:

```
# Use DATA_DIR as configured in Weblate settings.py, it is /app/data in the Docker
HOME=${DATA_DIR}/home svn co https://svn.example.com/example
```

Veja também:

DATA_DIR

1.11.9 Local files

Novo na versão 3.8.

Weblate can also operate without a remote VCS. The initial translations are imported by uploading them. Later you can replace individual files by file upload, or add translation strings directly from Weblate (currently available only for monolingual translations).

In the background Weblate creates a Git repository for you and all changes are tracked in. In case you later decide to use a VCS to store the translations, you already have a repository within Weblate can base your integration on.

1.12 Weblate's REST API

Novo na versão 2.6: The REST API is available since Weblate 2.6.

The API is accessible on the `/api/` URL and it is based on Django REST framework. You can use it directly or by *Cliente Weblate*.

1.12.1 Authentication and generic parameters

The public project API is available without authentication, though unauthenticated requests are heavily throttled (by default to 100 requests per day), so it is recommended to use authentication. The authentication uses a token, which you can get in your profile. Use it in the `Authorization` header:

ANY /

Generic request behaviour for the API, the headers, status codes and parameters here apply to all endpoints as well.

Query Parameters

- **format** – Response format (overrides Accept). Possible values depends on REST framework setup, by default `json` and `api` are supported. The latter provides web browser interface for API.

Request Headers

- Accept – the response content type depends on Accept header
- Authorization – optional token to authenticate

Response Headers

- Content-Type – this depends on Accept header of request
- Allow – list of allowed HTTP methods on object

Response JSON Object

- **detail** (*string*) – verbose description of failure (for HTTP status codes other than 200 OK)
- **count** (*int*) – total item count for object lists
- **next** (*string*) – next page URL for object lists
- **previous** (*string*) – previous page URL for object lists
- **results** (*array*) – results for object lists
- **url** (*string*) – URL to access this resource using API
- **web_url** (*string*) – URL to access this resource using web browser

Status Codes

- 200 OK – when request was correctly handled

- 400 Bad Request – when form parameters are missing
- 403 Forbidden – when access is denied
- 429 Too Many Requests – when throttling is in place

Authentication examples

Example request:

```
GET /api/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
Authorization: Token YOUR-TOKEN
```

Example response:

```
HTTP/1.0 200 OK
Date: Fri, 25 Mar 2016 09:46:12 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS

{
    "projects":"http://example.com/api/projects/",
    "components":"http://example.com/api/components/",
    "translations":"http://example.com/api/translations/",
    "languages":"http://example.com/api/languages/"
}
```

CURL example:

```
curl \
    -H "Authorization: Token TOKEN" \
    https://example.com/api/
```

Passing Parameters Examples

For the POST method the parameters can be specified either as form submission (*application/x-www-form-urlencoded*) or as JSON (*application/json*).

Form request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Token TOKEN

operation=pull
```

JSON request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
```

(continues on next page)

(continuação da página anterior)

```
Content-Length: 20

{"operation":"pull"}
```

CURL example:

```
curl \
    -d operation=pull \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

CURL JSON example:

```
curl \
    --data-binary '{"operation":"pull"}' \
    -H "Content-Type: application/json" \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

Limitação de taxa

The API requests are rate limited; the default configuration limits it to 100 requests per day for anonymous users and 5000 requests per hour for authenticated users.

Rate limiting can be adjusted in the `settings.py`; see Throttling in Django REST framework documentation for more details how to configure it.

The status of rate limiting is reported in following headers:

`X-RateLimit-Limit`	Rate limiting limit of requests to perform
`X-RateLimit-Remaining`	Remaining limit of requests
`X-RateLimit-Reset`	Number of seconds until ratelimit window resets

Alterado na versão 4.1: Added ratelimiting status headers.

1.12.2 API Entry Point

`GET /api/`
The API root entry point.

Example request:

```
GET /api/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
Authorization: Token YOUR-TOKEN
```

Example response:

```
HTTP/1.0 200 OK
Date: Fri, 25 Mar 2016 09:46:12 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS
```

(continues on next page)

(continuação da página anterior)

```
{
    "projects":"http://example.com/api/projects/",
    "components":"http://example.com/api/components/",
    "translations":"http://example.com/api/translations/",
    "languages":"http://example.com/api/languages/"
}
```

1.12.3 Utilizadores

Novo na versão 4.0.

GET /api/users/
Returns a list of users if you have permissions to see manage users. If not, then you get to see only your own details.

Veja também:

Users object attributes are documented at *GET /api/users/(str:username)/*.

POST /api/users/
Creates a new user.

Parameters

- **username** (*string*) – Nome de utilizador
- **full_name** (*string*) – User full name
- **email** (*string*) – User email
- **is_superuser** (*boolean*) – Is user superuser? (optional)
- **is_active** (*boolean*) – Is user active? (optional)

GET /api/users/(str: *username*)/
Returns information about users.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **username** (*string*) – username of a user
- **full_name** (*string*) – full name of a user
- **email** (*string*) – email of a user
- **is_superuser** (*boolean*) – whether the user is a super user
- **is_active** (*boolean*) – whether the user is active
- **date_joined** (*string*) – date the user is created
- **groups** (*array*) – link to associated groups; see *GET /api/groups/(int:id)/*

Example JSON data:

```
{
    "email": "user@example.com",
    "full_name": "Example User",
    "username": "exampleusername",
    "groups": [
        "http://example.com/api/groups/2/",
        "http://example.com/api/groups/3/"
    ],
```

(continues on next page)

(continuação da página anterior)

```
    "is_superuser": true,
    "is_active": true,
    "date_joined": "2020-03-29T18:42:42.617681Z",
    "url": "http://example.com/api/users/exampleusername/",
    "statistics_url": "http://example.com/api/users/exampleusername/statistics/
↪"
}
```

PUT /api/users/(**str:** *username*)**/**
Changes the user parameters.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **username** (*string*) – username of a user
- **full_name** (*string*) – full name of a user
- **email** (*string*) – email of a user
- **is_superuser** (*boolean*) – whether the user is a super user
- **is_active** (*boolean*) – whether the user is active
- **date_joined** (*string*) – date the user is created

PATCH /api/users/(**str:** *username*)**/**
Changes the user parameters.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **username** (*string*) – username of a user
- **full_name** (*string*) – full name of a user
- **email** (*string*) – email of a user
- **is_superuser** (*boolean*) – whether the user is a super user
- **is_active** (*boolean*) – whether the user is active
- **date_joined** (*string*) – date the user is created

DELETE /api/users/(**str:** *username*)**/**
Deletes all user information and marks the user inactive.

Parameters

- **username** (*string*) – User's username

POST /api/users/(**str:** *username*)**/groups/**
Associate groups with a user.

Parameters

- **username** (*string*) – User's username

Form Parameters

- **string group_id** – The unique group ID

GET /api/users/(**str:** *username*)**/statistics/**
List statistics of a user.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **translated** (*int*) – Quantidade de traduções por utilizador
- **suggested** (*int*) – Quantidade de sugestões por utilizador
- **uploaded** (*int*) – Quantidade de envios por utilizador
- **commented** (*int*) – Quantidade de comentários por utilizador
- **languages** (*int*) – Quantidade de idiomas que o utilizador pode traduzir

GET /api/users/(str: *username*)/notifications/
List subscriptions of a user.

Parameters

- **username** (*string*) – User's username

POST /api/users/(str: *username*)/notifications/
Associate subscriptions with a user.

Parameters

- **username** (*string*) – User's username

Request JSON Object

- **notification** (*string*) – Name of notification registered
- **scope** (*int*) – Scope of notification from the available choices
- **frequency** (*int*) – Frequency choices for notifications

GET /api/users/(str: *username*)/notifications/
int: *subscription_id*/ Get a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** (*int*) – ID da notificação registada

PUT /api/users/(str: *username*)/notifications/
int: *subscription_id*/ Edit a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** (*int*) – ID da notificação registada

Request JSON Object

- **notification** (*string*) – Name of notification registered
- **scope** (*int*) – Scope of notification from the available choices
- **frequency** (*int*) – Frequency choices for notifications

PATCH /api/users/(str: *username*)/notifications/
int: *subscription_id*/ Edit a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** (*int*) – ID da notificação registada

Request JSON Object

- **notification** (*string*) – Name of notification registered
- **scope** (*int*) – Scope of notification from the available choices

- **frequency** (*int*) – Frequency choices for notifications

DELETE /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Delete a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** – Name of notification registered
- **subscription_id** – int

1.12.4 Grupos

Novo na versão 4.0.

GET /api/groups/
Returns a list of groups if you have permissions to see manage groups. If not, then you get to see only the groups the user is a part of.

Veja também:

Group object attributes are documented at *GET /api/groups/(int:id)/*.

POST /api/groups/
Creates a new group.

Parameters

- **name** (*string*) – Nome do grupo
- **project_selection** (*int*) – Group of project selection from given options
- **language_selection** (*int*) – Group of languages selected from given options

GET /api/groups/(**int:** *id*)**/**
Returns information about group.

Parameters

- **id** (*int*) – Group's ID

Response JSON Object

- **name** (*string*) – name of a group
- **project_selection** (*int*) – integer corresponding to group of projects
- **language_selection** (*int*) – integer corresponding to group of languages
- **roles** (*array*) – link to associated roles; see *GET /api/roles/(int:id)/*
- **projects** (*array*) – link to associated projects; see *GET /api/projects/(string:project)/*
- **components** (*array*) – link to associated components; see *GET /api/components/(string:project)/(string:component)/*
- **componentlist** (*array*) – link to associated componentlist; see *GET /api/component-lists/(str:slug)/*

Example JSON data:

```
{
    "name": "Guests",
    "project_selection": 3,
    "language_selection": 1,
    "url": "http://example.com/api/groups/1/",
```

(continues on next page)

(continuação da página anterior)

```
    "roles": [
        "http://example.com/api/roles/1/",
        "http://example.com/api/roles/2/"
    ],
    "languages": [
        "http://example.com/api/languages/en/",
        "http://example.com/api/languages/cs/",
    ],
    "projects": [
        "http://example.com/api/projects/demo1/",
        "http://example.com/api/projects/demo/"
    ],
    "componentlist": "http://example.com/api/component-lists/new/",
    "components": [
        "http://example.com/api/components/demo/weblate/"
    ]
}
```

PUT /api/groups/(int: *id*)/
Changes the group parameters.

Parameters

- **id** (*int*) – Group's ID

Response JSON Object

- **name** (*string*) – name of a group
- **project_selection** (*int*) – integer corresponding to group of projects
- **language_selection** (*int*) – integer corresponding to group of Languages

PATCH /api/groups/(int: *id*)/
Changes the group parameters.

Parameters

- **id** (*int*) – Group's ID

Response JSON Object

- **name** (*string*) – name of a group
- **project_selection** (*int*) – integer corresponding to group of projects
- **language_selection** (*int*) – integer corresponding to group of languages

DELETE /api/groups/(int: *id*)/
Deletes the group.

Parameters

- **id** (*int*) – Group's ID

POST /api/groups/(int: *id*)/roles/
Associate roles with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string role_id** – The unique role ID

POST /api/groups/(int: *id*)/components/
Associate components with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string component_id** – The unique component ID

DELETE /api/groups/(**int:** *id*)**/components/**
int: *component_id* Delete component from a group.

Parameters

- **id** (*int*) – Group's ID
- **component_id** (*int*) – The unique component ID

POST /api/groups/(**int:** *id*)**/projects/**
Associate projects with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string project_id** – The unique project ID

DELETE /api/groups/(**int:** *id*)**/projects/**
int: *project_id* Delete project from a group.

Parameters

- **id** (*int*) – Group's ID
- **project_id** (*int*) – The unique project ID

POST /api/groups/(**int:** *id*)**/languages/**
Associate languages with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string language_code** – The unique language code

DELETE /api/groups/(**int:** *id*)**/languages/**
string: *language_code* Delete language from a group.

Parameters

- **id** (*int*) – Group's ID
- **language_code** (*string*) – The unique language code

POST /api/groups/(**int:** *id*)**/componentlists/**
Associate componentlists with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string component_list_id** – The unique componentlist ID

DELETE /api/groups/(**int:** *id*)**/componentlists/**
int: *component_list_id* Delete componentlist from a group.

Parameters

- **id** (*int*) – Group's ID
- **component_list_id** (*int*) – The unique componentlist ID

1.12.5 Funções

GET /api/roles/
Returns a list of all roles associated with user. If user is superuser, then list of all existing roles is returned.

Veja também:

Roles object attributes are documented at *GET /api/roles/(int:id)/*.

POST /api/roles/
Creates a new role.

Parameters

- **name** (*string*) – Role name
- **permissions** (*array*) – List of codenames of permissions

GET /api/roles/(int: *id*)/
Returns information about a role.

Parameters

- **id** (*int*) – Role ID

Response JSON Object

- **name** (*string*) – Role name
- **permissions** (*array*) – list of codenames of permissions

Example JSON data:

```
{
    "name": "Access repository",
    "permissions": [
        "vcs.access",
        "vcs.view"
    ],
    "url": "http://example.com/api/roles/1/",
}
```

PUT /api/roles/(int: *id*)/
Changes the role parameters.

Parameters

- **id** (*int*) – Role's ID

Response JSON Object

- **name** (*string*) – Role name
- **permissions** (*array*) – list of codenames of permissions

PATCH /api/roles/(int: *id*)/
Changes the role parameters.

Parameters

- **id** (*int*) – Role's ID

Response JSON Object

- **name** (*string*) – Role name
- **permissions** (*array*) – list of codenames of permissions

DELETE /api/roles/(int: *id*)/
Deletes the role.

Parameters

- **id** (*int*) – Role's ID

1.12.6 Idiomas

GET /api/languages/
Returns a list of all languages.

Veja também:

Language object attributes are documented at *GET /api/languages/(string:language)/*.

POST /api/languages/
Creates a new language.

Parameters

- **code** (*string*) – Nome do idioma
- **name** (*string*) – Nome do idioma
- **direction** (*string*) – Language direction
- **plural** (*object*) – Language plural formula and number

GET /api/languages/(string: *language*)/
Returns information about a language.

Parameters

- **language** (*string*) – Código do idioma

Response JSON Object

- **code** (*string*) – Código do idioma
- **direction** (*string*) – Direção do texto
- **plural** (*object*) – Object of language plural information
- **aliases** (*array*) – Array of aliases for language

Example JSON data:

```
{
    "code": "en",
    "direction": "ltr",
    "name": "English",
    "plural": {
        "id": 75,
        "source": 0,
        "number": 2,
        "formula": "n != 1",
        "type": 1
    },
    "aliases": [
        "english",
        "en_en",
        "base",
        "source",
        "eng"
    ],
    "url": "http://example.com/api/languages/en/",
    "web_url": "http://example.com/languages/en/",
    "statistics_url": "http://example.com/api/languages/en/statistics/"
}
```

PUT /api/languages/(string: *language*)/
Changes the language parameters.

Parameters

- **language** (*string*) – Language's code

Request JSON Object

- **name** (*string*) – Nome do idioma
- **direction** (*string*) – Language direction
- **plural** (*object*) – Language plural details

PATCH /api/languages/(string: language)/
Changes the language parameters.

Parameters

- **language** (*string*) – Language's code

Request JSON Object

- **name** (*string*) – Nome do idioma
- **direction** (*string*) – Language direction
- **plural** (*object*) – Language plural details

DELETE /api/languages/(string: language)/
Deletes the Language.

Parameters

- **language** (*string*) – Language's code

GET /api/languages/(string: language)/statistics/
Returns statistics for a language.

Parameters

- **language** (*string*) – Código do idioma

Response JSON Object

- **total** (*int*) – total number of strings
- **total_words** (*int*) – total number of words
- **last_change** (*timestamp*) – last changes in the language
- **recent_changes** (*int*) – total number of changes
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **translated_words** (*int*) – number of translated words
- **translated_words_percent** (*int*) – percentage of translated words
- **translated_chars** (*int*) – number of translated characters
- **translated_chars_percent** (*int*) – percentage of translated characters
- **total_chars** (*int*) – number of total characters
- **fuzzy** (*int*) – number of fuzzy strings
- **fuzzy_percent** (*int*) – percentage of fuzzy strings
- **failing** (*int*) – number of failing strings
- **failing** – percentage of failing strings

1.12.7 Projetos

GET /api/projects/
Returns a list of all projects.

Veja também:

Project object attributes are documented at *GET /api/projects/(string:project)/*.

POST /api/projects/
Novo na versão 3.9.

Creates a new project.

Parameters

- **name** (*string*) – Nome do projeto
- **slug** (*string*) – Project slug
- **web** (*string*) – Site da Web do Projeto

GET /api/projects/(string: *project*)/
Returns information about a project.

Parameters

- **project** (*string*) – URL semântico do projeto

Response JSON Object

- **name** (*string*) – project name
- **slug** (*string*) – project slug
- **web** (*string*) – project website
- **components_list_url** (*string*) – URL to components list; see *GET /api/projects/(string:project)/components/*
- **repository_url** (*string*) – URL to repository status; see *GET /api/projects/(string:project)/repository/*
- **changes_list_url** (*string*) – URL to changes list; see *GET /api/projects/(string:project)/changes/*

Example JSON data:

```
{
    "name": "Hello",
    "slug": "hello",
    "url": "http://example.com/api/projects/hello/",
    "web": "https://weblate.org/",
    "web_url": "http://example.com/projects/hello/"
}
```

PATCH /api/projects/(string: *project*)/
Novo na versão 4.3.

Edit a project by a patch request.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

PUT /api/projects/(string: *project*)/
Novo na versão 4.3.

Edit a project by a put request.

Parameters

- **project** (*string*) – URL semântico do projeto

DELETE /api/projects/(string: project)/
Novo na versão 3.9.

Deletes a project.

Parameters

- **project** (*string*) – URL semântico do projeto

GET /api/projects/(string: project)/changes/
Returns a list of project changes. This is essentially a project scoped *GET /api/changes/* accepting same params.

Parameters

- **project** (*string*) – URL semântico do projeto

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/changes/(int:id)/*

GET /api/projects/(string: project)/repository/
Returns information about VCS repository status. This endpoint contains only an overall summary for all repositories for the project. To get more detailed status use *GET /api/components/(string:project)/(string:component)/repository/*.

Parameters

- **project** (*string*) – URL semântico do projeto

Response JSON Object

- **needs_commit** (*boolean*) – whether there are any pending changes to commit
- **needs_merge** (*boolean*) – whether there are any upstream changes to merge
- **needs_push** (*boolean*) – whether there are any local changes to push

Example JSON data:

```
{
    "needs_commit": true,
    "needs_merge": false,
    "needs_push": true
}
```

POST /api/projects/(string: project)/repository/
Performs given operation on the VCS repository.

Parameters

- **project** (*string*) – URL semântico do projeto

Request JSON Object

- **operation** (*string*) – Operation to perform: one of `push`, `pull`, `commit`, `reset`, `cleanup`

Response JSON Object

- **result** (*boolean*) – result of the operation

CURL example:

```
curl \
    -d operation=pull \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/repository/
```

JSON request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"result":true}
```

GET /api/projects/(string: *project*)/components/
Returns a list of translation components in the given project.

Parameters

- **project** (*string*) – URL semântico do projeto

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/components/(string:project)/(string:component)/*

POST /api/projects/(string: *project*)/components/
Novo na versão 3.9.

Alterado na versão 4.3: The `zipfile` and `docfile` parameters are now accepted for VCS less components, see *Local files*.

Creates translation components in the given project.

Parameters

- **project** (*string*) – URL semântico do projeto

Request JSON Object

- **zipfile** (*file*) – ZIP file to upload into Weblate for translations initialization
- **docfile** (*file*) – Documento para traduzir

Response JSON Object

- **result** (*object*) – Created component object; see *GET /api/components/(string:project)/(string:component)/*

CURL example:

```
curl \
    --data-binary '{
        "branch": "master",
        "file_format": "po",
        "filemask": "po/*.po",
        "git_export": "",
        "license": "",
        "license_url": "",
        "name": "Weblate",
        "slug": "weblate",
        "repo": "file:///home/nijel/work/weblate-hello",
        "template": "",
        "new_base": "",
        "vcs": "git"
    }' \
    -H "Content-Type: application/json" \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/components/
```

JSON request example:

```
POST /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "vcs": "git"
}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
```

(continues on next page)

(continuação da página anterior)


```
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
        },
        "url": "http://example.com/api/projects/hello/",
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```


GET /api/projects/(string: *project*)/languages/
Returns paginated statistics for all languages within a project.

Novo na versão 3.8.

Parameters

- **project** (*string*) – URL semântico do projeto

Response JSON Object

- **results** (*array*) – array of translation statistics objects
- **language** (*string*) – language name
- **code** (*string*) – language code
- **total** (*int*) – total number of strings
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **total_words** (*int*) – total number of words
- **translated_words** (*int*) – number of translated words
- **words_percent** (*float*) – percentage of translated words

GET /api/projects/(string: *project*)/statistics/
Returns statistics for a project.

Novo na versão 3.8.

Parameters

- **project** (*string*) – URL semântico do projeto

Response JSON Object

- **total** (*int*) – total number of strings
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **total_words** (*int*) – total number of words

- **translated_words** (*int*) – number of translated words
- **words_percent** (*float*) – percentage of translated words

1.12.8 Componentes

GET /api/components/
Returns a list of translation components.

Veja também:

Component object attributes are documented at *GET /api/components/(string:project)/(string:component)/*.

GET /api/components/(string: *project*)/(string: *component*)/
Returns information about translation component.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **project** (*object*) – the translation project; see *GET /api/projects/(string:project)/*
- **name** (*string*) – *Nome do componente*
- **slug** (*string*) – *Component slug*
- **vcs** (*string*) – *Sistema de controlo de versões*
- **repo** (*string*) – *Repositório do código-fonte*
- **git_export** (*string*) – *URL do repositório exportado*
- **branch** (*string*) – *Ramo do repositório*
- **push_branch** (*string*) – *Ramo do push*
- **filemask** (*string*) – *File mask*
- **template** (*string*) – *Ficheiro de idioma base monolingue*
- **edit_template** (*string*) – *Editar ficheiro base*
- **intermediate** (*string*) – *Ficheiro de idioma intermédio*
- **new_base** (*string*) – *Modelo para novas traduções*
- **file_format** (*string*) – *Formato de ficheiro*
- **license** (*string*) – *Licença da tradução*
- **agreement** (*string*) – *Acordo de contribuidor*
- **new_lang** (*string*) – *Adicionar nova tradução*
- **language_code_style** (*string*) – *Estilo de código de idioma*
- **source_language** (*object*) – source language object; see *GET /api/languages/(string:language)/*
- **push** (*string*) – *URL de submissão do repositório*
- **check_flags** (*string*) – *Marcadores de tradução*
- **priority** (*string*) – *Prioridade*
- **enforced_checks** (*string*) – *Verificações impostas*

- **restricted** (*string*) – *Restricted access*
- **repoweb** (*string*) – *Navegador do repositório*
- **report_source_bugs** (*string*) – *Endereço para reportar erros na cadeia fonte*
- **merge_style** (*string*) – *Estilo de união*
- **commit_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **add_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **delete_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **merge_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **addon_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **allow_translation_propagation** (*string*) – *Permitir propagação da tradução*
- **enable_suggestions** (*string*) – *Ativar sugestões*
- **suggestion_voting** (*string*) – *Votação de sugestão*
- **suggestion_autoaccept** (*string*) – *Aceitar sugestões automaticamente*
- **push_on_commit** (*string*) – *Enviar ao submeter*
- **commit_pending_age** (*string*) – *Idade das alterações a fazer commit*
- **auto_lock_error** (*string*) – *Bloquear com erro*
- **language_regex** (*string*) – *Filtro de idioma*
- **variant_regex** (*string*) – *Expressão regular das variantes*
- **repository_url** (*string*) – URL to repository status; see *GET /api/components/(string:project)/(string:component)/repository/*
- **translations_url** (*string*) – URL to translations list; see *GET /api/components/(string:project)/(string:component)/translations/*
- **lock_url** (*string*) – URL to lock status; see *GET /api/components/(string:project)/(string:component)/lock/*
- **changes_list_url** (*string*) – URL to changes list; see *GET /api/components/(string:project)/(string:component)/changes/*

Example JSON data:


```
{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
```


(continues on next page)

(continuação da página anterior)

```
        },
        "url": "http://example.com/api/projects/hello/",
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
    },
    "source_language": {
        "code": "en",
        "direction": "ltr",
        "name": "English",
        "url": "http://example.com/api/languages/en/",
        "web_url": "http://example.com/languages/en/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```

PATCH /api/components/(string: *project*)/(string: *component*)/ Edit a component by a patch request.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **source_language** (*string*) – Project source language code (optional)

Request JSON Object

- **name** (*string*) – name of component
- **slug** (*string*) – slug of component
- **repo** (*string*) – VCS repository URL

CURL example:

```
curl \
    --data-binary '{"name": "new name"}' \
    -H "Content-Type: application/json" \
    -H "Authorization: Token TOKEN" \
    PATCH http://example.com/api/projects/hello/components/
```

JSON request example:

```
PATCH /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{
    "name": "new name"
}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
```

(continues on next page)

(continuação da página anterior)

```
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "new name",
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
        },
        "url": "http://example.com/api/projects/hello/",
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```

PUT /api/components/(string: *project*)/(string: *component*)/

Edit a component by a put request.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Request JSON Object

- **branch** (*string*) – VCS repository branch
- **file_format** (*string*) – file format of translations
- **filemask** (*string*) – mask of translation files in the repository
- **name** (*string*) – name of component
- **slug** (*string*) – slug of component
- **repo** (*string*) – VCS repository URL
- **template** (*string*) – base file for monolingual translations
- **new_base** (*string*) – base file for adding new translations
- **vcs** (*string*) – version control system

DELETE /api/components/(**string:** *project*)**/**
string: *component***/** Novo na versão 3.9.

Deletes a component.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

GET /api/components/(**string:** *project*)**/**
string: *component***/changes/** Returns a list of component changes. This is essentially a component scoped *GET /api/changes/* accepting same params.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/changes/(int:id)/*

GET /api/components/(**string:** *project*)**/**
string: *component***/screenshots/** Returns a list of component screenshots.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **results** (*array*) – array of component screenshots; see *GET /api/screenshots/(int:id)/*

GET /api/components/(**string:** *project*)**/**
string: *component***/lock/** Returns component lock status.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **locked** (*boolean*) – whether component is locked for updates

Example JSON data:

```
{
    "locked": false
}
```

POST /api/components/(**string:** *project*)**/**
string: *component***/lock/** Sets component lock status.

Response is same as *GET /api/components/(string:project)/(string:component)/lock/*.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Request JSON Object

- **lock** – Boolean whether to lock or not.

CURL example:

```
curl \
    -d lock=true \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

JSON request example:

```
POST /api/components/hello/weblate/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"lock": true}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"locked":true}
```

GET /api/components/(string: *project*)/(string: *component*)/repository/ Returns information about VCS repository status.

The response is same as for *GET /api/projects/(string:project)/repository/*.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **needs_commit** (*boolean*) – whether there are any pending changes to commit
- **needs_merge** (*boolean*) – whether there are any upstream changes to merge
- **needs_push** (*boolean*) – whether there are any local changes to push
- **remote_commit** (*string*) – Remote commit information
- **status** (*string*) – VCS repository status as reported by VCS
- **merge_failure** – Text describing merge failure or null if there is none

POST /api/components/(string: *project*)/(string: *component*)/repository/ Performs the given operation on a VCS repository.

See *POST /api/projects/(string:project)/repository/* for documentation.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Request JSON Object

- **operation** (*string*) – Operation to perform: one of `push`, `pull`, `commit`, `reset`, `cleanup`

Response JSON Object

- **result** (*boolean*) – result of the operation

CURL example:

```
curl \
    -d operation=pull \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

JSON request example:

```
POST /api/components/hello/weblate/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"result":true}
```

GET /api/components/(string: *project*)/(string: *component*)/monolingual_base/ Downloads base file for monolingual translations.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

GET /api/components/(string: *project*)/(string: *component*)/new_template/ Downloads template file for new translations.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

GET /api/components/(string: *project*)/(string: *component*)/translations/ Returns a list of translation objects in the given component.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **results** (*array*) – array of translation objects; see *GET /api/translations/(string:project)/(string:component)/(string:language)/*

POST /api/components/(string: *project*)/
string: *component*/translations/ Creates new translation in the given component.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Request JSON Object

- **language_code** (*string*) – translation language code; see *GET /api/languages/(string:language)/*

Response JSON Object

- **result** (*object*) – new translation object created

CURL example:

```
curl \
    -d language_code=cs \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/components/
```

JSON request example:

```
POST /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"language_code": "cs"}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "failing_checks": 0,
    "failing_checks_percent": 0,
    "failing_checks_words": 0,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "is_source": false,
    "language": {
        "code": "cs",
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
```

(continues on next page)

(continuação da página anterior)


```
    },
    "language_code": "cs",
    "id": 125,
    "last_author": null,
    "last_change": null,
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
    "translate_url": "http://example.com/translate/hello/weblate/cs/",
    "translated": 0,
    "translated_percent": 0.0,
    "translated_words": 0,
    "url": "http://example.com/api/translations/hello/weblate/cs/",
    "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```


GET /api/components/(string: *project*)/
string: *component*/statistics/ Returns paginated statistics for all translations within component.

Novo na versão 2.7.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente

Response JSON Object

- **results** (*array*) – array of translation statistics objects; see *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/*

1.12.9 Traduções

GET /api/translations/
Returns a list of translations.

Veja também:

Translation object attributes are documented at *GET /api/translations/(string:project)/(string:component)/(string:language)/*.

GET /api/translations/(string: *project*)/
string: *component*/string: *language*/ Returns information about a translation.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Response JSON Object

- **component** (*object*) – component object; see *GET /api/components/(string:project)/(string:component)/*
- **failing_checks** (*int*) – quantidade de cadeias com verificações falhadas
- **failing_checks_percent** (*float*) – Cadeias traduzidas com quaisquer verificações falhadas
- **failing_checks_words** (*int*) – quantidade de verificações falhadas
- **filename** (*string*) – translation filename

- **fuzzy** (*int*) – number of strings marked for review
- **fuzzy_percent** (*float*) – percentage of strings marked for review
- **fuzzy_words** (*int*) – number of words marked for review
- **have_comment** (*int*) – number of strings with comment
- **have_suggestion** (*int*) – number of strings with suggestion
- **is_template** (*boolean*) – se a tradução tem uma base monolingue
- **language** (*object*) – source language object; see *GET /api/languages/(string:language)/*
- **language_code** (*string*) – language code used in the repository; this can be different from language code in the language object
- **last_author** (*string*) – nome do úlitmo autor
- **last_change** (*timestamp*) – last change timestamp
- **revision** (*string*) – revision hash for the file
- **share_url** (*string*) – URL for sharing leading to engagement page
- **total** (*int*) – total number of strings
- **total_words** (*int*) – total number of words
- **translate_url** (*string*) – URL for translating
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **translated_words** (*int*) – number of translated words
- **repository_url** (*string*) – URL to repository status; see *GET /api/translations/(string:project)/(string:component)/(string:language)/repository/*
- **file_url** (*string*) – URL to file object; see *GET /api/translations/(string:project)/(string:component)/(string:language)/file/*
- **changes_list_url** (*string*) – URL to changes list; see *GET /api/translations/(string:project)/(string:component)/(string:language)/changes/*
- **units_list_url** (*string*) – URL to strings list; see *GET /api/translations/(string:project)/(string:component)/(string:language)/units/*

Example JSON data:

```
{
    "component": {
        "branch": "master",
        "file_format": "po",
        "filemask": "po/*.po",
        "git_export": "",
        "license": "",
        "license_url": "",
        "name": "Weblate",
        "new_base": "",
        "project": {
            "name": "Hello",
            "slug": "hello",
            "source_language": {
```

(continues on next page)

(continuação da página anterior)


```
                "code": "en",
                "direction": "ltr",
                "name": "English",
                "url": "http://example.com/api/languages/en/",
                "web_url": "http://example.com/languages/en/"
            },
            "url": "http://example.com/api/projects/hello/",
            "web": "https://weblate.org/",
            "web_url": "http://example.com/projects/hello/"
        },
        "repo": "file:///home/nijel/work/weblate-hello",
        "slug": "weblate",
        "template": "",
        "url": "http://example.com/api/components/hello/weblate/",
        "vcs": "git",
        "web_url": "http://example.com/projects/hello/weblate/"
    },
    "failing_checks": 3,
    "failing_checks_percent": 75.0,
    "failing_checks_words": 11,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "language": {
        "code": "cs",
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
    },
    "language_code": "cs",
    "last_author": "Weblate Admin",
    "last_change": "2016-03-07T10:20:05.499",
    "revision": "7ddfafe6daaf57fc8654cc852ea6be212b015792",
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
    "translate_url": "http://example.com/translate/hello/weblate/cs/",
    "translated": 4,
    "translated_percent": 100.0,
    "translated_words": 15,
    "url": "http://example.com/api/translations/hello/weblate/cs/",
    "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```


DELETE /api/translations/(**string:** *project*)**/**
string: *component***/string:** *language***/** Novo na versão 3.9.

Deletes a translation.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

GET /api/translations/(**string:** *project*)**/**
string: *component***/string:** *language***/changes/** Returns a list of translation changes. This is

essentially a translations-scoped *GET /api/changes/* accepting the same parameters.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/changes/(int:id)/*

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/units/ Returns a list of translation units.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code
- **q** (*string*) – Search query string *Searching* (optional)

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/units/(int:id)/*

POST /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/units/ Add new monolingual unit.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Request JSON Object

- **key** (*string*) – Name of translation unit
- **value** (*string*) – The translation unit value

POST /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/autotranslate/ Trigger automatic translation.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Request JSON Object

- **mode** (*string*) – Modo de tradução automático
- **filter_type** (*string*) – Automatic translation filter type
- **auto_source** (*string*) – Fonte da tradução automática
- **component** (*string*) – Ativar a contribuição para a memória de tradução compartilhada para que o projeto tenha acesso a componentes adicionais.
- **engines** (*string*) – Motores de tradução automática
- **threshold** (*string*) – Limite de pontuação

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/file/ Download current translation file as stored in VCS (without `format` parameter) or as converted to a standard format (currently supported: Gettext PO, MO, XLIFF and TBX).

Nota: This API endpoint uses different logic for output than rest of API as it operates on whole file rather than on data. Set of accepted `format` parameter differs and without such parameter you get translation file as stored in VCS.

Query Parameters

- **format** – Formato de ficheiro a usar; se não for especificado nenhuma conversão de formato acontecerá; formatos de ficheiro suportados: `po`, `mo`, `xliff`, `xliff11`, `tbx`, `csv`, `xlsx`, `json`, `aresource`, `strings`

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

POST /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/file/ Upload new file with translations.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Form Parameters

- **string conflicts** – How to deal with conflicts (`ignore`, `replace-translated` or `replace-approved`)
- **file file** – Uploaded file
- **string email** – E-mail do autor
- **string author** – Nome do autor
- **string method** – Upload method (`translate`, `approve`, `suggest`, `fuzzy`, `replace`, `source`), see *Métodos de importação*
- **string fuzzy** – Fuzzy strings processing (*empty*, `process`, `approve`)

CURL example:

```
curl -X POST \
    -F file=@strings.xml \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/translations/hello/android/cs/file/
```

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/repository/ Returns information about VCS repository status.

The response is same as for *GET /api/components/(string:project)/(string:component)/repository/*.

Parameters

- **project** (*string*) – URL semântico do projeto

- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

POST /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/repository/ Performs given operation on the VCS repository.

See *POST /api/projects/(string:project)/repository/* for documentation.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Request JSON Object

- **operation** (*string*) – Operation to perform: one of `push`, `pull`, `commit`, `reset`, `cleanup`

Response JSON Object

- **result** (*boolean*) – result of the operation

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/statistics/ Returns detailed translation statistics.

Novo na versão 2.7.

Parameters

- **project** (*string*) – URL semântico do projeto
- **component** (*string*) – URL semântico do componente
- **language** (*string*) – Translation language code

Response JSON Object

- **code** (*string*) – language code
- **failing** (*int*) – number of failing checks
- **failing_percent** (*float*) – percentage of failing checks
- **fuzzy** (*int*) – number of strings needing review
- **fuzzy_percent** (*float*) – percentage of strings needing review
- **total_words** (*int*) – total number of words
- **translated_words** (*int*) – number of translated words
- **last_author** (*string*) – nome do úlitmo autor
- **last_change** (*timestamp*) – date of last change
- **name** (*string*) – language name
- **total** (*int*) – total number of strings
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **url** (*string*) – URL to access the translation (engagement URL)
- **url_translate** (*string*) – URL to access the translation (real translation URL)

1.12.10 Units

Novo na versão 2.10.

GET /api/units/
Returns list of translation units.

Veja também:

Unit object attributes are documented at *GET /api/units/(int:id)/*.

GET /api/units/(int: *id*)/
Alterado na versão 4.3: The `target` and `source` are now arrays to properly handle plural strings.

Returns information about translation unit.

Parameters

- **id** (*int*) – Unit ID

Response JSON Object

- **translation** (*string*) – URL of a related translation object
- **source** (*array*) – source string
- **previous_source** (*string*) – previous source string used for fuzzy matching
- **target** (*array*) – target string
- **id_hash** (*string*) – unique identifier of the unit
- **content_hash** (*string*) – unique identifier of the source string
- **location** (*string*) – location of the unit in source code
- **context** (*string*) – translation unit context
- **note** (*string*) – translation unit note
- **flags** (*string*) – translation unit flags
- **state** (*int*) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only
- **fuzzy** (*boolean*) – se a unidade está confusa ou marcada para revisão
- **translated** (*boolean*) – Percentagem traduzido
- **approved** (*boolean*) – Tradução aprovada
- **position** (*int*) – unit position in translation file
- **has_suggestion** (*boolean*) – Cadeia tem sugestão
- **has_comment** (*boolean*) – Cadeia tem comentário
- **has_failing_check** (*boolean*) – Cadeia tem verificação falhada
- **num_words** (*int*) – number of source words
- **priority** (*int*) – translation priority; 100 is default
- **id** (*int*) – unit identifier
- **explanation** (*string*) – String explanation, available on source units, see *Additional info on source strings*
- **extra_flags** (*string*) – Sinalizadores de cadeias adicionais, disponíveis nas unidades de fonte, veja *Personalizar o comportamento*
- **web_url** (*string*) – URL where the unit can be edited
- **souce_unit** (*string*) – Source unit link; see *GET /api/units/(int:id)/*

PATCH /api/units/(**int:** *id*)**/**
Novo na versão 4.3.

Realiza uma atualização parcial na unidade de tradução.

Parameters

- **id** (*int*) – Unit ID

Request JSON Object

- **state** (*int*) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only
- **target** (*array*) – target string
- **explanation** (*string*) – String explanation, available on source units, see *Additional info on source strings*
- **extra_flags** (*string*) – Sinalizadores de cadeias adicionais, disponíveis nas unidades de fonte, veja *Personalizar o comportamento*

PUT /api/units/(**int:** *id*)**/**
Novo na versão 4.3.

Realiza a atualização completa da unidade de tradução.

Parameters

- **id** (*int*) – Unit ID

Request JSON Object

- **state** (*int*) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only
- **target** (*array*) – target string
- **explanation** (*string*) – String explanation, available on source units, see *Additional info on source strings*
- **extra_flags** (*string*) – Sinalizadores de cadeias adicionais, disponíveis nas unidades de fonte, veja *Personalizar o comportamento*

DELETE /api/units/(**int:** *id*)**/**
Novo na versão 4.3.

Apaga a unidade de tradução.

Parameters

- **id** (*int*) – Unit ID

1.12.11 Alterações

Novo na versão 2.10.

GET /api/changes/
Alterado na versão 4.1: Filtering of changes was introduced in the 4.1 release.

Returns a list of translation changes.

Veja também:

Change object attributes are documented at *GET /api/changes/(int:id)/*.

Query Parameters

- **user** (*string*) – Username of user to filters
- **action** (*int*) – Action to filter, can be used several times

- **timestamp_after** (*timestamp*) – ISO 8601 formatted timestamp to list changes after
- **timestamp_before** (*timestamp*) – ISO 8601 formatted timestamp to list changes before

GET /api/changes/(int: *id*)/
Returns information about translation change.

Parameters

- **id** (*int*) – Change ID

Response JSON Object

- **unit** (*string*) – URL of a related unit object
- **translation** (*string*) – URL of a related translation object
- **component** (*string*) – URL of a related component object
- **glossary_term** (*string*) – URL of a related glossary term object
- **user** (*string*) – URL of a related user object
- **author** (*string*) – URL of a related author object
- **timestamp** (*timestamp*) – event timestamp
- **action** (*int*) – numeric identification of action
- **action_name** (*string*) – text description of action
- **target** (*string*) – event changed text or detail
- **id** (*int*) – change identifier

1.12.12 Capturas de ecrã

Novo na versão 2.14.

GET /api/screenshots/
Returns a list of screenshot string information.

Veja também:

Screenshot object attributes are documented at *GET /api/screenshots/(int:id)/*.

GET /api/screenshots/(int: *id*)/
Returns information about screenshot information.

Parameters

- **id** (*int*) – Screenshot ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

GET /api/screenshots/(int: *id*)/file/
Download the screenshot image.

Parameters

- **id** (*int*) – Screenshot ID

POST /api/screenshots/(int: *id*)/file/
Replace screenshot image.

Parameters

- **id** (*int*) – Screenshot ID

Form Parameters

- **file image** – Uploaded file

CURL example:

```
curl -X POST \
    -F image=@image.png \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/screenshots/1/file/
```

POST /api/screenshots/(int: *id*)/units/
Associate source string with screenshot.

Parameters

- **id** (*int*) – Screenshot ID

Form Parameters

- **string unit_id** – Unit ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **translation** (*string*) – URL of a related translation object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

DELETE /api/screenshots/(int: *id*)/units/
int: *unit_id* Remove a associação de texto fonte com captura de ecrã.

Parameters

- **id** (*int*) – Screenshot ID
- **unit_id** – ID da unidade do texto fonte

POST /api/screenshots/
Creates a new screenshot.

Form Parameters

- **file image** – Uploaded file
- **string name** – Nome da captura do ecrã
- **string project_slug** – Project slug
- **string component_slug** – Component slug
- **string language_code** – Código do idioma

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object

- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

PATCH /api/screenshots/(int: *id*)/
Edita informações parciais sobre captura de ecrã.

Parameters

- **id** (*int*) – Screenshot ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

PUT /api/screenshots/(int: *id*)/
Edita informações completas sobre captura de ecrã.

Parameters

- **id** (*int*) – Screenshot ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

DELETE /api/screenshots/(int: *id*)/
Apagar captura de ecrã.

Parameters

- **id** (*int*) – Screenshot ID

1.12.13 Listas de componentes

Novo na versão 4.0.

GET /api/component-lists/
Returns a list of component lists.

Veja também:

Component list object attributes are documented at *GET /api/component-lists/(str:slug)/*.

GET /api/component-lists/(str: *slug*)/
Returns information about component list.

Parameters

- **slug** (*string*) – Component list slug

Response JSON Object

- **name** (*string*) – name of a component list
- **slug** (*string*) – slug of a component list
- **show_dashboard** (*boolean*) – whether to show it on a dashboard
- **components** (*array*) – link to associated components; see *GET /api/components/(string:project)/(string:component)/*
- **auto_assign** (*array*) – automatic assignment rules

PUT /api/component-lists/(str: *slug*)/
Changes the component list parameters.

Parameters

- **slug** (*string*) – Component list slug

Request JSON Object

- **name** (*string*) – name of a component list
- **slug** (*string*) – slug of a component list
- **show_dashboard** (*boolean*) – whether to show it on a dashboard

PATCH /api/component-lists/(str: *slug*)/
Changes the component list parameters.

Parameters

- **slug** (*string*) – Component list slug

Request JSON Object

- **name** (*string*) – name of a component list
- **slug** (*string*) – slug of a component list
- **show_dashboard** (*boolean*) – whether to show it on a dashboard

DELETE /api/component-lists/(str: *slug*)/
Deletes the component list.

Parameters

- **slug** (*string*) – Component list slug

POST /api/component-lists/(str: *slug*)/components/
Associate component with a component list.

Parameters

- **slug** (*string*) – Component list slug

Form Parameters

- **string component_id** – Component ID

DELETE /api/component-lists/(str: *slug*)/components/
str: *component_slug* Disassociate a component from the component list.

Parameters

- **slug** (*string*) – Component list slug
- **component_slug** (*string*) – Component slug

1.12.14 Glossário

GET /api/glossary/
Returns a list of all glossaries which are associated with a project that user has access to.

Veja também:

Language object attributes are documented at *GET /api/languages/(string:language)/*.

GET /api/glossary/(int: *id*)/
Retorna informações sobre um glossário.

Parameters

- **id** (*int*) – Id do glossário

Response JSON Object

- **name** (*string*) – Código do idioma
- **color** (*string*) – Direção do texto
- **source_language** (*object*) – Object of language plural information
- **projects** (*array*) – link to associated projects; see *GET /api/projects/(string:project)/*

Example JSON data:

```
{
    "name": "Hello",
    "id": 1,
    "color": "silver",
    "source_language": {
        "code": "en",
        "name": "English",
        "plural": {
            "id": 75,
            "source": 0,
            "number": 2,
            "formula": "n != 1",
            "type": 1
        },
        "aliases": [
            "english",
            "en_en",
            "base",
            "source",
            "eng"
        ],
        "direction": "ltr",
        "web_url": "http://example.com/languages/en/",
        "url": "http://example.com/api/languages/en/",
        "statistics_url": "http://example.com/api/languages/en/statistics/"
    },
    "project": {
        "name": "Hello",
        "slug": "hello",
        "id": 1,
        "source_language": {
            "code": "en",
            "name": "English",
            "plural": {
                "id": 75,
                "source": 0,
                "number": 2,
```

(continues on next page)

(continuação da página anterior)


```
                "formula": "n != 1",
                "type": 1
            },
            "aliases": [
                "english",
                "en_en",
                "base",
                "source",
                "eng"
            ],
            "direction": "ltr",
            "web_url": "http://example.com/languages/en/",
            "url": "http://example.com/api/languages/en/",
            "statistics_url": "http://example.com/api/languages/en/statistics/"
        },
        "web_url": "http://example.com/projects/demo1/",
        "url": "http://example.com/api/projects/demo1/",
        "components_list_url": "http://example.com/api/projects/demo1/
↪components/",
        "repository_url": "http://example.com/api/projects/demo1/repository/",
        "statistics_url": "http://example.com/api/projects/demo1/statistics/",
        "changes_list_url": "http://example.com/api/projects/demo1/changes/",
        "languages_url": "http://example.com/api/projects/demo1/languages/"
    },
    "projects_url": "http://example.com/api/glossary/7/projects/",
    "terms_url": "http://example.com/api/glossary/7/terms/",
    "url": "http://example.com/api/glossary/7/"
}
```


PUT /api/glossary/(int: *id*)/
Altera os parâmetros do glossário.

Parameters

- **id** (*int*) – Id do glossário

Request JSON Object

- **name** (*string*) – Nome do idioma
- **color** (*string*) – Language direction
- **source_language** (*object*) – Language plural details

PATCH /api/glossary/(int: *id*)/
Altera os parâmetros do glossário.

Parameters

- **id** (*int*) – Id do glossário

Request JSON Object

- **name** (*string*) – Nome do idioma
- **color** (*string*) – Language direction
- **source_language** (*object*) – Language plural details

DELETE /api/glossary/(int: *id*)/
Apaga o glossário.

Parameters

- **id** (*int*) – Id do glossário

GET /api/glossary/(int: *id*)/projects/
Returns projects linked with a glossary.

Parameters

- **id** (*int*) – Id do glossário

Response JSON Object

- **projects** (*array*) – associated projects; see *GET /api/projects/(string:project)/*

POST /api/glossary/(int: *id*)/projects/
Associate project with a glossary.

Parameters

- **id** (*int*) – Id do glossário

Form Parameters

- **string project_slug** – Project slug

DELETE /api/glossary/(int: *id*)/projects/
Apaga a associação de um projeto com um glossário.

Parameters

- **id** (*int*) – Id do glossário

Form Parameters

- **string project_slug** – Project slug

GET /api/glossary/(int: *id*)/terms/
Liste os termos de um glossário.

Parameters

- **id** (*int*) – Id do glossário

POST /api/glossary/(int: *id*)/terms/
Associar termos com um glossário.

Parameters

- **id** (*int*) – Id do glossário

Request JSON Object

- **language** (*object*) – Idioma do termo
- **source** (*string*) – Cadeia de fonte para o termo
- **target** (*string*) – Cadeia de destino para o termo

GET /api/glossary/(int: *id*)/terms/
int: *term_id*/ Obtenha um termo associado a um glossário.

Parameters

- **id** (*int*) – Id do glossário
- **term_id** (*int*) – ID of term

PUT /api/glossary/(int: *id*)/terms/
int: *term_id*/ Edite um termo associado a um glossário.

Parameters

- **id** (*int*) – Id do glossário
- **term_id** (*int*) – ID of term

Request JSON Object

- **language** (*object*) – Idioma do termo

- **source** (*string*) – Cadeia de fonte para o termo
- **target** (*string*) – Cadeia de destino para o termo

PATCH /api/glossary/(**int:** *id*)**/terms/**
int: *term_id***/** Edite um termo associado a um glossário.

Parameters

- **id** (*int*) – Id do glossário
- **term_id** (*int*) – ID of term

Request JSON Object

- **language** (*object*) – Idioma do termo
- **source** (*string*) – Cadeia de fonte para o termo
- **target** (*string*) – Cadeia de destino para o termo

DELETE /api/glossary/(**int:** *id*)**/terms/**
int: *term_id***/** Delete a term associated with a glossary.

Parameters

- **id** (*int*) – Id do glossário
- **term_id** (*int*) – ID of term

1.12.15 Hooks de notificação

Notification hooks allow external applications to notify Weblate that the VCS repository has been updated.

You can use repository endpoints for projects, components and translations to update individual repositories; see *POST /api/projects/(string:project)/repository/* for documentation.

GET /hooks/update/(**string:** *project*)**/**
string: *component***/** Obsoleto desde a versão 2.6: Please use *POST /api/components/(string:project)/(string:component)/repository/* instead which works properly with authentication for ACL limited projects.

Triggers update of a component (pulling from VCS and scanning for translation changes).

GET /hooks/update/(**string:** *project*)**/**
Obsoleto desde a versão 2.6: Please use *POST /api/projects/(string:project)/repository/* instead which works properly with authentication for ACL limited projects.

Triggers update of all components in a project (pulling from VCS and scanning for translation changes).

POST /hooks/github/
Special hook for handling GitHub notifications and automatically updating matching components.

Nota: GitHub includes direct support for notifying Weblate: enable Weblate service hook in repository settings and set the URL to the URL of your Weblate installation.

Veja também:

Receber alterações do GitHub automaticamente For instruction on setting up GitHub integration

https://docs.github.com/en/github/extending-github/about-webhooks Generic information about GitHub Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitlab/
Special hook for handling GitLab notifications and automatically updating matching components.

Veja também:

Receber alterações do GitLab automaticamente For instruction on setting up GitLab integration

https://docs.gitlab.com/ce/user/project/integrations/webhooks.html Generic information about GitLab Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/bitbucket/
Special hook for handling Bitbucket notifications and automatically updating matching components.

Veja também:

Receber alterações do Bitbucket automaticamente For instruction on setting up Bitbucket integration

https://confluence.atlassian.com/bitbucket/manage-webhooks-735643732.html Generic information about Bitbucket Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/pagure/
Novo na versão 3.3.

Special hook for handling Pagure notifications and automatically updating matching components.

Veja também:

Receber alterações do Pagure automaticamente For instruction on setting up Pagure integration

https://docs.pagure.org/pagure/usage/using_webhooks.html Generic information about Pagure Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/azure/
Novo na versão 3.8.

Special hook for handling Azure Repos notifications and automatically updating matching components.

Veja também:

Receber alterações dos Azure Repos automaticamente For instruction on setting up Azure integration

https://docs.microsoft.com/en-us/azure/devops/service-hooks/services/webhooks Generic information about Azure Repos Web Hooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitea/
Novo na versão 3.9.

Special hook for handling Gitea Webhook notifications and automatically updating matching components.

Veja também:

Receber alterações dos Gitea Repos automaticamente For instruction on setting up Gitea integration

https://docs.gitea.io/en-us/webhooks/ Generic information about Gitea Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitee/

Novo na versão 3.9.

Special hook for handling Gitee Webhook notifications and automatically updating matching components.

Veja também:

Receber alterações de Gitee Repos automaticamente For instruction on setting up Gitee integration

https://gitee.com/help/categories/40 Generic information about Gitee Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

1.12.16 Exports

Weblate provides various exports to allow you to further process the data.

GET /exports/stats/(**string:** *project*)**/**
string: *component***/**

Query Parameters

- **format** (*string*) – Output format: either `json` or `csv`

Obsoleto desde a versão 2.6: Please use *GET /api/components/(string:project)/(string:component)/statistics/* and *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/* instead; it allows access to ACL controlled projects as well.

Retrieves statistics for given component in given format.

Example request:

```
GET /exports/stats/weblate/master/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

[
    {
        "code": "cs",
        "failing": 0,
        "failing_percent": 0.0,
        "fuzzy": 0,
        "fuzzy_percent": 0.0,
        "last_author": "Michal Čihař",
        "last_change": "2012-03-28T15:07:38+00:00",
        "name": "Czech",
        "total": 436,
        "total_words": 15271,
        "translated": 436,
        "translated_percent": 100.0,
        "translated_words": 3201,
        "url": "http://hosted.weblate.org/engage/weblate/cs/",
        "url_translate": "http://hosted.weblate.org/projects/weblate/master/cs/
↪"
    },
    {
        "code": "nl",
```

(continues on next page)

(continuação da página anterior)


```
        "failing": 21,
        "failing_percent": 4.8,
        "fuzzy": 11,
        "fuzzy_percent": 2.5,
        "last_author": null,
        "last_change": null,
        "name": "Dutch",
        "total": 436,
        "total_words": 15271,
        "translated": 319,
        "translated_percent": 73.2,
        "translated_words": 3201,
        "url": "http://hosted.weblate.org/engage/weblate/nl/",
        "url_translate": "http://hosted.weblate.org/projects/weblate/master/nl/
↪"
    },
    {
        "code": "el",
        "failing": 11,
        "failing_percent": 2.5,
        "fuzzy": 21,
        "fuzzy_percent": 4.8,
        "last_author": null,
        "last_change": null,
        "name": "Greek",
        "total": 436,
        "total_words": 15271,
        "translated": 312,
        "translated_percent": 71.6,
        "translated_words": 3201,
        "url": "http://hosted.weblate.org/engage/weblate/el/",
        "url_translate": "http://hosted.weblate.org/projects/weblate/master/el/
↪"
    }
]
```


1.12.17 Feeds RSS

Changes in translations are exported in RSS feeds.

GET /exports/rss/(**string:** *project*)**/**
string: *component***/string:** *language***/** Retrieves RSS feed with recent changes for a translation.

GET /exports/rss/(**string:** *project*)**/**
string: *component***/** Retrieves RSS feed with recent changes for a component.

GET /exports/rss/(**string:** *project*)**/**
Retrieves RSS feed with recent changes for a project.

GET /exports/rss/language/(**string:** *language*)**/**
Retrieves RSS feed with recent changes for a language.

GET /exports/rss/
Retrieves RSS feed with recent changes for Weblate instance.

Veja também:

RSS on wikipedia

1.13 Cliente Weblate

Novo na versão 2.7: Há suporte total do utilitário wlc desde o Weblate 2.7. Se estiver a usar uma versão mais antiga, algumas incompatibilidades com a API podem ocorrer.

1.13.1 Instalação

O cliente Weblate é enviado separadamente e inclui o módulo Python. Para usar os comandos abaixo, precisa instalar *wlc*:

```
pip3 install wlc
```

1.13.2 Uso do Docker

The Weblate Client is also available as a Docker image.

The image is published on Docker Hub: https://hub.docker.com/r/weblate/wlc

Instalar:

```
docker pull weblate/wlc
```

The Docker container uses Weblate's default settings and connects to the API deployed in localhost. The API URL and API_KEY can be configured through the arguments accepted by Weblate.

The command to launch the container uses the following syntax:

```
docker run --rm weblate/wlc [WLC_ARGS]
```

Exemplo:

```
docker run --rm weblate/wlc --url https://hosted.weblate.org/api/ list-projects
```

1.13.3 Primeiros Passos

A configuração do wlc é armazenada em ``~/.config/weblate``(veja *Ficheiros de configuração* para outros locais), por favor, crie-a para corresponder ao seu ambiente:

```
[weblate]
url = https://hosted.weblate.org/api/

[keys]
https://hosted.weblate.org/api/ = APIKEY
```

Depois pode invocar comandos no servidor predefinido:

```
wlc ls
wlc commit sandbox/hello-world
```

Veja também:

Ficheiros de configuração

1.13.4 Sinopse

```
wlc [arguments] <command> [options]
```

Os comandos indicam, na verdade, qual operação deve ser realizada.

1.13.5 Descrição

O cliente Weblate é uma biblioteca Python e utilitário de linha de comando para gerir o Weblate remotamente a usar a API. O utilitário de linha de comando pode ser invocado como **wlc** e está embutido em *wlc*.

Argumentos

O programa aceita os seguintes argumentos que definem o formato de saída ou qual a instância do Weblate a utilizar. Estes devem ser introduzidos antes de qualquer comando.

--format {csv,json,text,html}
: Especifica o formato de saída.

--url URL
: Especifica a URL da API. Substitui qualquer valor encontrado no ficheiro de configuração, consulte *Ficheiros de configuração*. A URL deve terminar com `/api/`, por exemplo, `https://hosted.weblate.org/api/`.

--key KEY
: Especifica a chave do utilizador de API a ser usada. Substitui qualquer valor encontrado no ficheiro de configuração, consulte *Ficheiros de configuração*. Pode encontrar a sua chave no seu perfil no Weblate.

--config PATH
: Substitui o caminho do ficheiro de configuração, consulte *Ficheiros de configuração*.

--config-section SECTION
: Substitui a secção de ficheiros de configuração em uso, consulte *Ficheiros de configuração*.

Comandos

Os comandos seguintes estão disponíveis:

version
: Imprime a versão atual.

list-languages
: Lista os idiomas usados no Weblate.

list-projects
: Lista os projetos no Weblate.

list-components
: Lista os componentes no Weblate.

list-translations
: Lista as traduções no Weblate.

show
: Mostra o objeto do Weblate (tradução, componente ou projeto).

ls
: Lista o objeto do Weblate (tradução, componente ou projeto).

commit
: Faz um commit das alterações feitas num objeto Weblate (tradução, componente ou projeto).

pull
Faz um pull das alterações remotas do repositório no objeto Weblate (tradução, componente ou projeto).

push
Faz um push das alterações do objeto Weblate ao repositório remoto (tradução, componente ou projeto).

reset
Novo na versão 0.7: Suportado desde o wlc 0.7.

Redefine as alterações no objeto Weblate para corresponder ao repositório remoto (tradução, componente ou projeto).

cleanup
Novo na versão 0.9: Suportado desde o wlc 0.9.

Remove todas as alterações não rastreadas num objeto Weblate para corresponder ao repositório remoto (tradução, componente ou projeto).

repo
Exibe o status do repositório para um determinado objeto do Weblate (tradução, componente ou projeto).

statistics
Exibe estatísticas detalhadas para um determinado objeto Weblate (tradução, componente ou projeto).

lock-status
Novo na versão 0.5: Suportado desde o wlc 0.5.

Exibe o status do bloqueio.

lock
Novo na versão 0.5: Suportado desde o wlc 0.5.

Bloqueia o componente de tradução posterior no Weblate.

unlock
Novo na versão 0.5: Suportado desde o wlc 0.5.

Desbloqueia a tradução do componente Weblate.

changes
Novo na versão 0.7: Suportado desde o wlc 0.7 e o Weblate 2.10.

Exibe alterações para um determinado objeto.

download
Novo na versão 0.7: Suportado desde o wlc 0.7.

Descarrega um ficheiro de tradução.

--convert
Converte o formato do ficheiro, se nenhuma conversão não especificada for feita no servidor e o ficheiro for descarregado como está no repositório.

--output
Especifica o ficheiro para gravar a saída e se não for especificado é impresso na stdout (saída predefinida).

upload
Novo na versão 0.9: Suportado desde o wlc 0.9.

Descarrega um ficheiro de tradução.

--overwrite
Substitua as traduções existentes ao enviar.

--input
Ficheiro a partir do qual o conteúdo é lido, se não for especificado é lido de stdin (entrada predefinida).

Dica: You can get more detailed information on invoking individual commands by passing `--help`, for example: `wlc ls --help`.

1.13.6 Ficheiros de configuração

`.weblate`, `.weblate.ini`, `weblate.ini` Alterado na versão 1.6: The files with *.ini* extension are accepted as well.

Por ficheiro de configuração de projeto

`C:\Users\NAME\AppData\weblate.ini` Novo na versão 1.6.

Ficheiro de configuração do utilizador no Windows.

`~/.config/weblate` Ficheiro de configuração do utilizador

`/etc/xdg/weblate` Ficheiro de configuração para todo o sistema

O programa segue a especificação XDG, para que possa ajustar a colocação de ficheiros de configuração por variáveis de ambiente `XDG_CONFIG_HOME` ou `XDG_CONFIG_DIRS`. No diretório do Windows `APPDATA`` é o local preferido para o ficheiro de configuração.

As configurações seguintes podem ser configuradas na secção `[weblate]` (pode personalizar-lo por *`--config-section`*):

`key`
Chave de API para acessar o Weblate.

`url`
URL de API do servidor, a predefinição sendo `http://127.0.0.1:8000/api/`.

`translation`
Caminho à tradução predefinida - componente ou projeto.

O ficheiro de configuração é um ficheiro INI, por exemplo:

```
[weblate]
url = https://hosted.weblate.org/api/
key = APIKEY
translation = weblate/master
```

Além disso, as chaves de API podem ser armazenadas na secção `[keys]`:

```
[keys]
https://hosted.weblate.org/api/ = APIKEY
```

Isso permite que armazene chaves nas suas configurações pessoais, enquanto usa a configuração do `.weblate` no repositório VCS para que o wlc saiba com qual servidor ele deve comunicar.

1.13.7 Exemplos

Imprimir a versão atual do programa:

```
$ wlc version
version: 0.1
```

Listar todos os projetos:

```
$ wlc list-projects
name: Hello
slug: hello
url: http://example.com/api/projects/hello/
web: https://weblate.org/
web_url: http://example.com/projects/hello/
```

Também pode designar em qual projeto o wlc deve trabalhar:

```
$ cat .weblate
[weblate]
url = https://hosted.weblate.org/api/
translation = weblate/master

$ wlc show
branch: master
file_format: po
source_language: en
filemask: weblate/locale/*/LC_MESSAGES/django.po
git_export: https://hosted.weblate.org/git/weblate/master/
license: GPL-3.0+
license_url: https://spdx.org/licenses/GPL-3.0+
name: master
new_base: weblate/locale/django.pot
project: weblate
repo: git://github.com/WeblateOrg/weblate.git
slug: master
template:
url: https://hosted.weblate.org/api/components/weblate/master/
vcs: git
web_url: https://hosted.weblate.org/projects/weblate/master/
```

Com esta configuração é fácil fazer um commit de alterações pendentes no projeto atual:

```
$ wlc commit
```

1.14 Weblate's Python API

1.14.1 Instalação

The Python API is shipped separately, you need to install the *Cliente Weblate*: (wlc) to have it.

```
pip install wlc
```

1.14.2 `wlc`

`WeblateException`

exception wlc.**WeblateException**
Base class for all exceptions.

Weblate

class wlc.**Weblate**(*key=''*, *url=None*, *config=None*)

Parâmetros

- **key** (*str*) – User key
- **url** (*str*) – API server URL, if not specified default is used
- **config** (wlc.config.WeblateConfig) – Configuration object, overrides any other parameters.

Access class to the API, define API key and optionally API URL.

get(*path*)

Parâmetros **path** (*str*) – Request path

Tipo de retorno object

Performs a single API GET call.

post(*path*, ***kwargs*)

Parâmetros **path** (*str*) – Request path

Tipo de retorno object

Performs a single API GET call.

1.14.3 wlc.config

WeblateConfig

class wlc.config.**WeblateConfig**(*section='wlc'*)

Parâmetros **section** (*str*) – Configuration section to use

Configuration file parser following XDG specification.

load(*path=None*)

Parâmetros **path** (*str*) – Path from which to load configuration.

Loads configuration from a file, if none is specified, it loads from the *wlc* configuration file (`~/.config/wlc`) placed in your XDG configuration path (`/etc/xdg/wlc`).

1.14.4 wlc.main

wlc.main.**main**(*settings=None*, *stdout=None*, *args=None*)

Parâmetros

- **settings** (*list*) – Settings to override as list of tuples
- **stdout** (*object*) – stdout file object for printing output, uses `sys.stdout` as default
- **args** (*list*) – Command-line arguments to process, uses `sys.args` as default

Main entry point for command-line interface.

@wlc.main.**register_command**(*command*)
Decorator to register *Command* class in main parser used by *main()*.

Command

class wlc.main.**Command**(*args*, *config*, *stdout=None*)
Main class for invoking commands.

CAPÍTULO 2

Documentação de administrador

2.1 Instruções de configuração

2.1.1 Instalar o Weblate

Installing using Docker

With dockerized Weblate deployment you can get your personal Weblate instance up and running in seconds. All of Weblate's dependencies are already included. PostgreSQL is set up as the default database.

Hardware requirements

Weblate should run on all contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

Nota: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Instalação

The following examples assume you have a working Docker environment, with `docker-compose` installed. Please check the Docker documentation for instructions.

1. Clone the weblate-docker repo:

```
git clone https://github.com/WeblateOrg/docker-compose.git weblate-docker
cd weblate-docker
```

2. Create a `docker-compose.override.yml` file with your settings. See *Docker environment variables* for full list of environment variables.

```
version: '3'
services:
  weblate:
    ports:
      - 80:8080
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SERVER_EMAIL: weblate@example.com
      WEBLATE_DEFAULT_FROM_EMAIL: weblate@example.com
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for the admin user
      WEBLATE_ADMIN_EMAIL: weblate.admin@example.com
```

Nota: If *WEBLATE_ADMIN_PASSWORD* is not set, the admin user is created with a random password shown on first startup.

The provided example makes Weblate listen on port 80, edit the port mapping in the `docker-compose.override.yml` file to change it.

---

3. Start Weblate containers:

```
docker-compose up
```

Enjoy your Weblate deployment, it's accessible on port 80 of the `weblate` container.

Alterado na versão 2.15-2: The setup has changed recently, priorly there was separate web server container, since 2.15-2 the web server is embedded in the Weblate container.

Alterado na versão 3.7.1-6: In July 2019 (starting with the 3.7.1-6 tag), the containers are not running as a root user. This has changed the exposed port from 80 to 8080.

Veja também:

Invoking management commands

Docker container with HTTPS support

Please see *Instalação* for generic deployment instructions, this section only mentions differences compared to it.

Using own SSL certificates

Novo na versão 3.8-3.

In case you have own SSL certificate you want to use, simply place the files into the Weblate data volume (see *Docker container volumes*):

- `ssl/fullchain.pem` containing the certificate including any needed CA certificates
- `ssl/privkey.pem` containing the private key

Both of these files must be owned by the same user as the one starting the docker container and have file mask set to `600` (readable and writable only by the owning user).

Additionally, Weblate container will now accept SSL connections on port 4443, you will want to include the port forwarding for HTTPS in docker compose override:

```
version: '3'
services:
  weblate:
    ports:
      - 80:8080
      - 443:4443
```

If you already host other sites on the same server, it is likely ports `80` and `443` are used by a reverse proxy, such as NGINX. To pass the HTTPS connection from NGINX to the docker container, you can use the following configuration:

```
server {
    listen 443;
    listen [::]:443;

    server_name <SITE_URL>;
    ssl_certificate /etc/letsencrypt/live/<SITE>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<SITE>/privkey.pem;

    location / {
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto https;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Host $server_name;
            proxy_pass https://127.0.0.1:<EXPOSED_DOCKER_PORT>;
    }
}
```

Replace `<SITE_URL>`, `<SITE>` and `<EXPOSED_DOCKER_PORT>` with actual values from your environment.

Automatic SSL certificates using Let's Encrypt

In case you want to use Let's Encrypt automatically generated SSL certificates on public installation, you need to add a reverse HTTPS proxy an additional Docker container, https-portal will be used for that. This is made use of in the `docker-compose-https.yml` file. Then create a `docker-compose-https.override.yml` file with your settings:

```
version: '3'
services:
  weblate:
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for admin user
  https-portal:
    environment:
      DOMAINS: 'weblate.example.com -> http://weblate:8080'
```

Whenever invoking **`docker-compose`** you need to pass both files to it, and then do:

```
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml␣
↪build
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml up
```

Upgrading the Docker container

Usually it is good idea to only update the Weblate container and keep the PostgreSQL container at the version you have, as upgrading PostgreSQL is quite painful and in most cases does not bring many benefits.

You can do this by sticking with the existing docker-compose and just pull the latest images and then restart:

```
docker-compose stop
docker-compose pull
docker-compose up
```

The Weblate database should be automatically migrated on first startup, and there should be no need for additional manual actions.

Nota: Upgrades across 3.0 are not supported by Weblate. If you are on 2.x series and want to upgrade to 3.x, first upgrade to the latest 3.0.1-x (at time of writing this it is the `3.0.1-7`) image, which will do the migration and then continue upgrading to newer versions.

You might also want to update the `docker-compose` repository, though it's not needed in most case. Please beware of PostgreSQL version changes in this case as it's not straightforward to upgrade the database, see GitHub issue for more info.

Admin login

After container setup, you can sign in as *admin* user with password provided in *WEBLATE_ADMIN_PASSWORD*, or a random password generated on first start if that was not set.

To reset *admin* password, restart the container with *WEBLATE_ADMIN_PASSWORD* set to new password.

Veja também:

WEBLATE_ADMIN_PASSWORD, *WEBLATE_ADMIN_NAME*, *WEBLATE_ADMIN_EMAIL*

Docker environment variables

Many of Weblate's *Configuração* can be set in the Docker container using environment variables:

Generic settings

WEBLATE_DEBUG
Configures Django debug mode using *DEBUG*.

Example:

```
environment:
  WEBLATE_DEBUG: 1
```

Veja também:

Desativar o modo de depuração.

WEBLATE_LOGLEVEL
Configures the logging verbosity.

WEBLATE_SITE_TITLE
Changes the site-title shown in the header of all pages.

WEBLATE_SITE_DOMAIN
Configura o domínio do site.

Dica: In case it is not set, the first item from *WEBLATE_ALLOWED_HOSTS* is used.

Veja também:

Definir domínio correto do site, *SITE_DOMAIN*

WEBLATE_ADMIN_NAME

WEBLATE_ADMIN_EMAIL
Configures the site-admin's name and e-mail. It is used for both *ADMINS* setting and creating *admin* user (see *WEBLATE_ADMIN_PASSWORD* for more info on that).

Example:

```
environment:
  WEBLATE_ADMIN_NAME: Weblate admin
  WEBLATE_ADMIN_EMAIL: noreply@example.com
```

Veja também:

Admin login, *Configurar administradores corretamente*, *ADMINS*

WEBLATE_ADMIN_PASSWORD
Sets the password for the *admin* user.

- If not set and *admin* user does not exist, it is created with a random password shown on first container startup.
- If not set and *admin* user exists, no action is performed.
- If set the *admin* user is adjusted on every container startup to match *WEBLATE_ADMIN_PASSWORD*, *WEBLATE_ADMIN_NAME* and *WEBLATE_ADMIN_EMAIL*.

Aviso: It might be a security risk to store password in the configuration file. Consider using this variable only for initial setup (or let Weblate generate random password on initial startup) or for password recovery.

Veja também:

Admin login, *WEBLATE_ADMIN_PASSWORD*, *WEBLATE_ADMIN_NAME*, *WEBLATE_ADMIN_EMAIL*

WEBLATE_SERVER_EMAIL

WEBLATE_DEFAULT_FROM_EMAIL

Configures the address for outgoing e-mails.

Veja também:

Configurar envio de e-mail

WEBLATE_ALLOWED_HOSTS

Configures allowed HTTP hostnames using *ALLOWED_HOSTS*.

Defaults to `*` which allows all hostnames.

Example:

```
environment:
  WEBLATE_ALLOWED_HOSTS: weblate.example.com,example.com
```

Veja também:

ALLOWED_HOSTS, *Configuração de hosts permitidos*, *Definir domínio correto do site*

WEBLATE_REGISTRATION_OPEN

Configures whether registrations are open by toggling *REGISTRATION_OPEN*.

Example:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
```

WEBLATE_REGISTRATION_ALLOW_BACKENDS

Configure which authentication methods can be used to create new account via *REGISTRATION_ALLOW_BACKENDS*.

Example:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
  WEBLATE_REGISTRATION_ALLOW_BACKENDS: azuread-oauth2,azuread-tenant-
↪oauth2
```

WEBLATE_TIME_ZONE

Configures the used time zone in Weblate, see `TIME_ZONE`.

Nota: To change the time zone of the Docker container itself, use the `TZ` environment variable.

Example:

```
environment:
  WEBLATE_TIME_ZONE: Europe/Prague
```

WEBLATE_ENABLE_HTTPS

Makes Weblate assume it is operated behind a reverse HTTPS proxy, it makes Weblate use HTTPS in e-mail and API links or set secure flags on cookies.

Nota: This does not make the Weblate container accept HTTPS connections, you need to configure that as well, see *Docker container with HTTPS support* for examples.

Example:

```
environment:
  WEBLATE_ENABLE_HTTPS: 1
```

Veja também:

Definir domínio correto do site

WEBLATE_IP_PROXY_HEADER

Lets Weblate fetch the IP address from any given HTTP header. Use this when using a reverse proxy in front of the Weblate container.

Enables *IP_BEHIND_REVERSE_PROXY* and sets *IP_PROXY_HEADER*.

Nota: The format must conform to Django's expectations. Django transforms raw HTTP header names as follows:

- converts all characters to uppercase
- replaces any hyphens with underscores
- prepends `HTTP_` prefix

So `X-Forwarded-For` would be mapped to `HTTP_X_FORWARDED_FOR`.

Example:

```
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```

WEBLATE_SECURE_PROXY_SSL_HEADER

A tuple representing a HTTP header/value combination that signifies a request is secure. This is needed when Weblate is running behind a reverse proxy doing SSL termination which does not pass standard HTTPS headers.

Example:

```
environment:
  WEBLATE_SECURE_PROXY_SSL_HEADER: HTTP_X_FORWARDED_PROTO,https
```

Veja também:

`SECURE_PROXY_SSL_HEADER`

WEBLATE_REQUIRE_LOGIN

Configures login required for the whole of the Weblate installation using *LOGIN_REQUIRED_URLS*.

Example:

```
environment:
  WEBLATE_REQUIRE_LOGIN: 1
```

WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS
Adds URL exceptions for login required for the whole Weblate installation using *LOGIN_REQUIRED_URLS_EXCEPTIONS*.

You can either replace whole settings, or modify default value using `ADD` and `REMOVE` variables.

WEBLATE_GOOGLE_ANALYTICS_ID
Configures ID for Google Analytics by changing *GOOGLE_ANALYTICS_ID*.

WEBLATE_GITHUB_USERNAME
Configures GitHub username for GitHub pull-requests by changing *GITHUB_USERNAME*.

Veja também:

GitHub

WEBLATE_GITHUB_TOKEN
Novo na versão 4.3.

Configures GitHub personal access token for GitHub pull-requests via API by changing *GITHUB_TOKEN*.

Veja também:

GitHub

WEBLATE_GITLAB_USERNAME
Configures GitLab username for GitLab merge-requests by changing *GITLAB_USERNAME*

Veja também:

GitLab

WEBLATE_GITLAB_TOKEN
Configures GitLab personal access token for GitLab merge-requests via API by changing *GITLAB_TOKEN*

Veja também:

GitLab

WEBLATE_PAGURE_USERNAME
Configures Pagure username for Pagure merge-requests by changing *PAGURE_USERNAME*

Veja também:

Pagure

WEBLATE_PAGURE_TOKEN
Configures Pagure personal access token for Pagure merge-requests via API by changing *PAGURE_TOKEN*

Veja também:

Pagure

WEBLATE_SIMPLIFY_LANGUAGES
Configures the language simplification policy, see *SIMPLIFY_LANGUAGES*.

WEBLATE_DEFAULT_ACCESS_CONTROL
Configures the default *Controlo de acesso* for new projects, see *DEFAULT_ACCESS_CONTROL*.

WEBLATE_DEFAULT_RESTRICTED_COMPONENT
Configures the default value for *Restricted access* for new components, see *DEFAULT_RESTRICTED_COMPONENT*.

WEBLATE_DEFAULT_TRANSLATION_PROPAGATION
Configures the default value for *Permitir propagação da tradução* for new components, see *DEFAULT_TRANSLATION_PROPAGATION*.

WEBLATE_DEFAULT_COMMITER_EMAIL
Configura *DEFAULT_COMMITER_EMAIL*.

WEBLATE_DEFAULT_COMMITER_NAME
Configura *DEFAULT_COMMITER_NAME*.

WEBLATE_AKISMET_API_KEY
Configures the Akismet API key, see *AKISMET_API_KEY*.

WEBLATE_GPG_IDENTITY
Configures GPG signing of commits, see *WEBLATE_GPG_IDENTITY*.

Veja também:

Signing Git commits with GnuPG

WEBLATE_URL_PREFIX
Configures URL prefix where Weblate is running, see *URL_PREFIX*.

WEBLATE_SILENCED_SYSTEM_CHECKS
Configures checks which you do not want to be displayed, see SILENCED_SYSTEM_CHECKS.

WEBLATE_CSP_SCRIPT_SRC

WEBLATE_CSP_IMG_SRC

WEBLATE_CSP_CONNECT_SRC

WEBLATE_CSP_STYLE_SRC

WEBLATE_CSP_FONT_SRC
Allows to customize `Content-Security-Policy` HTTP header.

Veja também:

Política de segurança de conteúdo, *CSP_SCRIPT_SRC*, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

WEBLATE_LICENSE_FILTER
Configura *LICENSE_FILTER*.

WEBLATE_HIDE_VERSION
Configura *HIDE_VERSION*.

Machine translation settings

WEBLATE_MT_APERTIUM_APY
Enables *Apertium* machine translation and sets *MT_APERTIUM_APY*

WEBLATE_MT_AWS_REGION

WEBLATE_MT_AWS_ACCESS_KEY_ID

WEBLATE_MT_AWS_SECRET_ACCESS_KEY
Configures *AWS* machine translation.

```
environment:
  WEBLATE_MT_AWS_REGION: us-east-1
  WEBLATE_MT_AWS_ACCESS_KEY_ID: AKIAIOSFODNN7EXAMPLE
  WEBLATE_MT_AWS_SECRET_ACCESS_KEY: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

WEBLATE_MT_DEEPL_KEY
Enables *DeepL* machine translation and sets *MT_DEEPL_KEY*

WEBLATE_MT_DEEPL_API_VERSION
Configures *DeepL* API version to use, see *MT_DEEPL_API_VERSION*.

WEBLATE_MT_GOOGLE_KEY
Enables *Google Translate* and sets *MT_GOOGLE_KEY*

WEBLATE_MT_MICROSOFT_COGNITIVE_KEY
Enables *Microsoft Cognitive Services Translator* and sets *MT_MICROSOFT_COGNITIVE_KEY*

WEBLATE_MT_MICROSOFT_ENDPOINT_URL
Sets *MT_MICROSOFT_ENDPOINT_URL*, please note this is supposed to contain domain name only.

WEBLATE_MT_MICROSOFT_REGION
Defina *MT_MICROSOFT_REGION*

WEBLATE_MT_MICROSOFT_BASE_URL
Defina *MT_MICROSOFT_BASE_URL*

WEBLATE_MT_MODERNMT_KEY
Enables *ModernMT* and sets *MT_MODERNMT_KEY*.

WEBLATE_MT_MYMEMORY_ENABLED

Enables *MyMemory* machine translation and sets *MT_MYMEMORY_EMAIL* to *WEBLATE_ADMIN_EMAIL*.

Example:

```
environment:
  WEBLATE_MT_MYMEMORY_ENABLED: 1
```

WEBLATE_MT_GLOSBE_ENABLED
Enables *Glosbe* machine translation.

```
environment:
  WEBLATE_MT_GLOSBE_ENABLED: 1
```

WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED
Enables *Microsoft Terminology Service* machine translation.

```
environment:
  WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED: 1
```

WEBLATE_MT_SAP_BASE_URL

WEBLATE_MT_SAP_SANDBOX_APIKEY

WEBLATE_MT_SAP_USERNAME

WEBLATE_MT_SAP_PASSWORD

WEBLATE_MT_SAP_USE_MT
Configures *SAP Translation Hub* machine translation.

```
environment:
    WEBLATE_MT_SAP_BASE_URL: "https://example.hana.ondemand.com/translationhub/
↪api/v1/"
    WEBLATE_MT_SAP_USERNAME: "user"
    WEBLATE_MT_SAP_PASSWORD: "password"
    WEBLATE_MT_SAP_USE_MT: 1
```

Authentication settings

LDAP

WEBLATE_AUTH_LDAP_SERVER_URI

WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE

WEBLATE_AUTH_LDAP_USER_ATTR_MAP

WEBLATE_AUTH_LDAP_BIND_DN

WEBLATE_AUTH_LDAP_BIND_PASSWORD

WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS

WEBLATE_AUTH_LDAP_USER_SEARCH

WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER

WEBLATE_AUTH_LDAP_USER_SEARCH_UNION

WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER

LDAP authentication configuration.

Example for direct bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE: uid=%(user)s,ou=People,dc=example,dc=net
  # map weblate 'full_name' to ldap 'name' and weblate 'email' attribute to
↪'mail' ldap attribute.
  # another example that can be used with OpenLDAP: 'full_name:cn,email:mail'
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
```

Example for search and bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
```

Example for union search and bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH_UNION: ou=users,dc=example,
↪dc=com|ou=otherusers,dc=example,dc=com
```

Example with search and bind against Active Directory:

```
environment:
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS: 0
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER: (sAMAccountName=%(user)s)
```

Veja também:

Autenticação por LDAP

GitHub

WEBLATE_SOCIAL_AUTH_GITHUB_KEY

WEBLATE_SOCIAL_AUTH_GITHUB_SECRET
Enables *Autenticação por GitHub*.

Bitbucket

WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY

WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET
Enables *Autenticação por Bitbucket*.

Facebook

WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY

WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET
Enables *OAuth 2 do Facebook*.

Google

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS
Enables *OAuth 2 do Google*.

GitLab

WEBLATE_SOCIAL_AUTH_GITLAB_KEY

WEBLATE_SOCIAL_AUTH_GITLAB_SECRET

WEBLATE_SOCIAL_AUTH_GITLAB_API_URL
Enables *OAuth 2 do GitLab*.

Azure Active Directory

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET
Enables Azure Active Directory authentication, see *Active Directory do Microsoft Azure*.

Azure Active Directory with Tenant support

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID
Enables Azure Active Directory authentication with Tenant support, see *Active Directory do Microsoft Azure*.

Keycloak

WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY

WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET

WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY

WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM

WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URL

WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL
Enables Keycloak authentication, see documentation.

Linux vendors

You can enable authentication using Linux vendors authentication services by setting following variables to any value.

WEBLATE_SOCIAL_AUTH_FEDORA

WEBLATE_SOCIAL_AUTH_OPENSUSE

WEBLATE_SOCIAL_AUTH_UBUNTU

Slack

WEBLATE_SOCIAL_AUTH_SLACK_KEY

SOCIAL_AUTH_SLACK_SECRET
Enables Slack authentication, see *Slack*.

SAML

Self-signed SAML keys are automatically generated on first container startup. In case you want to use own keys, place the certificate and private key in `/app/data/ssl/saml.crt` and `/app/data/ssl/saml.key`.

WEBLATE_SAML_IDP_ENTITY_ID

WEBLATE_SAML_IDP_URL

WEBLATE_SAML_IDP_X509CERT
SAML Identity Provider settings, see *Autenticação por SAML*.

Other authentication settings

WEBLATE_NO_EMAIL_AUTH
Disables e-mail authentication when set to any value.

PostgreSQL database setup

The database is created by `docker-compose.yml`, so these settings affect both Weblate and PostgreSQL containers.

Veja também:

Configuração de banco de dados para o Weblate

POSTGRES_PASSWORD
PostgreSQL password.

POSTGRES_USER
PostgreSQL username.

POSTGRES_DATABASE
PostgreSQL database name.

POSTGRES_HOST
PostgreSQL server hostname or IP address. Defaults to `database`.

POSTGRES_PORT
PostgreSQL server port. Defaults to none (uses the default value).

POSTGRES_SSL_MODE
Configure how PostgreSQL handles SSL in connection to the server, for possible choices see SSL Mode Descriptions

POSTGRES_ALTER_ROLE
Configures name of role to alter during migrations, see *Configurar Weblate para usar PostgreSQL*.

Configurações de backup de base de dados

Veja também:

Dados despejados para backups

WEBLATE_DATABASE_BACKUP
Configures the daily database dump using *DATABASE_BACKUP*. Defaults to `plain`.

Caching server setup

Using Redis is strongly recommended by Weblate and you have to provide a Redis instance when running Weblate in Docker.

Veja também:

Ativar o cache

REDIS_HOST
The Redis server hostname or IP address. Defaults to `cache`.

REDIS_PORT
The Redis server port. Defaults to `6379`.

REDIS_DB
The Redis database number, defaults to `1`.

REDIS_PASSWORD
The Redis server password, not used by default.

REDIS_TLS
Enables using SSL for Redis connection.

REDIS_VERIFY_SSL
Can be used to disable SSL certificate verification for Redis connection.

Email server setup

To make outgoing e-mail work, you need to provide a mail server.

Example TLS configuration:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
```

Example SSL configuration:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_PORT: 465
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
    WEBLATE_EMAIL_USE_TLS: 0
    WEBLATE_EMAIL_USE_SSL: 1
```

Veja também:

Configuração de e-mail de saída

WEBLATE_EMAIL_HOST
Mail server hostname or IP address.

Veja também:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, *WEBLATE_EMAIL_USE_TLS*, EMAIL_HOST

WEBLATE_EMAIL_PORT
Mail server port, defaults to 25.

Veja também:

EMAIL_PORT

WEBLATE_EMAIL_HOST_USER
Utilizador da autenticação por e-mail.

Veja também:

EMAIL_HOST_USER

WEBLATE_EMAIL_HOST_PASSWORD
Palavra-passe da autenticação por e-mail.

Veja também:

EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_USE_SSL
Whether to use an implicit TLS (secure) connection when talking to the SMTP server. In most e-mail documentation, this type of TLS connection is referred to as SSL. It is generally used on port 465. If you are experiencing problems, see the explicit TLS setting *WEBLATE_EMAIL_USE_TLS*.

Veja também:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_TLS*, EMAIL_USE_SSL

WEBLATE_EMAIL_USE_TLS
Whether to use a TLS (secure) connection when talking to the SMTP server. This is used for explicit TLS connections, generally on port 587 or 25. If you are experiencing connections that hang, see the implicit TLS setting *WEBLATE_EMAIL_USE_SSL*.

Veja também:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, EMAIL_USE_TLS

WEBLATE_EMAIL_BACKEND
Configures Django back-end to use for sending e-mails.

Veja também:

Configurar envio de e-mail, EMAIL_BACKEND

Error reporting

It is recommended to collect errors from the installation systematically, see *Collecting error reports*.

To enable support for Rollbar, set the following:

ROLLBAR_KEY
Your Rollbar post server access token.

ROLLBAR_ENVIRONMENT
Your Rollbar environment, defaults to `production`.

To enable support for Sentry, set following:

SENTRY_DSN
Your Sentry DSN.

SENTRY_ENVIRONMENT
Your Sentry Environment (optional).

CDN de localização

WEBLATE_LOCALIZE_CDN_URL

WEBLATE_LOCALIZE_CDN_PATH
Novo na versão 4.2.1.

Configuração para *CDN de localização JavaScript*.

The *WEBLATE_LOCALIZE_CDN_PATH* is path within the container. It should be stored on the persistent volume and not in the transient storage.

One of possibilities is storing that inside the Weblate data dir:

```
environment:
  WEBLATE_LOCALIZE_CDN_URL: https://cdn.example.com/
  WEBLATE_LOCALIZE_CDN_PATH: /app/data/l10n-cdn
```

Nota: You are responsible for setting up serving of the files generated by Weblate, it only does stores the files in configured location.

Veja também:

Traduzir HTML e JavaScript a usar CDN Weblate, *LOCALIZE_CDN_URL*, *LOCALIZE_CDN_PATH*

Changing enabled apps, checks, addons or autofixes

Novo na versão 3.8-5.

The built-in configuration of enabled checks, addons or autofixes can be adjusted by the following variables:

WEBLATE_ADD_APPS

WEBLATE_REMOVE_APPS

WEBLATE_ADD_CHECK

WEBLATE_REMOVE_CHECK

WEBLATE_ADD_AUTOFIX

WEBLATE_REMOVE_AUTOFIX

WEBLATE_ADD_ADDONS

WEBLATE_REMOVE_ADDONS

Example:

```
environment:
  WEBLATE_REMOVE_AUTOFIX: weblate.trans.autofixes.whitespace.
↪SameBookendingWhitespace
  WEBLATE_ADD_ADDONS: customize.addons.MyAddon,customize.addons.OtherAddon
```

Veja também:

CHECK_LIST, *AUTOFIX_LIST*, *WEBLATE_ADDONS*, INSTALLED_APPS

Configurações do contentor

CELERY_MAIN_OPTIONS

CELERY_NOTIFY_OPTIONS

CELERY_MEMORY_OPTIONS

CELERY_TRANSLATE_OPTIONS

CELERY_BACKUP_OPTIONS

CELERY_BEAT_OPTIONS

These variables allow you to adjust Celery worker options. It can be useful to adjust concurrency (`--concurrency 16`) or use different pool implementation (`--pool=gevent`).

By default, the number of concurrent workers matches the number of processors (except the backup worker, which is supposed to run only once).

Example:

```
environment:
  CELERY_MAIN_OPTIONS: --concurrency 16
```

Veja também:

Celery worker options, *Tarefas de fundo a usar o Celery*

UWSGI_WORKERS

Configure how many uWSGI workers should be executed.

It defaults to number of processors + 1.

Example:

```
environment:
  UWSGI_WORKERS: 32
```

In case you have lot of CPU cores and hit out of memory issues, try reducing number of workers:

```
environment:
  UWSGI_WORKERS: 4
  CELERY_MAIN_OPTIONS: --concurrency 2
  CELERY_NOTIFY_OPTIONS: --concurrency 1
  CELERY_TRANSLATE_OPTIONS: --concurrency 1
```

Docker container volumes

There is single data volume exported by the Weblate container. The other service containers (PostgreSQL or Redis) have their data volumes as well, but those are not covered by this document.

The data volume is used to store Weblate persistent data such as cloned repositories or to customize Weblate installation.

The placement of the Docker volume on host system depends on your Docker configuration, but usually it is stored in `/var/lib/docker/volumes/weblate-docker_weblate-data/_data/`. In the container it is mounted as `/app/data`.

Veja também:

Docker volumes documentation

Further configuration customization

You can further customize Weblate installation in the data volume, see *Docker container volumes*.

Custom configuration files

You can additionally override the configuration in `/app/data/settings-override.py` (see *Docker container volumes*). This is executed after all environment settings are loaded, so it gets completely set up, and can be used to customize anything.

Replacing logo and other static files

Novo na versão 3.8-5.

The static files coming with Weblate can be overridden by placing into `/app/data/python/customize/static` (see *Docker container volumes*). For example creating `/app/data/python/customize/static/favicon.ico` will replace the favicon.

Dica: The files are copied to the corresponding location upon container startup, so a restart of Weblate is needed after changing the content of the volume.

Alternatively you can also include own module (see *Personalizar o Weblate*) and add it as separate volume to the Docker container, for example:

```
weblate:
  volumes:
    - weblate-data:/app/data
    - ./weblate_customization/weblate_customization:/app/data/python/weblate_
↪customization
```

(continues on next page)

(continuação da página anterior)

```
  environment:
    WEBLATE_ADD_APPS: weblate_customization
```

Adding own Python modules

Novo na versão 3.8-5.

You can place own Python modules in `/app/data/python/` (see *Docker container volumes*) and they can be then loaded by Weblate, most likely by using *Custom configuration files*.

Veja também:

Personalizar o Weblate

Select your machine - local or cloud providers

With Docker Machine you can create your Weblate deployment either on your local machine, or on any large number of cloud-based deployments on e.g. Amazon AWS, Greenhost, and many other providers.

Installing on Debian and Ubuntu

Hardware requirements

Weblate should run on all contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

Nota: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Instalação

System requirements

Install the dependencies needed to build the Python modules (see *Requisitos de software*):

```
apt install \
   libxml2-dev libxslt-dev libfreetype6-dev libjpeg-dev libz-dev libyaml-dev \
   libcairo-dev gir1.2-pango-1.0 libgirepository1.0-dev libacl1-dev libssl-dev \
   build-essential python3-gdbm python3-dev python3-pip python3-virtualenv␣
↪virtualenv git
```

Install wanted optional dependencies depending on features you intend to use (see *Dependências opcionais*):

```
apt install tesseract-ocr libtesseract-dev libleptonica-dev
```

Optionally install software for running production server, see *Executar o servidor*, *Configuração de banco de dados para o Weblate*, *Tarefas de fundo a usar o Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
apt install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
apt install apache2 libapache2-mod-wsgi

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```

Python modules

Dica: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Dependências opcionais*):

   ```
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

Configuring Weblate

Nota: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see *Ajustar a configuração*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Configuração de banco de dados para o Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Executar o servidor* and *Servir ficheiros estáticos*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Comprimir os ativos do cliente*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Tarefas de fundo a usar o Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Executar o servidor* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Formatos de ficheiros suportados* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on SUSE and openSUSE

Hardware requirements

Weblate should run on all contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

Nota: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Instalação

System requirements

Install the dependencies needed to build the Python modules (see *Requisitos de software*):

```
zypper install \
   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-
↪devel \
   cairo-devel typelib-1_0-Pango-1_0 gobject-introspection-devel libacl-devel \
   python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Dependências opcionais*):

```
zypper install tesseract-ocr tesseract-devel leptonica-devel
```

Optionally install software for running production server, see *Executar o servidor*, *Configuração de banco de dados para o Weblate*, *Tarefas de fundo a usar o Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
zypper install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
zypper install apache2 apache2-mod_wsgi

# Caching backend: Redis
zypper install redis-server

# Database server: PostgreSQL
zypper install postgresql postgresql-contrib

# SMTP server
zypper install postfix
```

Python modules

Dica: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Dependências opcionais*):

   ```
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

Configuring Weblate

Nota: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see *Ajustar a configuração*.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Configuração de banco de dados para o Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Executar o servidor* and *Servir ficheiros estáticos*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Comprimir os ativos do cliente*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Tarefas de fundo a usar o Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Executar o servidor* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Formatos de ficheiros suportados* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on RedHat, Fedora and CentOS

Hardware requirements

Weblate should run on all contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

Nota: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Instalação

System requirements

Install the dependencies needed to build the Python modules (see *Requisitos de software*):

```
dnf install \
   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-
↪devel \
   cairo-devel pango-devel gobject-introspection-devel libacl-devel \
   python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Dependências opcionais*):

```
dnf install tesseract-langpack-eng tesseract-devel leptonica-devel
```

Optionally install software for running production server, see *Executar o servidor*, *Configuração de banco de dados para o Weblate*, *Tarefas de fundo a usar o Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
dnf install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
dnf install apache2 apache2-mod_wsgi

# Caching backend: Redis
dnf install redis

# Database server: PostgreSQL
dnf install postgresql postgresql-contrib

# SMTP server
dnf install postfix
```

Python modules

Dica: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

```
virtualenv --python=python3 ~/weblate-env
```

2. Activate the virtualenv for Weblate:

```
. ~/weblate-env/bin/activate
```

3. Install Weblate including all dependencies:

```
pip install Weblate
```

4. Install database driver:

```
pip install psycopg2-binary
```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Dependências opcionais*):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

Nota: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see *Ajustar a configuração*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Configuração de banco de dados para o Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Executar o servidor* and *Servir ficheiros estáticos*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Comprimir os ativos do cliente*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Tarefas de fundo a usar o Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Executar o servidor* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Formatos de ficheiros suportados* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on macOS

Hardware requirements

Weblate should run on all contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

Nota: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Instalação

System requirements

Install the dependencies needed to build the Python modules (see *Requisitos de software*):

```
brew install pango libjpeg python git libyaml gobject-introspection
pip3 install virtualenv
```

Make sure pip will be able to find the `libffi` version provided by homebrew — this will be needed during the installation build step.

```
export PKG_CONFIG_PATH="/usr/local/opt/libffi/lib/pkgconfig"
```

Install wanted optional dependencies depending on features you intend to use (see *Dependências opcionais*):

```
brew install tesseract
```

Optionally install software for running production server, see *Executar o servidor*, *Configuração de banco de dados para o Weblate*, *Tarefas de fundo a usar o Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
brew install nginx uwsgi

# Web server option 2: Apache with ``mod_wsgi``
brew install httpd

# Caching backend: Redis
brew install redis

# Database server: PostgreSQL
brew install postgresql
```

Python modules

Dica: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Dependências opcionais*):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

Nota: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see *Ajustar a configuração*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Configuração de banco de dados para o Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Executar o servidor* and *Servir ficheiros estáticos*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Comprimir os ativos do cliente*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Tarefas de fundo a usar o Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Executar o servidor* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Formatos de ficheiros suportados* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing from sources

1. Please follow the installation instructions for your system first:

   - *Installing on Debian and Ubuntu*
   - *Installing on SUSE and openSUSE*
   - *Installing on RedHat, Fedora and CentOS*

2. Grab the latest Weblate sources using Git (or download a tarball and unpack that):

   ```
   git clone https://github.com/WeblateOrg/weblate.git weblate-src
   ```

   Alternatively you can use released archives. You can download them from our website <https://weblate.org/>. Those downloads are cryptographically signed, please see *Verificar assinaturas de lançamento*.

3. Install current Weblate code into the virtualenv:

   ```
   . ~/weblate-env/bin/activate
   pip install -e weblate-src
   ```

4. Copy `weblate/settings_example.py` to `weblate/settings.py`.

5. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see *Ajustar a configuração*.

6. Create the database used by Weblate, see *Configuração de banco de dados para o Weblate*.

7. Build Django tables, static files and initial data (see *Preencher o banco de dados* and *Servir ficheiros estáticos*):

   ```
   weblate migrate
   weblate collectstatic
   weblate compress
   weblate compilemessages
   ```

   Nota: This step should be repeated whenever you update the repository.

Installing on OpenShift

With the OpenShift Weblate template you can get your personal Weblate instance up and running in seconds. All of Weblate's dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

Instalação

The following examples assume you have a working OpenShift v3.x environment, with `oc` client tool installed. Please check the OpenShift documentation for instructions.

Web Console

Copy the raw content from `template.yml` and import them into your project, then use the `Create` button in the OpenShift web console to create your application. The web console will prompt you for the values for all of the parameters used by the template.

CLI

To upload the Weblate template to your current project's template library, pass the `template.yml` file with the following command:

```
$ oc create -f https://raw.githubusercontent.com/WeblateOrg/openshift/main/
↪template.yml \
   -n <PROJECT>
```

The template is now available for selection using the web console or the CLI.

Parameters

The list of parameters that you can override are listed in the parameters section of the template. You can list them with the CLI by using the following command and specifying the file to be used:

```
$ oc process --parameters -f https://raw.githubusercontent.com/WeblateOrg/
↪openshift/main/template.yml

# If the template is already uploaded
$ oc process --parameters -n <PROJECT> weblate
```

Provisioning

You can also use the CLI to process templates and use the configuration that is generated to create objects immediately.

```
$ oc process -f https://raw.githubusercontent.com/WeblateOrg/openshift/main/
↪template.yml \
    -p APPLICATION_NAME=weblate \
    -p WEBLATE_VERSION=4.3.1-1 \
    -p WEBLATE_SITE_DOMAIN=weblate.app-openshift.example.com \
    -p POSTGRESQL_IMAGE=docker-registry.default.svc:5000/openshift/postgresql:9.6 \
    -p REDIS_IMAGE=docker-registry.default.svc:5000/openshift/redis:3.2 \
    | oc create -f
```

The Weblate instance should be available after successful migration and deployment at the specified `WEBLATE_SITE_DOMAIN` parameter.

After container setup, you can sign in as *admin* user with password provided in `WEBLATE_ADMIN_PASSWORD`, or a random password generated on first start if that was not set.

To reset *admin* password, restart the container with `WEBLATE_ADMIN_PASSWORD` set to new password in the respective `Secret`.

Eliminate

```
$ oc delete all -l app=<APPLICATION_NAME>
$ oc delete configmap -l app= <APPLICATION_NAME>
$ oc delete secret -l app=<APPLICATION_NAME>
# ATTTENTION! The following command is only optional and will permanently delete␣
↪all of your data.
$ oc delete pvc -l app=<APPLICATION_NAME>

$ oc delete all -l app=weblate \
    && oc delete secret -l app=weblate \
    && oc delete configmap -l app=weblate \
    && oc delete pvc -l app=weblate
```

Configuração

By processing the template a respective `ConfigMap` will be created and which can be used to customize the Weblate image. The `ConfigMap` is directly mounted as environment variables and triggers a new deployment every time it is changed. For further configuration options, see *Docker environment variables* for full list of environment variables.

Dependendo da sua configuração e experiência, escolha um método de instalação apropriado para si:

- *Installing using Docker*, recomendado para configurações de produção.
- Instalação virtualenv, recomendada para configurações de produção:
  - *Installing on Debian and Ubuntu*
  - *Installing on SUSE and openSUSE*
  - *Installing on RedHat, Fedora and CentOS*
  - *Installing on macOS*
- *Installing from sources*, recomendado para o desenvolvimento.
- *Installing on OpenShift*.

2.1.2 Requisitos de software

Sistema operacional

Weblate é conhecido por funcionar no Linux, FreeBSD e macOS. Outros sistemas como o Unix provavelmente funcionarão também.

O Weblate não é suportado no Windows. Mas ainda pode funcionar e patches são aceitos alegremente.

Outros serviços

Weblate está a usar outros serviços para a operação dele. Precisará pelo menos os seguintes serviços em execução:

- Servidor de banco de dados PostgreSQL, consulte *Configuração de banco de dados para o Weblate*.
- Servidor Redis para cache e fila de tarefas, consulte *Tarefas de fundo a usar o Celery*.
- Servidor SMTP para e-mails de saída, consulte *Configuração de e-mail de saída*.

Dependências Python

Weblate é escrito em Python e tem suporte de Python 3.6 ou mais novo. Pode instalar dependências a usar pip ou dos pacotes de distribuição deles, a lista completa está disponível em `requirements.txt`.

As dependências mais notáveis:

Django https://www.djangoproject.com/

Celery https://docs.celeryproject.org/

Translate Toolkit https://toolkit.translatehouse.org/

translation-finder https://github.com/WeblateOrg/translation-finder

Python Social Auth https://python-social-auth.readthedocs.io/

Django REST Framework https://www.django-rest-framework.org/

Dependências opcionais

Os módulos seguintes são necessários para alguns recursos do Weblate. Pode encontrar todos em `requirements-optional.txt`.

`Mercurial` (opcional para suporte de repositórios Mercurial) https://www.mercurial-scm.org/

`phply` (opcional para suporte de PHP) https://github.com/viraptor/phply

`tesserocr` (opcional para OCR de capturas de ecrã) https://github.com/sirfz/tesserocr

`akismet` (opcional para a sugestão de proteção de spam) https://github.com/ubernostrum/akismet

`ruamel.yaml` (opcional para *YAML files*) https://pypi.org/project/ruamel.yaml/

`Zeep` (opcional para *Microsoft Terminology Service*) https://docs.python-zeep.org/

`aeidon` (opcional para *Subtitle files*) https://pypi.org/project/aeidon/

Dependências de backend de banco de dados

O Weblate tem suporte de PostgreSQL, MySQL e MariaDB, consulte *Configuração de banco de dados para o Weblate* e a documentação dos backends para mais detalhes.

Outros requisitos do sistema

As dependências seguintes devem ser instaladas no sistema:

`Git` https://git-scm.com/

Pango, Cairo e ficheiros de cabeçalho relacionados e dados de introspecção gir https://cairographics.org/, https://pango.gnome.org/, veja *Pango e Cairo*

`git-review` (opcional para suporte de Gerrit) https://pypi.org/project/git-review/

`git-svn` (opcional para suporte de Subversion) https://git-scm.com/docs/git-svn

`tesseract` e os dados dele (opcional para OCR de capturas de ecrã) https://github.com/tesseract-ocr/tesseract

`licensee` (opcional para detetar a licença ao criar o componente) https://github.com/licensee/licensee

Dependências ao tempo de compilação

Para compilar algumas das *dependências do Python*, pode precisar de instalar as dependências delas. Isso depende de como as instala, por isso consulte pacotes individuais para obter a documentação. Não precisará deles se usar `Wheels` pré-compilado durante a instalação a usar `pip` ou quando usa pacotes de distribuição.

Pango e Cairo

Alterado na versão 3.7.

O Weblate usa Pango e Cairo para renderizar widgets de bitmap (ver *Promoting the translation*) e verificações de renderização (ver *Gerir letras*). Para instalar as ligações Python corretamente para esses, precisa de instalar bibliotecas de sistemas primeiro - precisa tanto do Cairo quanto do Pango, que por sua vez precisam de GLib. Todos esses devem ser instalados com ficheiros de desenvolvimento e dados de introspecção GObject.

2.1.3 Verificar assinaturas de lançamento

Os lançamentos do Weblate são criptograficamente assinados pelo programador que os lançou. Atualmente é Michal Čihař. A impressão digital da chave PGP é:

```
63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D
```

e pode obter mais informações de identificação de <https://keybase.io/nijel>.

Deve verificar se a assinatura corresponde ao ficheiro que descarregou. Desta forma, pode ter certeza de que está a usar o mesmo código que foi lançado. Também deve verificar a data da assinatura para ter certeza de que descarregou a versão mais recente.

Cada arquivo é acompanhado de ficheiros `.asc`, os quais contêm a assinatura PGP para ele. Uma vez que tenha ambos na mesma pasta, pode verificar a assinatura:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Can't check signature: public key not found
```

Como pode ver, o GPG reclama que não conhece a chave pública. Neste ponto deve fazer um dos seguintes passos:

- Use *wkd* para descarregar a chave:

```
$ gpg --auto-key-locate wkd --locate-keys michal@cihar.com
pub   rsa4096 2009-06-17 [SC]
      63CB1DF1EF12CF2AC0EE5A329C27B31342B7511D
uid           [ultimate] Michal Čihař <michal@cihar.com>
uid           [ultimate] Michal Čihař <nijel@debian.org>
uid           [ultimate] [jpeg image of size 8848]
uid           [ultimate] Michal Čihař (Braiins) <michal.cihar@braiins.cz>
sub   rsa4096 2009-06-17 [E]
sub   rsa4096 2015-09-09 [S]
```

- Descarregue o chaveiro do servidor do Michal e importe-o com:

```
$ gpg --import wmxth3chu9jfxdxywj1skpmhsj311mzm
```

- Descarregue e importe a chave de um dos servidores principais:

```
$ gpg --keyserver hkp://pgp.mit.edu --recv-keys␣
↪87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: key 9C27B31342B7511D: "Michal Čihař <michal@cihar.com>" imported
gpg: Total number processed: 1
gpg:              unchanged: 1
```

Isso vai melhorar a situação um pouco - neste momento pode verificar que a assinatura da chave dada está correta, mas ainda não pode confiar no nome usado na chave:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:                 aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:                 aka "[jpeg image of size 8848]" [ultimate]
gpg:                 aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>"␣
↪[ultimate]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 63CB 1DF1 EF12 CF2A C0EE  5A32 9C27 B313 42B7 511D
```

O problema aqui é que qualquer um poderia emitir a chave com este nome. Precisa garantir que a chave é realmente a propriedade da pessoa mencionada. O Manual de Privacidade do GNU aborda este tópico no capítulo Validating other keys on your public keyring. O método mais confiável é de conhecer o programador pessoalmente e trocar impressões digitais importantes, no entanto também pode confiar na rede de confiança. Dessa forma, pode confiar na chave transitivamente através de assinaturas de outros, que conheceram o programador pessoalmente.

Uma vez que a chave seja confiável, o aviso não ocorrerá:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:                 aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:                 aka "[jpeg image of size 8848]" [ultimate]
gpg:                 aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>"␣
↪[ultimate]
```

Se a assinatura for inválida (o ficheiro foi alterado), obteria um erro claro, independentemente do fato de que a chave é confiável ou não:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
```

(continues on next page)

(continuação da página anterior)

```
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: BAD signature from "Michal Čihař <michal@cihar.com>" [ultimate]
```

2.1.4 Permissões do sistema de ficheiros

O processo de Weblate precisa ser capaz de ler e escrever no diretório onde mantém os dados - *DATA_DIR*. Todos os ficheiros dentro deste diretório devem ser de propriedade e graváveis pelo utilizador que executa o Weblate.

A configuração predefinida põe-os na mesma árvore que as fontes do Weblate, no entanto, pode preferir movê-los para um local melhor, como `/var/lib/weblate`.

O Weblate tenta criar esses diretórios automaticamente, mas ele falhará quando não tiver permissões para fazê-lo.

Também deve tomar cuidado ao executar *Management commands*, pois eles devem ser executados sob o mesmo utilizador que o Weblate em si está a ser executado, caso contrário, permissões em alguns ficheiros podem estar erradas.

In the Docker container, all files in the `/app/data` volume have to be owned by weblate user inside the container (UID 1000).

Veja também:

Servir ficheiros estáticos

2.1.5 Configuração de banco de dados para o Weblate

Recomenda-se a executar o Weblate com um servidor de banco de dados PostgreSQL.

Veja também:

Usar um poderoso mecanismo de banco de dados, Databases, *Migrating from other databases to PostgreSQL*

PostgreSQL

PostgreSQL é geralmente a melhor escolha para sites baseados em Django. É o banco de dados de referência usado para implementar a camada de banco de dados Django.

Nota: O Weblate usa a extensão trigram que deve ser instalada separadamente em alguns casos. Procure por `postgresql-contrib` ou um pacote com nome similar.

Veja também:

PostgreSQL notes

Criar um banco de dados no PostgreSQL

Geralmente é uma boa ideia executar o Weblate num banco de dados separado e separar a conta do utilizador:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser --superuser --pwprompt weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -O weblate weblate
```

Dica: Se não quiser fazer do utilizador do Weblate um superutilizador no PostgreSQL, pode omiti-lo. Nesse caso, terá que executar algumas das etapas de migração manualmente como um superutilizador do PostgreSQL no esquema Weblate usará:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm WITH SCHEMA weblate;
```

Configurar Weblate para usar PostgreSQL

O trecho `settings.py` para PostgreSQL:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for authentication.
        # "ALTER_ROLE": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
    }
}
```

O código de migração assume que o nome da função combina com o nome de usuário enquanto autentica, caso não combine, por favor configure `ALTER_ROLE`. Caso o contrário, recebe um erro do PostrgreSQL sobre função não existente durante a migração do banco de dados `psycopg2.errors.UndefinedObject: função "weblate@nomedominio" não existe`).

MySQL e MariaDB

Weblate também pode ser usado com MySQL ou MariaDB, consulte MySQL notes e MariaDB notes para ressalvas ao uso do Django com elas.

Dica: Alguns recursos do Weblate terão melhor desempenho com *PostgreSQL*. Isso inclui a memória de pesquisa e tradução, que ambos utilizam recursos de texto completo no banco de dados e a implementação do PostgreSQL é superior.

Por causa disso, recomenda-se usar *PostgreSQL* para novas instalações.

A configuração seguinte é recomendada para Weblate:

- Use o conjunto de caracteres `utf8mb4` para permitir a representação de planos Unicode mais altos (por exemplo, emojis).
- Configure o servidor com `Innodb_large_prefix` para permitir índices mais longos em campos de texto.
- Defina o nível de isolamento para `READ COMMITTED`.
- O modo SQL deve ser definido como `STRICT_TRANS_TABLES`.

2.1.6 Outras configurações

Configuração de e-mail de saída

O Weblate envia e-mails em várias ocasiões - para a ativação de contas e sobre várias notificações configuradas pelos utilizadores. Para isso, precisa de acesso a um servidor de SMTP.

A configuração do servidor de e-mail é configurada a usar essas configurações: `EMAIL_HOST`, `EMAIL_HOST_PASSWORD`, `EMAIL_HOST_USER` e `EMAIL_PORT`. Os nomes deles são bastante autoexplicativos, mas pode encontrar mais informações na documentação do Django.

Nota: Pode verificar se o e-mail de saída está a funcionar corretamente pelo comando de gestão `sendtestemail` (veja *Invoking management commands* para instruções sobre como invocá-lo em ambientes diferentes).

Executar por trás de um proxy reverso

Vários recursos no Weblate dependem de ser capaz de obter o endereço IP do cliente. Isso inclui *Limitação de taxa*, *Spam protection* ou *Registo de auditoria*.

Na configuração predefinida, o Weblate analisa o endereço IP de `REMOTE_ADDR` que é definido pelo manipulador WSGI.

Se estiver a usar um proxy reverso, este campo provavelmente conterá o seu endereço. Precisa configurar o Weblate para confiar em cabeçalhos HTTP adicionais e analisar o endereço IP destes. Isso não pode ser ativado por predefinição, porque permitiria a falsificação de endereços IP para instalações que não usam um proxy reverso. Ativar *`IP_BEHIND_REVERSE_PROXY`* pode ser suficiente para as configurações mais usuais, mas podia precisar de ajustar *`IP_PROXY_HEADER`* e *`IP_PROXY_OFFSET`* também.

Veja também:

Spam protection, *Limitação de taxa*, *Registo de auditoria*, *`IP_BEHIND_REVERSE_PROXY`*, *`IP_PROXY_HEADER`*, *`IP_PROXY_OFFSET`*, `SECURE_PROXY_SSL_HEADER`

Proxy HTTP

O Weblate executa comandos VCS e esses que aceitam a configuração proxy do ambiente. A abordagem recomendada é definir configurações de proxy em `settings.py`:

```
import os
os.environ['http_proxy'] = "http://proxy.example.com:8080"
os.environ['HTTPS_PROXY'] = "http://proxy.example.com:8080"
```

Veja também:

Variáveis de Ambiente de Proxy

2.1.7 Ajustar a configuração

Veja também:

Sample configuration

Copie `weblate/settings_example.py` para `weblate/settings.py` e ajuste-o para corresponder à configuração. Provavelmente irá ajustar as opções a seguir: `ADMINS`

> Lista de administradores de sites para receber notificações quando algo dá errado, por exemplo, notificações em mesclagens fracassadas ou erros de Django.
>
> **Veja também:**

ADMINS

ALLOWED_HOSTS

Precisa definir isso para listar os hosts que o seu site deve servir. Por exemplo:

```
ALLOWED_HOSTS = ['demo.weblate.org']
```

Alternativamente, pode incluir curinga:

```
ALLOWED_HOSTS = ['*']
```

Veja também:

ALLOWED_HOSTS, *WEBLATE_ALLOWED_HOSTS*, *Configuração de hosts permitidos*

SESSION_ENGINE

Configure como as suas sessões serão armazenadas. Caso mantenha o mecanismo de backend do banco de dados predefinido, deve agendar: **`weblate clearsessions`** para remover dados de sessão obsoletos do banco de dados.

Se estiver a usar o Redis como cache (veja *Ativar o cache*) é recomendado também usá-lo para sessões:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
```

Veja também:

Configuring the session engine, SESSION_ENGINE

DATABASES

Conetividade ao servidor de banco de dados, verifique a documentação do Django para obter mais detalhes.

Veja também:

Configuração de banco de dados para o Weblate, DATABASES, Databases

DEBUG

Desative isto para qualquer servidor de produção. Com o modo de depuração ativado, o Django mostrará backtraces em caso de erro aos utilizadores, quando desativá-lo, erros serão enviados por e-mail para ADMINS (veja acima).

O modo de depuração também desacelera o Weblate, já que o Django armazena muito mais informações internamente neste caso.

Veja também:

DEBUG

DEFAULT_FROM_EMAIL

Endereço de remetente de e-mail para e-mail de saída, por exemplo, e-mails de registo.

Veja também:

DEFAULT_FROM_EMAIL

SECRET_KEY

Chave usada por Django para assinar informações em cookies, consulte *Chave secreta do Django* para obter mais informações.

Veja também:

SECRET_KEY

SERVER_EMAIL

E-mail usado como endereço de remetente para envio de e-mails ao administrador, por exemplo, notificações em mesclagens falhadas.

Veja também:

`SERVER_EMAIL`

2.1.8 Preencher o banco de dados

Depois que a sua configuração estiver pronta, pode executar `weblate migrate` para criar a estrutura do banco de dados. Agora deve ser capaz de criar projetos de tradução a usar a interface administrativa.

Caso quere executar uma instalação não interativamente, pode usar `weblate migrate --noinput` e depois criar um utilizador administrativo pelo comando *`createadmin`*.

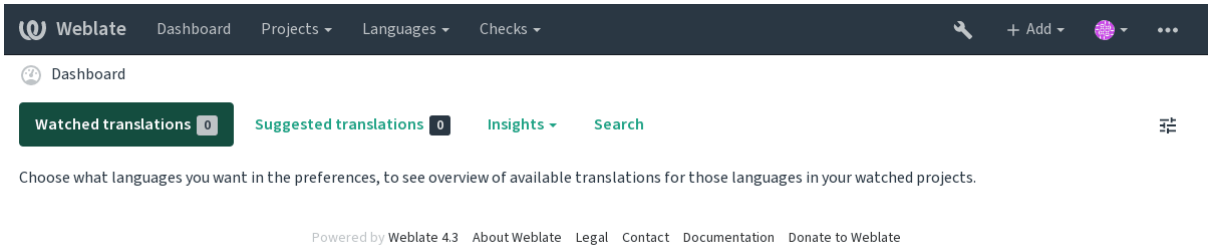
Uma vez feito, também deve verificar o *Relatório de desempenho* na interface administrativa, o que lhe dará dicas de configuração potencial não ideal no seu site.

Veja também:

Configuração, *Controlo de acesso*

2.1.9 Configuração de produção

Para uma configuração de produção, deve realizar ajustes descritos nas seções a seguir. As configurações mais críticas acionarão um aviso, que é indicado por um ponto de exclamação na barra superior se esitver conectado como um superutilizador:



Também é recomendado inspecionar verificações desencadeadas por Django (embora possa não precisar corrigir todas):

```
weblate check --deploy
```

Veja também:

Deployment checklist

Desativar o modo de depuração

Desative o modo de depuração do Django (*`DEBUG`*) com:

```
DEBUG = False
```

Com o modo de depuração ativado, o Django armazena todas as consultas executadas e mostra aos utilizadores backtraces de erros, o que não é desejado numa configuração de produção.

Veja também:

Ajustar a configuração

Configurar administradores corretamente

Defina os endereços de administração corretos à configuração *ADMINS* para definir quem receberá e-mails caso algo dê errado no servidor, por exemplo:

```
ADMINS = (
    ('Your Name', 'your_email@example.com'),
)
```

Veja também:

Ajustar a configuração

Definir domínio correto do site

Ajuste o nome e o domínio do site na interface administrativa, caso contrário, ligações no RSS ou e-mails de registo não funcionarão. Isto é configurado usando *SITE_DOMAIN* que deve conter o nome de domínio do site.

Alterado na versão 4.2: Antes da versão 4.2, a estrutura de sites do Django era usada em vez disso, consulte The "sites" framework.

Veja também:

Configuração de hosts permitidos, *Configurar HTTPS corretamente* *SITE_DOMAIN*, *WEBLATE_SITE_DOMAIN*, *ENABLE_HTTPS*

Configurar HTTPS corretamente

É fortemente recomendado executar Weblate a com o protocolo criptografado HTTPS. Depois de ativá-lo, deve definir *ENABLE_HTTPS* nas configurações:

```
ENABLE_HTTPS = True
```

Dica: Pode também configurar o HSTS, consulte SSL/HTTPS para obter mais detalhes.

Veja também:

ENABLE_HTTPS, *Configuração de hosts permitidos*, *Definir domínio correto do site*

Definir SECURE_HSTS_SECONDS corretamente

Se o seu site for servido sobre SSL, deve considerar definir um valor para `SECURE_HSTS_SECONDS` no `settings.py` para ativar HTTP Strict Transport Security. Por predefinição, está definido para 0 como mostrado abaixo.

```
SECURE_HSTS_SECONDS = 0
```

Se for definido como um valor inteiro não-zero, o cabeçalho `django.middleware.security.SecurityMiddleware` define o cabeçalho HTTP Strict Transport Security em todas as respostas que ainda não o possuem.

Aviso: Definir isto incorretamente pode quebrar irreversivelmente (por algum tempo) o seu site. Leia primeiro a documentação HTTP Strict Transport Security.

Usar um poderoso mecanismo de banco de dados

Por favor, use PostgreSQL para um ambiente de produção, consulte *Configuração de banco de dados para o Weblate* para obter mais informações.

Veja também:

Configuração de banco de dados para o Weblate, *Migrating from other databases to PostgreSQL*, *Ajustar a configuração*, Databases

Ativar o cache

Se for possível, use Redis do Django e ajuste a variável de configuração `CACHES`, por exemplo:

```
CACHES = {
    'default': {
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'redis://127.0.0.1:6379/0',
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # 'LOCATION': 'unix:///var/run/redis/redis.sock?db=0',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            'PARSER_CLASS': 'redis.connection.HiredisParser',
        }
    }
}
```

Veja também:

Cache de avatares, Django's cache framework

Cache de avatares

Além do cache de Django, Weblate realiza cache de avatares. Recomenda-se usar um cache separado, baseado em ficheiros para este fim:

```
CACHES = {
    'default': {
        # Default caching backend setup, see above
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'unix:///var/run/redis/redis.sock?db=0',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            'PARSER_CLASS': 'redis.connection.HiredisParser',
        }
    },
    'avatar': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': os.path.join(DATA_DIR, 'avatar-cache'),
        'TIMEOUT': 604800,
        'OPTIONS': {
            'MAX_ENTRIES': 1000,
        },
    }
```

Veja também:

ENABLE_AVATARS, *AVATAR_URL_PREFIX*, *Avatars*, *Ativar o cache*, Django's cache framework

Configurar envio de e-mail

O Weblate precisa enviar e-mails em várias ocasiões e esses e-mails devem ter um endereço de remetente correto, por favor, configure **:configuração:`SERVER_EMAIL`** e *DEFAULT_FROM_EMAIL* para combinar com o seu ambiente, por exemplo:

```
SERVER_EMAIL = 'admin@example.org'
DEFAULT_FROM_EMAIL = 'weblate@example.org'
```

Nota: Para desativar o envio de e-mails pelo Weblate, defina EMAIL_BACKEND a `django.core.mail.backends.dummy.EmailBackend`.

Isso desativará *toda* a entrega de e-mail, incluindo e-mails de registo ou redefinição de palavra-passe.

Veja também:

Ajustar a configuração, *Configuração de e-mail de saída*, EMAIL_BACKEND, DEFAULT_FROM_EMAIL, SERVER_EMAIL

Configuração de hosts permitidos

Django requer *ALLOWED_HOSTS* para manter uma lista de nomes de domínio que o seu site pode servir, deixá-lo vazio bloqueará todas solicitações.

Caso isso não esteja configurado para corresponder ao seu servidor HTTP, terá erros como `Invalid HTTP_HOST header: '1.1.1.1'. Pode ter que adicionar '1.1.1.1' ao ALLOWED_HOSTS.`

Dica: No contentor Docker, isso está disponível como *WEBLATE_ALLOWED_HOSTS*.

Veja também:

ALLOWED_HOSTS, *WEBLATE_ALLOWED_HOSTS*, *Definir domínio correto do site*

Chave secreta do Django

A configuração *SECRET_KEY* é usada pelo Django para assinar cookies e você deve realmente gerar o seu próprio valor em vez de usar o da configuração do exemplo.

Pode gerar uma nova chave por `weblate/examples/generate-secret-key`, que vem com o Weblate.

Veja também:

SECRET_KEY

Directório inicial

Alterado na versão 2.1: Isso não é mais necessário, agora o Weblate armazena todos os seus dados em *DATA_DIR*.

O diretório home do utilizador que executa o Weblate deve existir e ser gravável por este utilizador. Isso é especialmente necessário se quiser usar o SSH para acessar repositórios privados, mas o Git pode precisar acessar este diretório também (dependendo da versão git que usa).

Pode alterar o diretório usado pelo Weblate em `settings.py`, por exemplo, para defini-lo como diretório `configuration` na árvore do Weblate:

```
os.environ['HOME'] = os.path.join(BASE_DIR, 'configuration')
```

Nota: No Linux e em outros sistemas como UNIX, o caminho ao diretório home do utilizador é definido em `/etc/passwd`. Muitas distribuições usam um diretório sem permissão de escrita como predefinição para utilizadores para servir conteúdo web (como `apache`, `www-data` ou `wwwrun`), então tem que executar o Weblate sob um utilizador diferente ou alterar essa configuração.

Veja também:

Accessing repositories

Carregar modelos

Recomenda-se usar um carregador de modelo em cache para Django. Armazena modelos analisados e evita a necessidade de analizar cada solicitação. Pode configurá-lo a usar o trecho a seguir (a configuração `loaders` é importante aqui):

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join(BASE_DIR, 'templates'),
        ],
        'OPTIONS': {
            'context_processors': [
                'django.contrib.auth.context_processors.auth',
                'django.template.context_processors.debug',
                'django.template.context_processors.i18n',
                'django.template.context_processors.request',
                'django.template.context_processors.csrf',
                'django.contrib.messages.context_processors.messages',
                'weblate.trans.context_processors.weblate_context',
            ],
            'loaders': [
                ('django.template.loaders.cached.Loader', [
                    'django.template.loaders.filesystem.Loader',
                    'django.template.loaders.app_directories.Loader',
                ]),
            ],
        },
    },
]
```

Veja também:

`django.template.loaders.cached.Loader`

Executar tarefas de manutenção

Para um desempenho ideal, é uma boa ideia executar algumas tarefas de manutenção em segundo plano. Isso agora é feito automaticamente por *Tarefas de fundo a usar o Celery* e abrange as seguintes tarefas:

- Verificação de saúde da configuração (de hora em hora).
- Realização de commits de alterações pendentes (de hora em hora), consulte *Commits adiados* e *commit_pending*.
- Atualização de alertas de componentes (dialy).
- Atualização dos ramos remotos (nightly), consulte *AUTO_UPDATE*.
- Backup de memória de tradução para JSON (diariamente), consulte *dump_memory*.

- Tarefas de manutenção de texto completo e banco de dados (tarefas diárias e semanais), consulte *cleanup-trans*.

Alterado na versão 3.2: Desde a versão 3.2, a maneira predefinida de executar essas tarefas é usar o Celery e o Weblate já vem com a configuração adequada, consulte *Tarefas de fundo a usar o Celery*.

Codificação e localidades do sistema

As localidades do sistema devem ser configuradas para UTF-8. Na maioria das distribuições Linux, esta é a configuração predefinida. Se não é o caso no seu sistema, altere as localidades para a variante UTF-8.

Por exemplo, a editar `/etc/default/locale` e a definir lá `LANG="C.UTF-8"`.

Em alguns casos, os serviços individuais têm uma configuração separada para locais. Por exemplo, ao usar o Apache, pode defini-lo em `/etc/apache2/envvars`:

```
export LANG='en_US.UTF-8'
export LC_ALL='en_US.UTF-8'
```

Usar uma autoridade certificadora personalizada

O Weblate verifica os certificados SSL durante as solicitações HTTP. Caso esteja a usar uma autoridade de certificação personalizada que não seja confiável em maços predefinidos, terá que adicionar o seu certificado como confiável.

A abordagem preferida é fazer isso no nível do sistema. Consulte a documentação da sua distro para mais detalhes (por exemplo, no Debian isso pode ser feito a por o certificado da AC em `/usr/local/share/ca-certificates/` e executando **`update-ca-certificates`**).

Uma vez feito isso, as ferramentas do sistema confiarão no certificado e isso inclui o Git.

Para código em Python, precisará configurar solicitações para usar o pacote de AC do sistema em vez do fornecido . Isso pode ser conseguido pondo os seguintes trechos em `settings.py` (o caminho é específico do Debian):

```
import os
os.environ["REQUESTS_CA_BUNDLE"] = "/etc/ssl/certs/ca-certificates.crt"
```

Comprimir os ativos do cliente

O Weblate vem com um monte de ficheiros JavaScript e CSS. Por razões de desempenho, é bom comprimi-los antes de enviar para um cliente. Na configuração predefinida isso é feito rapidamente ao custo de pouca sobrecarga. Em grandes instalações, recomenda-se ativar o modo de compressão offline. Isso precisa ser feito na configuração e a compressão tem que ser acionada em cada atualização do Weblate.

A mudança da configuração é simples ao ativar `django.conf.settings.COMPRESS_OFFLINE` e configuração `django.conf.settings.COMPRESS_OFFLINE_CONTEXT` (este último já está incluído na configuração do exemplo):

```
COMPRESS_OFFLINE = True
```

Em cada implantação precisa compactar os ficheiros para corresponder à versão atual:

```
weblate compress
```

Dica: A imagem oficial do Docker já tem este recurso ativado.

Veja também:

Common Deployment Scenarios, *Servir ficheiros estáticos*

2.1.10 Executar o servidor

Precisará de vários serviços para executar o Weblate, a configuração recomendada consiste em:

- Servidor de banco de dados (consulte *Configuração de banco de dados para o Weblate*)
- Servidor de cache (consulte *Ativar o cache*)
- Servidor web frontend para ficheiros estáticos e terminação SSL (consulte *Servir ficheiros estáticos*)
- Servidor Wsgi para conteúdo dinâmico (consulte *Configuração de amostra para NGINX e uWSGI*)
- Celery para executar tarefas em segundo plano (consulte *Tarefas de fundo a usar o Celery*)

Nota: Existem algumas dependências entre os serviços, por exemplo, o cache e o banco de dados devem estar em execução ao iniciar os processos de Celery ou uwsgi.

Na maioria dos casos, executará todos os serviços num único servidor (virtual), mas se a sua instalação estar muito carregada, pode dividir os serviços. A única limitação disso é que os servidores Celery e Wsgi precisam acessar `DATA_DIR`.

Executar um servidor web

Executar o Weblate não é diferente de executar qualquer outro programa baseado em Django. Django é geralmente executado como uWSGI ou fcgi (consulte exemplos para diferentes servidores web abaixo).

Para fins de teste, pode usar o servidor web incorporado no Django:

```
weblate runserver
```

Aviso: NÃO USE ESTE SERVIDOR NUMA CONFIGURAÇÃO DE PRODUÇÃO. Não passou por auditorias de segurança ou testes de desempenho. Veja também a documentação de Django no `runserver`.

Dica: O servidor embutido do Django serve apenas ficheiros estáticos com `DEBUG` ativado, pois é destinado apenas ao desenvolvimento. Para uso em produção, consulte as configurações de wsgi em *Configuração de amostra para NGINX e uWSGI*, *Configuração de amostra para Apache*, *Configuração de amostra para Apache and Gunicorn* e *Servir ficheiros estáticos*.

Servir ficheiros estáticos

Alterado na versão 2.4: Antes da versão 2.4, o Weblate não usava a estrutura de ficheiros estáticos do Django corretamente e a configuração era mais complexa.

Django precisa coletar os ficheiros estáticos dele num único diretório. Para isso, execute `weblate collectstatic --noinput`. Isso copiará os ficheiros estáticos num diretório especificado pela configuração `STATIC_ROOT` (isso é a predefinição para um diretório `static` dentro de `DATA_DIR`).

Recomenda-se servir ficheiros estáticos diretamente do seu servidor web. Deve usá-los para os seguintes caminhos:

`/static/` Serve ficheiros estáticos para Weblate e a interface de administração (definida por `STATIC_ROOT`).

`/media/` Usado para o envio de mídia pelo utilizador (por exemplo, capturas de ecrã).

`/favicon.ico` Deve ser reescrito para reescrever uma regra para servir `/static/favicon.ico`.

Veja também:

Comprimir os ativos do cliente, Deploying Django, Deploying static files

Política de segurança de conteúdo

A configuração predefinida do Weblate ativa o middleware `weblate.middleware.SecurityMiddleware` que define cabeçalhos HTTP relacionados à segurança como `Content-Security-Policy` ou `X-XSS-Protection`. São configurados por predefinição para funcionar com o Weblate e a configuração dele, mas isso pode precisar de personalização no seu ambiente.

Veja também:

CSP_SCRIPT_SRC, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

Configuração de amostra para NGINX e uWSGI

Para executar o servidor web de produção, use o wrapper wsgi instalado com Weblate (no caso de ambiente virtual é instalado como `~/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py`). Também não se esqueça de definir o caminho de pesquisa Python para o seu virtualenv (por exemplo, usando *virtualenv = /home/user/weblate-env*` no uWSGI).

A configuração a seguir executa o Weblate como uWSGI sob o servidor web NGINX.

A configuração para NGINX (também disponível como `weblate/examples/weblate.nginx.conf`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
server {
    listen 80;
    server_name weblate;
    # Not used
    root /var/www/html;

    location ~ ^/favicon.ico$ {
        # DATA_DIR/static/favicon.ico
        alias /home/weblate/data/static/favicon.ico;
        expires 30d;
    }

    location /static/ {
        # DATA_DIR/static/
        alias /home/weblate/data/static/;
        expires 30d;
    }

    location /media/ {
        # DATA_DIR/media/
        alias /home/weblate/data/media/;
        expires 30d;
    }

    location / {
        include uwsgi_params;
        # Needed for long running operations in admin interface
        uwsgi_read_timeout 3600;
        # Adjust based to uwsgi configuration:
        uwsgi_pass unix:///run/uwsgi/app/weblate/socket;
        # uwsgi_pass 127.0.0.1:8080;
    }
}
```

Configuração para uWSGI (também disponível como `weblate/examples/weblate.uwsgi.ini`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
[uwsgi]
plugins       = python3
master        = true
protocol      = uwsgi
socket        = 127.0.0.1:8080
wsgi-file     = /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/wsgi.
↪py

# Add path to Weblate checkout if you did not install
# Weblate by pip
# python-path   = /path/to/weblate

# In case you're using virtualenv uncomment this:
virtualenv = /home/weblate/weblate-env

# Needed for OAuth/OpenID
buffer-size   = 8192

# Reload when consuming too much of memory
reload-on-rss = 250

# Increase number of workers for heavily loaded sites
workers       = 8

# Enable threads for Sentry error submission
enable-threads = true

# Child processes do not need file descriptors
close-on-exec = true

# Avoid default 0000 umask
umask = 0022

# Run as weblate user
uid = weblate
gid = weblate

# Enable harakiri mode (kill requests after some time)
# harakiri = 3600
# harakiri-verbose = true

# Enable uWSGI stats server
# stats = :1717
# stats-http = true

# Do not log some errors caused by client disconnects
ignore-sigpipe = true
ignore-write-errors = true
disable-write-exception = true
```

Veja também:

How to use Django with uWSGI

Configuração de amostra para Apache

A configuração seguinte executa Weblate como WSGI, precisa ter mod_wsgi ativado (disponível como `weblate/examples/apache.conf`):

```
#
# VirtualHost for Weblate
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

    WSGIScriptAlias / /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py process-group=weblate
    WSGIPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
        Require all granted
        </Files>
    </Directory>

</VirtualHost>
```

Nota: Weblate requires Python 3, so please make sure you are running Python 3 variant of the modwsgi. Usually it is available as a separate package, for example `libapache2-mod-wsgi-py3`.

Veja também:

Codificação e localidades do sistema, How to use Django with Apache and mod_wsgi

Configuração de amostra para Apache and Gunicorn

A configuração seguinte executa o Weblate em Gunicorn and Apache 2.4 (disponível como `weblate/examples/apache.gunicorn.conf`):

```
#
# VirtualHost for Weblate using gunicorn on localhost:8000
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:443>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/https_cert.cert
    SSLCertificateKeyFile /etc/apache2/ssl/https_key.pem
    SSLProxyEngine On

    ProxyPass /favicon.ico !
    ProxyPass /static/ !
    ProxyPass /media/ !

    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
    ProxyPreserveHost On
</VirtualHost>
```

Veja também:

How to use Django with Gunicorn

A executar o Weblate sob o caminho

Alterado na versão 1.3: This is supported since Weblate 1.3.

A sample Apache configuration to serve Weblate under `/weblate`. Again using mod_wsgi (also available as `weblate/examples/apache-path.conf`):

```
#
# VirtualHost for Weblate, running under /weblate path
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
```

(continues on next page)

(continuação da página anterior)

```
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /weblate/favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /weblate/static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /weblate/media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

    WSGIScriptAlias /weblate /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py process-group=weblate
    WSGIPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
        Require all granted
        </Files>
    </Directory>

</VirtualHost>
```

Additionally, you will have to adjust `weblate/settings.py`:

```
URL_PREFIX = '/weblate'
```

2.1.11 Tarefas de fundo a usar o Celery

Novo na versão 3.2.

Weblate uses Celery to process background tasks. The example settings come with eager configuration, which does process all tasks in place, but you want to change this to something more reasonable for a production setup.

A typical setup using Redis as a backend looks like this:

```
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = 'redis://localhost:6379'
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
```

You should also start the Celery worker to process the tasks and start scheduled tasks, this can be done directly on the command line (which is mostly useful when debugging or developing):

```
./weblate/examples/celery start
./weblate/examples/celery stop
```

Running Celery as system service

Most likely you will want to run Celery as a daemon and that is covered by Daemonization. For the most common Linux setup using systemd, you can use the example files shipped in the `examples` folder listed below.

Systemd unit to be placed as `/etc/systemd/system/celery-weblate.service`:

```
[Unit]
Description=Celery Service (Weblate)
After=network.target

[Service]
Type=forking
User=weblate
Group=weblate
EnvironmentFile=/etc/default/celery-weblate
WorkingDirectory=/home/weblate
RuntimeDirectory=celery
RuntimeDirectoryPreserve=restart
LogsDirectory=celery
ExecStart=/bin/sh -c '${CELERY_BIN} multi start ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'
ExecStop=/bin/sh -c '${CELERY_BIN} multi stopwait ${CELERYD_NODES} \
  --pidfile=${CELERYD_PID_FILE}'
ExecReload=/bin/sh -c '${CELERY_BIN} multi restart ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'

[Install]
WantedBy=multi-user.target
```

Environment configuration to be placed as `/etc/default/celery-weblate`:

```
# Name of nodes to start
CELERYD_NODES="celery notify memory backup translate"

# Absolute or relative path to the 'celery' command:
CELERY_BIN="/home/weblate/weblate-env/bin/celery"

# App instance to use
# comment out this line if you don't use an app
CELERY_APP="weblate.utils"

# Extra command-line arguments to the worker,
# increase concurency if you get weblate.E019
CELERYD_OPTS="--beat:celery --queues:celery=celery --prefetch-multiplier:celery=4 \
    --queues:notify=notify --prefetch-multiplier:notify=10 \
    --queues:memory=memory --prefetch-multiplier:memory=10 \
    --queues:translate=translate --prefetch-multiplier:translate=4 \
    --concurrency:backup=1 --queues:backup=backup  --prefetch-multiplier:backup=2"

# Logging configuration
# - %n will be replaced with the first part of the nodename.
# - %I will be replaced with the current child process index
#   and is important when using the prefork pool to avoid race conditions.
CELERYD_PID_FILE="/var/run/celery/weblate-%n.pid"
CELERYD_LOG_FILE="/var/log/celery/weblate-%n%I.log"
```

(continues on next page)

(continuação da página anterior)

```
CELERYD_LOG_LEVEL="INFO"

# Internal Weblate variable to indicate we're running inside Celery
CELERY_WORKER_RUNNING="1"
```

Logrotate configuration to be placed as `/etc/logrotate.d/celery`:

```
/var/log/celery/*.log {
        weekly
        missingok
        rotate 12
        compress
        notifempty
}
```

Nota: The Celery process has to be executed under the same user as Weblate and the WSGI process, otherwise files in the *DATA_DIR* will be stored with mixed ownership, leading to runtime issues.

Periodic tasks using Celery beat

Weblate comes with built-in setup for scheduled tasks. You can however define additional tasks in `settings.py`, for example see *Commits adiados*.

The tasks are supposed to be executed by Celery beats daemon. In case it is not working properly, it might not be running or its database was corrupted. Check the Celery startup logs in such case to figure out root cause.

Monitoring Celery status

You can use *celery_queues* to see current length of Celery task queues. In case the queue will get too long, you will also get configuration error in the admin interface.

Aviso: The Celery errors are by default only logged into Celery log and are not visible to user. In case you want to have overview on such failures, it is recommended to configure *Collecting error reports*.

Veja também:

Configuration and defaults, Workers Guide, Daemonization, Monitoring and Management Guide, *celery_queues*

2.1.12 Monitoring Weblate

Weblate provides the `/healthz/` URL to be used in simple health checks, for example using Kubernetes.

2.1.13 Collecting error reports

Weblate, as any other software, can fail. In order to collect useful failure states we recommend to use third party services to collect such information. This is especially useful in case of failing Celery tasks, which would otherwise only report error to the logs and you won't get notified on them. Weblate has support for the following services:

Sentry

Weblate has built-in support for Sentry. To use it, it's enough to set *SENTRY_DSN* in the `settings.py`:

```
SENTRY_DSN = "https://id@your.sentry.example.com/"
```

Rollbar

Weblate has built-in support for Rollbar. To use it, it's enough to follow instructions for Rollbar notifier for Python.

In short, you need to adjust `settings.py`:

```
# Add rollbar as last middleware:
MIDDLEWARE = [
    # … other middleware classes …
    'rollbar.contrib.django.middleware.RollbarNotifierMiddleware',
]

# Configure client access
ROLLBAR = {
    'access_token': 'POST_SERVER_ITEM_ACCESS_TOKEN',
    'client_token': 'POST_CLIENT_ITEM_ACCESS_TOKEN',
    'environment': 'development' if DEBUG else 'production',
    'branch': 'master',
    'root': '/absolute/path/to/code/root',
}
```

Everything else is integrated automatically, you will now collect both server and client side errors.

2.1.14 Migrating Weblate to another server

Migrating Weblate to another server should be pretty easy, however it stores data in few locations which you should migrate carefully. The best approach is to stop Weblate for the migration.

Migrating database

Depending on your database backend, you might have several options to migrate the database. The most straightforward one is to dump the database on one server and import it on the new one. Alternatively you can use replication in case your database supports it.

The best approach is to use database native tools, as they are usually the most effective (e.g. **`mysqldump`** or **`pg_dump`**). If you want to migrate between different databases, the only option might be to use Django management to dump and import the database:

```
# Export current data
weblate dumpdata > /tmp/weblate.dump
# Import dump
weblate loaddata /tmp/weblate.dump
```

Migrating VCS repositories

The VCS repositories stored under *DATA_DIR* need to be migrated as well. You can simply copy them or use **rsync** to do the migration more effectively.

Other notes

Don't forget to move other services Weblate might have been using like Redis, Cron jobs or custom authentication backends.

2.2 Implantações de Weblate

O Weblate pode ser facilmente instalado na sua nuvem. Encontre um guia detalhado para sua plataforma:

- *Installing using Docker*
- *Installing on OpenShift*

2.2.1 Gráfico do Helm

Pode instalar Weblate em Kubernetes a usar Helm. Consulte <https://github.com/WeblateOrg/helm/tree/master/charts/weblate> para as instruções detalhadas.

2.2.2 Pilha Weblate para Bitnami

Bitnami fornece uma pilha Weblate para muitas plataformas em <https://bitnami.com/stack/weblate>. A configuração será ajustada durante a instalação, consulte <https://bitnami.com/stack/weblate/README.txt> para mais documentação.

2.2.3 Weblate no YunoHost

O projeto de hospedagem própria YunoHost fornece um pacote para Weblate. Uma vez que tenha a sua instalação YunoHost, pode instalar o Weblate como qualquer outra aplicação. Ele fornecerá uma pilha de trabalho completo com backup e restauração, mas ainda pode ter que editar seu ficheiro de configurações para usos específicos.

Pode usar a sua interface de administração ou este botão (vai levá-lo ao seu servidor):

Também é possível usar a interface da linha de comando:

```
yunohost app install https://github.com/YunoHost-Apps/weblate_ynh
```

2.3 Upgrading Weblate

2.3.1 Docker image upgrades

The official Docker image (see *Installing using Docker*) has all upgrade steps integrated. There are no manual step besides pulling latest version.

2.3.2 Generic upgrade instructions

Before upgrading, please check the current *Requisitos de software* as they might have changed. Once all requirements are installed or updated, please adjust your `settings.py` to match changes in the configuration (consult `settings_example.py` for correct values).

Always check *Version specific instructions* before upgrade. In case you are skipping some versions, please follow instructions for all versions you are skipping in the upgrade. Sometimes it's better to upgrade to some intermediate version to ensure a smooth migration. Upgrading across multiple releases should work, but is not as well tested as single version upgrades.

Nota: It is recommended to perform a full database backup prior to upgrade so that you can roll back the database in case upgrade fails, see *Fazer backup e mover o Weblate*.

---

1. Stop wsgi and Celery processes. The upgrade can perform incompatible changes in the database, so it is always safer to avoid old processes running while upgrading.
2. Upgrade Weblate code.

   For pip installs it can be achieved by:

   ```
   pip install -U Weblate
   ```

   With Git checkout you need to fetch new source code and update your installation:

   ```
   cd weblate-src
   git pull
   # Update Weblate inside your virtualenv
   . ~/weblate-env/bin/pip install -e .
   # Install dependencies directly when not using virtualenv
   pip install --upgrade -r requirements.txt
   ```

3. Upgrade configuration file, refer to `settings_example.py` or *Version specific instructions* for needed steps.
4. Upgrade database structure:

   ```
   weblate migrate --noinput
   ```

5. Collect updated static files (see *Executar o servidor* and *Servir ficheiros estáticos*):

   ```
   weblate collectstatic --noinput
   ```

6. Compress JavaScript and CSS files (optional, see *Comprimir os ativos do cliente*):

   ```
   weblate compress
   ```

7. If you are running version from Git, you should also regenerate locale files every time you are upgrading. You can do this by invoking:

   ```
   weblate compilemessages
   ```

8. Verify that your setup is sane (see also *Configuração de produção*):

```
weblate check --deploy
```

9. Restart celery worker (see *Tarefas de fundo a usar o Celery*).

2.3.3 Version specific instructions

Upgrade from 2.x

If you are upgrading from 2.x release, always first upgrade to 3.0.1 and then continue upgrading in the 3.x series. Upgrades skipping this step are not supported and will break.

Veja também:

Upgrade from 2.20 to 3.0 in Weblate 3.0 documentation

Upgrade from 3.x

If you are upgrading from 3.x release, always first upgrade to 4.0.4 or 4.1.1 and then continue upgrading in the 4.x series. Upgrades skipping this step are not supported and will break.

Veja também:

Upgrade from 3.11 to 4.0 in Weblate 4.0 documentation

Upgrade from 4.0 to 4.1

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in `settings_example.py`, most notable middleware changes, please adjust your settings accordingly.
- There are new file formats, you might want to include them in case you modified the *WEBLATE_FORMATS*.
- There are new quality checks, you might want to include them in case you modified the *CHECK_LIST*.
- There is change in `DEFAULT_THROTTLE_CLASSES` setting to allow reporting of rate limiting in the API.
- There are some new and updated requirements.
- There is a change in `INSTALLED_APPS`.
- The *DeepL* machine translation now defaults to v2 API, you might need to adjust *MT_DEEPL_API_VERSION* in case your current DeepL subscription does not support that.

Veja também:

Generic upgrade instructions

Upgrade from 4.1 to 4.2

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- Upgrade from 3.x releases is not longer supported, please upgrade to 4.0 or 4.1 first.
- There are some new and updated requirements.
- There are several changes in `settings_example.py`, most notable new middleware and changed application ordering.
- The keys for JSON based formats no longer include leading dot. The strings are adjusted during the database migration, but external components might need adjustment in case you rely on keys in exports or API.
- The Celery configuration was changed to no longer use `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.
- The Weblate domain is now configured in the settings, see *SITE_DOMAIN* (or *WEBLATE_SITE_DOMAIN*). You will have to configure it before running Weblate.
- The username and email fields on user database now should be case insensitive unique. It was mistakenly not enforced with PostgreSQL.

Veja também:

Generic upgrade instructions

Upgrade from 4.2 to 4.3

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are some changes in quality checks, you might want to include them in case you modified the *CHECK_LIST*.
- The source language attribute was moved from project to a component what is exposed in the API. You will need to update *Cliente Weblate* in case you are using it.
- The database migration to 4.3 might take long depending on number of strings you are translating (expect around one hour of migration time per 100,000 source strings).
- There is a change in `INSTALLED_APPS`.
- There is a new setting *SESSION_COOKIE_AGE_AUTHENTICATED* which complements `SESSION_COOKIE_AGE`.
- In case you were using **`hub`** or **`lab`** to integrate with GitHub or GitLab, you will need to reconfigure this, see *GITHUB_CREDENTIALS* and *GITLAB_CREDENTIALS*.
- **Changed in 4.3.1:** The Celery configuration was changed to add `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.

Veja também:

Generic upgrade instructions

2.3.4 Upgrading from Python 2 to Python 3

Weblate no longer supports Python older than 3.5. In case you are still running on older version, please perform migration to Python 3 first on existing version and upgrade later. See Upgrading from Python 2 to Python 3 in the Weblate 3.11.1 documentation.

2.3.5 Migrating from other databases to PostgreSQL

If you are running Weblate on other dabatase than PostgreSQL, you should migrate to PostgreSQL as that will be the only supported database backend in the 4.0 release. The following steps will guide you in migrating your data between the databases. Please remember to stop both web and Celery servers prior to the migration, otherwise you might end up with inconsistent data.

Criar um banco de dados no PostgreSQL

Geralmente é uma boa ideia executar o Weblate num banco de dados separado e separar a conta do utilizador:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser -D -P weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -O weblate weblate
```

Migrating using Django JSON dumps

The simplest approach for migration is to utilize Django JSON dumps. This works well for smaller installations. On bigger sites you might want to use pgloader instead, see *Migrating to PotsgreSQL using pgloader*.

1. Add PostgreSQL as additional database connection to the `settings.py`:

```
DATABASES = {
    'default': {
        # Database engine
        'ENGINE': 'django.db.backends.mysql',
        # Database name
        'NAME': 'weblate',
        # Database user
        'USER': 'weblate',
        # Database password
        'PASSWORD': 'password',
        # Set to empty string for localhost
        'HOST': 'database.example.com',
        # Set to empty string for default
        'PORT': '',
        # Additional database options
        'OPTIONS': {
            # In case of using an older MySQL server, which has MyISAM as a␣
↪default storage
            # 'init_command': 'SET storage_engine=INNODB',
            # Uncomment for MySQL older than 5.7:
            # 'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
            # If your server supports it, see the Unicode issues above
           'charset': 'utf8mb4',
           # Change connection timeout in case you get MySQL gone away error:
           'connect_timeout': 28800,
```

(continues on next page)

(continuação da página anterior)

```
        }
    },
    'postgresql': {
        # Database engine
        'ENGINE': 'django.db.backends.postgresql',
        # Database name
        'NAME': 'weblate',
        # Database user
        'USER': 'weblate',
        # Database password
        'PASSWORD': 'password',
        # Set to empty string for localhost
        'HOST': 'database.example.com',
        # Set to empty string for default
        'PORT': '',
    }
}
```

2. Run migrations and drop any data inserted into the tables:

```
weblate migrate --database=postgresql
weblate sqlflush --database=postgresql | weblate dbshell --database=postgresql
```

3. Dump legacy database and import to PostgreSQL

```
weblate dumpdata --all --output weblate.json
weblate loaddata weblate.json --database=postgresql
```

4. Adjust `DATABASES` to use just PostgreSQL database as default, remove legacy connection.

Weblate should be now ready to run from the PostgreSQL database.

Migrating to PotsgreSQL using pgloader

The pgloader is a generic migration tool to migrate data to PostgreSQL. You can use it to migrate Weblate database.

1. Adjust your `settings.py` to use PostgreSQL as a database.
2. Migrate the schema in the PostgreSQL database:

```
weblate migrate
weblate sqlflush | weblate dbshell
```

3. Run the pgloader to transfer the data. The following script can be used to migrate the database, but you might want to learn more about pgloader to understand what it does and tweak it to match your setup:

```
LOAD DATABASE
     FROM      mysql://weblate:password@localhost/weblate
     INTO postgresql://weblate:password@localhost/weblate

WITH include no drop, truncate, create no tables, create no indexes, no␣
↪foreign keys, disable triggers, reset sequences, data only

ALTER SCHEMA 'weblate' RENAME TO 'public'
;
```

2.3.6 Migrating from Pootle

As Weblate was originally written as replacement from Pootle, it is supported to migrate user accounts from Pootle. You can dump the users from Pootle and import them using `importusers`.

2.4 Fazer backup e mover o Weblate

2.4.1 Backup automatizado pelo BorgBackup

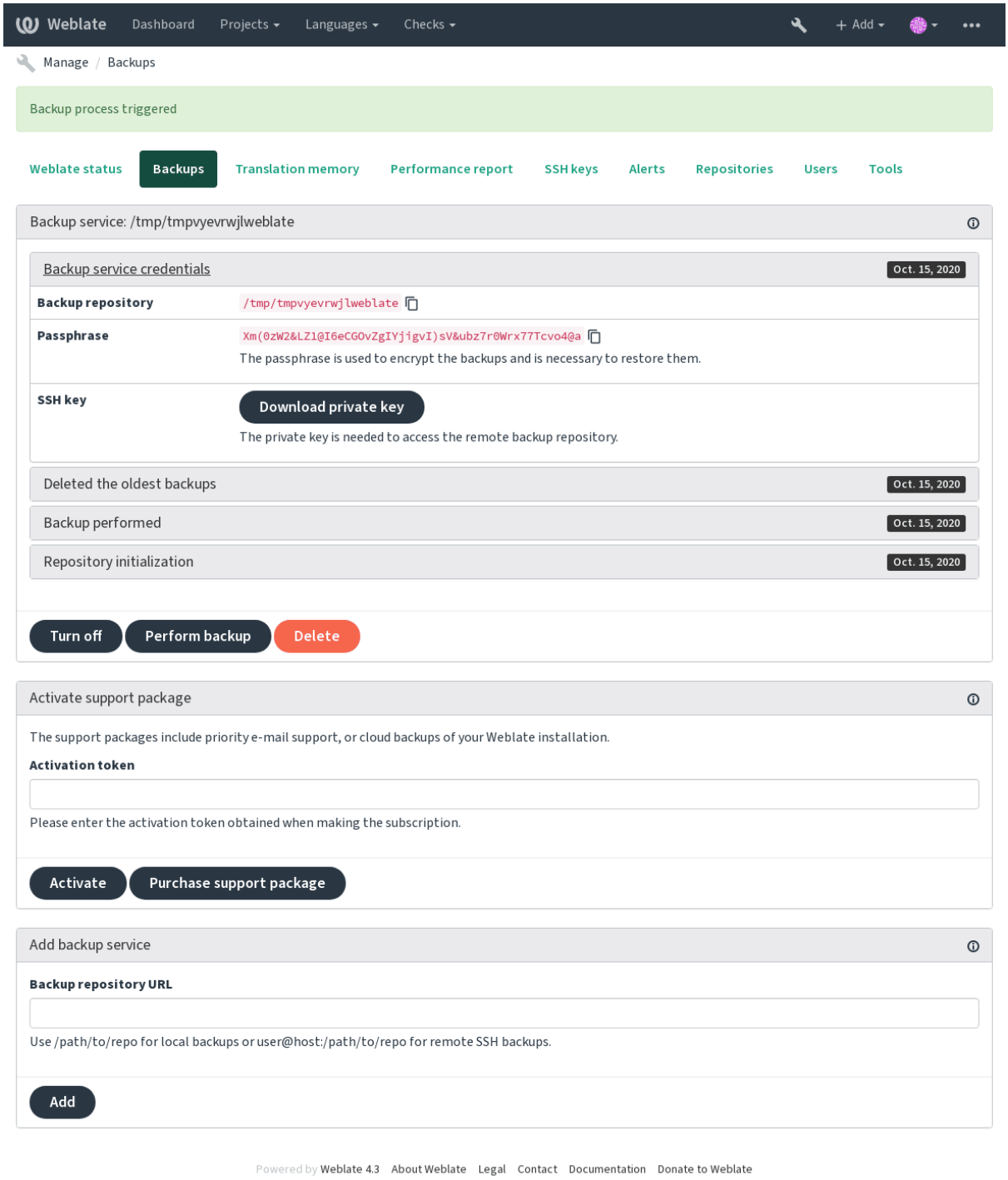
Novo na versão 3.9.

O Weblate tem suporte embutido para a criação de backups de serviços a usar BorgBackup. Borg cria backups encriptados, eficazes como espaço, que podem ser armazenados com segurança na nuvem. Os backups podem ser controlados na interface de gestão na guia *Backups*.

Aviso: Apenas o banco de dados PostgreSQL está incluído nos backups automatizados. Outros mecanismos de banco de dados devem ter os backups deles criados manualmente. É recomendado migrar ao PostgreSQL, veja em *Configuração de banco de dados para o Weblate* e *Migrating from other databases to PostgreSQL*.

Os backups que usam o Borg são incrementais e o Weblate é configurado para manter os seguintes backups:

- 14 backups diários
- 8 backups semanais
- 6 backups mensais



Chave de criptografia do Borg

BorgBackup cria backups criptografados e sem uma palavra-passe não será capaz de restaurar o backup. A palavra-passe é gerada ao adicionar um novo serviço de backup e deve copiá-lo e mantê-lo num lugar seguro.

No caso de que esteja a usar *Armazenamento de backup provisionado do Weblate*, também faça um backup da sua chave SSH privada — é usada para acessar os seus backups.

Veja também:

borg init

2.4.2 Armazenamento de backup provisionado do Weblate

A abordagem mais fácil para fazer backup da sua instância do Weblate é comprar o serviço de backup em weblate.org. O processo de ativação pode ser realizado em poucas etapas:

1. Compre o serviço de backup em https://weblate.org/support/#backup.
2. Insira a chave obtida na interface de gestão, veja *Integrando o apoio*.
3. Weblate vai conectar-se ao serviço de nuvem e obter informações de acesso aos backups.
4. Ativar a nova configuração de backup na guia *Backups*.
5. Faça backup das credenciais do Borg para conseguir restaurar os backups, veja *Chave de criptografia do Borg*.

Dica: Existe um passo manual a ativar para sua segurança. Sem o seu consentimento, nenhum dado é enviado ao repositório de backup obtido através do processo de registo.

2.4.3 Usar armazenamento de backup personalizado

Também pode usar o seu próprio armazenamento para backups. SSH pode ser usado para armazenar cópias de segurança no destino remoto, o servidor de destino precisa do BorgBackup instalado.

Veja também:

General na documentação do Borg

Sistema de ficheiros local

Recomenda-se especificar o caminho absoluto ao backup local, por exemplo */caminho/para/backup*. O diretório deve ser gravável pelo utilizador a executar o Weblate (veja *Permissões do sistema de ficheiros*). Caso não exista, o Weblate tentará criá-lo, mas precisa de permissões para fazê-lo.

Dica: Ao executar o Weblate no Docker, certifique-se de que o local de backup seja exposto como um volume do contentor Weblate. Caso contrário, os backups seriam descartados pelo Docker na reinicialização do contentor.

Uma opção é pôr backups no volume existente. Por exemplo, escolha `/app/data/borgbackup`. Este é o volume existente no contentor.

Também pode adicionar um novo contentor para os backups no ficheiro de composição do Docker e usar, por exemplo `/borgbackup`:

```
services:
  weblate:
    volumes:
      - /home/weblate/data:/app/data
      - /home/weblate/borgbackup:/borgbackup
```

The directory where backups will be stored have to be owned by UID 1000, otherwise Weblate will not be able to write the backups there.

Backups remotos

Há suporte para backups remotos a usar o SSH. O servidor SSH precisa ter BorgBackup instalado. O Weblate conecta-se ao servidor a usar uma chave SSH, então certifique-se de que a chave SSH do Weblate seja aceita pelo servidor (veja *Weblate SSH key*).

Dica: *Armazenamento de backup provisionado do Weblate* fornece backups remotos automatizados.

2.4.4 Restaurar do BorgBackup

1. Restaurar o acesso ao repositório de backup e preparar a sua palavra-passe de backup.
2. Liste o backup existente no servidor a usar `borg list REPOSITÓRIO`.
3. Restaure o backup desejado para o diretório atual a usar `borg extract REPOSITÓRIO::PACOTE`.
4. Restaure o banco de dados do despejo de SQL posto no diretório `backup` no diretório de dados do Weblate (veja :ref:"backup-dumps").
5. Copie a configuração do Weblate (`backups/settings.py`, veja *Dados despejados para backups*) até o local correto, veja *Ajustar a configuração*.
6. Copie todo o diretório de dados restaurados para o local configurado por *DATA_DIR*.

A sessão de Borg podia parecer com isso:

```
$ borg list /tmp/xxx
Enter passphrase for key /tmp/xxx:
2019-09-26T14:56:08                  Thu, 2019-09-26 14:56:08␣
↪[de0e0f13643635d5090e9896bdaceb92a023050749ad3f3350e788f1a65576a5]
$ borg extract /tmp/xxx::2019-09-26T14:56:08
Enter passphrase for key /tmp/xxx:
```

Veja também:

borg list, borg extract

2.4.5 Backup manual

Dependendo do que deseja gravar, faça backup do tipo de dados que o Weblate armazena em cada lugar.

Dica: No caso de fazer backups manuais, pode silenciar avisos do Weblate sobre a falta de backups a adicionar `weblate.I028` a `SILENCED_SYSTEM_CHECKS` em `settings.py` ou *WEBLATE_SILENCED_SYSTEM_CHECKS* ao Docker.

```
SILENCED_SYSTEM_CHECKS.append("weblate.I028")
```

Banco de dados

O local de armazenamento real depende da configuração do seu banco de dados.

O banco de dados é o armazenamento mais importante. Configure backups regulares do seu banco de dados, sem o qual toda a sua configuração de tradução desaparecerá.

Backup nativo do banco de dados

A abordagem recomendada é de fazer o despejo do banco de dados a usar ferramentas nativas, tais como **`pg_dump`** ou **`msqldump`**. Esta abordagem normalmente tem um desempenho melhor do que o backup do Django e restaura tabelas completas com todos os dados.

Pode restaurar esse backup na versão mais nova do Weblate, ele executará quaisquer migrações necessárias ao executar em `migrate`. Consulte *Upgrading Weblate* sobre informações mais detalhadas sobre como realizar a atualização entre as versões.

Backup do banco de dados do Django

Alternativamente, pode fazer backup do banco de dados pelo comando `dumpdata` do Django. Dessa forma o backup é agnóstico de banco de dados e pode ser usado caso queira alterar o backend do banco de dados.

Antes de restaurar, deve estar a usar exatamente a mesma versão do Weblate que foi usada ao fazer backups. Isso é necessário, pois a estrutura do banco de dados muda entre as versões e acabaria corrompendo os dados de alguma forma. Depois de instalar a mesma versão, execute todas as migrações do banco de dados a usar `migrate`.

Uma vez feito isto, algumas entradas já serão criadas no banco de dados e as terá no backup do banco de dados também. A abordagem recomendada é apagar essas entradas manualmente a usar o shell de gestão (veja *Invoking management commands*):

```
weblate shell
>>> from weblate.auth.models import User
>>> User.objects.get(username='anonymous').delete()
```

Ficheiros

Se tiver espaço de backup suficiente, basta fazer backup de todo o *DATA_DIR*. Esta é uma situação segura, mesmo que inclua alguns ficheiros que não queira. As secções a seguir descrevem em detalhes do que deve fazer backup e o que pode pular.

Dados despejados para backups

Armazenados em *DATA_DIR* `/backups`.

O Weblate despeja vários dados aqui e pode incluir esses ficheiros para backups mais completos. Os ficheiros são atualizados diariamente (requer um servidor de «beats» do Celery em execução, consulte *Tarefas de fundo a usar o Celery*). Atualmente, isto inclui:

- Configurações do Weblate como `settings.py` (existe também a versão expandida em `settings-expanded.py`).
- Backup de banco de dados PostgreSQL como `database.sql`.

Os backups do banco de dados são gravados como texto simples por predefinição, mas também podem ser comprimidos ou totalmente ignorados a usar *DATABASE_BACKUP*.

Repositórios de controle de versão

Armazenados em *DATA_DIR* /vcs.

Os repositórios de controle de versão contêm uma cópia dos seus repositórios upstream com alterações do Weblate. Se tiver o push ao fazer commit ativado para todos os seus componentes de tradução, todas as alterações do Weblate são incluídas upstream e não precisa fazer backup dos repositórios no lado do Weblate. Eles podem ser clonados novamente dos locais upstream sem perda de dados.

Chaves SSH e GPG

Armazenados em *DATA_DIR* /ssh e *DATA_DIR* /home.

Se está a usar chaves de SSH ou GPG geradas pelo Weblate, deve fazer backup destes locais; caso contrário, vai perder as chaves privadas e terá que gerar outras novamente.

Ficheiros enviados pelo utilizador

Armazenados em *DATA_DIR* /media.

Deve fazer o backup dos ficheiros enviados pelo utilizador (por exemplo, *Visual context for strings*).

Tarefas do Celery

A fila de tarefas do Celery pode conter algumas informações, mas geralmente não é necessária para um backup. No máximo, perderá atualizações que ainda não foram processadas para a memória de tradução. Recomenda-se de realizar as atualizações de texto completo ou repositório ao restaurar de qualquer maneira, de modo que não há problema em perdê-las.

Veja também:

Tarefas de fundo a usar o Celery

Linha de comando para backup manual

Através de uma tarefa de cron pode configurar um comando bash a ser executado diariamente, por exemplo:

```
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$(date -u +%Y-%m-%d_%H%M%S).xz␣
↪backups vcs ssh home media fonts secret
```

A cadeia entre aspas após XZ_OPT permite que escolha as suas opções do xz, por exemplo, a quantidade de memória utilizada para compressão; veja https://linux.die.net/man/1/xz

Pode ajustar a lista de pastas e ficheiros às suas necessidades. Por exemplo, para evitar gravar a memória de tradução (na pasta backups), pode usar:

```
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$(date -u +%Y-%m-%d_%H%M%S).xz␣
↪backups/database.sql backups/settings.py vcs ssh home media fonts secret
```

2.4.6 Restaurar backup manual

1. Restaure todos os dados dos quais tenha feito backup.
2. Atualize todos repositórios a usar o *updategit*.

```
weblate updategit --all
```

2.4.7 Mover uma instalação do Weblate

Realoque a instalação de um sistema diferente, seguindo as instruções de backup e restauração acima.

Veja também:

Upgrading from Python 2 to Python 3, *Migrating from other databases to PostgreSQL*

2.5 Autenticação

2.5.1 Registo de utilizador

A configuração predefinida para Weblate é usar python-social-auth, um formulário no site para lidar com o registo de novos utilizadores. Depois de confirmar o seu e-mail, um novo utilizador pode contribuir ou autenticar a usar um dos serviços de terceiros.

Também pode desativar o registo de novos utilizadores configurando *REGISTRATION_OPEN*.

As tentativas de autenticação estão sujeitas a *Limitação de taxa*.

2.5.2 Backends de autenticação

A solução embutida do Django é utilizada para autenticação, incluindo várias opções sociais para fazê-lo. Utilizando-a, pode importar o banco de dados de utilizadores de outros projetos baseados no Django (veja *Migrating from Pootle*).

Django pode, adicionalmente, ser configurado para autenticar em outros meios também.

Veja também:

Authentication settings descreve como configurar a autenticação na imagem oficial do Docker.

2.5.3 Autenticação social

Graças ao Welcome to Python Social Auth's documentation!, o Weblate tem suporte a autenticação utilizando muitos serviços de terceiros, tais como GitLab, Ubuntu, Fedora, etc.

Por favor, verifique a documentação deles por instruções de configuração genéricas em Django Framework.

Nota: Por predefinição, o Weblate conta com serviços de autenticação de terceiros para fornecer um endereço de e-mail validado. Se alguns dos serviços que deseja usar não suportarem isto, por favor aplique a validação de e-mail no lado Weblate configurando FORCE_EMAIL_VALIDATION para eles. Por exemplo:

```
SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True
```

Veja também:

Pipeline

Permitir backends individuais é bastante fácil, é apenas uma questão de adicionar uma entrada à configuração AUTHENTICATION_BACKENDS e possivelmente adicionar chaves necessárias para um determinado método de autenticação. Por favor, note que alguns backends não fornecem e-mails do utilizador por predefinição, tem que solicitá-lo explicitamente, caso contrário o Weblate não será capaz de corretamente dar mérito às contribuições que os utilizadores fazem.

Veja também:

Backend de Python Social Auth

Autenticação por OpenID

Para serviços baseados em OpenID, geralmente é apenas uma questão de ativá-los. A secção a seguir permite a autenticação OpenID para OpenSUSE, Fedora e Ubuntu:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.email.EmailAuth',
    'social_core.backends.suse.OpenSUSEOpenId',
    'social_core.backends.ubuntu.UbuntuOpenId',
    'social_core.backends.fedora.FedoraOpenId',
    'weblate.accounts.auth.WeblateUserBackend',
)
```

Veja também:

OpenID

Autenticação por GitHub

Precisa registar uma aplicação no GitHub e dar todos os segredos dele ao Weblate:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.github.GithubOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = 'GitHub Client ID'
SOCIAL_AUTH_GITHUB_SECRET = 'GitHub Client Secret'
SOCIAL_AUTH_GITHUB_SCOPE = ['user:email']
```

O GitHub deve ser configurado para ter URL de um retorno de chamada como `https://example.com/accounts/complete/github/`.

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

GitHub

Autenticação por Bitbucket

Precisa registar uma aplicação no Bitbucket e dar todos os segredos dele ao Weblate:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.bitbucket.BitbucketOAuth',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_BITBUCKET_KEY = 'Bitbucket Client ID'
SOCIAL_AUTH_BITBUCKET_SECRET = 'Bitbucket Client Secret'
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True
```

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

Bitbucket

OAuth 2 do Google

Para usar o OAuth 2 do Google, precisa registar-se numa aplicação em <https://console.developers.google.com/> e ativar a API do Google+.

A URL de redirecionamento é `https://SERVIDOR WEBLATE/accounts/complete/google-oauth2/`

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.google.GoogleOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = 'Client ID'
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = 'Client secret'
```

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

Google

OAuth 2 do Facebook

Como de costume com os serviços OAuth 2, precisa registar a sua aplicação no Facebook. Uma vez feito, pode configurar o Weblate para usá-lo:

A URL de redirecionamento é `https://SERVIDOR WEBLATE/accounts/complete/facebook/`

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.facebook.FacebookOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_FACEBOOK_KEY = 'key'
SOCIAL_AUTH_FACEBOOK_SECRET = 'secret'
SOCIAL_AUTH_FACEBOOK_SCOPE = ['email', 'public_profile']
```

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

Facebook

OAuth 2 do GitLab

Para usar o OAuth 2 do GitLab, precisa registar uma aplicação em <https://gitlab.com/profile/applications>.

A URL de redirecionamento é `https://SERVIDOR WEBLATE/accounts/complete/gitlab/` e garantir que marque o escopo *read_user*.

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.gitlab.GitLabOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_GITLAB_KEY = 'Application ID'
SOCIAL_AUTH_GITLAB_SECRET = 'Secret'
SOCIAL_AUTH_GITLAB_SCOPE = ['read_user']

# If you are using your own GitLab
# SOCIAL_AUTH_GITLAB_API_URL = 'https://gitlab.example.com/'
```

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

GitLab

Active Directory do Microsoft Azure

Weblate pode ser configurado para usar inquilinos comuns ou específicos para autenticação.

O URL de redirecionamento é `https://SERVIDOR WEBLATE/accounts/complete/azuread-oauth2/` para autenticação comum e `https://SERVIDOR WEBLATE/accounts/complete/azuread-tenant-oauth2/` para autenticação específica do inquilino.

```
# Azure AD common

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread.AzureADOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET = ""
```

```
# Azure AD Tenant

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread_tenant.AzureADTenantOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET = ""
# Tenant ID
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID = ""
```

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

Microsoft Azure Active Directory

Slack

Para usar o OAuth 2 do Slack, precisa registar uma aplicação em <https://api.slack.com/apps>.

A URL de redirecionamento é `https://SERVIDOR WEBLATE/accounts/complete/slack/`.

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.slack.SlackOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_SLACK_KEY = ''
SOCIAL_AUTH_SLACK_SECRET = ''
```

Nota: O Weblate fornecia URL de retorno de chamada durante a autenticação inclui domínio configurado. No caso de obter erros sobre incompatibilidade de URL, pode corrigir isso, consulte *Definir domínio correto do site*.

Veja também:

Slack

Desativar autenticação por palavra-passe

Autenticação por e-mail e palavra-passe pode ser desativada através da remoção de `social_core.backends.email.EmailAuth` de `AUTHENTICATION_BACKENDS`. Mantenha sempre `weblate.accounts.auth.WeblateUserBackend` lá, pois é necessário para a funcionalidade central do Weblate.

Dica: Ainda pode usar autenticação por palavra-passe para a interface administrativa, para utilizadores que lá cria manualmente. Basta navegar para `/admin/`.

Por exemplo, a autenticação a usar apenas o provedor Open ID do openSUSE pode ser alcançada a usar o seguinte:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.suse.OpenSUSEOpenId',
    'weblate.accounts.auth.WeblateUserBackend',
)
```

2.5.4 Autenticação por palavra-passe

A predefinição `settings.py` vem com um razoável conjunto de `AUTH_PASSWORD_VALIDATORS`:

- As palavras-passe não podem ser muito similares com as suas outras informações pessoais.
- As palavras-passe devem conter no mínimo de 10 caracteres.
- As palavras-passe não podem ser palaras-passe comumente usadas.
- As palavras-passe não podem ser inteiramente numéricas.
- As palavras-passe não podem consistir num único caractere ou apenas espaço em branco.
- As palavras-passe não podem corresponder a uma palavra-passe que já usou no passado.

Pode personalizar esta configuração para corresponder à sua política de palavra-passe.

Além disso, também pode instalar o django-zxcvbn-password o que dá bastante estimativas realistas de complexidade da palavra-passe e permite rejeitar palavras-passe abaixo de um determinado limite.

2.5.5 Autenticação por SAML

Novo na versão 4.1.1.

Siga as instruções do Python Social Auth para configuração. Diferenças notáveis:

- Weblate tem suporte a único IDP que tem de ser chamado de `weblate` em `SOCIAL_AUTH_SAML_ENABLED_IDPS`.
- A URL de metadados XML de SAML é `/accounts/metadata/saml/`.
- As configurações a seguir são preenchidas automaticamente: `SOCIAL_AUTH_SAML_SP_ENTITY_ID`, `SOCIAL_AUTH_SAML_TECHNICAL_CONTACT`, `SOCIAL_AUTH_SAML_SUPPORT_CONTACT`

Exemplo de configuração:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    "social_core.backends.saml.SAMLAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = "-----BEGIN CERTIFICATE-----"
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = "-----BEGIN PRIVATE KEY-----"
SOCIAL_AUTH_SAML_ENABLED_IDPS = {
    "weblate": {
        "entity_id": "https://idp.testshib.org/idp/shibboleth",
        "url": "https://idp.testshib.org/idp/profile/SAML2/Redirect/SSO",
        "x509cert": "MIIEDjCCAvagAwIBAgIBADA ... 8Bbnl+ev0peYzxFyF5sQA==",
        "attr_name": "full_name",
        "attr_username": "username",
        "attr_email": "email",
    }
}
```

Veja também:

Configurando SAML no Docker, SAML

2.5.6 Autenticação por LDAP

A autenticação por LDAP pode ser melhor alcançada utilizando o pacote *django-auth-ldap*. Pode instalá-lo através dos meios habituais:

```
# Using PyPI
pip install django-auth-ldap>=1.3.0

# Using apt-get
apt-get install python-django-auth-ldap
```

Aviso: Com django-auth-ldap anterior a 1.3.0, o *Atribuições de grupo automáticas* não funcionarão corretamente para utilizadores recentemente criados.

Nota: Há algumas incompatibilidades no módulo Python LDAP 3.1.0, o que o pode impedir de usar essa versão. Se obter o erro AttributeError: "module" object has no attribute "_trace_level", fayendo o downgrade para python-ldap 3.0.0 pode ajudar.

Uma vez que tenha o pacote instalado, pode ligá-lo à autenticação do Django:

```
# Add LDAP backed, keep Django one if you want to be able to login
# even without LDAP for admin account
AUTHENTICATION_BACKENDS = (
    'django_auth_ldap.backend.LDAPBackend',
    'weblate.accounts.auth.WeblateUserBackend',
)

# LDAP server address
AUTH_LDAP_SERVER_URI = 'ldaps://ldap.example.net'

# DN to use for authentication
```

(continues on next page)

(continuação da página anterior)

```
AUTH_LDAP_USER_DN_TEMPLATE = 'cn=%(user)s,o=Example'
# Depending on your LDAP server, you might use a different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users,dc=example,dc=com'

# List of attributes to import from LDAP upon login
# Weblate stores full name of the user in the full_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    'full_name': 'name',
    # Use the following if your LDAP server does not have full name
    # Weblate will merge them later
    # 'first_name': 'givenName',
    # 'last_name': 'sn',
    # Email is required for Weblate (used in VCS commits)
    'email': 'mail',
}

# Hide the registration form
REGISTRATION_OPEN = False
```

Nota: Deve remover `'social_core.backends.email.EmailAuth'` da configuração AUTHENTICATION_BACKENDS, caso contrário, os utilizadores poderão definir a palavra-passe deles no Weblate e autenticar a usar-a. Manter `'weblate.accounts.auth.WeblateUserBackend'` ainda é necessário para fazer permissões e facilitar utilizadores anónimos. Também permitirá que faça login a usar uma conta administrativa local, se a criou (por exemplo, a usar *createadmin*).

Usando palavra-passe associada

Se não puder usar a associação direta para autenticação, precisará usar a pesquisa e fornecer um utilizador para associar à pesquisa. Por exemplo:

```
import ldap
from django_auth_ldap.config import LDAPSearch

AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_USER_SEARCH = LDAPSearch("ou=users,dc=example,dc=com",
    ldap.SCOPE_SUBTREE, "(uid=%(user)s)")
```

Integração com o Active Directory

```
import ldap
from django_auth_ldap.config import LDAPSearch, NestedActiveDirectoryGroupType

AUTH_LDAP_BIND_DN = "CN=ldap,CN=Users,DC=example,DC=com"
AUTH_LDAP_BIND_PASSWORD = "password"

# User and group search objects and types
AUTH_LDAP_USER_SEARCH = LDAPSearch("CN=Users,DC=example,DC=com", ldap.SCOPE_
↪SUBTREE, "(sAMAccountName=%(user)s)")

# Make selected group a superuser in Weblate
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
    # is_superuser means user has all permissions
    "is_superuser": "CN=weblate_AdminUsers,OU=Groups,DC=example,DC=com",
```

(continues on next page)

(continuação da página anterior)

```
}

# Map groups from AD to Weblate
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU=Groups,DC=example,DC=com", ldap.SCOPE_
↪SUBTREE, "(objectClass=group)")
AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True

# Optionally enable group mirroring from LDAP to Weblate
# AUTH_LDAP_MIRROR_GROUPS = True
```

Veja também:

Django Authentication Using LDAP, Authentication

2.5.7 Autenticação por CAS

A autenticação por CAS pode ser alcançada a usar um pacote como o *django-cas-ng*.

O primeiro passo é divulgar o campo de e-mail do utilizador via CAS. Isso tem que ser configurado no próprio servidor CAS e requer que utilize pelo menos CAS v2, já que o CAS v1 não tem suporte a atributos.

O segundo passo é atualizar a Weblate para utilizar o seu servidor CAS e os seus atributos.

Para instalar *django-cas-ng*:

```
pip install django-cas-ng
```

Uma vez que o pacote está instalado, pode conectá-lo ao sistema de autenticação do Django a modificar o ficheiro `settings.py`:

```
# Add CAS backed, keep the Django one if you want to be able to sign in
# even without LDAP for the admin account
AUTHENTICATION_BACKENDS = (
    'django_cas_ng.backends.CASBackend',
    'weblate.accounts.auth.WeblateUserBackend',
)

# CAS server address
CAS_SERVER_URL = 'https://cas.example.net/cas/'

# Add django_cas_ng somewhere in the list of INSTALLED_APPS
INSTALLED_APPS = (
    ...,
    'django_cas_ng'
)
```

Finalmente, um sinal pode ser usado para mapear o campo de e-mail para o objeto do utilizador. Para que isso funcione, tem que importar o sinal do pacote *django-cas-ng* e conectar o seu código com este sinal. Fazer isto em configurações de ficheiro pode causar problemas, portanto, é sugerido pôr-lo:

- No método `django.apps.AppConfig.ready()` da configuração do seu app
- No ficheiro `urls.py` do projeto (quando não há modelos)

```
from django_cas_ng.signals import cas_user_authenticated
from django.dispatch import receiver
@receiver(cas_user_authenticated)
def update_user_email_address(sender, user=None, attributes=None, **kwargs):
    # If your CAS server does not always include the email attribute
    # you can wrap the next two lines of code in a try/catch block.
```

(continues on next page)

(continuação da página anterior)

```
    user.email = attributes['email']
    user.save()
```

Veja também:

Django CAS NG

2.5.8 Configurando autenticação por Django de terceiros

Geralmente, qualquer extensão de autenticação Django deve funcionar com Weblate. Basta seguir as instruções da extensão, lembrando de manter o backend do utilizador Weblate instalado.

Veja também:

Autenticação por LDAP, *Autenticação por CAS*

Normalmente, a instalação consiste em adicionar uma autenticação de backend a `AUTHENTICATION_BACKENDS`e a instalar uma app de autenticação (se houver) no :setting:`django:INSTALLED_APPS`:

```
AUTHENTICATION_BACKENDS = (
    # Add authentication backend here
    'weblate.accounts.auth.WeblateUserBackend',
)

INSTALLED_APPS = (
    ...
    'weblate',
    # Install authentication app here
)
```

2.6 Controlo de acesso

Alterado na versão 3.0: Antes do Weblate 3.0, o sistema de privilégios era baseado no Django, mas agora é especificamente construído para Weblate. Se estiver a usar uma versão mais antiga, consulte a documentação para essa versão, as informações aqui não se aplicarão.

O Weblate vem com um sistema de privilégios fino para atribuir permissões ao utilizador para toda a instância ou num escopo limitado.

O sistema de autorização baseado nos grupos e funções, onde as funções de definir um conjunto de permissões, grupos e atribuir-lhes aos utilizadores e traduções, veja *Utilizadores, funções, grupos e permissões* para mais detalhes.

Após a instalação, um conjunto de grupos predefinido é criado e pode usá-los para atribuir funções de utilizadores para toda a instância (ver *Grupos e funções predefinidos*). Além disso, quando acl`está ativado, pode atribuir utilizadores a projetos de tradução específicos. Configurações mais refinadas podem ser alcançadas a usar :ref:`custom-acl.

2.6.1 Configurações comuns

Bloqueando o Weblate

Para bloquear a sua instalação de Weblate completamente, pode usar o `REQUIRE_LOGIN` para forçar os utilizadores a entrar e o `REGISTRATION_OPEN` para impedir novos registos.

Permissões para todo o site

Para gerir permissões para uma instância inteira, basta adicionar utilizadores aos grupos *Utilizadores* (isso é feito por predefinição, a usar o *Atribuições de grupo automáticas*), *Revisores* e *Gestores*. Manter todos os projetos configurados como «Público» (veja *Controlo de acesso por projeto*).

Permissões por projeto

Nota: This feature is not available for projects running the Hosted Libre plan.

Defina os seus projetos a *Protegido* ou *Privado* e gira utilizadores por projeto na interface do Weblate.

Adicionar permissões a idiomas, componentes ou projetos

Nota: This feature is not available for projects running the Hosted Libre plan.

Além disso pode conceder permissões a qualquer utilizador com base no projeto, componente ou conjunto de idiomas. Para conseguir isso, crie um grupo (por exemplo, *tradutores de tcheco*) e configure-o para um determinado recurso. Quaisquer permissões atribuídas serão concedidas aos membros desse grupo para os recursos selecionados.

Isto funcionará muito bem sem configuração adicional, se for usado por permissões de projeto. Para permissões em toda a instância, provavelmente também vai querer remover essas permissões do grupo *Utilizadores* ou alterar a atribuição automática de todos os utilizadores para esse grupo (ver *Atribuições de grupo automáticas*).

Veja também:

Verificação de permissão

2.6.2 Controlo de acesso por projeto

Nota: Ao ativar a ACL, todos os utilizadores são proibidos de acessar qualquer coisa dentro de um determinado projeto, a menos que adicione as permissões para que eles façam exatamente isso.

Nota: This feature is not available for projects running the Hosted Libre plan.

Pode limitar o acesso do utilizador a projetos individuais. Este recurso é ligado por *Controlo de acesso* na configuração de cada projeto. Isto cria automaticamente vários grupos para este projeto, consulte *Grupos predefinidos*.

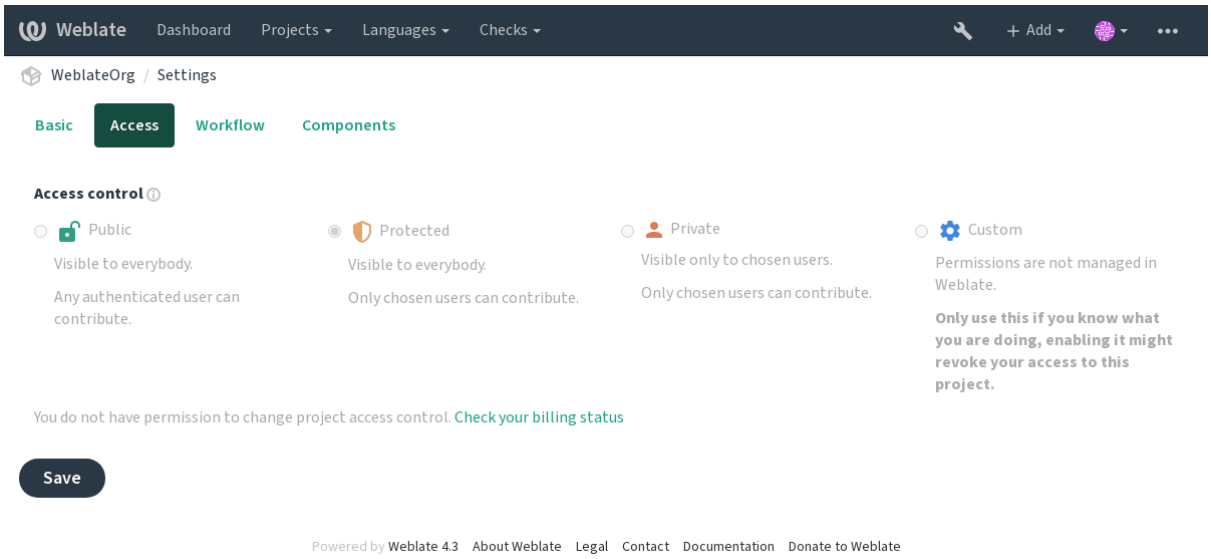
Existem as seguintes opções para *Controle de acesso*:

Pública Publicamente visíveis e traduzíveis

Protegido Visível publicamente, mas traduzível apenas por utilizadores selecionados

Privado Visível e traduzível apenas por utilizadores selecionados

Personalizado Weblate não gere os utilizadores, consulte: *Controle de acesso personalizado*.



Para permitir o acesso a este projeto, tem de adicionar o privilégio diretamente ao utilizador, ou grupo de utilizadores na interface administrativa do Django, ou utilizando a gestão do utilizador na página do projeto, conforme descrito em *Gerindo controle de acesso por projeto*.

Nota: Mesmo com a ACL ativada, alguma informação do resumo irá estar disponível sobre o seu projeto:

- Estatísticas para toda a instância, incluindo as contagens para todos os projetos.
- Resumo do idioma para toda a instância, incluindo contagens para todos os projetos.

2.6.3 Atribuições de grupo automáticas

Pode configurar o Weblate para adicionar automaticamente os utilizadores aos grupos com base no endereço de e-mail dos mesmos. Esta atribuição automática acontece apenas no momento da criação da conta.

Isto pode ser configurado na interface administrativa do Django para cada grupo (na secção *Autenticação*).

Nota: A atribuição automática de grupo aos grupos *Utilizadores* e *Visualizadores* será criada sempre pelo Weblate após as migrações; caso queira desativá-la, basta definir a expressão regular para `^$`, que nunca corresponderá.

2.6.4 Utilizadores, funções, grupos e permissões

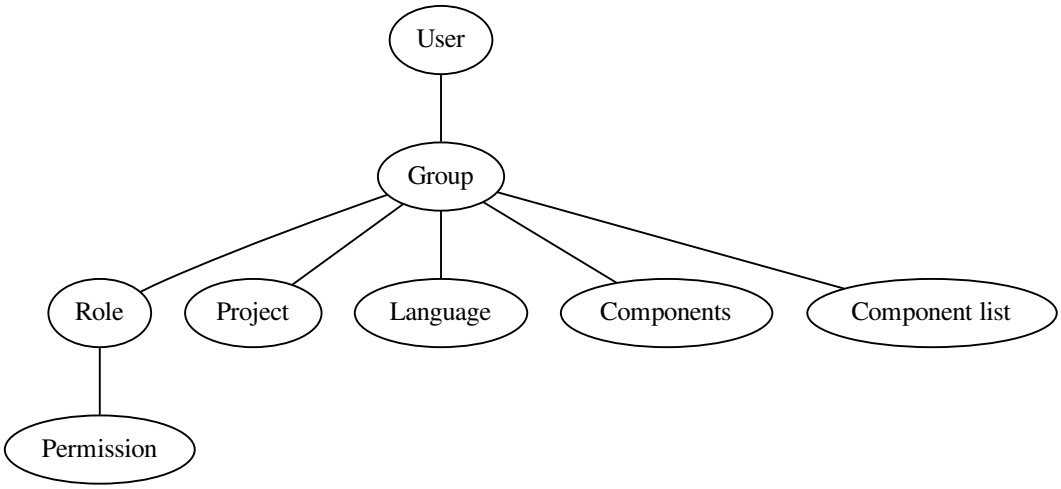
Os modelos de autenticação consistem em vários objetos:

Permissão Permissões individuais definidas pelo Weblate. Não pode atribuir permissões individuais, isto só pode ser efetuado através da atribuição de funções.

Função A função define um conjunto de permissões. Isto permite a reutilização destes conjuntos em vários lugares e facilita a administração.

Utilizador Os utilizadores podem ser membros de vários grupos.

Grupo Grupos associam funções, utilizadores e objetos de autenticação (projetos, idiomas e listas de componentes).



Verificação de permissão

Sempre que uma permissão é verificada para decidir se alguém é capaz de realizar uma determinada ação, a verificação é realizada de acordo com o âmbito e as seguintes verificações são realizadas na ordem:

1. *Lista de componentes* são comparadas com o componente ou o projeto.
2. *Componentes* são comparados com o componente ou o projeto.
3. *Projetos* são comparados com o projeto.

Como pode ver, a concessão de acesso a um componente concede automaticamente o acesso do utilizador a um projeto que contém o componente.

Nota: Apenas será utilizada a primeira regra. Assim, se definir todas da *Lista de Componentes*, *Componentes* e *Projeto*, apenas será aplicada a *Lista de componentes*.

Uma etapa adicional é executada se estiver a verificar a permissão para a tradução:

4. *Languages* are matched against the scope of translations if set, if not set, this does not match any language.

Dica: Pode utilizar *Seleção de Idioma* ou *Seleção de Projeto* para automatizar a inclusão de todos os idiomas ou projetos.

Verificar acesso a um projeto

Um utilizador tem que ser um membro de um grupo vinculado ao projeto ou qualquer componente dentro dele. Apenas a adesão é suficiente, permissões específicas não são necessárias para acessar um projeto (isso é usado no grupo predefinido *Visualizadores*, consulte *Grupos e funções predefinidos*).

Verificar acesso a um componente

Um utilizador pode acessar o componente irrestrito assim que puder acessar o projeto de contenção. Com o *Restricted access* ativado, o acesso ao componente requer permissão explícita no componente (ou que contenha lista de componentes).

2.6.5 Gerir utilizadores e grupos

Todos os utilizadores e grupos podem ser geridos a usar-se a interface administrativa do Django, disponível na URL `/admin/`.

Gerindo controle de acesso por projeto

Nota: Este recurso só funciona para projetos controlados por ACL, veja *Controlo de acesso por projeto*.

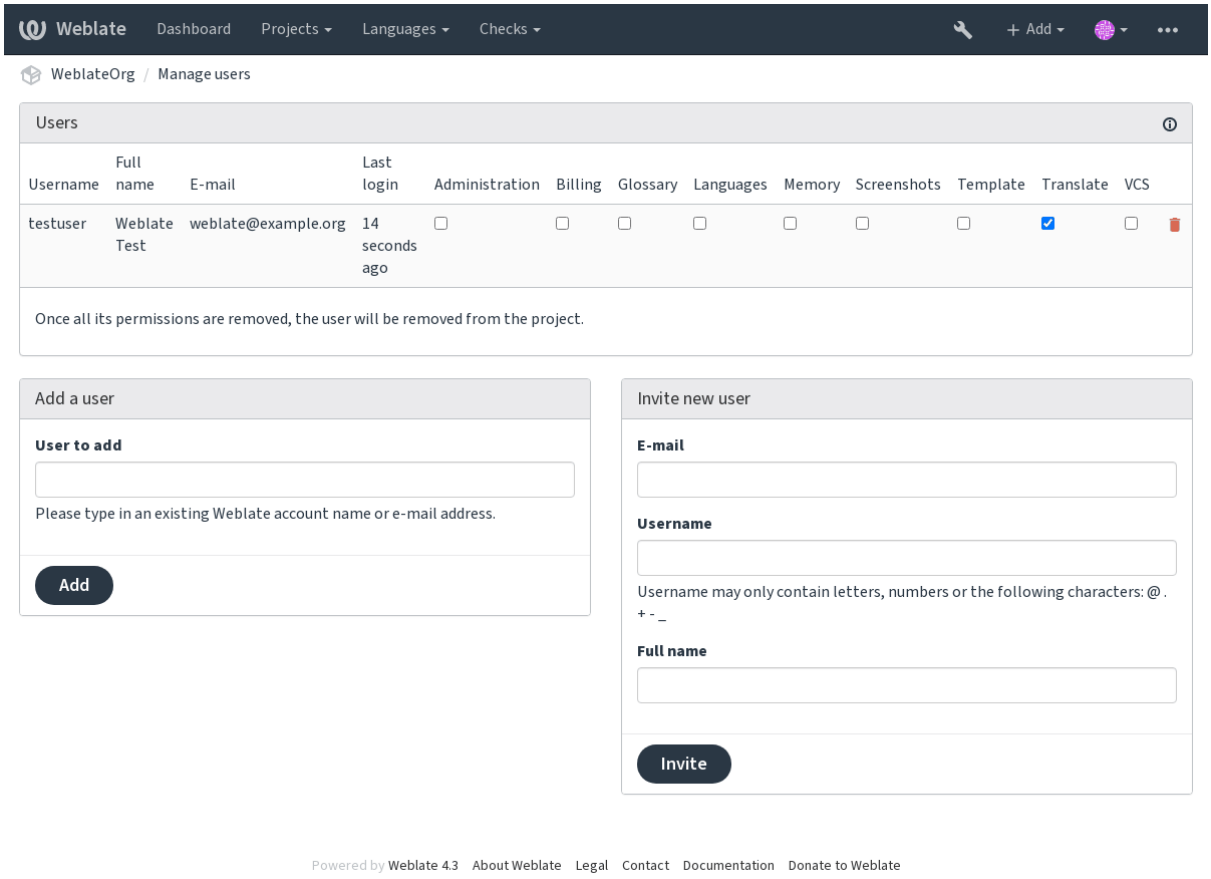
Os utilizadores com o privilégio *Pode gerir regras ACL para um projeto* (veja a *Controlo de acesso*) também pode gerir utilizadores em projetos de controle de acesso ativado através da página do projeto. A interface permite que:

- Adicionar utilizadores existentes ao projeto
- Convidar novos utilizadores ao projeto
- Alterar permissões dos utilizadores
- Revogar acesso dos utilizadores

Novo na versão 3.11.

- Reenvie convites de e-mail do utilizador, a invalidar todos convitesanteriormente enviados

A gestão de utilizadores está disponível no menu *Gerir* de um projeto:



Veja também:

Controlo de acesso por projeto

Grupos predefinidos

Weblate vem com um conjunto predefinido de grupos para um projeto, onde pode atribuir utilizadores.

`Administration`
: Tem todas as permissões disponíveis no projeto.

`Glossary`
: Pode gerir glossário (adicionar ou remover entradas ou enviar).

`Languages`
: Pode gerir idiomas traduzidos - adicionar ou remover traduções.

`Screenshots`
: Pode gerir capturas de ecrã - adicioná-las ou removê-las e associá-las a cadeias fonte.

`Sources`
: Can edit source strings in *Componentes monolínguas* and source string info.

`Translate`
: Pode traduzir o projeto e enviar traduções feitas offline.

`VCS`
: Pode gerir VCS e acessar o repositório exportado.

`Review`
: Pode aprovar traduções durante a revisão.

`Billing`
: Pode acessar informações de faturamento (consulte *Faturação*).

2.6.6 Controle de acesso personalizado

Ao escolher *Personalizado* como *Controlo de acesso*, o Weblate deixará de gerir o acesso para um determinado projeto e pode configurar regras personalizadas na interface administrativa do Django. Isto pode ser usado para definir um controlo de acesso mais complexo ou configurar uma política de acesso compartilhada com todos os projetos numa única instância do Weblate. Se quiser ativar isto para todos os projetos por predefinição, configure `DEFAULT_ACCESS_CONTROL`.

Aviso: Ao ativar isto, o Weblate removerá todos os *Controlo de acesso por projeto* que ele criou para este projeto. Se estiver a fazer isto sem permissão administrativa da instância, perderá instantaneamente o seu acesso para gerir o projeto.

2.6.7 Grupos e funções predefinidos

Lista de privilégios

Faturamento (consulte *Faturação*) Visualizar informações de faturamento [*Administração*, *Faturamento*]

Alterações Descarrgar alterações [*Administração*]

Comentários Publicar comentário [*Administração*, *Editar fonte*, *Utiliyador avançado*, *Revisar cadeias*, *Traduzir*]

Apagar comentário [*Administração*]

Componente Editar configurações do componente [*Administração*]

Bloquear componente, impedindo traduções [*Administração*]

Glossário Adicionar entrada do glossário [*Administração*, *Gerir glossário*, *Utilizador avançado*]

Editar entrada do glossário [*Administração*, *Gerir glossário*, *Utilizador avançado*]

Apagar entrada do glossário [*Administração*, *Gerir glossário*, *Utilizador avançado*]

Enviar entradas do glossário [*Administração*, *Gerir glossário*, *Utilizador avançado*]

Sugestões automáticas Usar sugestões automáticas [*Administração*, *Utilizador avançado*]

Projetos Editar configurações do projeto [*Administração*]

Gerir acesso do projeto [*Administração*]

Relatórios Descarragar relatórios [*Administração*]

Capturas de ecrã Adicionar captura de ecrã [*Administração*, *Gerir capturas de ecrã*]

Editar captura de ecrã [*Administração*, *Gerir capturas de ecrã*]

Apagar captura de ecrã [*Administração*, *Gerir capturas de ecrã*]

Cadeias fonte Editar informações de cadeias fonte [*Administração*, *Editar fonte*]

Cadeias Adicionar novas cadeias [*Administração*]

Ignorar verificações com falha [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

Editar cadeias [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

Revisar cadeias [*Administração*, *Revisar cadeias*]

Editar cadeias quando as sugestões são forçadas [*Administração*, *Revisar cadeias*]

Editar cadeias fonte [*Administração*, *Editar fonte*, *Utilizador avançado*]

Sugestões Aceitar sugestões [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

Adicionar sugestões [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

Apagar sugestões [*Administração*]

Votar em sugestões [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

Traduções Iniciar nova tradução [*Administração*, *Gerir idiomas*, *Utilizador avançado*]

Efetuar tradução automática [*Administração*, *Gerir idiomas*]

Apagar traduções existentes [*Administração*, *Gerir idiomas*]

Iniciar tradução para um novo idioma [*Administração*, *Gerir idiomas*]

Envios Definir o autor da tradução enviada [*Administração*]

Sobrescrever cadeias existentes com um envio [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

Enviar cadeias de tradução [*Administração*, *Editar fonte*, *Utilizador avançado*, *Revisar cadeias*, *Traduzir*]

VCS Acessar o repositório interno [*Acessar repositório*, *Administração*, *Gerir repositório*, *Utilizador avançado*]

Submeter as alterações ao repositório interno [*Administração*, *Geriar repositório*]

Empurrar alterações do repositório interno [*Administração*, *Gerir repositório*]

Redefinir as alterações no repositório interno [*Administração*, *Gerir repositório*]

Ver o local do repositório upstream [*Acessar repositório*, *Administração*, *Gerir repositório*, *Utilizador avançado*]

Atualizar o repositório interno [*Administração*, *Gerir repositório*]

Privilégios para todo o site Utilizar a interface de gestão

Adicionar novos projetos

Adicionar definições de idioma

Gerir definições de idioma

Gerir grupos

Gerir utilizadores

Gerir funções

Gerir anúncios

Gerir a memória de tradução

Gerir as listas de componentes

Nota: Os privilégios para todo o site não são concedidos a nenhuma função predfinida. Eles são poderosos e muito próximos do estado de superutilizador — a maioria deles afetam todos os projetos da sua instalação do Weblate.

Lista de grupos

Os grupos seguintes são criados após a instalação (ou após a execução de `setupgroups`):

Convidados Define permissões para utilizadores não autenticados.

Este grupo contém apenas utilizadores anônimos (consulte `ANONYMOUS_USER_NAME`).

Pode remover funções deste grupo para limitar as permissões para utilizadores não autenticados.

Funções predefinidas: *Adicionar sugestão*, *Acessar repositório*

Visualizadores Esta função garante a visibilidade de projetos públicos para todos os utilizadores. Por predefinição, todos os utilizadores são membros deste grupo.

Por predefinição, todos os utilizadores são membros deste grupo, a usar *Atribuições de grupo automáticas*.

Funções predefinidas: nenhuma

Utilizadores Grupo predefinido para todos os utilizadores.

Por predefinição, todos os utilizadores são membros deste grupo a usar *Atribuições de grupo automáticas*.

Funções predefinidas: *Utilizador avançado*

Revisores Grupo para revisores (consulte *Fluxos de trabalho de tradução*).

Funções predefinidas: *Revisar cadeias*

Gestores Grupo pra administradores.

Funções predefinidas: *Administração*

Aviso: Nunca remova os grupos e utilizadores predefinidos do Weblate, pois isso pode levar a problemas inesperados. Se não quiser usar esses recursos, basta remover todos os privilégios deles.

2.7 Projetos de tradução

2.7.1 Translation organization

Weblate organizes translatable VCS content of project/components into a tree-like structure.

- The bottom level object is *Project configuration*, which should hold all translations belonging together (for example translation of an application in several versions and/or accompanying documentation).
- On the level above, *Component configuration*, which is actually the component to translate, you define the VCS repository to use, and the mask of files to translate.
- Above *Component configuration* there are individual translations, handled automatically by Weblate as translation files (which match the mask defined in *Component configuration*) appear in the VCS repository.

Weblate supports a wide range of translation formats (both bilingual and monolingual ones) supported by Translate Toolkit, see *Formatos de ficheiros suportados*.

Nota: You can share cloned VCS repositories using *Weblate internal URLs*. Using this feature is highly recommended when you have many components sharing the same VCS. It improves performance and decreases required disk space.

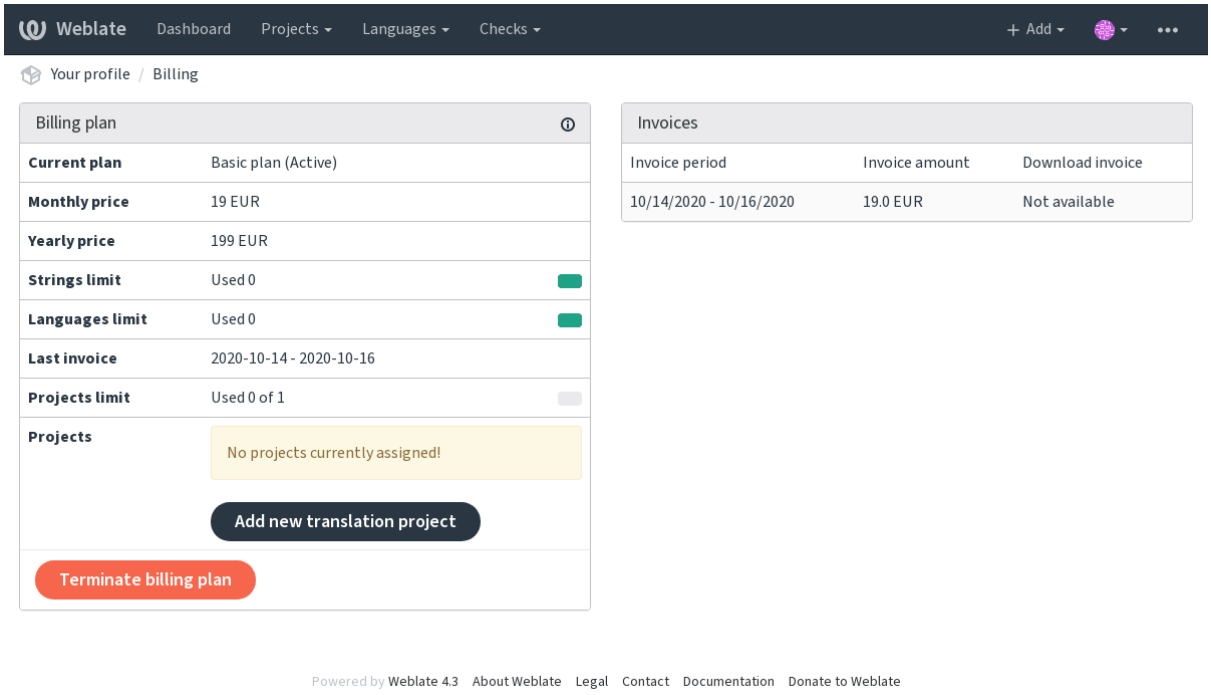
2.7.2 Adding translation projects and components

Alterado na versão 3.2: An interface for adding projects and components is included, and you no longer have to use *A interface administrativa do Django*.

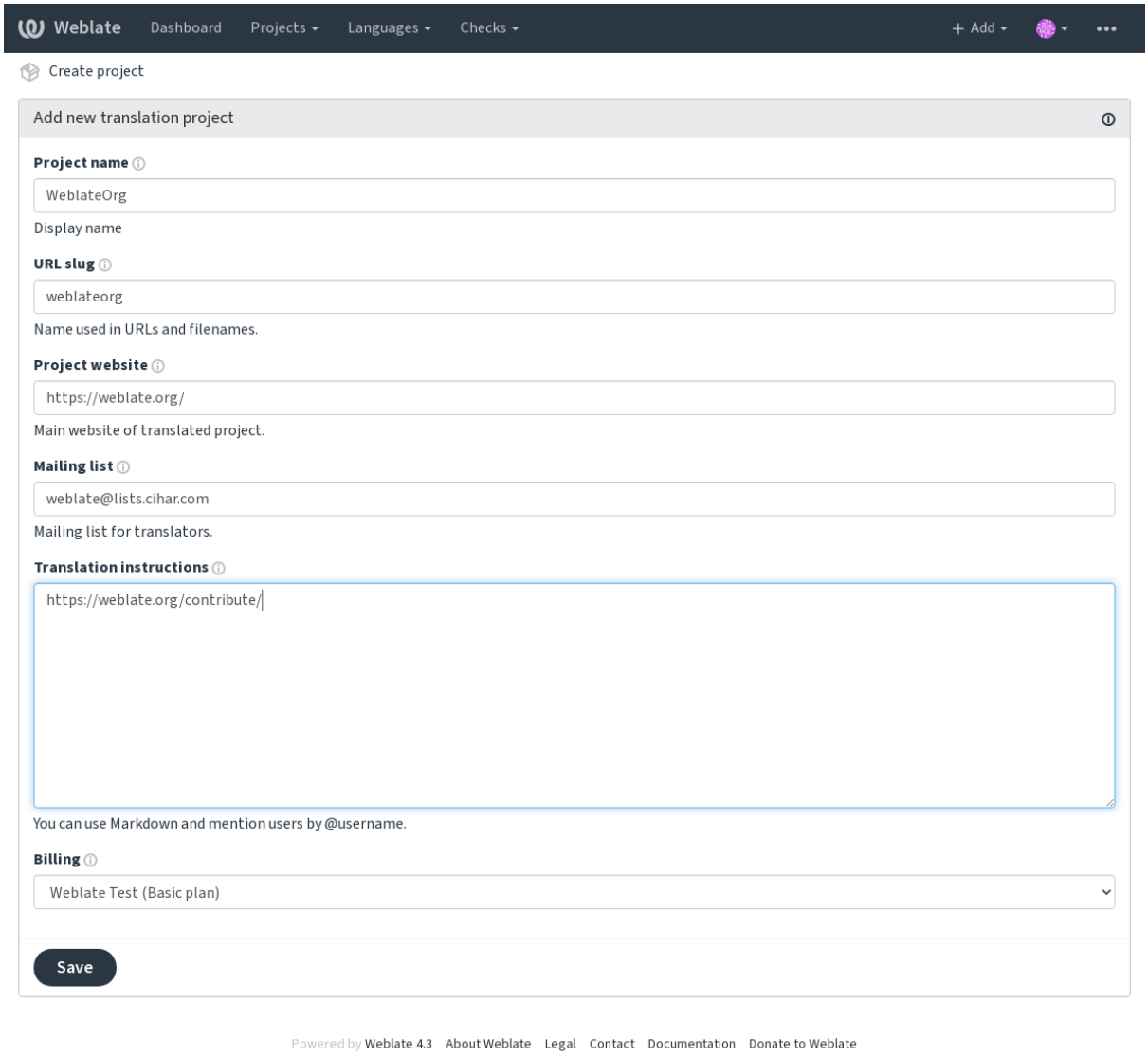
Alterado na versão 3.4: The process of adding components is now multi staged, with automated discovery of most parameters.

Based on your permissions, new translation projects and components can be created. It is always permitted for users with the *Add new projects* permission, and if your instance uses billing (e.g. like https://hosted.weblate.org/ see *Faturação*), you can also create those based on your plans allowance from the user account that manages billing.

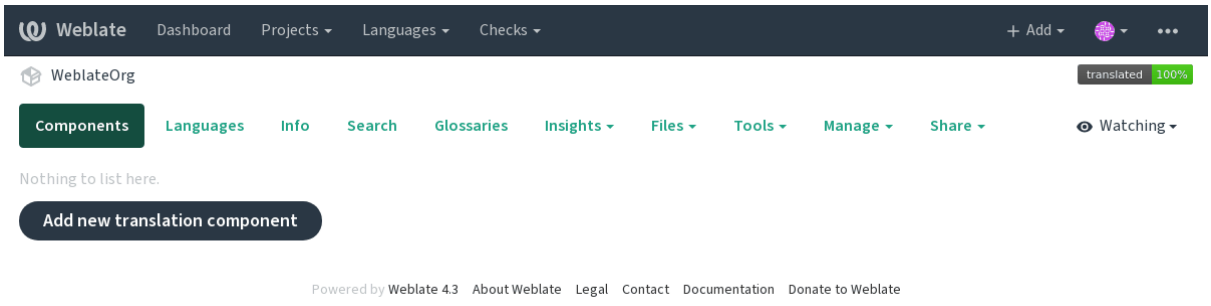
You can view your current billing plan on a separate page:



The project creation can be initiated from there, or using the menu in the navigation bar, filling in basic info about the translation project to complete addition of it:



After creating the project, you are taken directly to the project page:



Creating a new translation component can be initiated via a single click there. The process of creating a component is multi-staged and automatically detects most translation parameters. There are several approaches to creating component:

Do controle de versão Creates component from remote version control repository.

Do componente existente Creates additional component to existing one by choosing different files.

Ramo adicional Creates additional component to existing one, just for different branch.

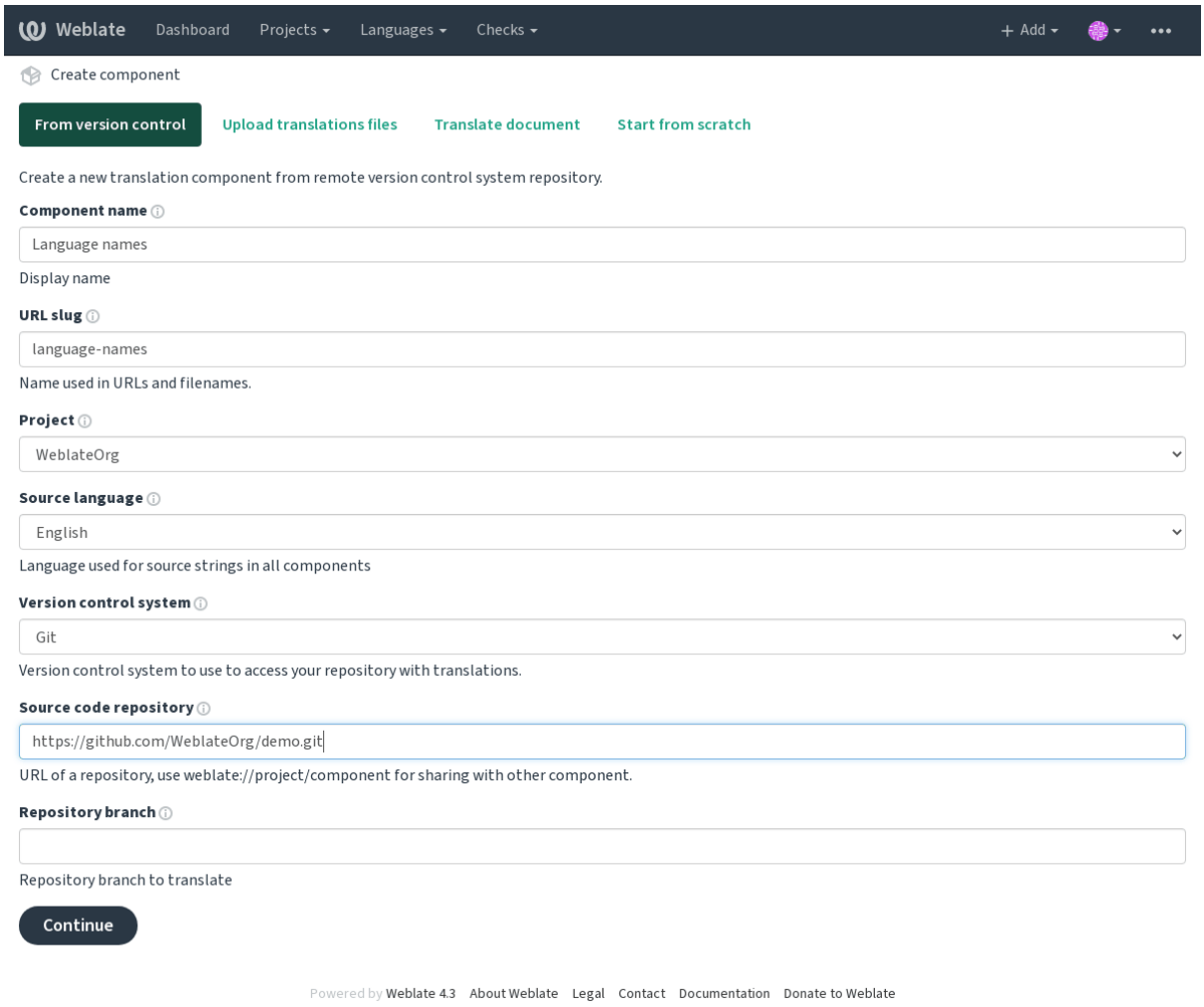
Enviar ficheiros de tradução Upload translation files to Weblate in case you do not have version control or do not want to integrate it with Weblate. You can later update the content using the web interface or API.

Traduzir documento Upload single document and translate that.

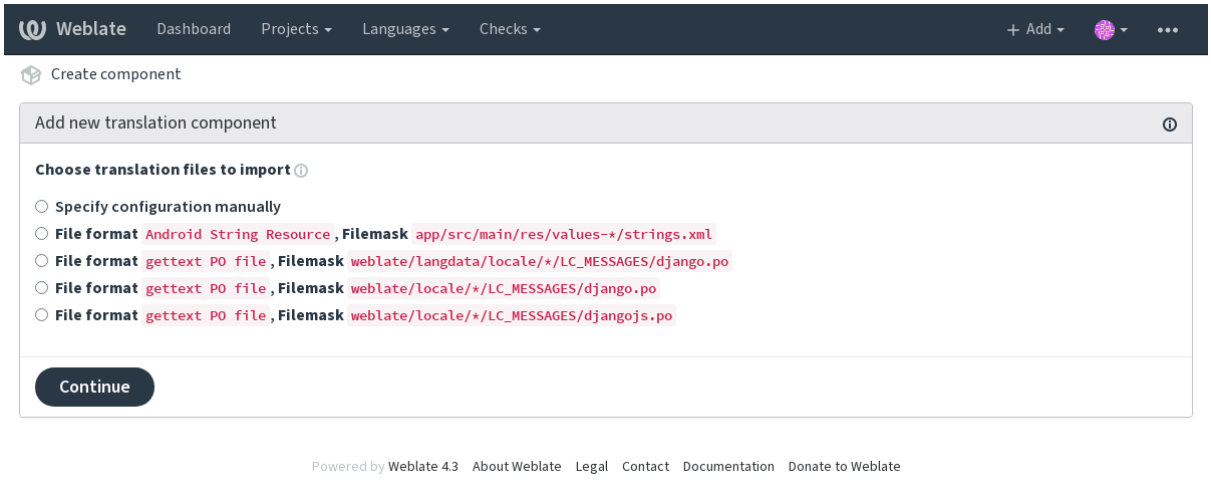
Começar do zero Create blank translation project and add strings manually.

Once you have existing translation components, you can also easily add new ones for additional files or branches using same repository.

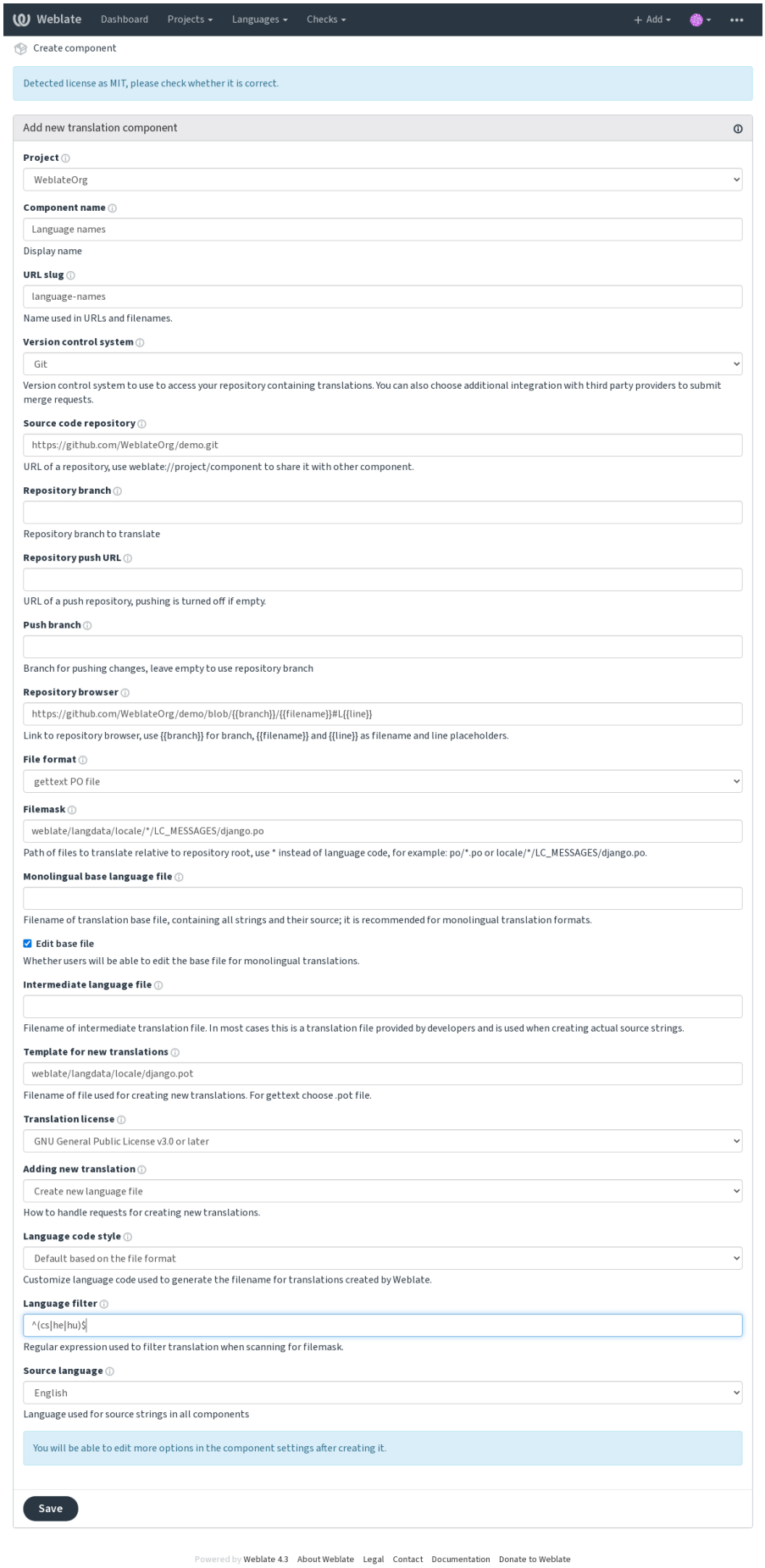
First you need to fill in name and repository location:



On the next page, you are presented with a list of discovered translatable resources:



As a last step, you review the translation component info and fill in optional details:


Weblate
Dashboard
Projects
Languages
Checks
Add
Create component
Detected license as MIT, please check whether it is correct.
Add new translation component
Project
WeblateOrg
Component name
Language names
Display name
URL slug
language-names
Name used in URLs and filenames.
Version control system
Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.
Source code repository
https://github.com/WeblateOrg/demo.git
URL of a repository, use weblate://project/component to share it with other component.
Repository branch
Repository branch to translate
Repository push URL
URL of a push repository, pushing is turned off if empty.
Push branch
Branch for pushing changes, leave empty to use repository branch
Repository browser
https://github.com/WeblateOrg/demo/blob/{{branch}}/{{filename}}#L{{line}}
Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders.
File format
gettext PO file
Filemask
weblate/langdata/locale/*/LC_MESSAGES/django.po
Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/django.po.
Monolingual base language file
Filename of translation base file, containing all strings and their source; it is recommended for monolingual translation formats.
Edit base file
Whether users will be able to edit the base file for monolingual translations.
Intermediate language file
Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.
Template for new translations
weblate/langdata/locale/django.pot
Filename of file used for creating new translations. For gettext choose .pot file.
Translation license
GNU General Public License v3.0 or later
Adding new translation
Create new language file
How to handle requests for creating new translations.
Language code style
Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.
Language filter
^(cs|he|hu)$
Regular expression used to filter translation when scanning for filemask.
Source language
English
Language used for source strings in all components
You will be able to edit more options in the component settings after creating it.
Save
Powered by Weblate 4.3
About Weblate
Legal
Contact
Documentation
Donate to Weblate

Veja também:

A interface administrativa do Django, *Project configuration*, *Component configuration*

2.7.3 Project configuration

Create a translation project and then add a new component for translation in it. The project is like a shelf, in which real translations are stacked. All components in the same project share suggestions and their dictionary; the translations are also automatically propagated through all components in a single project (unless turned off in the component configuration), see Memory Management.

These basic attributes set up and inform translators of a project:

Nome do projeto

Verbose project name, used to display the project name.

Project slug

Project name suitable for URLs.

Site da Web do Projeto

URL where translators can find more info about the project.

Lista de correio

Mailing list where translators can discuss or comment translations.

Instruções para tradução

URL to more site with more detailed instructions for translators.

Set Language-Team header

Whether Weblate should manage the `Language-Team` header (this is a *GNU gettext* only feature right now).

Utilizar memória de tradução partilhada

Whether to use shared translation memory, see *Memória de tradução compartilhada* for more details.

Contribuir à memória de tradução compartilhada

Whether to contribute to shared translation memory, see *Memória de tradução compartilhada* for more details.

Controlo de acesso

Configure per project access control, see *Controlo de acesso por projeto* for more details.

Default value can be changed by `DEFAULT_ACCESS_CONTROL`.

Activar revisões

Enable review workflow for translations, see *Revisores dedicados*.

Ativar revisões de fontes

Enable review workflow for source strings, see *Revisões de cadeias fonte*.

Ativar hooks

Whether unauthenticated *Hooks de notificação* are to be used for this repository.

Veja também:

Ficheiro de idioma intermédio, *Portal de qualidade para cadeias fonte*, *Bilingual and monolingual formats*, *Language definitions*

Aliases do idioma

Define language codes mapping when importing translations into Weblate. Use this when language codes are inconsistent in your repositories and you want to get a consistent view in Weblate.

The typical use case might be mapping American English to English: `en_US:en`

Multiple mappings to be separated by comma: `en_GB:en,en_US:en`

Dica: The language codes are mapped when matching the translation files and the matches are case sensitive, so make sure you use the source language codes in same form as used in the filenames.

Veja também:

Parsing language codes

2.7.4 Component configuration

A component is a grouping of something for translation. You enter a VCS repository location and file mask for which files you want translated, and Weblate automatically fetches from this VCS, and finds all matching translatable files.

You can find some examples of typical configurations in the *Formatos de ficheiros suportados*.

Nota: It is recommended to keep translation components to a reasonable size - split the translation by anything that makes sense in your case (individual apps or addons, book chapters or websites).

Weblate easily handles translations with 10000s of strings, but it is harder to split work and coordinate among translators with such large translation components.

Should the language definition for a translation be missing, an empty definition is created and named as «cs_CZ (generated)». You should adjust the definition and report this back to the Weblate authors, so that the missing languages can be included in next release.

The component contains all important parameters for working with the VCS, and for getting translations out of it:

Nome do componente

Verbose component name, used to display the component name.

Component slug

Component name suitable for URLs.

Component project

Project configuration where the component belongs.

Sistema de controlo de versões

VCS to use, see *Integração de controlo de versões* for details.

Repositório do código-fonte

VCS repository used to pull changes.

Veja também:

See *Accessing repositories* for more details on specifying URLs.

Dica: This can either be a real VCS URL or `weblate://project/component` indicating that the repository should be shared with another component. See *Weblate internal URLs* for more details.

URL de submissão do repositório

Repository URL used for pushing. This setting is used only for *Git* and *Mercurial* and push support is turned off for these when this is empty.

Veja também:

See *Accessing repositories* for more details on how to specify a repository URL and *Fazendo push das alterações do Weblate* for more details on pushing changes from Weblate.

Navegador do repositório

URL of repository browser used to display source files (location of used messages). When empty, no such links will be generated. You can use *Template markup*.

For example on GitHub, use something like: `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename}}#L{{line}}`

In case your paths are relative to different folder, you might want to strip leading directory by `parentdir` filter (see *Template markup*): `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}`

URL do repositório exportado

URL where changes made by Weblate are exported. This is important when *Tradução contínua* is not used, or when there is a need to manually merge changes. You can use *Git exporter* to automate this for Git repositories.

Ramo do repositório

Which branch to checkout from the VCS, and where to look for translations.

Ramo do push

Branch for pushing changes, leave empty to use *Ramo do repositório*.

Nota: This is currently only supported for Git, GitLab and GitHub, it is ignored for other VCS integrations.

File mask

Mask of files to translate, including path. It should include one «*» replacing language code (see *Language definitions* for info on how this is processed). In case your repository contains more than one translation file (e.g. more gettext domains), you need to create a component for each of them.

For example `po/*.po` or `locale/*/LC_MESSAGES/django.po`.

In case your filename contains special characters such as `[`, `]`, these need to be escaped as `[[]` or `[]]`.

Veja também:

Bilingual and monolingual formats, *What does mean «There are more files for the single language (en)»?*

Ficheiro de idioma base monolingue

Base file containing string definitions for *Componentes monolínguas*.

Veja também:

Bilingual and monolingual formats, *What does mean «There are more files for the single language (en)»?*

Editar ficheiro base

Whether to allow editing the base file for *Componentes monolínguas*.

Ficheiro de idioma intermédio

Intermediate language file for *Componentes monolínguas*. In most cases this is a translation file provided by developers and is used when creating actual source strings.

When set, the source translation is based on this file, but all others are based on *Ficheiro de idioma base monolingue*. In case the string is not translated in source translation, translating to other languages is prohibited. This provides *Portal de qualidade para cadeias fonte*.

Veja também:

Portal de qualidade para cadeias fonte, *Bilingual and monolingual formats*, *What does mean «There are more files for the single language (en)»?*

Modelo para novas traduções

Base file used to generate new translations, e.g. `.pot` file with gettext.

Dica: In many monolingual formats Weblate starts with blank file by default. Use this in case you want to have all strings present with empty value when creating new translation.

Veja também:

Adding new translations, *Adicionar nova tradução*, *Bilingual and monolingual formats*, *What does mean «There are more files for the single language (en)»?*

Formato de ficheiro

Translation file format, see also *Formatos de ficheiros suportados*.

Endereço para reportar erros na cadeia fonte

Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.

Permitir propagação da tradução

You can turn off propagation of translations to this component from other components within same project. This really depends on what you are translating, sometimes it's desirable to have make use of a translation more than once.

It's usually a good idea to turn this off for monolingual translations, unless you are using the same IDs across the whole project.

Default value can be changed by *DEFAULT_TRANSLATION_PROPAGATION*.

Ativar sugestões

Whether translation suggestions are accepted for this component.

Votação de sugestão

Turns on votecasting for suggestions, see *Votação de sugestão*.

Aceitar sugestões automaticamente

Automatically accept voted suggestions, see *Votação de sugestão*.

Marcadores de tradução

Customization of quality checks and other Weblate behavior, see *Personalizar o comportamento*.

Verificações impostas

List of checks which can not be ignored, see *Forçar verificações*.

Licença da tradução

License of the translation (does not need to be the same as the source code license).

Acordo de contribuidor

Acordo do utilizador que tem de ser aprovado antes do utilizador poder traduzir este componente.

Adicionar nova tradução

How to handle requests for creation of new languages. Available options:

Contactar gestores User can select desired language and the project maintainers will receive a notification about this. It is up to them to add (or not) the language to the repository.

Apontar para URL com instruções de tradução User is presented a link to page which describes process of starting new translations. Use this in case more formal process is desired (for example forming a team of people before starting actual translation).

Criar novo ficheiro de idioma User can select language and Weblate automatically creates the file for it and translation can begin.

Desativar adição de novas traduções There will be no option for user to start new translation.

Veja também:

Adding new translations.

Estilo de código de idioma

Customize language code used to generate the filename for translations created by Weblate, see *Adding new translations* for more details.

Estilo de união

You can configure how updates from the upstream repository are handled. This might not be supported for some VCSs. See *Mesclar ou rebase* for more details.

Default value can be changed by `DEFAULT_MERGE_STYLE`.

Commit, add, delete, merge and addon messages

Message used when committing a translation, see *Template markup*.

Default value can be changed by *DEFAULT_ADD_MESSAGE*, *DEFAULT_ADDON_MESSAGE*, *DEFAULT_COMMIT_MESSAGE*, *DEFAULT_DELETE_MESSAGE*, *DEFAULT_MERGE_MESSAGE*.

Nome do publicador

Name of the committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported.

Default value can be changed by *DEFAULT_COMMITER_NAME*.

E-mail do publicador

Email of committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported. The default value can be changed in *DEFAULT_COMMITER_EMAIL*.

Enviar ao submeter

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its internal repository (see *Commits adiados*). To actually enable pushing *Repository push URL* has to be configured as well.

Idade das alterações a fazer commit

Sets how old changes (in hours) are to get before they are committed by background task or *commit_pending* management command. All changes in a component are committed once there is at least one older than this period.

Default value can be changed by *COMMIT_PENDING_HOURS*.

Bloquear com erro

Enables locking the component on repository error (failed pull, push or merge). Locking in this situation avoids adding another conflict which would have to be resolved manually.

The component will be automatically unlocked once there are no repository errors left.

Idioma fonte

Language used for source strings. Change this if you are translating from something else than English.

Dica: In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, you might want to choose *English (Developer)* as a source language. To avoid conflict between name of the source language and existing translation.

For monolingual translations, you can use intermediate translation in this case, see *Ficheiro de idioma intermédio*.

Filtro de idioma

Regular expression used to filter the translation when scanning for filemask. This can be used to limit the list of languages managed by Weblate.

Nota: You need to list language codes as they appear in the filename.

Some examples of filtering:

Filter description	Expressão regular
Selected languages only	`^(cs|de|es)$`
Exclude languages	`^(?!(it|fr)$).+$`
Exclude non language files	`^(?!(blank)$).+$`
Include all files (default)	`^[^.]+$`

Expressão regular das variantes

Regular expression used to determine the variants of a string, see *String variants*.

Nota: Most of the fields can be edited by project owners or managers, in the Weblate interface.

Veja também:

Does Weblate support other VCSes than Git and Mercurial?, *Translation component alerts*

Prioridade

Componentes de prioridade mais elevada são oferecidos primeiro aos tradutores.

Restricted access

By default the component is visible to anybody who has access to the project, even if the person can not perform any changes in the component. This makes it easier to keep translation consistency within the project.

Enable this in case you want to grant access to this component explicitly - the project level permissions will not apply and you will have to specify component or component list level permission in order to grant access.

Default value can be changed by `DEFAULT_RESTRICTED_COMPONENT`.

Dica: This applies to project managers as well - please make sure you will not loose access to the component after toggling the status.

2.7.5 Template markup

Weblate uses simple markup language in several places where text rendering is needed. It is based on The Django template language, so it can be quite powerful.

Currently it is used in:

- Commit message formatting, see *Component configuration*
- **Several addons**
  - *Descoberta de componentes*
  - *Gerador de estatísticas*
  - *Executar scripts de extensões*

There following variables are available in the component templates:

{{ language_code }} Código do idioma

{{ language_name }} Nome do idioma

{{ component_name }} Nome do componente

{{ component_slug }} Component slug

{{ project_name }} Nome do projeto

{{ project_slug }} Project slug

{{ url }} Translation URL

{{ filename }} Nome do ficheiro de tradução

{{ stats }} Translation stats, this has further attributes, examples below.

{{ stats.all }} Total strings count

{{ stats.fuzzy }} Count of strings needing review

{{ stats.fuzzy_percent }} Percent of strings needing review

{{ stats.translated }} Translated strings count

{{ stats.translated_percent }} Translated strings percent

{{ stats.allchecks }} Number of strings with failing checks

{{ stats.allchecks_percent }} Percent of strings with failing checks

{{ author }} Author of current commit, available only in the commit scope.

{{ addon_name }} Name of currently executed addon, available only in the addon commit message.

The following variables are available in the repository browser or editor templates:

{{branch}} current branch

{{line}} line in file

{{filename}} filename, you can also strip leading parts using the `parentdir` filter, for example `{{filename|parentdir}}`

You can combine them with filters:

```
{{ component|title }}
```

You can use conditions:

```
{% if stats.translated_percent > 80 %}Well translated!{% endif %}
```

There is additional tag available for replacing characters:

```
{% replace component "-" " " %}
```

You can combine it with filters:

```
{% replace component|capfirst "-" " " %}
```

There are also additional filter to manipulate with filenames:

```
Directory of a file: {{ filename|dirname }}
File without extension: {{ filename|stripext }}
File in parent dir: {{ filename|parentdir }}
It can be used multiple times:  {{ filename|parentdir|parentdir }}
```

…and other Django template features.

2.7.6 Importing speed

Fetching VCS repository and importing translations to Weblate can be a lengthy process, depending on size of your translations. Here are some tips:

Optimize configuration

The default configuration is useful for testing and debugging Weblate, while for a production setup, you should do some adjustments. Many of them have quite a big impact on performance. Please check *Configuração de produção* for more details, especially:

- Configure Celery for executing background tasks (see *Tarefas de fundo a usar o Celery*)
- *Ativar o cache*
- *Usar um poderoso mecanismo de banco de dados*
- *Desativar o modo de depuração*

Check resource limits

If you are importing huge translations or repositories, you might be hit by resource limitations of your server.

- Check the amount of free memory, having translation files cached by the operating system will greatly improve performance.
- Disk operations might be bottleneck if there is a lot of strings to process—the disk is pushed by both Weblate and the database.
- Additional CPU cores might help improve performance of background tasks (see *Tarefas de fundo a usar o Celery*).

Disable unneeded checks

Some quality checks can be quite expensive, and if not needed, can save you some time during import if omitted. See `CHECK_LIST` for info on configuration.

2.7.7 Automatic creation of components

In case your project has dozen of translation files (e.g. for different gettext domains, or parts of Android apps), you might want to import them automatically. This can either be achieved from the command line by using *import_project* or *import_json*, or by installing the *Descoberta de componentes* addon.

To use the addon, you first need to create a component for one translation file (choose the one that is the least likely to be renamed or removed in future), and install the addon on this component.

For the management commands, you need to create a project which will contain all components and then run *import_project* or *import_json*.

Veja também:

Management commands, *Descoberta de componentes*

2.8 Language definitions

To present different translations properly, info about language name, text direction, plural definitions and language code is needed.

2.8.1 Parsing language codes

While parsing translations, Weblate attempts to map language code (usually the ISO 639-1 one) to any existing language object.

You can further adjust this mapping at project level by *Aliases do idioma*.

If no exact match can be found, an attempt will be made to best fit it into an existing language (e.g. ignoring the default country code for a given language—choosing `cs` instead of `cs_CZ`).

Should that also fail, a new language definition will be created using the defaults (left to right text direction, one plural) and naming of the language as *xx_XX (generated)*. You might want to change this in the admin interface later, (see *Changing language definitions*) and report it to the issue tracker (see *Contribuir ao Weblate*).

Dica: In case you see something unwanted as a language, you might want to adjust *Filtro de idioma* to ignore such file when parsing translations.

2.8.2 Changing language definitions

You can change language definitions in the languages interface (`/languages/` URL).

While editing, make sure all fields are correct (especially plurals and text direction), otherwise translators will be unable to properly edit those translations.

2.8.3 Built-in language definitions

Definitions for more than 550 languages are included in Weblate and the list is extended in every release. Whenever Weblate is upgraded (more specifically whenever **`weblate migrate`** is executed, see *Generic upgrade instructions*) the database of languages is updated to include all language definitions shipped in Weblate.

This feature can be disable using *UPDATE_LANGUAGES*. You can also enforce updating the database to match Weblate built-in data using *setuplang*.

2.8.4 Language definitions

Each language consists of following fields:

Código do idioma

Code identifying the language. Weblate prefers two letter codes as defined by ISO 639-1, but uses ISO 639-2 or ISO 639-3 codes for languages that do not have two letter code. It can also support extended codes as defined by BCP 47.

Veja também:

Parsing language codes

Nome do idioma

Visible name of the language. The language names included in Weblate are also being localized depending on user interface language.

Direção do texto

Determines whether language is written right to left or left to right. This property is autodetected correctly for most of the languages.

Plural number

Number of plurals used in the language.

Fórmula de plural

Gettext compatible plural formula used to determine which plural form is used for given count.

Veja também:

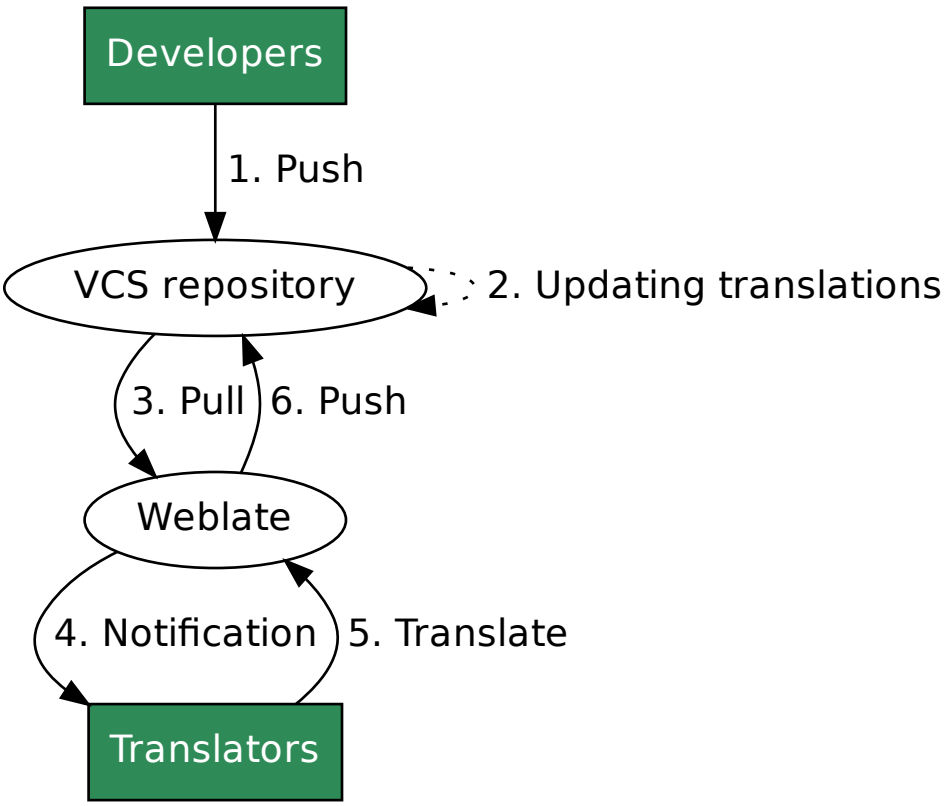
Plurais, GNU gettext utilities: Plural forms, Language Plural Rules by the Unicode Consortium

2.9 Tradução contínua

Há infraestrutura em vigor para que a sua tradução acompanhe o desenvolvimento de perto . Desta forma, os tradutores podem trabalhar em traduções o tempo todo, em vez de trabalhar com uma enorme quantidade de texto novo pouco antes do lançamento.

O processo é o seguinte:

1. Os programadores fazem alterações e fazem *push* delas so repositório VCS.
2. Opcionalmente, os ficheiros de tradução são atualizados (isso depende do formato do ficheiro, consulte *Why does Weblate still show old translation strings when I've updated the template?*).
3. O Weblate faz o *pull* das alterações do repositório VCS, consulte *Atualizar repositórios*.
4. Uma vez que o Weblate deteta alterações nas traduções, os tradutores são notificados com base na configurações de assinatura deles.
5. Os tradutores enviam traduções a usar a interface web do Weblate ou enviam alterações feitas offline.
6. Uma vez que os tradutores acabaram, o Weblate faz commit das alterações no repositório local (veja *Commits adiados*) e faz *push* delas de volta se tiver permissões para fazê-lo (veja *Fazendo push das alterações do Weblate*).



2.9.1 Atualizar repositórios

Deve configurar alguma maneira de atualizar repositórios de backend a partir da fonte dele.

- Use *Hooks de notificação* para integrar com a maioria dos serviços comuns de hospedagem de código
- Acione manualmente a atualização na gestão do repositório ou a usar API ou *Cliente Weblate*
- Ative `AUTO_UPDATE` para atualizar todos os componentes na sua instância Weblate automaticamente
- Execute `updategit` (com a seleção de um projeto ou *–all* para atualizar tudo)

Sempre que o Weblate atualiza o repositório, as extensões de pós-atualização serão acionadas, consulte *Extensões*.

Evitar conflitos de mesclagem

Os conflitos de mesclagem do Weblate surgem quando o mesmo ficheiro foi alterado tanto no Weblate quanto fora dele. Existem duas abordagens para lidar com isso - evitar edições fora do Weblate ou integrar o Weblate no seu processo de atualização, de modo que descarte alterações antes de atualizar os ficheiros fora do Weblate.

A primeira abordagem é fácil com ficheiros monolingues - pode adicionar novas cadeias no Weblate e deixar a edição completa dos ficheiros lá. Para ficheiros bilíngues, geralmente há algum tipo de processo de extração de mensagens para gerar ficheiros traduzíveis do código-fonte. Em alguns casos, isso pode ser dividido em duas partes - uma para a extração gera modelo (por exemplo, o GETTEXT POT é gerado a usar **`xgettext`**) e depois o processo a seguir mescla-o em traduções reais (os ficheiros GETTEXT PO são atualizados a usar **`msgmerge`**). Pode executar o segundo passo dentro do Weblate e garantirá que todas as alterações pendentes sejam incluídas antes desta operação.

A segunda abordagem pode ser alcançada a usar o API para forçar o Weblate a fazer push de todas as alterações pendentes e bloquear a tradução enquanto está fazendo alterações do seu lado.

O script para fazer atualizações pode ser assim:

```
# Lock Weblate translation
wlc lock
# Push changes from Weblate to upstream repository
wlc push
# Pull changes from upstream repository to your local copy
git pull
# Update translation files, this example is for Django
./manage.py makemessages --keep-pot -a
git commit -m 'Locale updates' -- locale
# Push changes to upstream repository
git push
# Tell Weblate to pull changes (not needed if Weblate follows your repo
# automatically)
wlc pull
# Unlock translations
wlc unlock
```

Se tiver vários componentes compartilhando o mesmo repositório, deve bloqueá-los todos separadamente:

```
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

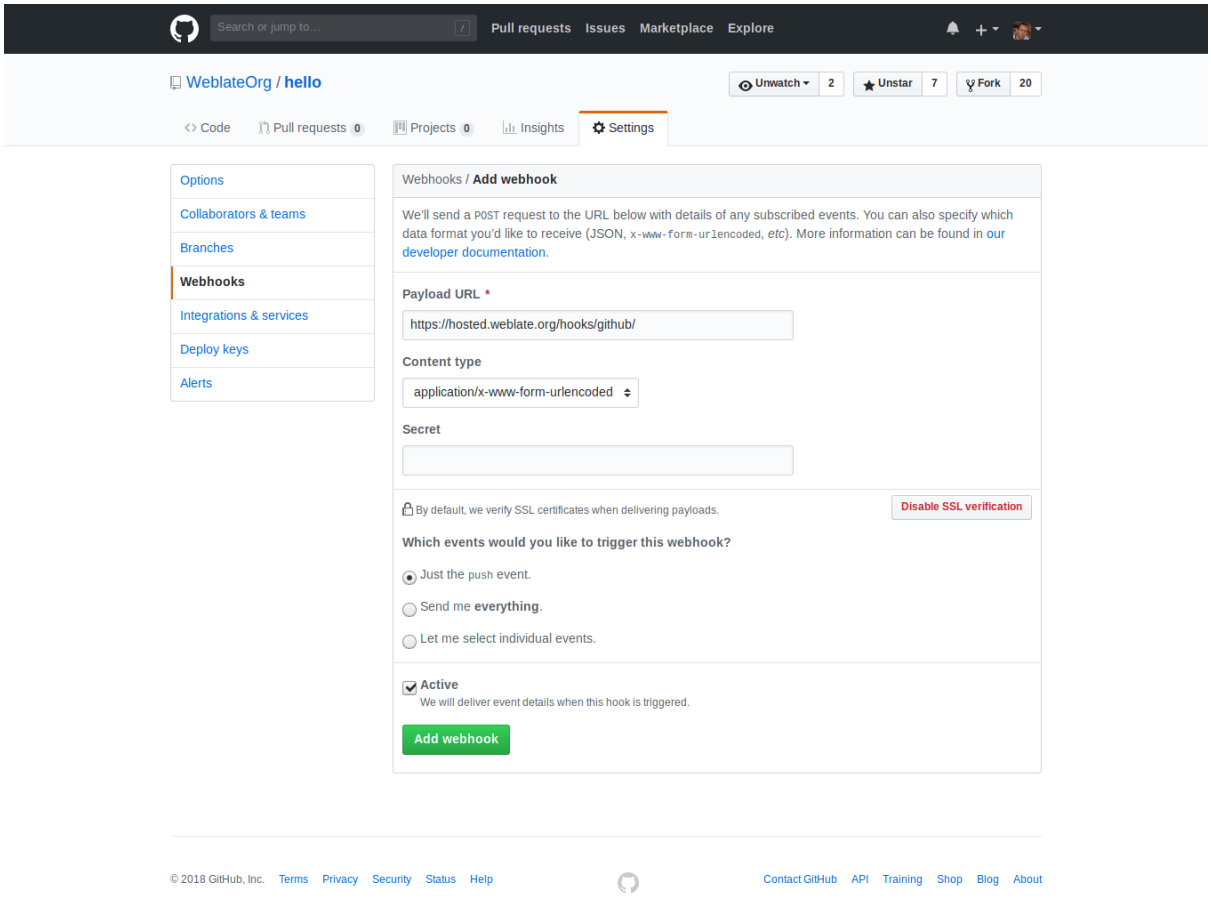
Nota: O exemplo usa *Cliente Weblate*, que precisa de configuração (chaves de API) para ser capaz de controlar o Weblate remotamente. Também pode conseguir isso a usar qualquer cliente HTTP em vez de wlc, por exemplo, curl, ver API.

Receber alterações do GitHub automaticamente

O Weblate vem com suporte nativo ao GitHub.

Se estiver a usar o Hosted Weblate, a abordagem recomendada é instalar o app Weblate, dessa forma terá a configuração correta sem ter que configurar muito. Também pode ser usado para fazer push de mudanças de volta.

Para receber notificações em cada push a um repositório do GitHub, adicione o webhook do Weblate nas configurações do repositório (*Webhooks*) como mostrado na imagem abaixo:



Para a URL de carga útil, anexar `/hooks/github/` à URL do Weblate, por exemplo, para o serviço Hosted Weblate, este é `https://hosted.weblate.org/hooks/github/`.

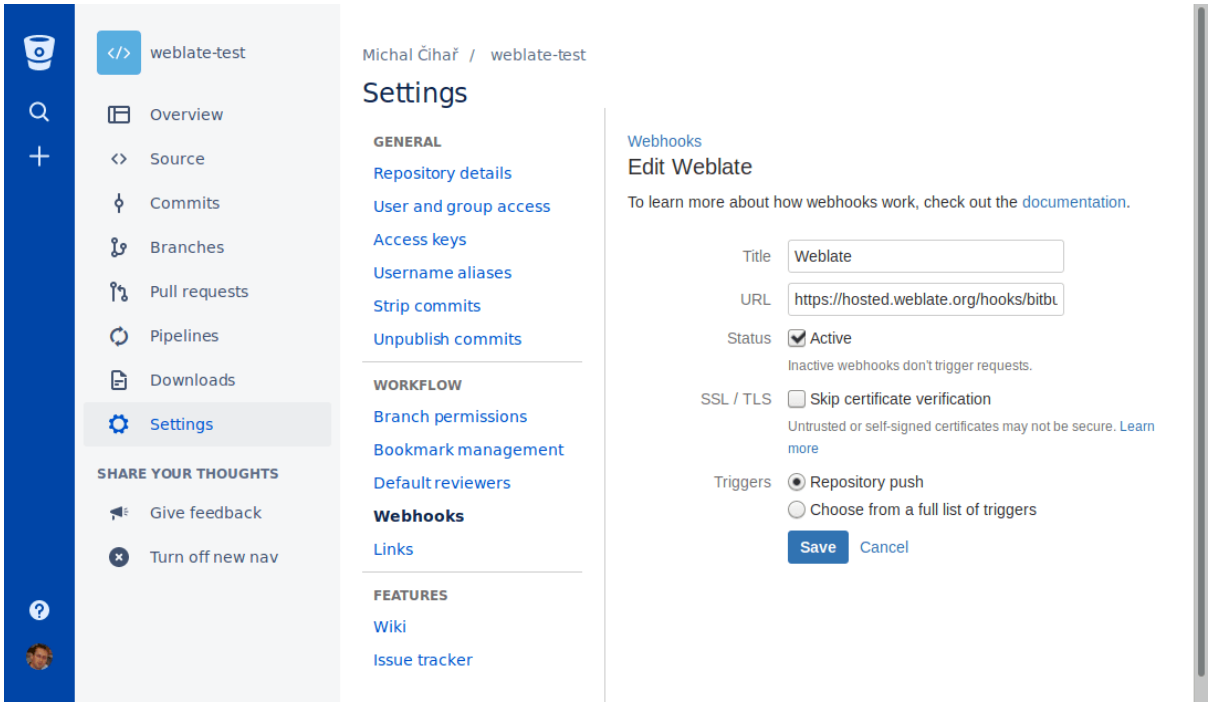
Pode deixar outros valores nas configurações predefinidas (o Weblate pode lidar com ambos os tipos de conteúdo e consome apenas o evento *push*).

Veja também:

`POST /hooks/github/`, *Accessing repositories from Hosted Weblate*

Receber alterações do Bitbucket automaticamente

O Weblate tem suporte para webhooks do Bitbucket, adicione um webhook que aciona no push do repositório, com destino a URL `/hooks/bitbucket/` na instalação do Weblate (por exemplo, `https://hosted.weblate.org/hooks/bitbucket/`).



Veja também:

`POST /hooks/bitbucket/`, *Accessing repositories from Hosted Weblate*

Receber alterações do GitLab automaticamente

O Weblate tem suporte para ganchos do GitLab, adiciona um webhook de projeto com destino a URL `/hooks/gitlab/` na instalação do Weblate (por exemplo, `https://hosted.weblate.org/hooks/gitlab/`).

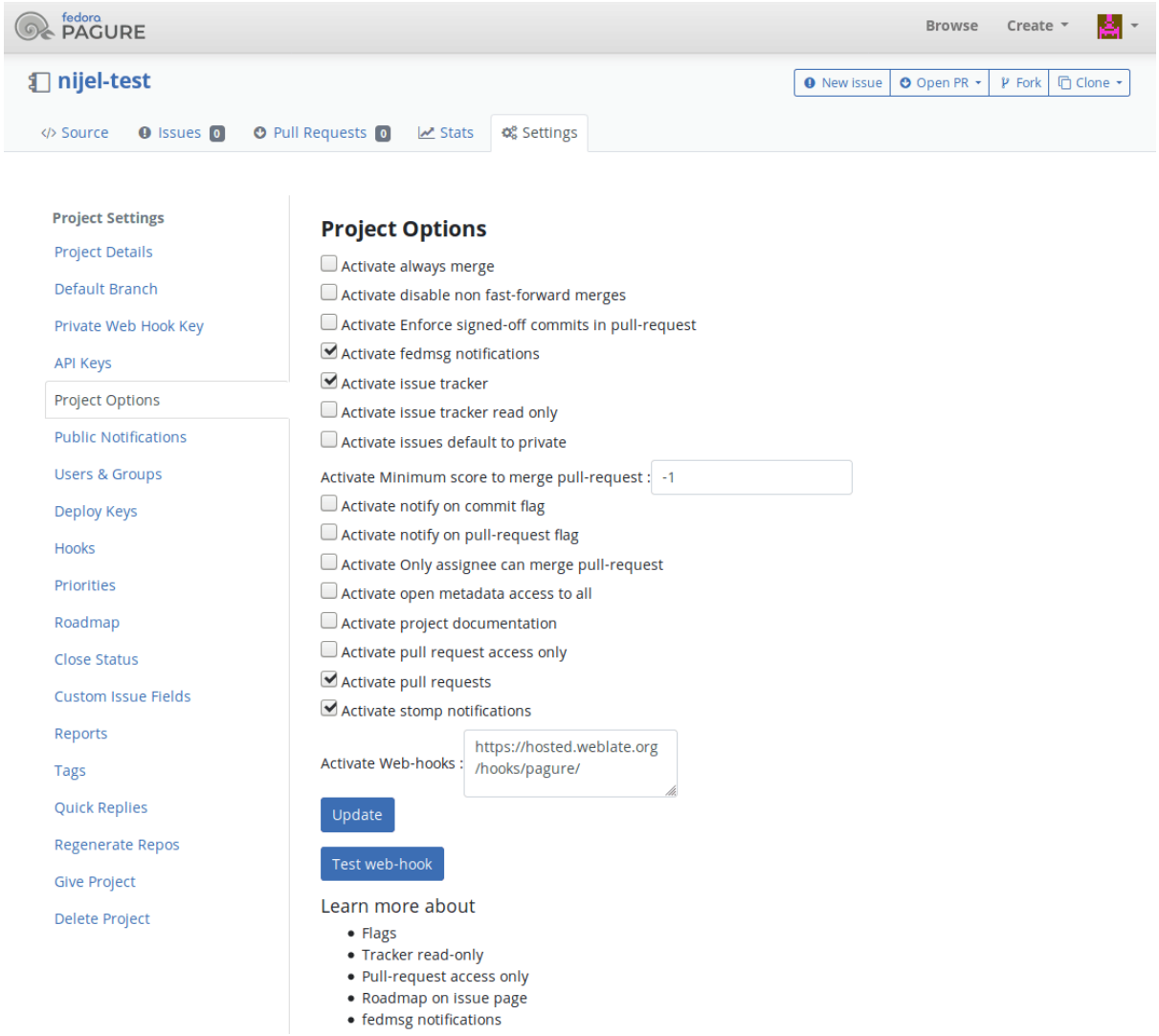
Veja também:

`POST /hooks/gitlab/`, *Accessing repositories from Hosted Weblate*

Receber alterações do Pagure automaticamente

Novo na versão 3.3.

O Weblate tem suporte para ganchos Pagure. Adicione um webhook com destino a URL `/hooks/pagure/` na instalação do Weblate (por exemplo, `https://hosted.weblate.org/hooks/pagure/`). Isso pode ser feito em *Web-hooks* em *Project options*:



Veja também:

`POST /hooks/pagure/`, *Accessing repositories from Hosted Weblate*

Receber alterações dos Azure Repos automaticamente

Novo na versão 3.8.

O Weblate tem suporte para webhooks dos Azure Repos, adicione um webhook para o evento *Code pushed* com destino para URL `/ganchos/azure/` na instalação do Weblate (por exemplo, `https://hosted.weblate.org/hooks/azure/`). Isso pode ser feito em *Service hooks ` em :guilabel:`Project settings*.

Veja também:

Webhooks no manual do Azure DevOps, `POST /hooks/azure/`, *Accessing repositories from Hosted Weblate*

Receber alterações dos Gitea Repos automaticamente

Novo na versão 3.9.

Weblate tem suporte para webhooks do Gitea, adicione um *Gitea Webhook* para o evento *Push events* com destino ao URL `/hooks/gitea/` na instalação do Weblate (por exemplo, `https://hosted.weblate.org/hooks/gitea/`). Isso pode ser feito no *Webhooks* em *Settings* do repositório.

Veja também:

Webhooks no manual do Gitea, *`POST /hooks/gitea/`*, *Accessing repositories from Hosted Weblate*

Receber alterações de Gitee Repos automaticamente

Novo na versão 3.9.

O Weblate tem suporte para webhooks Gitee, adicione um *WebHook* para o evento *Push* com destino para URL `/hooks/gitee/` na instalação do Weblate (por exemplo, `https://hosted.weblate.org/hooks/gitee/`). Isso pode ser feito em *WebHooks* sob *Management* do repositório.

Veja também:

Webhooks no manual do Gitee, *`POST /hooks/gitee/`*, *Accessing repositories from Hosted Weblate*

Atualizar repositórios *nightly* automaticamente

O Weblate busca automaticamente repositórios remotos *nightly* para melhorar o desempenho ao mesclar alterações mais tarde. Pode opcionalmente transformar isso em fazer mesclagens noturnas também, ativando *`AUTO_UPDATE`*.

2.9.2 Fazendo push das alterações do Weblate

Cada componente de tradução pode ter uma URL de push configurada (veja *URL de submissão do repositório*) e, nesse caso, o Weblate será capaz de fazer push da alteração ao repositório remoto. O Weblate também pode ser configurado para fazer push automaticamente das alterações em cada commit (isso é a predefinição, veja *Enviar ao submeter*). Se não quiser que seja feito push automático das alterações, pode fazer-lo manualmente em *Manutenção do repositório* ou a usar API via *`wlc push`*.

As opções de push diferem com base no *Integração de controlo de versões* usado, mais detalhes são encontrados nesse capítulo.

No caso de não querer pushes diretos pelo Weblate, há suporte para pull requests do *GitHub*, *GitLab*, *Pagure* ou reviews do *Gerrit*, pode ativá-los a escolher *GitHub*, *GitLab* ou *Gerrit* como *Sistema de controlo de versões* em *Component configuration*.

No geral, as opções a seguir estão disponíveis com Git, GitHub e GitLab:

Configuração desejada	*Sistema de controlo de versões*	*URL de submissão do repositório*	*Ramo do push*
Sem push	*Git*	*vazio*	*vazio*
Push diretamente	*Git*	URL de SSH	*vazio*
Empurrar para um ramo separado	*Git*	URL de SSH	Nome do ramo
Pull request de GitHub do fork	*GitHub*	*vazio*	*vazio*
Pull request de GitHub do ramo	*GitHub*	URL de SSH[1]	Nome do ramo
Merge request de GitLab do fork	*GitLab*	*vazio*	*vazio*
Merge request de GitLab do ramo	*GitLab*	URL de SSH[1]	Nome do ramo
Merge request de Pagure do fork	*Pagure*	*vazio*	*vazio*
Merge request de Pagure do ramo	*Pagure*	URL de SSH[1]	Nome do ramo

Nota: Também pode ativar o push automático de alterações após o Weblate fazer commit, isso pode ser feito em *Enviar ao submeter*.

Veja também:

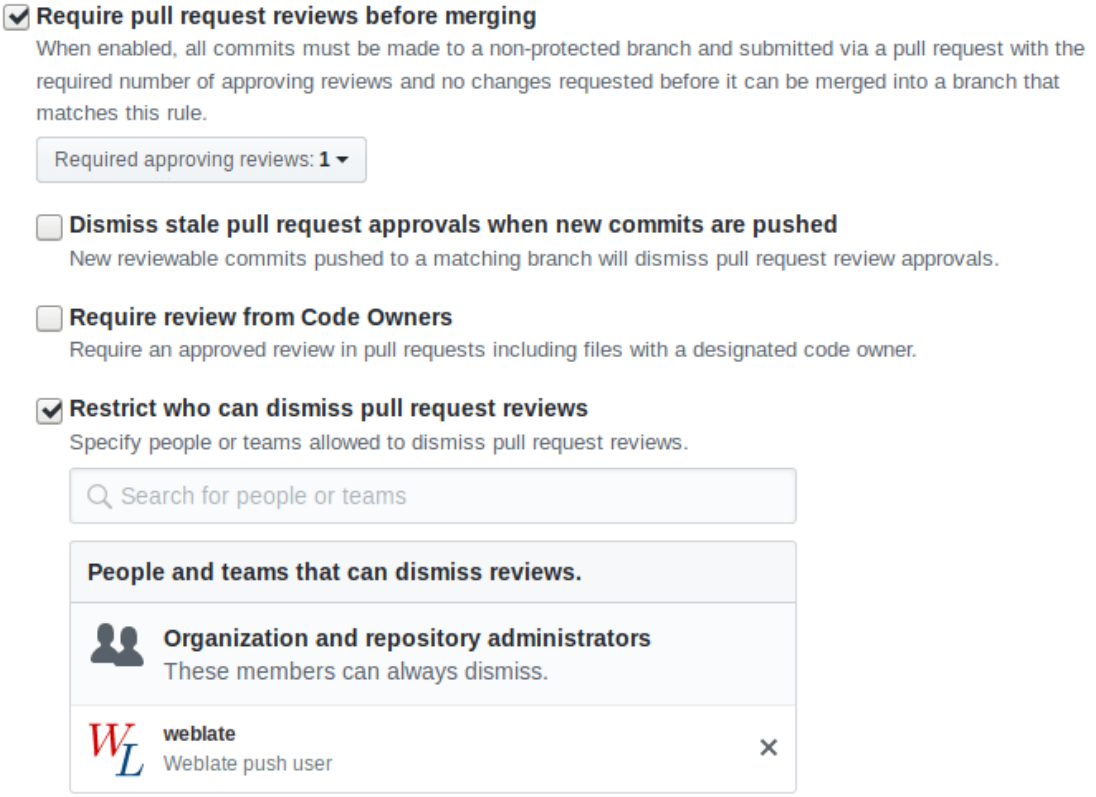
Consulte *Accessing repositories* para configurar chaves de SSH e *Commits adiados* para obter informações sobre quando o Weblate decide fazer commit de alterações.

Ramos protegidos

Se estiver a usar o Weblate em ramo protegido, pode configurá-lo para usar pull requests e executar revisão real sobre as traduções (o que pode ser problemático para idiomas que não conhece). Uma abordagem alternativa é abrir mão desta limitação em favor do utilizador de push no Weblate.

Por exemplo, no GitHub, isso pode ser feito na configuração do repositório:

[1] Pode estar vazia caso o *Repositório do código-fonte* tenha suporte a push.



2.9.3 Mesclar ou rebase

Por predefinição, o Weblate mescla o repositório upstream para dentro do seu próprio. Esta é a maneira mais segura no caso de também acessar o repositório subjacente por outros meios. Caso não precise disso, pode permitir fazer rebase de alterações em upstream, o que produzirá um histórico com menos compromissos de mesclagem.

Nota: Rebasing pode causar problemas em caso de mesclagens complicadas, então considere cuidadosamente se quer ou não ativá-los.

2.9.4 Interagir com os outros

O Weblate facilita a interação com outras pessoas a usar a API dele.

Veja também:

API

2.9.5 Commits adiados

O comportamento do Weblate é de agrupar commits do mesmo autor num só commit, se for possível. Isso reduz a quantidade de commits consideravelmente, no entanto, pode precisar de dizer explicitamente para fazer os commits no caso de querer deixar o repositório VCS em sincronia, por exemplo, para mesclarem (isso é por predefinição permitido para o grupo *Managers*, consulte *Controlo de acesso*).

As alterações neste modo têm o commit delas feitas assim que qualquer uma das seguintes condições são cumpridas:

- Outra pessoa altera uma cadeia já alterada.
- Um merge do upstream é feito.
- Um commit explícito é solicitado.
- A alteração é mais antiga do que o período definido como *Idade das alterações para fazer commit* em ref:*component*.

Dica: Os commits são criados para cada componente. Então, caso tenha muitos componentes, ainda verá muitos commits. Pode utilizar a extensão *Squash de commits git* neste caso.

Se quiser fazer commit das alterações com mais frequência e sem verificar a idade, pode agendar uma tarefa regular para realizar um commit:

```
CELERY_BEAT_SCHEDULE = {
    # Unconditionally commit all changes every 2 minutes
    "commit": {
        "task": "weblate.trans.tasks.commit_pending",
        # Ommiting hours will honor per component settings,
        # otherwise components with no changes older than this
        # won't be committed
        "kwargs": {"hours": 0},
        # How frequently to execute the job in seconds
        "schedule": 120,
    }
}
```

2.9.6 Processar repositório com scripts

A maneira de personalizar como o Weblate interage com o repositório é com *Extensões*. Consulte *Executar scripts de extensões* para obter informações sobre como executar scripts externos através de extensões.

2.9.7 Manter traduções iguais entre componentes

Uma vez que tenha vários componentes de tradução, pode garantir que as mesmas cadeias tenham a mesma tradução. Isso pode ser alcançado em vários níveis.

Propagação de tradução

Com a propagação de tradução ativada (que é a predefinição, consulte *Component configuration*), todas as novas traduções são feitas automaticamente em todos os componentes com cadeias correspondentes. Estas traduções são devidamente creditadas ao utilizador que traduz atualmente em todos os componentes.

Nota: A propagação de tradução requer a chave para ser compatível com formatos de tradução monolíngue, por isso tenha isso em mente ao criar chaves de tradução.

Verificação de consistência

A verificação check-inconsistente é acionada sempre que as cadeias são diferentes. Pode usar isso para rever tais diferenças manualmente e escolher a tradução certa.

Tradução automática

A tradução automática com base em diferentes componentes pode ser uma maneira de sincronizar as traduções entre os componentes. Pode acioná-la manualmente (veja *Tradução automática*) ou fazê-la ser executada automaticamente na atualização do repositório a usar uma extensão (veja *Tradução automática*).

2.10 Licenciar traduções

Pode especificar sob quais traduções de licença são contribuídas. Isto é especialmente importante se as traduções forem abertas ao público, para estipular para que elas possam ser usadas.

Deve especificar as informações da licença da *Component configuration*. Deve evitar exigir um contrato de licença de colaborador, embora seja possível.

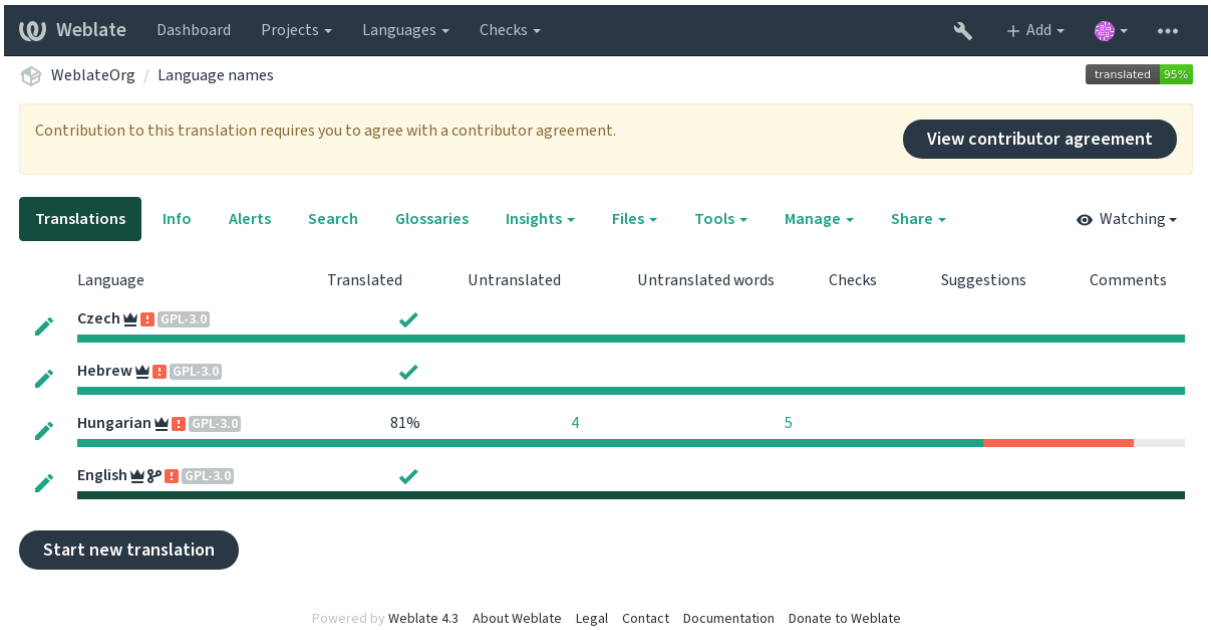
2.10.1 Informações de licença

Ao especificar informações de licenças (nome da licença e URL), essas informações são mostradas na secção de informações de tradução do respetivo *Component configuration*.

Normalmente este é o melhor lugar para publicar informações de licenciamento se nenhum consentimento explícito for necessário. Se o seu projeto ou tradução não for livre, provavelmente precisa de consentimento prévio.

2.10.2 Acordo de contribuidor

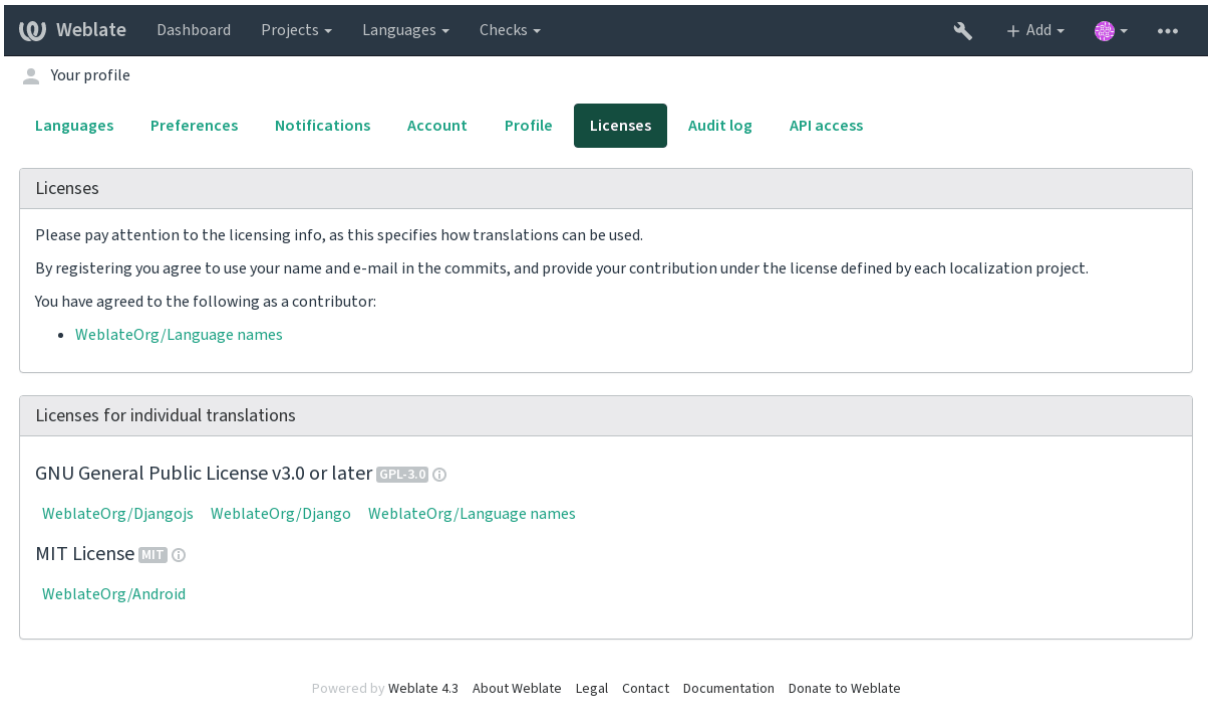
Se especificar um contrato de licença de colaborador, apenas os utilizadores que concordaram com ele poderão contribuir. Este é um passo claramente visível ao acessar a tradução:



O texto inserido é formatado em parágrafos e ligações externas podem ser incluídas. A marcação HTML não pode ser usada.

2.10.3 Licenças de utilizador

Todos utilizadores podem rever todas as licenças de tradução de todos os projetos públicos na instância do seu perfil:



2.11 Processo de tradução

2.11.1 Votação de sugestão

Everyone can add suggestions by default, to be accepted by signed in users. Suggestion voting can be used to make use of a string when more than signed in user agrees, by setting up the *Component configuration* configuration with *Suggestion voting* to turn on voting, and *Autoaccept suggestions* to set a threshold for accepted suggestions (this includes a vote from the user making the suggestion if it is cast).

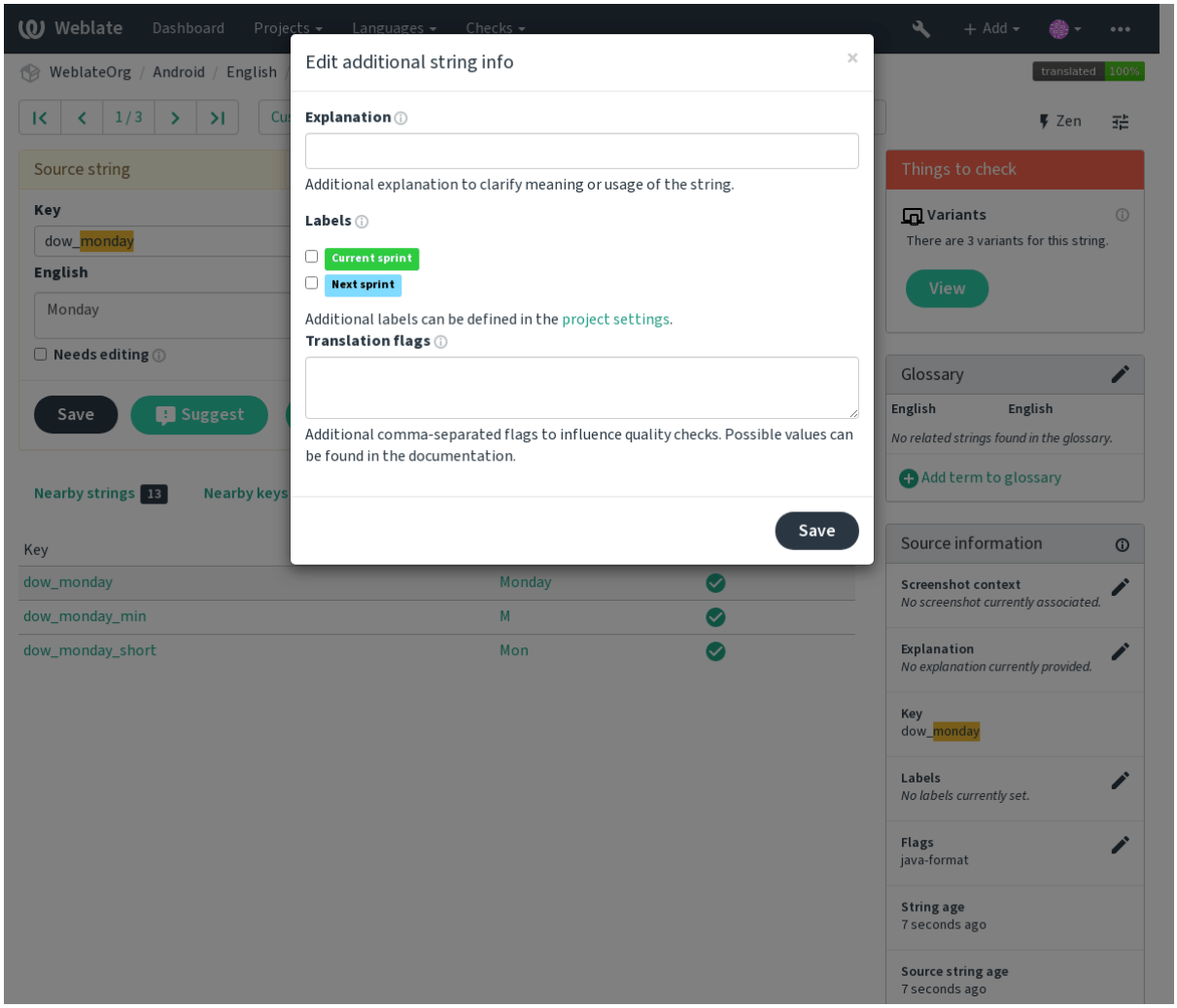
Nota: Once automatic acceptance is set up, normal users lose the privilege to directly save translations or accept suggestions. This can be overridden with the *Edit string when suggestions are enforced* privilege (see *Controlo de acesso*).

You can combine these with *Controlo de acesso* into one of the following setups:

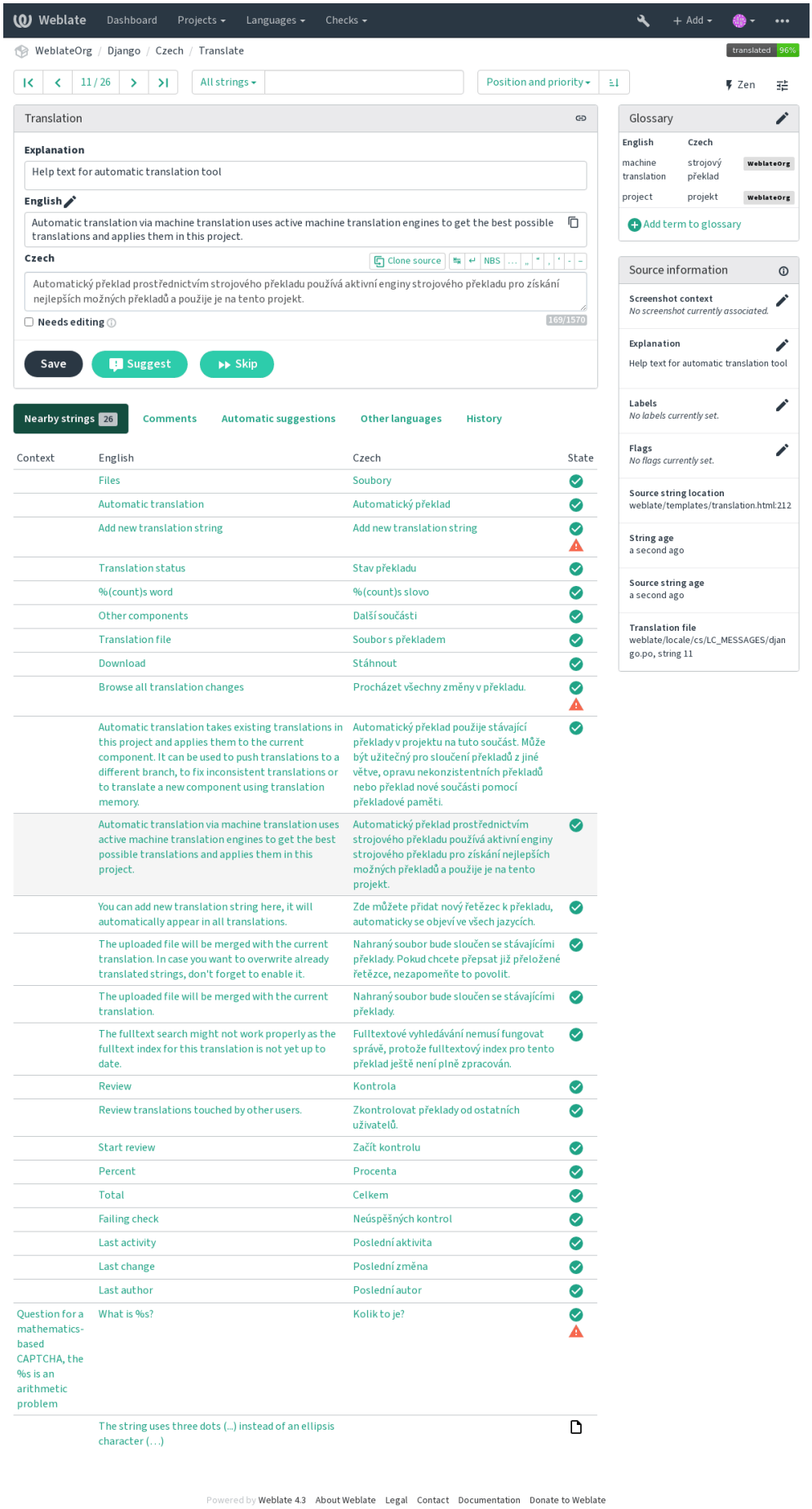
- Users suggest and vote for suggestions and a limited group controls what is accepted. - Turn on voting. - Turn off automatic acceptance. - Don't let users save translations.
- Users suggest and vote for suggestions with automatic acceptance once the defined number of them agree. - Turn on voting. - Set the desired number of votes for automatic acceptance.
- Optional voting for suggestions. (Can optionally be used by users when they are unsure about a translation by making multiple suggestions.) - Only turn on voting.

2.11.2 Additional info on source strings

Enhance the translation process with info available in the translation files. This includes explanation, string priority, check flags, or providing visual context. All these features can be set in the *Reviewing strings*:



Access this directly from the translation interface by clicking the «Edit» icon next to *Screenshot context* or *Flags*.


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Django / Czech / Translate
translated 96%
11 / 26
All strings
Position and priority
Zen
Translation
Explanation
Help text for automatic translation tool
English
Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.
Czech
Clone source
Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.
Needs editing
169/1570
Save
Suggest
Skip
Nearby strings 26
Comments
Automatic suggestions
Other languages
History
Context
English
Czech
State
Files
Soubory
Automatic translation
Automatický překlad
Add new translation string
Add new translation string
Translation status
Stav překladu
%(count)s word
%(count)s slovo
Other components
Další součásti
Translation file
Soubor s překladem
Download
Stáhnout
Browse all translation changes
Procházet všechny změny v překladu.
Automatic translation takes existing translations in this project and applies them to the current component. It can be used to push translations to a different branch, to fix inconsistent translations or to translate a new component using translation memory.
Automatický překlad použije stávající překlady v projektu na tuto součást. Může být užitečný pro sloučení překladů z jiné větve, opravu nekonzistentních překladů nebo překlad nové součásti pomocí překladové paměti.
Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.
Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.
You can add new translation string here, it will automatically appear in all translations.
Zde můžete přidat nový řetězec k překladu, automaticky se objeví ve všech jazycích.
The uploaded file will be merged with the current translation. In case you want to overwrite already translated strings, don't forget to enable it.
Nahraný soubor bude sloučen se stávajícími překlady. Pokud chcete přepsat již přeložené řetězce, nezapomeňte to povolit.
The uploaded file will be merged with the current translation.
Nahraný soubor bude sloučen se stávajícími překlady.
The fulltext search might not work properly as the fulltext index for this translation is not yet up to date.
Fulltextové vyhledávání nemusí fungovat správě, protože fulltextový index pro tento překlad ještě není plně zpracován.
Review
Kontrola
Review translations touched by other users.
Zkontrolovat překlady od ostatních uživatelů.
Start review
Začít kontrolu
Percent
Procenta
Total
Celkem
Failing check
Neúspěšných kontrol
Last activity
Poslední aktivita
Last change
Poslední změna
Last author
Poslední autor
Question for a mathematics-based CAPTCHA, the %s is an arithmetic problem
What is %s?
Kolik to je?
The string uses three dots (...) instead of an ellipsis character (…)
Glossary
English
Czech
machine translation
strojový překlad
WeblateOrg
project
projekt
WeblateOrg
Add term to glossary
Source information
Screenshot context
No screenshot currently associated.
Explanation
Help text for automatic translation tool
Labels
No labels currently set.
Flags
No flags currently set.
Source string location
weblate/templates/translation.html:212
String age
a second ago
Source string age
a second ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 11
Powered by Weblate 4.3
About Weblate
Legal
Contact
Documentation
Donate to Weblate

Strings prioritization

Novo na versão 2.0.

String priority can be changed to offer higher priority strings for translation earlier by using the `priority` flag

Dica: This can be used to order the flow of translation in a logical manner.

Veja também:

Verificações de qualidade

Marcadores de tradução

Novo na versão 2.4.

Alterado na versão 3.3: Previously called *Quality checks flags*, it no longer configures only checks.

The default set of translation flags is determined by the translation *Component configuration* and the translation file. However, you might want to use it to customize this per source string.

Veja também:

Verificações de qualidade

Explicação

Alterado na versão 4.1: In previous version this has been called extra context.

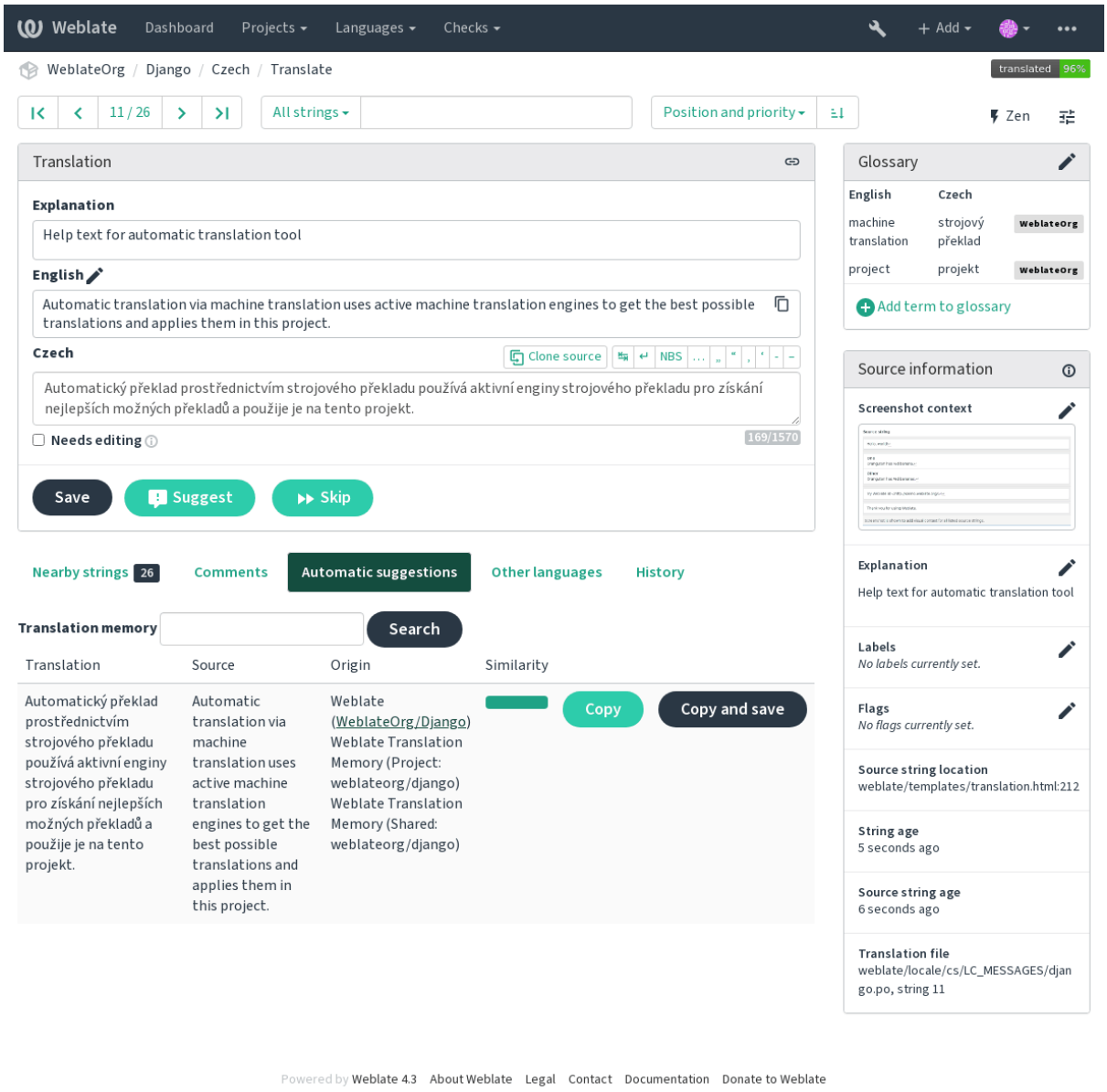
Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.

Visual context for strings

Novo na versão 2.9.

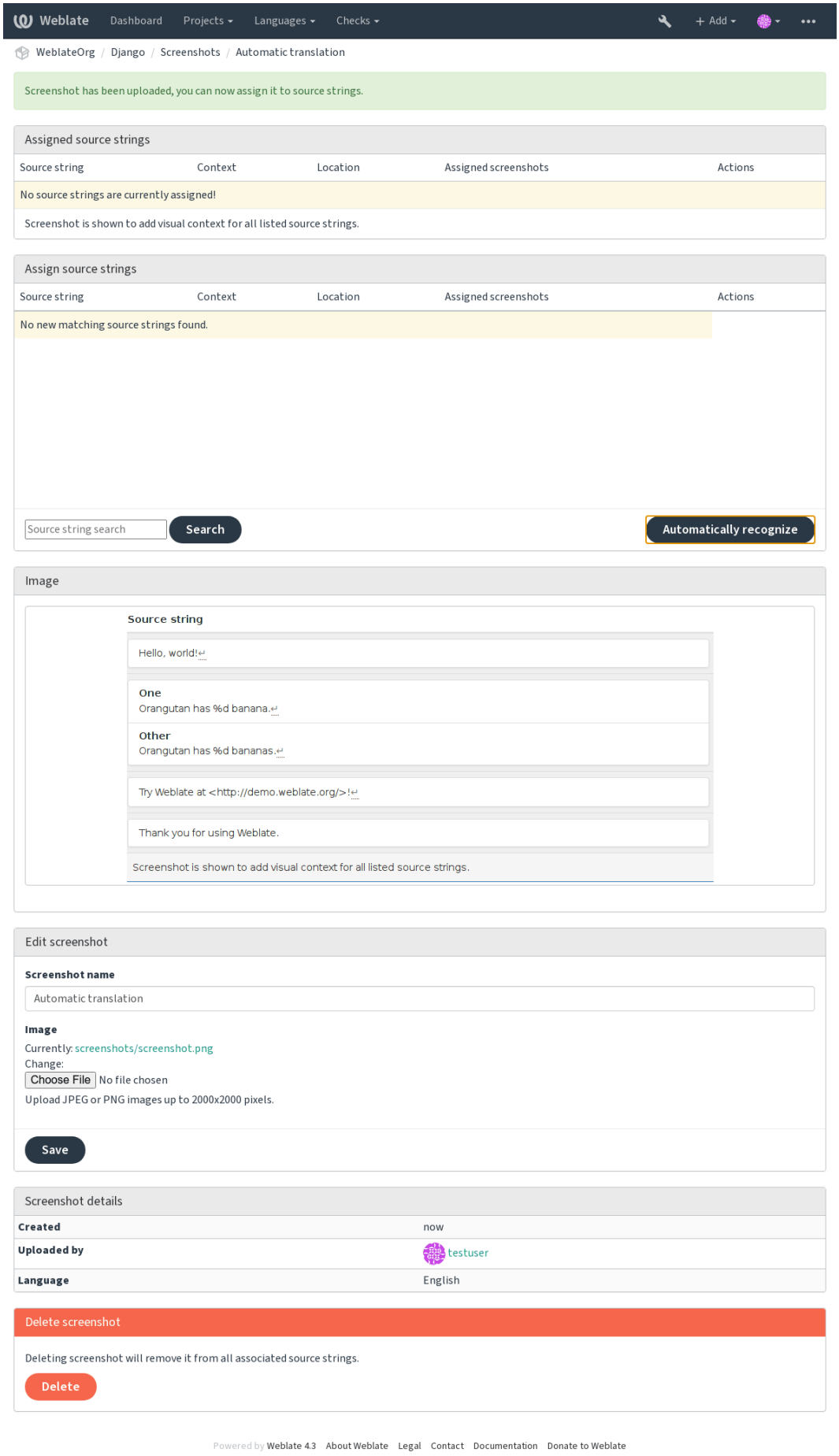
You can upload a screenshot showing a given source string in use within your program. This helps translators understand where it is used, and how it should be translated.

The uploaded screenshot is shown in the translation context sidebar:



In addition to *Reviewing strings*, screenshots have a separate management interface under the *Tools* menu. Upload screenshots, assign them to source strings manually, or use optical character recognition to do so.

Once a screenshot is uploaded, this interface handles management and source string association:


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Django / Screenshots / Automatic translation
Screenshot has been uploaded, you can now assign it to source strings.
Assigned source strings
Source string
Context
Location
Assigned screenshots
Actions
No source strings are currently assigned!
Screenshot is shown to add visual context for all listed source strings.
Assign source strings
Source string
Context
Location
Assigned screenshots
Actions
No new matching source strings found.
Source string search
Search
Automatically recognize
Image
Source string
Hello, world!
One
Orangutan has %d banana.
Other
Orangutan has %d bananas.
Try Weblate at <http://demo.weblate.org/>!
Thank you for using Weblate.
Screenshot is shown to add visual context for all listed source strings.
Edit screenshot
Screenshot name
Automatic translation
Image
Currently: screenshots/screenshot.png
Change:
Choose File
No file chosen
Upload JPEG or PNG images up to 2000x2000 pixels.
Save
Screenshot details
Created
now
Uploaded by
testuser
Language
English
Delete screenshot
Deleting screenshot will remove it from all associated source strings.
Delete
Powered by Weblate 4.3
About Weblate
Legal
Contact
Documentation
Donate to Weblate

2.12 Verificações e correções

2.12.1 Correções automáticas personalizadas

Também pode implementar a sua própria correção automática, além das predefinidas e incluí-las em *AUTOFIX_LIST*.

As correções automáticas são poderosas, mas também podem causar danos; tenha cuidado ao escrever uma.

Por exemplo, a correção automática a seguir iria substituir cada ocorrência da cadeia `foo`, numa tradução com `bar`:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#

from django.utils.translation import gettext_lazy as _

from weblate.trans.autofixes.base import AutoFix


class ReplaceFooWithBar(AutoFix):
    """Replace foo with bar."""

    name = _("Foobar")

    def fix_single_target(self, target, source, unit):
        if "foo" in target:
            return target.replace("foo", "bar"), True
        return target, False
```

Para instalar verificações personalizadas, forneça um caminho totalmente qualificado à classe Python em *AUTOFIX_LIST*, veja *Verificações de qualidade personalizadas, extensões e correções automáticas*.

2.12.2 Personalizar o comportamento

Pode ajustar o comportamento de Weblate (principalmente de verificações) para cada cadeia fonte (na revisão de cadeias fonte, veja *Additional info on source strings*) ou em *Component configuration* (*Marcadores de tradução*). Alguns formatos de ficheiro também permitem especificar sinalizadores diretamente no formato (veja *Formatos de ficheiros suportados*).

As etiquetas são separadas por vírgulas, os parâmetros são separados por caracteres de dois pontos. Pode usar aspas para incluir espaços em branco ou caracteres especiais na cadeia. Por exemplo:

```
placeholders:"special:value":"other value", regex:.*
```

Aqui está uma lista de sinalizadores atualmente aceitos:

rst-text Trata um texto como um documento reStructuredText, afeta *Tradução inalterada*.

md-text Trata o texto como um documento de Markdown.

dos-eol Usa marcadores de ponta de linha do DOS em vez dos Unix (`\r\n` em vez de `\n`).

url A cadeia deve consistir apenas numa URL.

safe-html A cadeia deve fazer seguro para HTML, veja *HTML inseguro*.

read-only A cadeia é somente leitura e não deve ser editada no Weblate, veja *Cadeias somente leitura*.

priority:N Prioridade da cadeia. As cadeias de maior prioridade são apresentados primeiro para tradução. A prioridade predefinida é 100, quanto maior prioridade que um texto tem, mais cedo é oferecido para tradução.

max-length:N Limita o comprimento máximo de uma cadeia a N caracteres, veja *Tamanho máximo da tradução*.

xml-text Trata o texto como documento XML, afeta *Sintaxe XML* e *Markup XML*.

font-family:NOME Define a família de letras para verificações da renderização, veja *Gerir letras*.

font-weight:PESO Define o peso da letra para verificações da renderização, veja *Gerir letras*.

font-size:SIZE Define o tamanho da letra para verificações da renderização, veja *Gerir letras*.

font-spacing:ESPAÇAMENTO Define o espaçamento da letra para verificações da renderização, veja *Gerir letras*.

placeholders:NOME Cadeias de espaço reservado esperados na tradução, veja *Espaços reservados*.

replacements:DE:PARA:DE2:PARA2...` Substituições para realizar ao verificar parâmetros de texto resultantes (por exemplo, em *Tamanho máximo da tradução* ou *Tamanho máximo da tradução*). O caso de uso típico para isso é expandir objetos colocáveis *(placeables)* para garantir que o texto se encaixe mesmo com nomes longos, por exemplo: `replacements:%s:"John Doe"`.

regex:EXPRESSÃO REGULAR Expressão regular para corresponder à tradução; veja *Expressão regular*.

python-format, c-format, php-format, python-brace-format, javascript-format, c-sharp-format, ja Trata todas as cadeias como a ser de formato, afeta *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Cadeias formatadas*, *Tradução inalterada*.

strict-same Faz com que «Tradução não alterada» evite usar a lista negra de palavras embutidas, veja *Tradução inalterada*.

ignore-bbcode Pular a verificação de qualidade «Markup de BBcode».

ignore-duplicate Pular a verificação de qualidade «Palavras consecutivas duplicadas».

ignore-double-space Pular a verificação de qualidade «Espaço duplo».

ignore-angularjs-format Pular a verificação de qualidade «Cadeia de interpolação AngularJS».

ignore-c-format Pular a verificação de qualidade «Formato C».

ignore-c-sharp-format Pular a verificação de qualidade «Formato C#».

ignore-es-format Pular a verificação de qualidade «Literais de modelo de ECMAScript».

ignore-i18next-interpolation Pular a verificação de qualidade «Interpolação de i18next».

ignore-java-format Pular a verificação de qualidade «Formato Java».

ignore-java-messageformat Pular a verificação de qualidade «MessageFormat do Java».

ignore-javascript-format Pular a verificação de qualidade «Formato JavaScript».

ignore-percent-placeholders Pular a verificação de qualidade «Espaços reservados de percentagem».

ignore-perl-format Pular a verificação de qualidade «Formato Perl».

ignore-php-format Pular a verificação de qualidade «Formato PHP».

ignore-python-brace-format Pular a verificação de qualidade «Formato de chaves Python».

ignore-python-format Pular a verificação de qualidade «Formato Python».

ignore-qt-format Pular a verificação de qualidade «Formato Qt».

ignore-qt-plural-format Pular a verificação de qualidade «Formato de plural Qt».

ignore-ruby-format Pular a verificação de qualidade «Formato Ruby».

ignore-vue-format Pular a verificação de qualidade «Formatação Vue I18n».

ignore-translated Pular a verificação de qualidade «Foi traduzido».

ignore-inconsistent Pular a verificação de qualidade «Inconsistente».

ignore-kashida Pular a verificação de qualidade «Letra Kashida usada».

ignore-md-link Pular a verificação de qualidade «Links Markdown».

ignore-md-reflink Pular a verificação de qualidade «Referências Markdown».

ignore-md-syntax Pular a verificação de qualidade «Sintaxe Markdown».

ignore-max-length Pular a verificação de qualidade «Comprimento máximo da tradução».

ignore-max-size Pular a verificação de qualidade «Tamanho máximo da tradução».

ignore-escaped-newline Ignora a verificação de qualidade «n não correspondente».

ignore-end-colon Ignora a verificação de qualidade «Caractere de dois pontos não correspondente».

ignore-end-ellipsis Pular a verificação de qualidade «Reticências não correspondentes».

ignore-end-exclamation Pular a verificação de qualidade «Ponto de exclamação não correspondente».

ignore-end-stop Pular a verificação de qualidade «Ponto final não correspondente».

ignore-end-question Pular a verificação de qualidade «Ponto de interrogação não correspondente».

ignore-end-semicolon Pular a verificação de qualidade «Ponto e vírgula não correspondente».

ignore-newline-count Pular a verificação de qualidade «Quebras de linha não correspondentes».

ignore-plurals Pular a verificação de qualidade «Faltam plurais».

ignore-placeholders Pular a verificação de qualidade «Espaços reservados».

ignore-punctuation-spacing Ignora a verificação de qualidade «Espaçamento de pontuação».

ignore-regex Pular a verificação de qualidade «Expressão regular».

ignore-same-plurals Pular a verificação de qualidade «Mesmos plurais».

ignore-begin-newline Pula a verificação de qualidade «Nova linha no início».

ignore-begin-space Pular a verificação de qualidade «Espaços no início».

ignore-end-newline Pular a verificação de qualidade «Nova linha no final».

ignore-end-space Pular a verificação de qualidade «Espaço no final».

ignore-same Ignora a verificação de qualidade «Tradução não alterada».

ignore-safe-html Pular a verificação de qualidade «HTML inseguro».

ignore-url Pular a verificação de qualidade «URL».

ignore-xml-tags Pular a verificação de qualidade «Marcação XML».

ignore-xml-invalid Pular a verificação de qualidade «Sintaxe XML».

ignore-zero-width-space Pular a verificação de qualidade «Espaço com largura zero».

ignore-ellipsis Pular a verificação de qualidade «Reticências».

ignore-long-untranslated Pular a verificação de qualidade «Não traduzido a muito tempo».

ignore-multiple-failures Pular a verificação de qualidade «Várias verificações com falha».

ignore-unnamed-format Pular a verificação de qualidade «Várias variáveis sem nome».

ignore-optional-plural Pular a verificação de qualidade «Não pluralizado».

Nota: Geralmente, a regra é chamada `ignore-*` para qualquer verificação, a usar o identificador dele, para que possa usá-la mesmo para as suas verificações personalizadas.

Essas etiquetas são entendidas tanto nas configurações de *Component configuration*, por configurações de cadeias fonte quanto no próprio ficheiro de tradução (por exemplo, no GNU gettext).

2.12.3 Forçar verificações

Novo na versão 3.11.

Pode configurar uma lista de verificações que não podem ser ignoradas a definir *Verificações impostas* em *Component configuration*. Cada verificação listada não pode ser ignorada na interface do utilizador e qualquer cadeia com falha nesta verificação é marcada como *Precisa de edição* (veja *Translation states*).

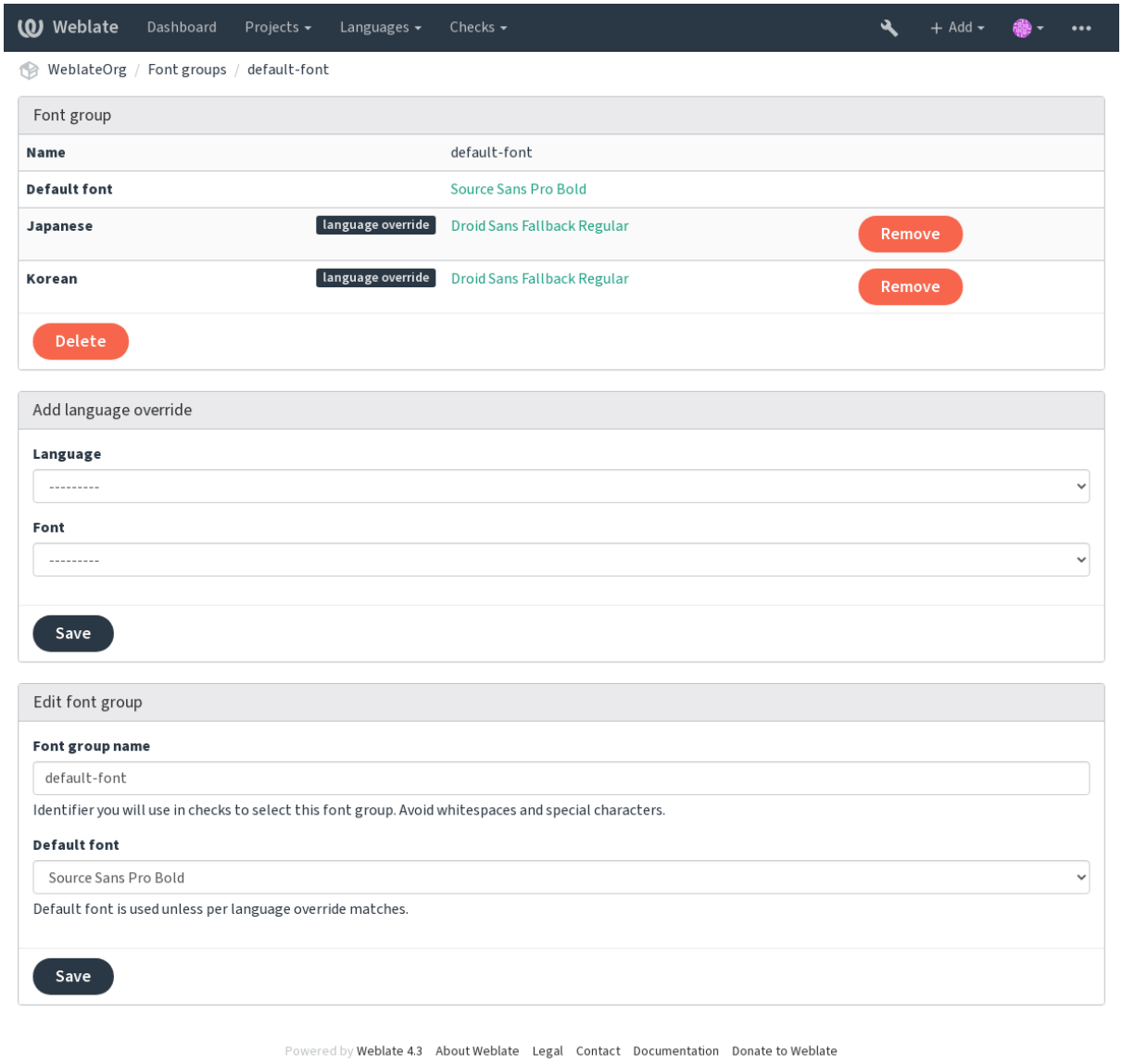
2.12.4 Gerir letras

Novo na versão 3.7.

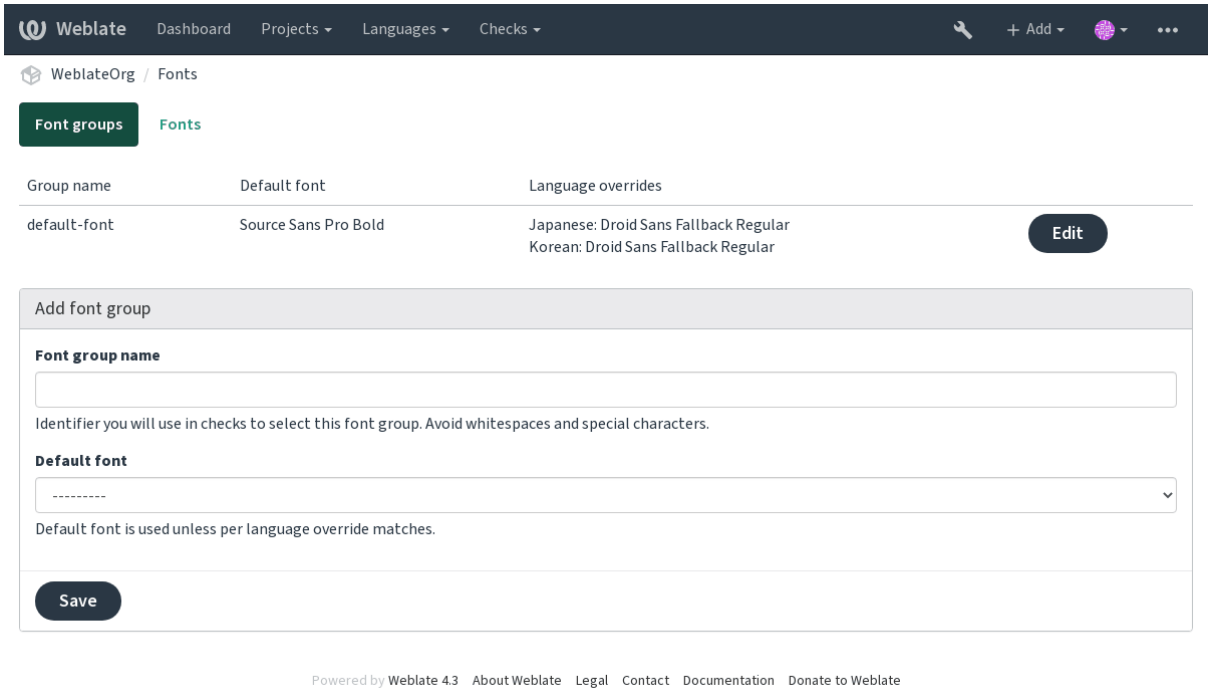
A verificação *Tamanho máximo da tradução* usada para calcular as dimensões do texto renderizado precisa de informações da letra para ser selecionada, o que pode ser feito na ferramenta de Weblate gestão de letra em *Fontes* sob *Gerir* do menu do seu projeto de tradução.

Letras de TrueType ou OpenType podem ser enviadas. Configure grupos de letras e use-as na verificação.

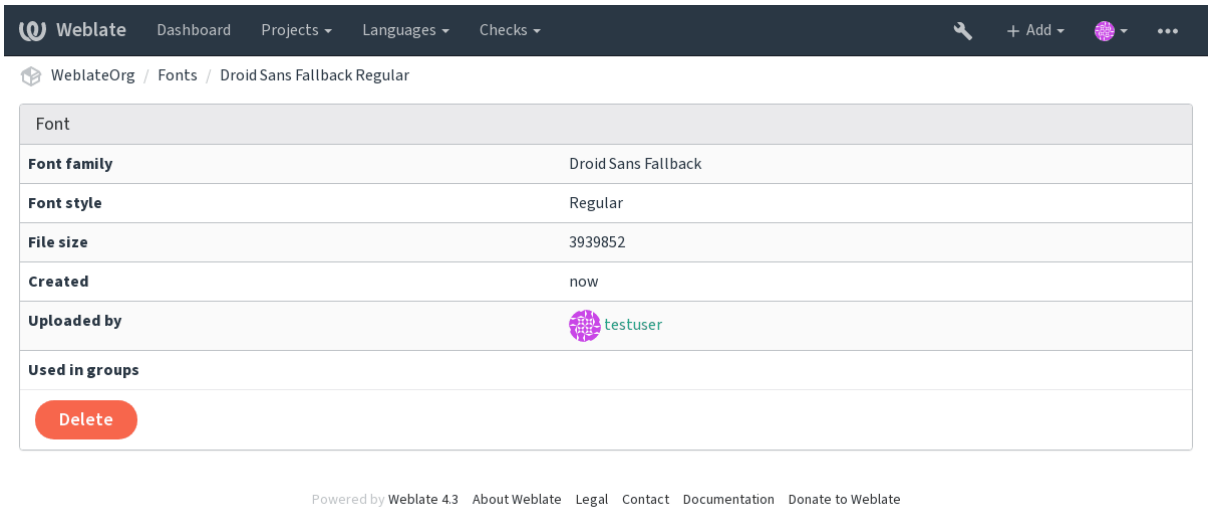
Os grupos de letras permitem definir letras diferentes para idiomas diferentes, o que é normalmente necessário para idiomas não-latinos:



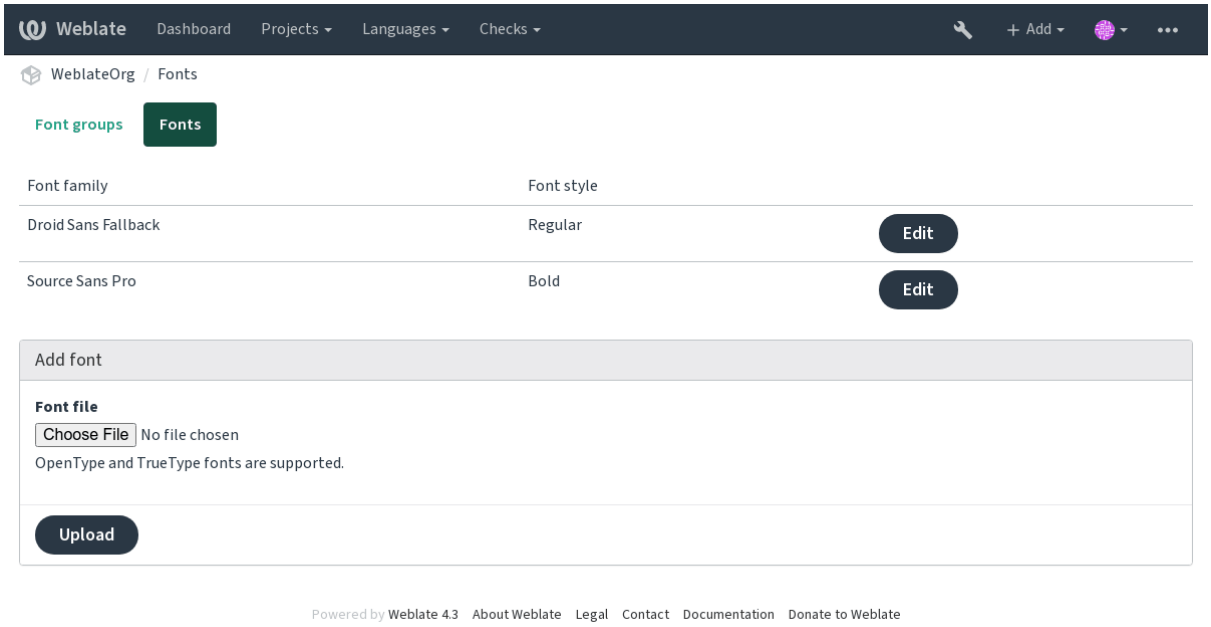
O grupos de letras são identificados pelo nome, que não pode conter espaços ou caracteres especiais, de modo que ele pode ser facilmente utilizado na definição da verificação:



A família de letras e o estilo são automaticamente reconhecidos após carregá-los:



Pode ter muitas letras carregadas para Weblate:



Para usar as letras para verificar o comprimento da cadeia, passe-a os sinalizadores apropriados (veja *Personalizar o comportamento*). Provavelmente precisará dos seguintes:

`max-size:500` Define o máximo de largura.

`font-family:ubuntu` Define o grupo de letras para usar especificando seu identificador.

`font-size:22` Define o tamanho da letra.

2.12.5 Escrever próprias verificações

Uma ampla gama de verificações de qualidade são incorporadas, (veja *Verificações de qualidade*), embora eles possam não cobrir tudo o que deseja verificar. A lista de verificações realizadas pode ser ajustada a usar `CHECK_LIST` e também pode adicionar verificações personalizadas.

1. Crie uma subclasse de *weblate.checks.Check*
2. Define alguns atributos.
3. Implemente o método `check` (se quiser lidar com plurais no seu código) ou o método `check_single` (que faz isso por si).

Alguns exemplos:

Para instalar verificações personalizadas, forneça um caminho totalmente qualificado à classe Python em `CHECK_LIST`, veja *Verificações de qualidade personalizadas, extensões e correções automáticas*.

Verificar se o texto de tradução não contém «foo»

Esta é uma verificação bastante simples que apenas verifica se a tradução não possui a cadeia «foo».

```
#
# Copyright © 2012 – 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
```

(continues on next page)

(continuação da página anterior)

```
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Simple quality check example."""

from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck


class FooCheck(TargetCheck):

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_single(self, source, target, unit):
        return "foo" in target
```

Verificando se os plurais de texto de tradução tcheca são diferentes

Usa as informações de idioma para verificar se as duas formas plurais no idioma tcheco não são os mesmos.

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Quality check example for Czech plurals."""

from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck
```

(continues on next page)

(continuação da página anterior)

```
class PluralCzechCheck(TargetCheck):

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_target_unit(self, sources, targets, unit):
        if self.is_language(unit, ("cs",)):
            return targets[1] == targets[2]
        return False

    def check_single(self, source, target, unit):
        """We don't check target strings here."""
        return False
```

2.13 Tradução automática

Built-in support for several machine translation services and can be turned on by the administrator using *MT_SERVICES* for each one. They come subject to their terms of use, so ensure you are allowed to use them how you want.

The source language can be configured at *Project configuration*.

2.13.1 amaGama

Special installation of *tmserver* run by the authors of Virtaal.

Turn on this service by adding `weblate.machinery.tmserver.AmagamaTranslation` to *MT_SERVICES*.

Veja também:

Installing amaGama, Amagama, amaGama Translation Memory

2.13.2 Apertium

A libre software machine translation platform providing translations to a limited set of languages.

The recommended way to use Apertium is to run your own Apertium-APy server.

Turn on this service by adding `weblate.machinery.apertium.ApertiumAPYTranslation` to *MT_SERVICES* and set *MT_APERTIUM_APY*.

Veja também:

MT_APERTIUM_APY, Apertium website, Apertium APy documentation

2.13.3 AWS

Novo na versão 3.1.

Amazon Translate is a neural machine translation service for translating text to and from English across a breadth of supported languages.

1. Turn on this service by adding `weblate.machinery.aws.AWSTranslation` to *MT_SERVICES*.
2. Install the *boto3* module.
3. Configure Weblate.

Veja também:

MT_AWS_REGION, *MT_AWS_ACCESS_KEY_ID*, *MT_AWS_SECRET_ACCESS_KEY* Amazon Translate Documentation

2.13.4 Baidu API machine translation

Novo na versão 3.2.

Machine translation service provided by Baidu.

This service uses an API and you need to obtain an ID and API key from Baidu to use it.

Turn on this service by adding `weblate.machinery.baidu.BaiduTranslation` to *MT_SERVICES* and set *MT_BAIDU_ID* and *MT_BAIDU_SECRET*.

Veja também:

MT_BAIDU_ID, *MT_BAIDU_SECRET* Baidu Translate API

2.13.5 DeepL

Novo na versão 2.20.

DeepL is paid service providing good machine translation for a few languages. You need to purchase *DeepL API* subscription or you can use legacy *DeepL Pro (classic)* plan.

Turn on this service by adding `weblate.machinery.deepl.DeepLTranslation` to *MT_SERVICES* and set *MT_DEEPL_KEY*.

Dica: In case you have subscription for CAT tools, you are supposed to use «v1 API» instead of default «v2» used by Weblate (it is not really an API version in this case). You can toggle this by *MT_DEEPL_API_VERSION*.

Veja também:

MT_DEEPL_KEY, *MT_DEEPL_API_VERSION*, DeepL website, DeepL pricing, DeepL API documentation

2.13.6 Glosbe

Free dictionary and translation memory for almost every living language.

The API is gratis to use, but subject to the used data source license. There is a limit of calls that may be done from one IP in a set period of time, to prevent abuse.

Turn on this service by adding `weblate.machinery.glosbe.GlosbeTranslation` to *MT_SERVICES*.

Veja também:

Glosbe website

2.13.7 Google Translate

Machine translation service provided by Google.

This service uses the Google Translation API, and you need to obtain an API key and turn on billing in the Google API console.

To turn on this service, add `weblate.machinery.google.GoogleTranslation` to *MT_SERVICES* and set *MT_GOOGLE_KEY*.

Veja também:

MT_GOOGLE_KEY, Google translate documentation

2.13.8 Google Translate API V3 (Advanced)

Machine translation service provided by Google Cloud services.

This service differs from the former one in how it authenticates. To enable service, add `weblate.machinery.googlev3.GoogleV3Translation` to *MT_SERVICES* and set

- *MT_GOOGLE_CREDENTIALS*
- *MT_GOOGLE_PROJECT*

If *location* fails, you may also need to specify *MT_GOOGLE_LOCATION*.

Veja também:

MT_GOOGLE_CREDENTIALS, *MT_GOOGLE_PROJECT*, *MT_GOOGLE_LOCATION* Google translate documentation

2.13.9 Microsoft Cognitive Services Translator

Novo na versão 2.10.

Machine translation service provided by Microsoft in Azure portal as a one of Cognitive Services.

Weblate implements Translator API V3.

To enable this service, add `weblate.machinery.microsoft.MicrosoftCognitiveTranslation` to *MT_SERVICES* and set *MT_MICROSOFT_COGNITIVE_KEY*.

Translator Text API V2

The key you use with Translator API V2 can be used with API 3.

Translator Text API V3

You need to register at Azure portal and use the key you obtain there. With new Azure keys, you also need to set *MT_MICROSOFT_REGION* to locale of your service.

Veja também:

MT_MICROSOFT_COGNITIVE_KEY, *MT_MICROSOFT_REGION*, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.13.10 Microsoft Terminology Service

Novo na versão 2.19.

The Microsoft Terminology Service API allows you to programmatically access the terminology, definitions and user interface (UI) strings available in the Language Portal through a web service.

Turn this service on by adding `weblate.machinery.microsoftterminology.MicrosoftTerminologyService` to *MT_SERVICES*.

Veja também:

Microsoft Terminology Service API

2.13.11 ModernMT

Novo na versão 4.2.

Turn this service on by adding `weblate.machinery.modernmt.ModernMTTranslation` to *MT_SERVICES* and configure *MT_MODERNMT_KEY*.

Veja também:

ModernMT API, *MT_MODERNMT_KEY*, *MT_MODERNMT_URL*

2.13.12 MyMemory

Huge translation memory with machine translation.

Free, anonymous usage is currently limited to 100 requests/day, or to 1000 requests/day when you provide a contact e-mail address in *MT_MYMEMORY_EMAIL*. You can also ask them for more.

Turn on this service by adding `weblate.machinery.mymemory.MyMemoryTranslation` to *MT_SERVICES* and set *MT_MYMEMORY_EMAIL*.

Veja também:

MT_MYMEMORY_EMAIL, *MT_MYMEMORY_USER*, *MT_MYMEMORY_KEY*, MyMemory website

2.13.13 NetEase Sight API machine translation

Novo na versão 3.3.

Machine translation service provided by Netease.

This service uses an API, and you need to obtain key and secret from NetEase.

Turn on this service by adding `weblate.machinery.youdao.NeteaseSightTranslation` to *MT_SERVICES* and set *MT_NETEASE_KEY* and *MT_NETEASE_SECRET*.

Veja também:

MT_NETEASE_KEY, *MT_NETEASE_SECRET* Netease Sight Translation Platform

2.13.14 tmserver

You can run your own translation memory server by using the one bundled with Translate-toolkit and let Weblate talk to it. You can also use it with an amaGama server, which is an enhanced version of tmserver.

1. First you will want to import some data to the translation memory:

2. Turn on this service by adding `weblate.machinery.tmserver.TMServerTranslation` to *MT_SERVICES*.

```
build_tmdb -d /var/lib/tm/db -s en -t cs locale/cs/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t de locale/de/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t fr locale/fr/LC_MESSAGES/django.po
```

3. Start tmserver to listen to your requests:

```
tmserver -d /var/lib/tm/db
```

4. Configure Weblate to talk to it:

```
MT_TMSERVER = 'http://localhost:8888/tmserver/'
```

Veja também:

MT_TMSERVER, tmserver Installing amaGama, Amagama, Amagama Translation Memory

2.13.15 Yandex Translate

Machine translation service provided by Yandex.

This service uses a Translation API, and you need to obtain an API key from Yandex.

Turn on this service by adding `weblate.machinery.yandex.YandexTranslation` to *MT_SERVICES*, and set *MT_YANDEX_KEY*.

Veja também:

MT_YANDEX_KEY, Yandex Translate API, Powered by Yandex.Translate

2.13.16 Youdao Zhiyun API machine translation

Novo na versão 3.2.

Machine translation service provided by Youdao.

This service uses an API, and you need to obtain an ID and an API key from Youdao.

Turn on this service by adding `weblate.machinery.youdao.YoudaoTranslation` to *MT_SERVICES* and set *MT_YOUDAO_ID* and *MT_YOUDAO_SECRET*.

Veja também:

MT_YOUDAO_ID, *MT_YOUDAO_SECRET* Youdao Zhiyun Natural Language Translation Service

2.13.17 Weblate

Weblate can be the source of machine translations as well. It is based on the Woosh fulltext engine, and provides both exact and inexact matches.

Turn on these services by adding `weblate.machinery.weblatetm.WeblateTranslation` to *MT_SERVICES*.

2.13.18 Weblate Translation Memory

Novo na versão 2.20.

The *Memória de Tradução* can be used as a source for machine translation suggestions as well.

Turn on these services by adding `weblate.memory.machine.WeblateMemory` to the *MT_SERVICES*. This service is turned on by default.

2.13.19 SAP Translation Hub

Machine translation service provided by SAP.

You need to have a SAP account (and enabled the SAP Translation Hub in the SAP Cloud Platform) to use this service.

Turn on this service by adding `weblate.machinery.saptranslationhub.SAPTranslationHub` to *MT_SERVICES* and set the appropriate access to either sandbox or the productive API.

Nota: To access the Sandbox API, you need to set *MT_SAP_BASE_URL* and *MT_SAP_SANDBOX_APIKEY*.

To access the productive API, you need to set *MT_SAP_BASE_URL*, *MT_SAP_USERNAME* and *MT_SAP_PASSWORD*.

Veja também:

MT_SAP_BASE_URL, *MT_SAP_SANDBOX_APIKEY*, *MT_SAP_USERNAME*, *MT_SAP_PASSWORD*, *MT_SAP_USE_MT* SAP Translation Hub API

2.13.20 Custom machine translation

You can also implement your own machine translation services using a few lines of Python code. This example implements machine translation in a fixed list of languages using `dictionary` Python module:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
```

(continues on next page)

(continuação da página anterior)

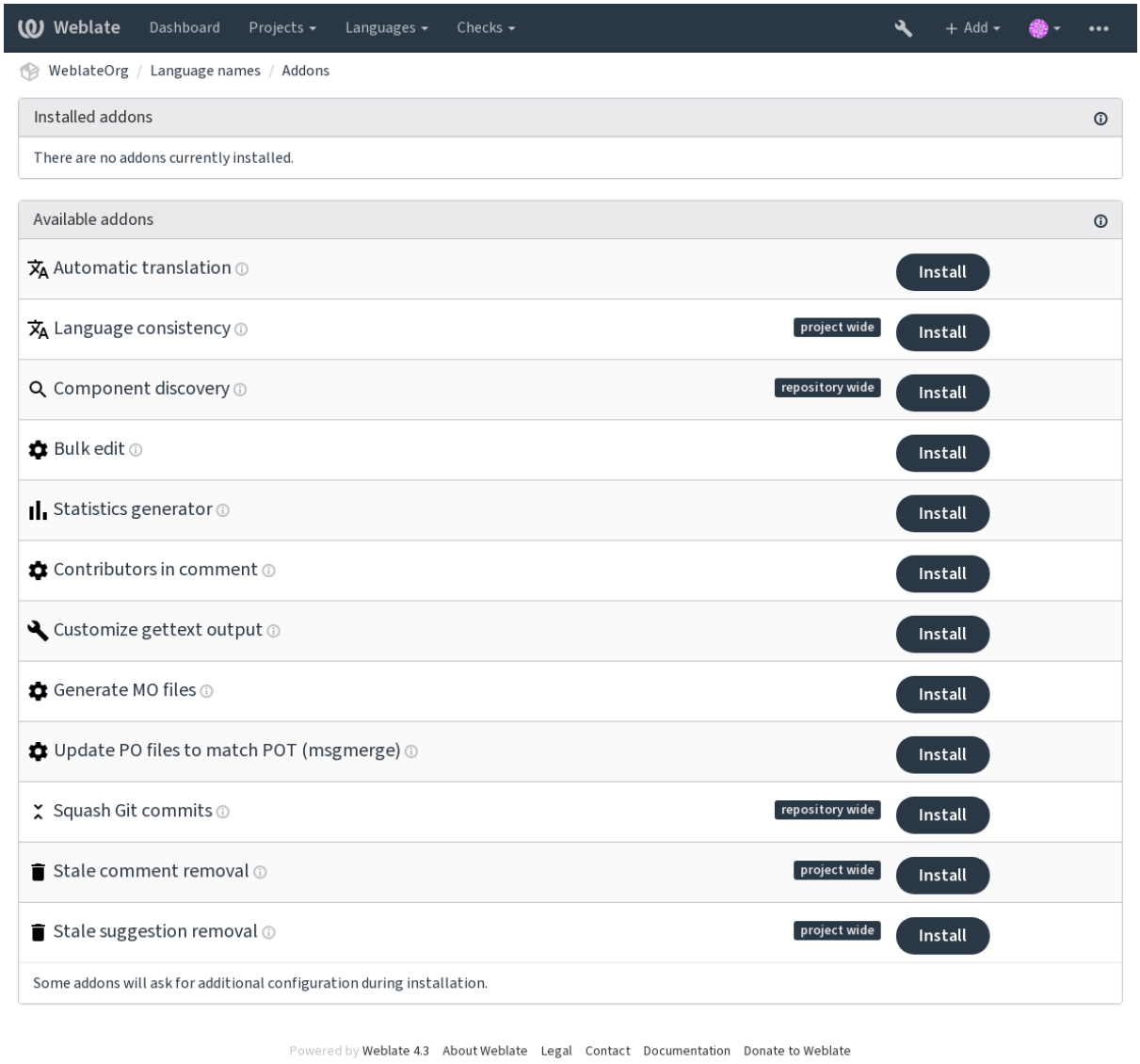
```
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Machine translation example."""

import dictionary

from weblate.machinery.base import MachineTranslation


class SampleTranslation(MachineTranslation):
    """Sample machine translation interface."""

    name = "Sample"

    def download_languages(self):
        """Return list of languages your machine translation supports."""
        return {"cs"}

    def download_translations(self, source, language, text, unit, user, search):
        """Return tuple with translations."""
        for t in dictionary.translate(text):
            yield {"text": t, "quality": 100, "service": self.name, "source": text}
```

You can list own class in *MT_SERVICES* and Weblate will start using that.

2.14 Extensões

Novo na versão 2.19.

Extensões fornecem maneiras para personalizar fluxo de trabalho de tradução. Elas podem ser instaladas na visão de componentes de tradução e trabalhar nos bastidores. Gestão de extensões está disponível do menu *Gerir* ↓ *Extensões* dos respetivos componentes de tradução para administradores.



2.14.1 Extensões embutidas

Tradução automática

Novo na versão 3.9.

Traduz automaticamente as cadeias utilizando a tradução automática ou outros componentes.

Esta extensão é acionada automaticamente quando novas cadeias aparecem num componente.

Veja também:

Tradução automática, *Manter traduções iguais entre componentes*

CDN de localização JavaScript

Novo na versão 4.2.

Adiciona o CDN de localização para localização em JavaScript ou HTML.

Pode ser usado para localizar páginas HTML estáticas ou para carregar a localização no código JavaScript.

Após a instalação, a extensão gera um URL exclusivo ao seu componente que pode incluir nos documentos HTML para localizar. Veja *Traduzir HTML e JavaScript a usar CDN Weblate* para mais detalhes.

Veja também:

Configurar extensão de Weblate CDN, *Traduzir HTML e JavaScript a usar CDN Weblate*, *Extração de cadeias para CDN Weblate*, *Localização de HTML a usar CDN Weblate*

Limpeza de ficheiros de tradução

Atualize todos os ficheiros de tradução para coincidirem com o ficheiro monolingue base. Para a maioria dos formatos de ficheiro, significa remover as chaves de tradução obsoletas que já não existem no ficheiro base.

Consistência do idioma

Garante que todos os componentes de um projeto tenham traduções para todos idiomas adicionados para tradução.

Cria traduções vazias em idiomas que têm componentes não adicionados.

Os idiomas ausentes são verificados uma vez a cada 24 horas e quando um novo idioma é adicionado no Weblate.

Ao contrário da maioria dos outros, esta extensão afeta todo o projeto.

Dica: Traduza as cadeias recém-adicionadas automaticamente com *Tradução automática*.

Descoberta de componentes

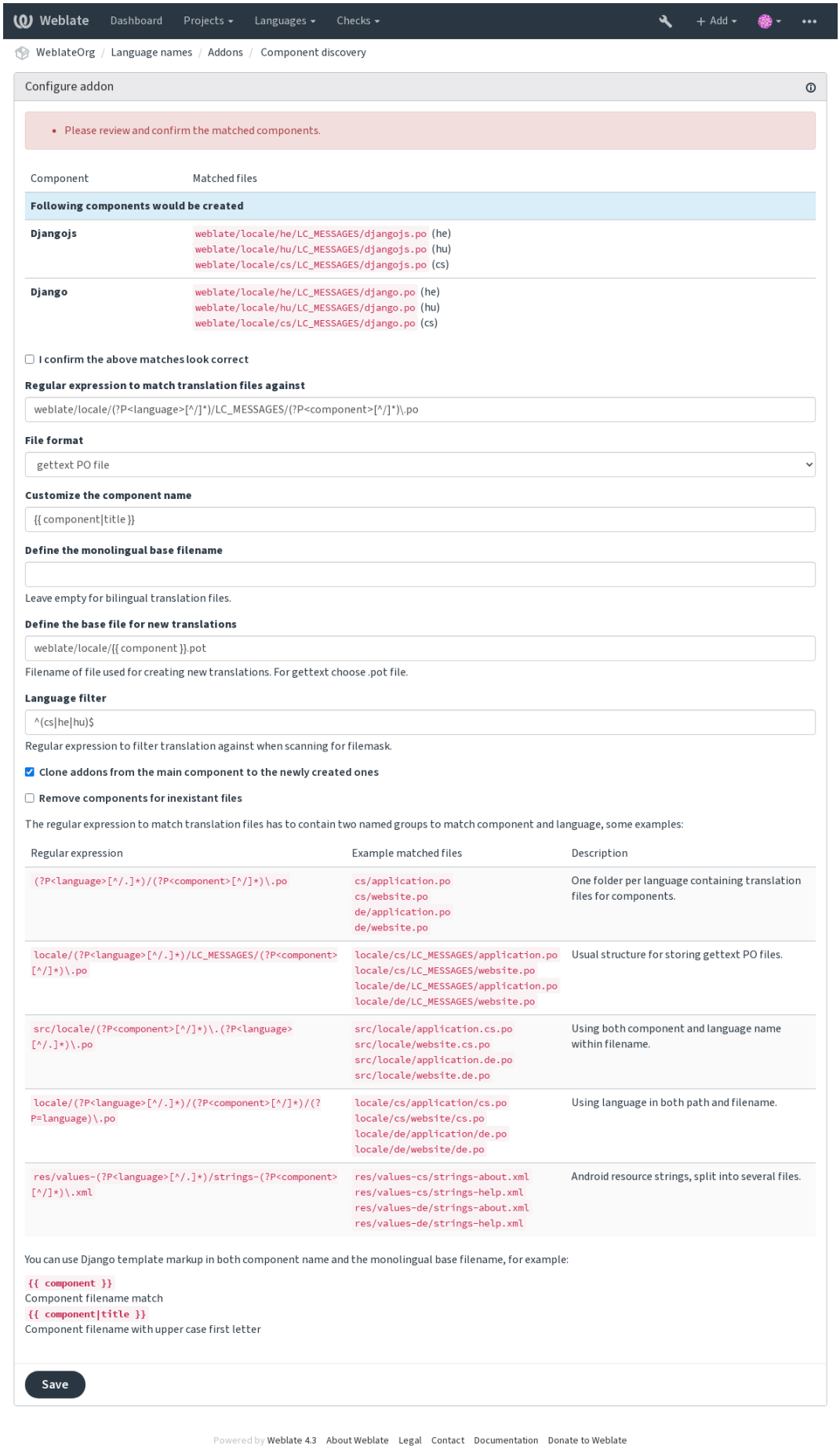
Adiciona ou remove automaticamente componentes do projeto com base em alterações de ficheiros no sistema de controlo de versão.

É acionada em todas as atualizações do VCS, de outra forma semelhante ao comando de gestão `import_project`. Desta forma, pode rastrear vários componentes de tradução dentro de um VCS.

Crie um componente principal menos provável de desaparecer no futuro e outros vão empregar *Weblate internal URLs* para ele como uma configuração VCS e vão configurá-lo para encontrar todos os componentes nele.

A correspondência é feita por expressões regulares, onde o poder é uma compensação para a complexidade na configuração. Alguns exemplos para casos de uso comum encontram-se na secção de ajuda de extensões.

Uma vez que acertar *Gravar*, uma prévia dos componentes correspondentes será apresentada, de onde pode verificar se a configuração realmente corresponde às suas necessidades:


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Language names / Addons / Component discovery
Configure addon
Please review and confirm the matched components.
Component
Matched files
Following components would be created
Djangojs
weblate/locale/he/LC_MESSAGES/djangojs.po (he)
weblate/locale/hu/LC_MESSAGES/djangojs.po (hu)
weblate/locale/cs/LC_MESSAGES/djangojs.po (cs)
Django
weblate/locale/he/LC_MESSAGES/django.po (he)
weblate/locale/hu/LC_MESSAGES/django.po (hu)
weblate/locale/cs/LC_MESSAGES/django.po (cs)
I confirm the above matches look correct
Regular expression to match translation files against
weblate/locale/(?P<language>[^/]*)/LC_MESSAGES/(?P<component>[^/]*)\.po
File format
gettext PO file
Customize the component name
{{ component|title }}
Define the monolingual base filename
Leave empty for bilingual translation files.
Define the base file for new translations
weblate/locale/{{ component }}.pot
Filename of file used for creating new translations. For gettext choose .pot file.
Language filter
^(cs|he|hu)$
Regular expression to filter translation against when scanning for filemask.
Clone addons from the main component to the newly created ones
Remove components for inexistant files
The regular expression to match translation files has to contain two named groups to match component and language, some examples:
Regular expression
Example matched files
Description
(?P<language>[^/.]*)/(?P<component>[^/]*)\.po
cs/application.po
cs/website.po
de/application.po
de/website.po
One folder per language containing translation files for components.
locale/(?P<language>[^/.]*)/LC_MESSAGES/(?P<component>[^/]*)\.po
locale/cs/LC_MESSAGES/application.po
locale/cs/LC_MESSAGES/website.po
locale/de/LC_MESSAGES/application.po
locale/de/LC_MESSAGES/website.po
Usual structure for storing gettext PO files.
src/locale/(?P<component>[^/]*)\.(?P<language>[^/.]*)\.po
src/locale/application.cs.po
src/locale/website.cs.po
src/locale/application.de.po
src/locale/website.de.po
Using both component and language name within filename.
locale/(?P<language>[^/.]*)/(?P<component>[^/]*)/(?P=language)\.po
locale/cs/application/cs.po
locale/cs/website/cs.po
locale/de/application/de.po
locale/de/website/de.po
Using language in both path and filename.
res/values-(?P<language>[^/.]*)/strings-(?P<component>[^/]*)\.xml
res/values-cs/strings-about.xml
res/values-cs/strings-help.xml
res/values-de/strings-about.xml
res/values-de/strings-help.xml
Android resource strings, split into several files.
You can use Django template markup in both component name and the monolingual base filename, for example:
{{ component }}
Component filename match
{{ component|title }}
Component filename with upper case first letter
Save
Powered by Weblate 4.3 About Weblate Legal Contact Documentation Donate to Weblate

Veja também:

Template markup

Edição em massa

Novo na versão 3.11.

Edição em série de marcadores, etiquetas, ou estados para cadeias.

Automatizar a etiquetagem de novos textos pode ser útil (comece com consulta de pesquisa `NOT has:label` e adicione etiquetas desejadas até que todas as cadeias sejam devidamente etiquetados). Também pode realizar quaisquer outras operações automatizadas para metadados do Weblate.

Veja também:

Edição em massa

Marcar as traduções inalteradas como «Precisa de edição»

Novo na versão 3.1.

Sempre que uma nova cadeia traduzível é importada de VCS e corresponde a uma cadeia fonte, esta é marcada como precisa de edição no Weblate. Isto é especialmente útil para os formatos de ficheiro que incluem todas as cadeias , mesmo que não traduzidas.

Marcar as novas cadeias fonte como «Precisa de edição»

Sempre que uma nova cadeia é importada de VCS, esta é marcada como precisa de edição no Weblate. Deste modo pode filtrar e editar facilmente as cadeias fonte escritas pelos programadores.

Marcar as novas traduções como «Precisa de edição»

Sempre que uma nova cadeia de tradução é importada de VCS, esta é marcada como precisa de edição no Weblate. Deste modo pode filtrar e editar facilmente as traduções criadas pelos programadores.

Gerador de estatísticas

Gera um ficheiro contendo a informação detalhada sobre a tradução.

Pode usar um modelo do Django, tanto de nome de ficheiro e conteúdo, veja Markdown para uma descrição detalhada de markup.

Por exemplo, a geração de ficheiro de resumo para cada tradução:

Nome do ficheiro gerado `locale/{{ language_code }}.json`

Conteúdo

```
{
    "language": "{{ language_code }}",
    "strings": "{{ stats.all }}",
    "translated": "{{ stats.translated }}",
    "last_changed": "{{ stats.last_changed }}",
    "last_author": "{{ stats.last_author }}",
}
```

Veja também:

Template markup

Contribuintes em comentários

Atualiza o comentário no cabeçalho do ficheiro PO para incluir nomes de colaboradores e anos de contribuições.

O cabeçalho do ficheiro PO conterá uma lista de contribuidores e anos contribuídos:

```
# Michal Čihař <michal@cihar.com>, 2012, 2018, 2019, 2020.
# Pavel Borecki <pavel@example.com>, 2018, 2019.
# Filip Hron <filip@example.com>, 2018, 2019.
# anonymous <noreply@weblate.org>, 2019.
```

Atualizar a variável ALL_LINGUAS no ficheiro «configure»

Atualiza a variável ALL_LINGUAS em ficheiros `configure`, `configure.in` ou `configure.ac`, quando uma nova tradução é adicionada.

Personalizar a saída gettext

Permite personalizar o comportamento da saída gettext, por exemplo, a quebra de linhas.

Oferece as seguintes opções:

- Quebrar linhas em 77 caracteres e em novas linhas
- Quebrar as linhas apenas nas novas linhas
- Sem quebra de linhas

Nota: Por predefinição, gettext quebra linhas em 77 caracteres e em novas linhas. Com o parâmetro `--no-wrap`, ele quebra apenas em novas linhas.

Atualizar ficheiro LINGUAS

Atualiza o ficheiro LINGUAS quando é adicionada uma nova tradução.

Gerar ficheiros MO

Gera automaticamente um ficheiro MO para cada ficheiro PO alterado.

Atualizar ficheiros PO para coincidir com POT (msgmerge)

Updates all PO files to match the POT file using msgmerge. Triggered whenever new changes are pulled from the upstream repository and updates all translation files to match *Modelo para novas traduções*. You can configure most of the msgmerge command line options through the addon configuration.

Squash de commits git

Comprimir as submissões Git antes de enviar as alterações.

Pode escolher um dos modos seguintes:

Novo na versão 3.4.

- Todos os commits num só
- Por idioma
- Por ficheiro

Novo na versão 3.5.

- Por autor

As mensagens de commit originais são mantidas, mas a autoria é perdida a menos que *Por autor* seja selecionada ou a mensagem de commit, seja personalizada para incluí-la.

Novo na versão 4.1.

As mensagens de commit originais podem opcionalmente ser substituídas por uma mensagem de commit personalizada.

Linhas finalizadoras (linhas de commits como `Co-authored-by: ...`) podem opcionalmente ser removidas das mensagens de commit originais e anexadas ao final da mensagem de compromisso após um squash. Isto também gera crédito próprio `Co-authored-by:` para cada tradutor.

Personalizar a saída JSON

Permite ajustar o comportamento da saída JSON, por exemplo, a indentação ou a ordenação.

Formata as propriedades do ficheiro Java

Ordena o ficheiro de propriedades Java.

Purga de comentários obsoletos

Novo na versão 3.7.

Definir um prazo para a remoção de comentários.

Isso pode ser útil para remover comentários antigos que podem ter ficado desatualizados. Use com cuidado, pois comentários sendo velhos não significam que perderam a importância deles.

Purga de sugestões obsoletas

Novo na versão 3.7.

Definir um prazo para a remoção de sugestões.

Isso pode ser muito útil em relação à votação em sugestão (ver *Revisão por pares*) para remover sugestões que não recebem votos positivos suficientes num determinado período.

Atualizar ficheiros RESX

Novo na versão 3.9.

Atualize todos os ficheiros de tradução para que correspondam ao ficheiro base monolingue original. As cadeias não utilizadas são removidas e as cadeias novas são adicionadas como cópias da cadeia fonte.

Dica: Use *Limpeza de ficheiros de tradução* se só quiser remover chaves de tradução obsoletas.

Personalizar a saída YAML

Novo na versão 3.10.2.

Permite ajustar o comportamento da saída YAML, por exemplo, o comprimento de linha ou novas linhas.

2.14.2 Personalizar a lista de extensões

A lista de extensões é configurada por *WEBLATE_ADDONS*. Para adicionar outra extensão, basta incluir o nome absoluto da classe nesta configuração.

2.14.3 Escrever extensões

Tambem pode escrever a sua própria extensão. Tudo o que precisa fazer é subclassear o *BaseAddon*, definir os metadados da extensão e implementar uma função de retorno que irá fazer o processamento.

Aqui está um exemplo de extensão:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#


from django.utils.translation import gettext_lazy as _

from weblate.addons.base import BaseAddon
from weblate.addons.events import EVENT_PRE_COMMIT


class ExampleAddon(BaseAddon):
    # Filter for compatible components, every key is
    # matched against property of component
    compat = {"file_format": {"po", "po-mono"}}
```

(continues on next page)

(continuação da página anterior)

```
    # List of events addon should receive
    events = (EVENT_PRE_COMMIT,)
    # Addon unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example addon")
    # Detailed addon description
    description = _("This addon does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

2.14.4 Executar scripts de extensões

Extenções também podem ser usados para executar scripts externos. Isto estava integrado no Weblate, mas agora tem que escrever código para embrulhar o seu script numa extensão.

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Example pre commit script."""


from django.utils.translation import gettext_lazy as _

from weblate.addons.events import EVENT_PRE_COMMIT
from weblate.addons.scripts import BaseScriptAddon


class ExamplePreAddon(BaseScriptAddon):
    # Event used to trigger the script
    events = (EVENT_PRE_COMMIT,)
    # Name of the addon, has to be unique
    name = "weblate.example.pre"
    # Verbose name and long descrption
    verbose = _("Execute script before commit")
    description = _("This addon executes a script.")

    # Script to execute
    script = "/bin/true"
    # File to add in commit (for pre commit event)
    # does not have to be set
    add_file = "po/{{ language_code }}.po"
```

Para instruções de instalação, veja *Verificações de qualidade personalizadas, extensões e correções automáticas*.

repositório VCS para qualquer componente.

Além disso, as seguintes variáveis de ambiente estão disponíveis:

WL_VCS
Sistema de controle de versão usado.

WL_REPO
URL do repositório upstream.

WL_PATH
Caminho absoluto ao repositório VCS.

WL_BRANCH
Novo na versão 2.11.

Ramo do repositório configurado no componente atual.

WL_FILEMASK
Máscara de ficheiro para componente atual.

WL_TEMPLATE
Nome de ficheiro de modelo para traduções monolínguas (pode estar vazio).

WL_NEW_BASE
Novo na versão 2.14.

Nome do ficheiro usado para criar novas traduções (pode estar vazio).

WL_FILE_FORMAT
Formato de ficheiro usado no componente atual.

WL_LANGUAGE
Idioma da tradução processada atualmente (não disponível para hooks de nível de componente).

WL_PREVIOUS_HEAD
HEAD anterior na atualização (disponível apenas ao executar o hook de atualização após).

WL_COMPONENT_SLUG
Novo na versão 3.9.

Slug do componente usado para construir a URL.

WL_PROJECT_SLUG
Novo na versão 3.9.

Slug de projeto usado para construir a URL.

WL_COMPONENT_NAME
Novo na versão 3.9.

Nome de componente.

WL_PROJECT_NAME
Novo na versão 3.9.

Nome do projeto.

WL_COMPONENT_URL
Novo na versão 3.9.

URL do componente.

WL_ENGAGE_URL
Novo na versão 3.9.

URL de engajamento do projeto.

Veja também:

Component configuration

Processamento de repositório pós-atualização

O processamento do repositório pós-atualização pode ser usado para atualizar ficheiros de tradução quando a fonte VCS do upstream alterar. Para conseguir isso, lembre-se que o Weblate só vê ficheiros dos quais se fez commit com o VCS, então precisa fazer commit das alterações como parte do script.

Por exemplo, com Gulp, pode fazê-lo a usar o código seguinte:

```
#! /bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

Processamento pré-commit de traduções

Use o script de commit para fazer alterações automaticamente na tradução antes de fazer commit dela ao repositório.

É passado como um parâmetro único que consiste o nome de uma tradução atual.

2.15 Memória de Tradução

Novo na versão 2.20.

Weblate comes with a built-in translation memory consisting of the following:

- Manually imported translation memory (see *User interface*).
- Automatically stored translations performed in Weblate (depending on *Translation memory scopes*).
- Automatically imported past translations.

Content in the translation memory can be applied one of two ways:

- Manually, *Sugestões automáticas* view while translating.
- Automatically, by translating strings using *Tradução automática*, or *Tradução automática* addon.

For installation tips, see *Weblate Translation Memory*, which is turned on by default.

2.15.1 Translation memory scopes

Novo na versão 3.2: In earlier versions translation memory could be only loaded from a file corresponding to the current imported translation memory scope.

The translation memory scopes are there to allow both privacy and sharing of translations, to suit the desired behavior.

Imported translation memory

Importing arbitrary translation memory data using the *import_memory* command makes memory content available to all users and projects.

Per user translation memory

Stores all user translations automatically in the personal translation memory of each respective user.

Per project translation memory

All translations within a project are automatically stored in a project translation memory only available for this project.

Memória de tradução compartilhada

All translation within projects with shared translation memory turned on are stored in a shared translation memory available to all projects.

Please consider carefully whether to turn this feature on for shared Weblate installations, as it can have severe implications:

- The translations can be used by anybody else.
- This might lead to disclosing secret information.

2.15.2 Managing translation memory

User interface

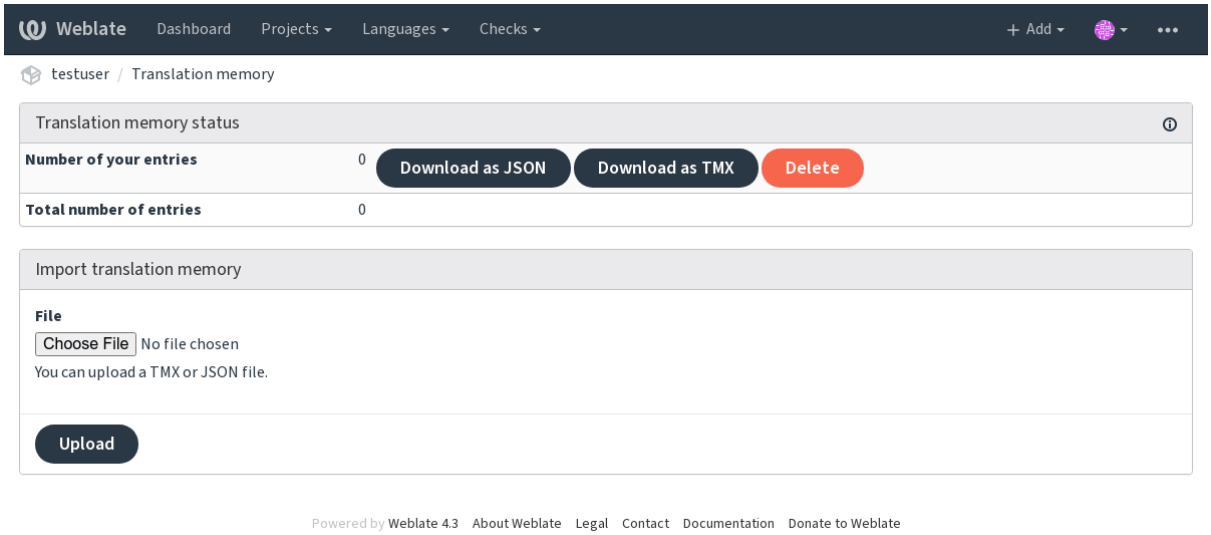
Novo na versão 3.2.

In the basic user interface you can manage per user and per project translation memories. It can be used to download, wipe or import translation memory.

Dica: Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

Veja também:

Esquema de memória de tradução Weblate



Interface de gestão

There are several management commands to manipulate the translation memory content. These operate on the translation memory as whole, unfiltered by scopes (unless requested by parameters):

dump_memory Exports the memory into JSON

import_memory Imports TMX or JSON files into the translation memory

2.16 Configuração

Todas as configurações estão armazenadas em `settings.py` (como é habitual no Django).

Nota: Após alterar qualquer uma dessas configurações, precisa reiniciar o Weblate - tanto os processos WSGI quanto os Celery.

Caso seja executado como mod_wsgi, precisa reiniciar o Apache para recarregar a configuração.

Veja também:

Verifique também :doc:`Django's documentation <django:ref/settings>`para parâmetros de configuração do próprio Django.

2.16.1 AKISMET_API_KEY

O Weblate pode usar o Akismet para procurar sugestões recebidas anonimamente por spam. Visite akismet.com para comprar uma chave API e associá-la a um site.

2.16.2 ANONYMOUS_USER_NAME

O nome de utilizadores não autenticados.

Veja também:

Controlo de acesso

2.16.3 AUDITLOG_EXPIRY

Novo na versão 3.6.

Quantos dias o Weblate deve manter registos de auditoria, que contêm informações sobre a atividade da conta.

A predefinição é de 180 dias.

2.16.4 AUTH_LOCK_ATTEMPTS

Novo na versão 2.14.

Quantidade máxima de tentativas de autenticação que falharam antes da aplicação da limitação de taxa.

Atualmente, isto é aplicado nos locais seguintes:

- Logins. Apaga a palavra-passe da conta, impedindo que o utilizador entre sem solicitar uma nova palavra-passe.
- Redefinições de palavra-passe. Impede que novos e-mails sejam enviados, evitando o envio de spam aos utilizadores com muitas tentativas de redefinição de palavra-passe.

A predefinição é 10.

Veja também:

Limitação de taxa,

2.16.5 AUTO_UPDATE

Novo na versão 3.2.

Alterado na versão 3.11: A opção original de ligar/desligar foi alterada para diferenciar quais cadeias são aceites.

Atualiza todos repositórios diariamente.

Dica: Útil se não estiver a user *Hooks de notificação* para atualizar os repositórios do Weblate automaticamente.

Nota: Existem opções de ligar/desligar, além da seleção de cadeias para compatibilidade com versões anteriores.

As opções são:

`"none"` Sem atualizações diárias.

`"remote"` e também `False` Atualizar apenas os repositórios remotos.

`"full"` e também `True` Atualizar repositórios remotos e mesclar a cópia de trabalho.

Nota: Isto requer que *Tarefas de fundo a usar o Celery* esteja a funcionar e entrará em vigor após ser reiniciado.

2.16.6 AVATAR_URL_PREFIX

Prefixo para construção de URLs de avatars: `${AVATAR_URL_PREFIX}/avatar/${MAIL_HASH}?${PARAMS}`. Os serviços seguintes funcionam:

Gravatar (predefinição), conforme https://gravatar.com/ `AVATAR_URL_PREFIX = 'https://www.gravatar.com/'`

Libravatar, conforme https://www.libravatar.org/ `AVATAR_URL_PREFIX = 'https://www.libravatar.org/'`

Veja também:

Cache de avatares, *ENABLE_AVATARS*, *Avatars*

2.16.7 AUTH_TOKEN_VALID

Novo na versão 2.14.

Por quanto tempo o token de autenticação e a palavra-passe temporária dos e-mails de redefinição de palavra-passe são válidos. Definido em número de segundos, usando 172800 (2 dias) como predefinição.

2.16.8 AUTH_PASSWORD_DAYS

Novo na versão 2.15.

Quantos dias a usar a mesma palavra-passe deve ser permitido.

Nota: Mudanças de palavra-passe feitas anteriormente ao Weblate 2.15 não serão consideradas para essa política.

A predefinição é de 180 dias.

2.16.9 AUTOFIX_LIST

Lista de correções automáticas para aplicar ao gravar uma cadeia.

Nota: Forneça um caminho totalmente qualificado à classe Python que implementa a interface de correção automática.

Correções disponíveis:

weblate.trans.autofixes.whitespace.SameBookendingWhitespace Corresponde o espaço em branco no início e no fim da cadeia com a fonte.

weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis Substitui pontos ao final (...) se a cadeia fonte tiver um carácter de reticências (…).

weblate.trans.autofixes.chars.RemoveZeroSpace Remove caracteres de espaço de largura zero se a fonte não contiver nenhum.

weblate.trans.autofixes.chars.RemoveControlChars Remove caracteres de controle se a fonte não contiver nenhum.

weblate.trans.autofixes.html.BleachHTML Remove a marcação HTML insegura das cadeias sinalizadas como `safe-html` (veja *HTML inseguro*).

Pode selecionar quais usar:

```
AUTOFIX_LIST = (
    'weblate.trans.autofixes.whitespace.SameBookendingWhitespace',
    'weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis',
)
```

Veja também:

Correções automáticas, *Correções automáticas personalizadas*

2.16.10 BASE_DIR

Diretório base onde as fontes do Weblate estão localizadas. Usado para derivar vários outros caminhos por predefinição:

- *DATA_DIR*

Valor predefinido: Diretório de nível superior de fontes do Weblate.

2.16.11 CSP_SCRIPT_SRC, CSP_IMG_SRC, CSP_CONNECT_SRC, CSP_STYLE_SRC, CSP_FONT_SRC

Personalize o cabeçalho `Content-Security-Policy` para Weblate. O cabeçalho é gerado automaticamente com base em integrações ativadas com serviços de terceiros (Matomo, Google Analytics, Sentry, …).

Todos esses tem uma lista vazia como predefinição.

** Exemplo:: **

```
# Enable Cloudflare Javascript optimizations
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

Veja também:

Política de segurança de conteúdo, Content Security Policy (CSP)

2.16.12 CHECK_LIST

Lista de verificações de qualidade para realizar numa tradução.

Nota: Forneça um caminho totalmente qualificado à classe Python que implementa a interface de verificação.

Ajuste a lista de verificações para incluir as relevantes para si.

Todas as *Verificações de qualidade* embutidas estão ativadas por predefinição, de onde pode alterar essas configurações. Por predefinição, eles são comentados em *Sample configuration* para que os valores predefinidos sejam usados. Novas verificações são realizadas para cada versão nova do Weblate.

Pode desativar todas as verificações:

```
CHECK_LIST = ()
```

Pode ativar apenas algumas:

```
CHECK_LIST = (
    'weblate.checks.chars.BeginNewlineCheck',
    'weblate.checks.chars.EndNewlineCheck',
    'weblate.checks.chars.MaxLengthCheck',
)
```

Nota: Alterar esta configuração afeta apenas as traduções recém-alteradas, as verificações existentes ainda serão armazenadas no banco de dados. Para também aplicar alterações nas traduções armazenadas, execute *update_checks*.

Veja também:

Verificações de qualidade, *Personalizar o comportamento*

2.16.13 COMMENT_CLEANUP_DAYS

Novo na versão 3.6.

Apaga comentários após uma determinada quantidade de dias. A predefinição é `None`, ou seja, nada apagado.

2.16.14 COMMIT_PENDING_HOURS

Novo na versão 2.10.

Quantidade de horas entre fazer o commit de alterações pendentes por meio da tarefa de segundo plano.

Veja também:

Component configuration, *Idade das alterações a fazer commit*, *Executar tarefas de manutenção*, *commit_pending*

2.16.15 DATA_DIR

A pasta na qual Weblate armazena todos os dados. Ela contém ligações para repositórios VCS, um índice de texto e vários ficheiros de configuração para ferramentas externas.

Os subdiretórios seguintes geralmente existem:

home O diretório pessoal usado para invocar scripts.

ssh Chaves e configuração de SSH.

static Localização predefinida para ficheiros estáticos de Django, especificados por `STATIC_ROOT`.

media Localização predefinida para ficheiros de mídia Django, especificado por `MEDIA_ROOT`.

vcs Repositórios de controle de versão.

backups Dados de backup diário. Confira *Dados despejados para backups* para detalhes.

Nota: Este diretório tem que ser escrito pelo Weblate. Executá-lo como uWSGI significa que o utilizador `www-data` deve ter acesso de escrita.

A maneira mais fácil de conseguir isto é fazer do utilizador o proprietário do diretório:

```
sudo chown www-data:www-data -R $DATA_DIR
```

A predefinição é `$BASE_DIR/data`.

Veja também:

BASE_DIR, *Fazer backup e mover o Weblate*

2.16.16 DATABASE_BACKUP

Novo na versão 3.1.

Se os backups de banco de dados devem ser armazenados como texto simples, compactado ou ignorado. Os valores autorizados são:

- `"plain"`
- `"compressed"`
- `"none"`

Veja também:

Fazer backup e mover o Weblate

2.16.17 DEFAULT_ACCESS_CONTROL

Novo na versão 3.3.

A configuração predefinida de controle de acesso para novos projetos:

`0` *Público*

`1` *Protegido*

`100` *Privado*

`200` *Personalizado*

Use *Personalizado* se está a gerir a ACL manualmente, o que significa não confiar na gestão interna do Weblate.

Veja também:

Controlo de acesso por projeto, *Controlo de acesso*, *Controlo de acesso*

2.16.18 DEFAULT_RESTRICTED_COMPONENT

Novo na versão 4.1.

O valor predefinido para a restrição de componentes.

Veja também:

Controlo de acesso por projeto, *Restricted access*, *Controlo de acesso*

2.16.19 DEFAULT_ADD_MESSAGE, DEFAULT_ADDON_MESSAGE, DEFAULT_COMMIT_MESSAGE, DEFAULT_DELETE_MESSAGE, DEFAULT_MERGE_MESSAGE

Enviar mensagens predefinidas para diferentes operações, consulte *Component configuration* para detalhes.

Veja também:

Template markup, *Component configuration*, *Commit, add, delete, merge and addon messages*

2.16.20 DEFAULT_ADDONS

Complementos predefinidos para instalar em cada componente criado.

Nota: Essa configuração afeta apenas componentes recém-criados.

Exemplo:

```
DEFAULT_ADDONS = {
     # Addon with no parameters
     "weblate.flags.target_edit": {},

     # Addon with parameters
     "weblate.autotranslate.autotranslate": {
          "mode": "suggest",
          "filter_type": "todo",
          "auto_source": "mt",
          "component": "",
          "engines": ["weblate-translation-memory"],
          "threshold": "80",
     }
}
```

Veja também:

install_addon

2.16.21 DEFAULT_COMMITER_EMAIL

Novo na versão 2.4.

Endereço de e-mail do committer para componentes de tradução criados com a predefinição `noreply@weblate.org`.

Veja também:

DEFAULT_COMMITER_NAME, *Component configuration*, *E-mail do publicador*

2.16.22 DEFAULT_COMMITER_NAME

Novo na versão 2.4.

Nome do committer para componentes de tradução criados com a predefinição `Weblate`.

Veja também:

DEFAULT_COMMITER_EMAIL, *Component configuration*, *Nome do publicador*

2.16.23 DEFAULT_LANGUAGE

Novo na versão 4.3.2.

Default source language to use for example in *Idioma fonte*.

Defaults to *en*. The matching language object needs to exist in the database.

Veja também:

Language definitions

2.16.24 DEFAULT_MERGE_STYLE

Novo na versão 3.4.

Mescla o estilo para quaisquer componentes novos.

- *rebase* - predefinição
- *merge*

Veja também:

Component configuration, *Estilo de união*

2.16.25 DEFAULT_TRANSLATION_PROPAGATION

Novo na versão 2.5.

Configuração predefinida para propagação de tradução, sendo a predefinição `True`.

Veja também:

Component configuration, *Permitir propagação da tradução*

2.16.26 DEFAULT_PULL_MESSAGE

Título para pull requests novas, sendo a predefinição `'Update from Weblate'`.

2.16.27 ENABLE_AVATARS

Se se deve ativar avatares baseados em Gravatar para os utilizadores. Por predefinição, isto está ativado.

Avatares são buscados e armazenados em cache no servidor, diminuindo o risco de vazamento de informações privadas, acelerando a experiência do utilizador.

Veja também:

Cache de avatares, *AVATAR_URL_PREFIX*, *Avatars*

2.16.28 ENABLE_HOOKS

Se se deve ativar ganchos remotos anônimos.

Veja também:

Hooks de notificação

2.16.29 ENABLE_HTTPS

Se se deve enviar ligações para Weblate como HTTPS ou HTTP. Esta configuração afeta os e-mails enviados e as URLs absolutas geradas.

Dica: Na configuração predefinida, isto também é usado para várias configurações de Django relacionadas ao HTTPS.

Veja também:

SESSION_COOKIE_SECURE, CSRF_COOKIE_SECURE, SECURE_SSL_REDIRECT, *Definir domínio correto do site*

2.16.30 ENABLE_SHARING

Ativa/desativa o menu *Compartilhar* para que os utilizadores possam compartilhar o progresso da tradução nas redes sociais.

2.16.31 GITLAB_CREDENTIALS

Novo na versão 4.3.

Lista para credenciais para servidores de GitLab.

Dica: Use isto no caso de querer que o Weblate interaja com mais deles, para um único ponto final do GitLab com *GITLAB_USERNAME* e *GITLAB_TOKEN*.

```
GITLAB_CREDENTIALS = {
    "gitlab.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "gitlab.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.32 GITLAB_USERNAME

O nome de utilizador GitLab para enviar merge requests para atualizações de tradução.

Veja também:

GITLAB_CREDENTIALS, *GitLab*

2.16.33 GITLAB_TOKEN

Novo na versão 4.3.

Token de acesso pessoal do GitLab usado para fazer chamadas API para atualizações de tradução.

Veja também:

GITLAB_CREDENTIALS,:ref:*vcs-gitlab*, GitLab: Ficha de acesso pessoal

2.16.34 GITHUB_CREDENTIALS

Novo na versão 4.3.

Lista para credenciais para servidores GitHub.

Dica: Use isto no caso de querer que o Weblate interaja com mais deles, para um único ponto final do GitHub com *GITHUB_USERNAME* e *GITHUB_TOKEN*.

```
GITHUB_CREDENTIALS = {
    "api.github.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "github.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.35 GITHUB_USERNAME

O nome de utilizador GitHub para enviar pull request para atualizações de tradução.

Veja também:

GITHUB_CREDENTIALS, *GitHub*

2.16.36 GITHUB_TOKEN

Novo na versão 4.3.

Token de acesso pessoal GitHub usado para fazer chamadas API para enviar pull requests de tradução.

Veja também:

GITHUB_CREDENTIALS,:ref:*vcs-github*, Criar um token de acesso pessoal

2.16.37 GOOGLE_ANALYTICS_ID

ID do Google Analytics para ativar o monitoramento do Weblate a usar o Google Analytics.

2.16.38 HIDE_REPO_CREDENTIALS

Oculta credenciais de repositório na interface web. No caso de ter o URL do repositório com utilizador e palavra-passe, o Weblate irá ocultá-la quando as informações relacionadas são mostradas aos utilizadores.

Por exemplo, em vez de `https://utilizador:palavra-passe@git.example.com/repo.git`, vai mostrar apenas “”https://git.example.com/repo.git””. Tenta limpar mensagens de erro VCS também de forma semelhante.

Nota: Isso está ativado por predefinição.

2.16.39 HIDE_VERSION

Novo na versão 4.3.1.

Hides version information from unauthenticated users. This also makes all documentation links point to latest version instead of the documentation matching currently installed version.

Ocultar a versão é uma prática de segurança recomendada em algumas empresas, mas não prevê invasores de descobrir a versão a sondar o comportamento.

Nota: Isto está desativado por predefinição.

2.16.40 IP_BEHIND_REVERSE_PROXY

Novo na versão 2.14.

Indica se o Weblate está a ser usado através de um proxy reverso.

Se for definido como `True`, o Weblate obtém o endereço IP de um cabeçalho definido por *IP_PROXY_HEADER*.

Aviso: Certifique-se de que está realmente a usar um proxy reverso e que ele define este cabeçalho, caso contrário, os utilizadores poderão falsificar o endereço IP.

Nota: Isto está ligado por predefinição.

Veja também:

Executar por trás de um proxy reverso, *Limitação de taxa*, *IP_PROXY_HEADER*, *IP_PROXY_OFFSET*

2.16.41 IP_PROXY_HEADER

Novo na versão 2.14.

Indica de qual cabeçalho o Weblate deve obter o endereço IP quando *IP_BEHIND_REVERSE_PROXY* está ativado.

A predefinição é `HTTP_X_FORWARDED_FOR`.

Veja também:

Executar por trás de um proxy reverso, *Limitação de taxa*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_OFFSET*

2.16.42 IP_PROXY_OFFSET

Novo na versão 2.14.

Indica qual parte de *IP_PROXY_HEADER* é usada como endereço IP do cliente.

Dependendo da configuração, este cabeçalho pode consistir em vários endereços IP (por exemplo, `X-Forwarded-For: a, b, client-ip`) e pode configurar qual endereço do cabeçalho é usado como endereço IP do cliente aqui.

Aviso: Configurar isto afeta a segurança da sua instalação, por isso deve configurá-la só para usar proxies confiáveis para determinar o endereço IP.

A predefinição é 0.

Veja também:

Executar por trás de um proxy reverso, *Limitação de taxa*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_HEADER*

2.16.43 LEGAL_URL

Novo na versão 3.5.

URL onde a sua instância de Weblate mostra os documentos legais dela.

Dica: Útil se hospeda os seus documentos legais fora do Weblate para incorporá-los ao Weblate, verifique *Legal* para obter detalhes.

Exemplo:

```
LEGAL_URL = "https://weblate.org/terms/"
```

2.16.44 LICENSE_EXTRA

Licenças adicionais para incluir nas opções de licença.

Nota: Cada definição de licença deve ser uma tupla do seu nome curto, um nome longo e uma URL.

Por exemplo:

```
LICENSE_EXTRA = [
    (
        "AGPL-3.0",
        "GNU Affero General Public License v3.0",
        "https://www.gnu.org/licenses/agpl-3.0-standalone.html",
    ),
]
```

2.16.45 LICENSE_FILTER

Alterado na versão 4.3: A configurar este para valor em branco desativa o alerta de licença.

Filter list of licenses to show. This also disables the license alert when set to empty.

Nota: Este filtro usa os nomes de licença curtos.

Por exemplo:

```
LICENSE_FILTER = {"AGPL-3.0", "GPL-3.0-or-later"}
```

O seguinte desativa o alerta de licença:

```
LICENSE_FILTER = set()
```

Veja também:

Translation component alerts

2.16.46 LICENSE_REQUIRED

Define se o atributo de licença em *Component configuration* é necessário.

Nota: Isto está desativado por predefinição.

2.16.47 LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH

Se o comprimento de uma determinada tradução deve ser limitado. A restrição é o comprimento da cadeia fonte * 10 caracteres.

Dica: Define isto como `False` para permitir traduções mais longas (até 10.000 caracteres) independentemente do comprimento da cadeia fonte.

Nota: A predefinição é «True».

2.16.48 LOCALIZE_CDN_URL e LOCALIZE_CDN_PATH

Essas configurações definem a extensão *CDN de localização JavaScript*. *LOCALIZE_CDN_URL* define a URL raiz onde o CDN de localização está disponível e *LOCALIZE_CDN_PATH* define o caminho onde o Weblate deve armazenar ficheiros gerados que serão servidos em *LOCALIZE_CDN_URL*.

Dica: O Hosted Weblate usa o `https://weblate-cdn.com/`.

Veja também:

CDN de localização JavaScript

2.16.49 LOGIN_REQUIRED_URLS

Uma lista de URLs para as quais deseja exigir autenticação. (Além das regras predefinidas incorporadas ao Weblate).

Dica: Isto permite que proteja toda a instalação com uma palavra-passe a usar:

```
LOGIN_REQUIRED_URLS = (
    r'/(.*)$',
)
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

Dica: É desejável bloquear o acesso à API também, como mostrado no exemplo acima.

2.16.50 LOGIN_REQUIRED_URLS_EXCEPTIONS

Lista de exceções para *LOGIN_REQUIRED_URLS*. Se não for especificado, os utilizadores podem acessar a página de autenticação.

Algumas das exceções que pode incluir:

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (
    r'/accounts/(.*)$', # Required for login
    r'/static/(.*)$',   # Required for development mode
    r'/widgets/(.*)$',  # Allowing public access to widgets
    r'/data/(.*)$',     # Allowing public access to data exports
    r'/hooks/(.*)$',    # Allowing public access to notification hooks
    r'/api/(.*)$',      # Allowing access to API
    r'/js/i18n/$',      # JavaScript localization
)
```

2.16.51 MATOMO_SITE_ID

ID de um site em Matomo (anteriormente Piwik) que quer rastrear.

Nota: Esta integração não suporta o Matomo Tag Manager.

Veja também:

MATOMO_URL

2.16.52 MATOMO_URL

URL completa (incluindo barra ao final) de uma instalação Matomo (anteriormente Piwik) que deseja usar para rastrear o uso do Weblate. Por favor, consulte <https://matomo.org/> para mais detalhes.

Dica: Esta integração não suporta o Matomo Tag Manager.

Por exemplo:

```
MATOMO_SITE_ID = 1
MATOMO_URL = "https://example.matomo.cloud/"
```

Veja também:

MATOMO_SITE_ID

2.16.53 MT_SERVICES

Alterado na versão 3.0: A configuração foi renomeada de `MACHINE_TRANSLATION_SERVICES` para `MT_SERVICES` para ser consistente com outras configurações de tradução de máquina.

Lista de serviços de tradução de máquina ativados para uso.

Nota: Muitos dos serviços precisam de configuração adicional, como chaves de API, consulte a sua documentação *Tradução automática* para mais detalhes.

```
MT_SERVICES = (
    'weblate.machinery.apertium.ApertiumAPYTranslation',
    'weblate.machinery.deepl.DeepLTranslation',
    'weblate.machinery.glosbe.GlosbeTranslation',
    'weblate.machinery.google.GoogleTranslation',
    'weblate.machinery.microsoft.MicrosoftCognitiveTranslation',
    'weblate.machinery.microsoftterminology.MicrosoftTerminologyService',
    'weblate.machinery.mymemory.MyMemoryTranslation',
    'weblate.machinery.tmserver.AmagamaTranslation',
    'weblate.machinery.tmserver.TMServerTranslation',
    'weblate.machinery.yandex.YandexTranslation',
    'weblate.machinery.weblatetm.WeblateTranslation',
    'weblate.machinery.saptranslationhub.SAPTranslationHub',
    'weblate.memory.machine.WeblateMemory',
)
```

Veja também:

Tradução automática, *Sugestões automáticas*

2.16.54 MT_APERTIUM_APY

URL do servidor Apertium-APy, https://wiki.apertium.org/wiki/Apertium-apy

Veja também:

Apertium, *Tradução automática*, *Sugestões automáticas*

2.16.55 MT_AWS_ACCESS_KEY_ID

ID da chave de acesso para Amazon Translate.

Veja também:

AWS, *Tradução automática*, *Sugestões automáticas*

2.16.56 MT_AWS_SECRET_ACCESS_KEY

Chave secreta da API para o Amazon Translate.

Veja também:

AWS, *Tradução automática*, *Sugestões automáticas*

2.16.57 MT_AWS_REGION

Nome da região para usar no Amazon Translate.

Veja também:

AWS, *Tradução automática*, *Sugestões automáticas*

2.16.58 MT_BAIDU_ID

ID do cliente para a API do Baidu Zhiyun, pode registar-se em https://api.fanyi.baidu.com/api/trans/product/index

Veja também:

Baidu API machine translation, *Tradução automática*, *Sugestões automáticas*

2.16.59 MT_BAIDU_SECRET

Segredo do cliente para a API do Baidu Zhiyun, pode registar-se em https://api.fanyi.baidu.com/api/trans/product/index

Veja também:

Baidu API machine translation, *Tradução automática*, *Sugestões automáticas*

2.16.60 MT_DEEPL_API_VERSION

Novo na versão 4.1.1.

Versão da API para usar com o serviço DeepL. A versão limita o escopo de uso:

v1 Destina-se a ferramentas CAT e é utilizável com assinatura baseada no utilizador.

v2 Destina-se ao uso da API e a assinatura é baseada em uso.

Anteriormente, o Weblate era classificado como uma ferramenta CAT pelo DeepL, por isso deveria usar a API v1, mas agora é entendido que deve usar a API v2. Portanto, a predefinição é v2 e pode alterar-lo para v1 no caso de ter uma assinatura CAT existente e querer que o Weblate use isso.

Veja também:

DeepL, *Tradução automática*, *Sugestões automáticas*

2.16.61 MT_DEEPL_KEY

Chave de API ao API do DeepL, pode registar-se em https://www.deepl.com/pro.html

Veja também:

DeepL, *Tradução automática*, *Sugestões automáticas*

2.16.62 MT_GOOGLE_KEY

Chave de API para a API v2 do Google Translate, pode registar-se em https://cloud.google.com/translate/docs

Veja também:

Google Translate, *Tradução automática*, *Sugestões automáticas*

2.16.63 MT_GOOGLE_CREDENTIALS

Ficheiro de credenciais da API v3 do JSON obtido no console de nuvem do Google. Por favor, forneça um caminho completo do sistema operacional. As credenciais são por conta de serviço afiliada ao projeto determinado . Por favor, verifique https://cloud.google.com/docs/authentication/getting-started para mais detalhes.

2.16.64 MT_GOOGLE_PROJECT

ID de projeto da API v3 do Google Cloud com serviço de tradução ativado e faturamento ativado. Por favor consulte https://cloud.google.com/appengine/docs/standard/nodejs/building-app/creating-project para mais detalhes

2.16.65 MT_GOOGLE_LOCATION

A API v3 do App Engine do Google Cloud pode ser específica para uma localidade. Altere conforme o caso, se a predefinição``global`` não lhe servir.

Consulte https://cloud.google.com/appengine/docs/locations para mais detalhes

Veja também:

Google Translate API V3 (Advanced)

2.16.66 MT_MICROSOFT_BASE_URL

Domínio de URL base da região conforme definido na secção «URLs base».

A predefinição é `api.cognitive.microsofttranslator.com` para o Azure Global.

Para Azure China use `api.translator.azure.cn`.

2.16.67 MT_MICROSOFT_COGNITIVE_KEY

Chave do cliente para a API do Microsoft Cognitive Services Translator.

Veja também:

Microsoft Cognitive Services Translator, *Tradução automática*, *Sugestões automáticas*, **`Cognitive Services - API de tradução de texto<https://azure.microsoft.com/services/cognitive-services/translator-text-api/>`_**, Microsoft Azure Portal

2.16.68 MT_MICROSOFT_REGION

Prefixo da região conforme definido na secção «Autenticar com um recurso de vários serviços».

2.16.69 MT_MICROSOFT_ENDPOINT_URL

Domínio de URL de extremidade da região para token de acesso definido na secção «Autenticando com um token de acesso».

A predefinição é `api.cognitive.microsoft.com` para Azure Global.

Para Azure China, use o desfecho do Portal do Azure.

2.16.70 MT_MODERNMT_KEY

Chave API ao motor de tradução automática ModernMT.

Veja também:

ModernMT `MT_MODERNMT_URL`

2.16.71 MT_MODERNMT_URL

URL de ModernMT. A predefinição é `https://api.modernmt.com/` para o serviço de nuvem.

Veja também:

ModernMT `MT_MODERNMT_KEY`

2.16.72 MT_MYMEMORY_EMAIL

Endereço de e-mail de identificação do myMemory. Permite 1000 solicitações por dia.

Veja também:

MyMemory, *Tradução automática*, *Sugestões automáticas*, MyMemory: API technical specifications

2.16.73 MT_MYMEMORY_KEY

Chave de acesso do MyMemory para memória de tradução privada. Use-a com `MT_MYMEMORY_USER`.

Veja também:

MyMemory, *Tradução automática*, *Sugestões automáticas*, MyMemory: API key generator

2.16.74 MT_MYMEMORY_USER

ID de utilizador do MyMemory para a memória de tradução privada. Use-o com `MT_MYMEMORY_KEY`.

Veja também:

MyMemory, *Tradução automática*, *Sugestões automáticas*, MyMemory: API key generator

2.16.75 MT_NETEASE_KEY

Chave de app para API da NetEase Sight, pode registar-se em https://sight.netease.com/

Veja também:

NetEase Sight API machine translation, *Tradução automática*, *Sugestões automáticas*

2.16.76 MT_NETEASE_SECRET

Segredo de app para a API da NetEase Sight, pode registar-se em https://sight.netease.com/

Veja também:

NetEase Sight API machine translation, *Tradução automática*, *Sugestões automáticas*

2.16.77 MT_TMSERVER

URL onde o tmserver está funcionando.

Veja também:

tmserver, *Tradução automática*, *Sugestões automáticas*, tmserver

2.16.78 MT_YANDEX_KEY

Chave de API para a API do Yandex Translate, pode registar-se em https://yandex.com/dev/translate/

Veja também:

Yandex Translate, *Tradução automática*, *Sugestões automáticas*

2.16.79 MT_YOUDAO_ID

ID do cliente para a API do Youdao Zhiyun, pode registar-se em https://ai.youdao.com/product-fanyi-text.s.

Veja também:

Youdao Zhiyun API machine translation, *Tradução automática*, *Sugestões automáticas*

2.16.80 MT_YOUDAO_SECRET

Segredo do cliente para a API do Youdao Zhiyun, pode registar-se em https://ai.youdao.com/product-fanyi-text.s.

Veja também:

Youdao Zhiyun API machine translation, *Tradução automática*, *Sugestões automáticas*

2.16.81 MT_SAP_BASE_URL

URL de API ao serviço SAP Translation Hub.

Veja também:

SAP Translation Hub, *Tradução automática*, *Sugestões automáticas*

2.16.82 MT_SAP_SANDBOX_APIKEY

Chave de API para uso de API em caixa de proteção

Veja também:

SAP Translation Hub, *Tradução automática*, *Sugestões automáticas*

2.16.83 MT_SAP_USERNAME

O seu nome de utilizador da SAP

Veja também:

SAP Translation Hub, *Tradução automática*, *Sugestões automáticas*

2.16.84 MT_SAP_PASSWORD

A sua palavra-passe da SAP

Veja também:

SAP Translation Hub, *Tradução automática*, *Sugestões automáticas*

2.16.85 MT_SAP_USE_MT

Se se deve também usar serviços de tradução de máquina, além do banco de dados de termos. Valores possíveis: `True` ou `False`

Veja também:

SAP Translation Hub, *Tradução automática*, *Sugestões automáticas*

2.16.86 NEARBY_MESSAGES

Quantas cadeia devem ser mostradas em torno da cadeia atualmente traduzida. Este é apenas um valor predefinido, os utilizadores podem ajustar-lo em *Perfil do utilizador*.

2.16.87 PAGURE_CREDENTIALS

Novo na versão 4.3.2.

Lista para credenciais para servidores de Pagure.

Dica: Use isto no caso de querer que o Weblate interaja com mais deles, para um único ponto final do Pagure com *PAGURE_USERNAME* e *PAGURE_TOKEN*.

```
PAGURE_CREDENTIALS = {
    "pagure.io": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "pagure.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.88 PAGURE_USERNAME

Novo na versão 4.3.2.

O nome de utilizador no Pagure para enviar merge requests para atualizações de tradução.

Veja também:

PAGURE_CREDENTIALS, *Pagure*

2.16.89 PAGURE_TOKEN

Novo na versão 4.3.2.

Token de acesso pessoal do Pagure usado para fazer chamadas API para atualizações de tradução.

Veja também:

PAGURE_CREDENTIALS, *Pagure*, Pagure API

2.16.90 RATELIMIT_ATTEMPTS

Novo na versão 3.2.

A quantidade máxima de tentativas de autenticação antes da limitação da taxa ser aplicada.

A predefinição é 5.

Veja também:

Limitação de taxa, *RATELIMIT_WINDOW*, *RATELIMIT_LOCKOUT*

2.16.91 RATELIMIT_WINDOW

Novo na versão 3.2.

Por quanto tempo a autenticação é aceita após a limitação da taxa ser aplicada.

Uma quantidade de segundos com a predefinição de 300 (5 minutos).

Veja também:

Limitação de taxa, *RATELIMIT_ATTEMPTS*, *RATELIMIT_LOCKOUT*

2.16.92 RATELIMIT_LOCKOUT

Novo na versão 3.2.

Por quanto tempo a autenticação é bloqueada após a limitação da taxa ser aplicada.

Uma quantidade de segundos com a predefinição de 600 (10 minutos).

Veja também:

Limitação de taxa, *RATELIMIT_ATTEMPTS*, *RATELIMIT_WINDOW*

2.16.93 REGISTRATION_ALLOW_BACKENDS

Novo na versão 4.1.

A lista de backends de autenticação de onde permite o registo. Isso só limita novos registos, os utilizadores ainda se podem autenticar e adicionar autenticação por todos os backends de autenticação configurados.

É recomendado para manter *REGISTRATION_OPEN* ativado enquanto limita os backends de registo, caso contrário, os utilizadores poderão se registar, mas o Weblate não mostrará ligações para se registar na interface do utilizador.

Exemplo:

```
REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]
```

Dica: Os nomes de backend correspondem aos nomes usados na URL para autenticação.

Veja também:

REGISTRATION_OPEN, *Autenticação*

2.16.94 REGISTRATION_CAPTCHA

Um valor de `True` ou `False` indicando se o registo de contas novas é protegido pelo CAPTCHA. Esta configuração é opcional e uma predifinição de `True` será presumido se não for fornecido.

Se for ativado, um CAPTCHA é adicionado a todas as páginas onde um utilizador digita o endereço de e-mail dele:

- Registo de uma conta nova.
- Recuperação de palavra-passe.
- Adição de uma e-mail a uma conta.
- Formulário de contacto para utilizadores que não estão autenticados.

2.16.95 REGISTRATION_EMAIL_MATCH

Novo na versão 2.17.

Permite filtrar quais endereços de e-mail podem ser registados.

A predefinição é `.*`, que permite que registar qualquer endereço de e-mail.

Pode usá-lo para restringir o registo a um único domínio de e-mail:

```
REGISTRATION_EMAIL_MATCH = r'^.*@weblate\.org$'
```

2.16.96 REGISTRATION_OPEN

Se o registo de contas novas é atualmente permitido. Esta configuração opcional pode permanecer com a predefinição `True` ou pode ser alterada para `Falsa`.

Esta configuração afeta a autenticação embutida por endereço de e-mail ou através do Python Social Auth (pode listar certos back-ends a usar *REGISTRATION_ALLOW_BACKENDS*).

Nota: Se estiver a usar métodos de autenticação de terceiros, como *Autenticação por LDAP*, ele apenas oculta o formulário de registo, mas novos utilizadores ainda conseguem se autenticar e criar contas.

Veja também:

REGISTRATION_ALLOW_BACKENDS, *REGISTRATION_EMAIL_MATCH*, *Autenticação*

2.16.97 REPOSITORY_ALERT_THRESHOLD

Novo na versão 4.0.2.

Limiar para acionar um alerta para repositórios desatualizados ou aqueles que contêm muitas alterações. A predefinçao é 25.

Veja também:

Translation component alerts

2.16.98 REQUIRE_LOGIN

Novo na versão 4.1.

Isso ativa :configuração`URLS_DE_LOGIN_NECESSÁRIOS` e configura o framework REST a requisitar login para todos os pontos finais da API.

Nota: Isto é implementado no: ref:*sample-configuration*. Para Docker, use `WEBLATE_REQUIRE_LOGIN`.

2.16.99 SENTRY_DSN

Novo na versão 3.9.

DSN do Sentry para usar para *Collecting error reports*.

Veja também:

Integração Django para o Sentry

2.16.100 SESSION_COOKIE_AGE_AUTHENTICATED

Novo na versão 4.3.

Set session expiry for authenticated users. This complements `SESSION_COOKIE_AGE` which is used for unauthenticated users.

Veja também:

`SESSION_COOKIE_AGE`

2.16.101 SIMPLIFY_LANGUAGES

Use códigos de idioma simples para combinações predfinidas de idioma/país. Por exemplo, uma tradução de `fr_FR` usará o código de idioma `fr`. Este é geralmente o comportamento desejado, pois simplifica a lista de idiomas para essas combinações predefinidas.

Desative isto se quiser traduções diferentes para cada variante.

2.16.102 SITE_DOMAIN

Configura o domínio do site. Isso é necessário para produzir ligações absolutas corretas em muitos escopos (por exemplo, ativação de e-mails, notificações ou feeds RSS).

No caso de o Weblate estar a ser executado num porte fora do padrão, inclua-a aqui também.

Exemplos::

```
# Production site with domain name
SITE_DOMAIN = "weblate.example.com"

# Local development with IP address and port
SITE_DOMAIN = "127.0.0.1:8000"
```

Nota: Esta configuração deve conter apenas o nome de domínio. Para configurar o protocolo (ativar e aplicar HTTPS), use *ENABLE_HTTPS* e para alterar o URL, use *URL_PREFIX*.

Dica: Num contentor Docker, o domínio do site é configurado através de *WEBLATE_ALLOWED_HOSTS*.

Veja também:

Definir domínio correto do site, *Configuração de hosts permitidos*, *Configurar HTTPS corretamente* *WEBLATE_SITE_DOMAIN*, *ENABLE_HTTPS*

2.16.103 SITE_TITLE

Título do site a ser usado para o site e e-mails enviados.

2.16.104 SPECIAL_CHARS

Caracteres adicionais para incluir no teclado visual, *Teclado visual*.

O valor predefinido é:

```
SPECIAL_CHARS = ('\t', '\n', '…')
```

2.16.105 SINGLE_PROJECT

Novo na versão 3.8.

Redireciona os utilizadores diretamente para um projeto ou componente em vez de mostrar o painel. Pode configurá-lo como `True` e, neste caso, só funciona no caso de haver realmente apenas um único projeto no Weblate. Alternativamente, define o projeto e redirecionará incondicionalmente para este projeto.

Alterado na versão 3.11: A configuração agora também aceita um slug de projeto, para forçar a exibição desse único projeto.

Exemplo:

```
SINGLE_PROJECT = "test"
```

2.16.106 STATUS_URL

A URL onde a sua instância de Weblate relata o estado dela.

2.16.107 SUGGESTION_CLEANUP_DAYS

Novo na versão 3.2.1.

Apaga sugestões automaticamente após uma determinada quantidade de dias. A predeinição é `None`, ou seja, sem exclusões.

2.16.108 UPDATE_LANGUAGES

Novo na versão 4.3.2.

Controls whether languages database should be updated when running database migration and is enabled by default. This setting has not effect on invocation of *`setuplang`*.

Veja também:

Built-in language definitions

2.16.109 URL_PREFIX

Esta configuração permite que execute Weblate em algum caminho (caso contrário, depende de ser executado a partir da raiz do servidor web).

Nota: Para usar esta configuração, também precisa configurar o seu servidor para remover este prefixo. Por exemplo, com o WSGI, isso pode ser alcançado definindo `WSGIScriptAlias`.

Dica: O prefixo deve iniciar com um `/`.

Exemplo:

```
URL_PREFIX = '/translations'
```

Nota: Esta configuração não funciona com o servidor embutido do Django, teria que ajustar `urls.py` para conter este prefixo.

2.16.110 VCS_BACKENDS

Configuração de backends VCS disponíveis.

Nota: Weblate tenta usar todos os back-ends suportados para os seus utilizadores.

Dica: Pode limitar escolhas ou adicionar back-ends VCS personalizados a usar isto.

```
VCS_BACKENDS = (
    'weblate.vcs.git.GitRepository',
)
```

Veja também:

Integração de controlo de versões

2.16.111 VCS_CLONE_DEPTH

Novo na versão 3.10.2.

Configura a profundidade a clonagem de repositórios Weblate deve ter.

Nota: Atualmente, isto só é suportado em *Git*. Por predefinição, o Weblate faz clones rasos dos repositórios para tornar a clonagem mais rápida e economizar espaço no disco. Dependendo do seu uso (por exemplo, ao usar o personalizado *Extensões*), pode aumentar a profundidade ou desligar os clones rasos completamente definindo isso para 0.

Dica: No caso de receber o erro `fatal: protocol error: expected old/new/ref, got 'shallow <hash de commit>'` ao fazer push do Weblate, desative clones rasos completamente configurando:

```
VCS_CLONE_DEPTH = 0
```

2.16.112 WEBLATE_ADDONS

Lista de extensões disponíveis para uso. Para usá-las, elas devem ser ativadas para um determinado componente de tradução. Por predefinição, isto inclui todas as extensões embutidas, ao estender a lista, provavelmente vai manter as existentes ativadas, por exemplo:

```
WEBLATE_ADDONS = (
    # Built-in addons
    "weblate.addons.gettext.GenerateMoAddon",
    "weblate.addons.gettext.UpdateLinguasAddon",
    "weblate.addons.gettext.UpdateConfigureAddon",
    "weblate.addons.gettext.MsgmergeAddon",
    "weblate.addons.gettext.GettextCustomizeAddon",
    "weblate.addons.gettext.GettextAuthorComments",
    "weblate.addons.cleanup.CleanupAddon",
    "weblate.addons.consistency.LangaugeConsistencyAddon",
    "weblate.addons.discovery.DiscoveryAddon",
    "weblate.addons.flags.SourceEditAddon",
    "weblate.addons.flags.TargetEditAddon",
    "weblate.addons.flags.SameEditAddon",
    "weblate.addons.flags.BulkEditAddon",
    "weblate.addons.generate.GenerateFileAddon",
    "weblate.addons.json.JSONCustomizeAddon",
    "weblate.addons.properties.PropertiesSortAddon",
    "weblate.addons.git.GitSquashAddon",
    "weblate.addons.removal.RemoveComments",
    "weblate.addons.removal.RemoveSuggestions",
    "weblate.addons.resx.ResxUpdateAddon",
    "weblate.addons.autotranslate.AutoTranslateAddon",
    "weblate.addons.yaml.YAMLCustomizeAddon",
    "weblate.addons.cdn.CDNJSAddon",
```

(continues on next page)

(continuação da página anterior)

```
    # Addon you want to include
    "weblate.addons.example.ExampleAddon",
)
```

Veja também:

Extensões

2.16.113 WEBLATE_EXPORTERS

Novo na versão 4.2.

Lista de exportadores disponíveis que oferecem descarregar traduções ou glossários em vários formatos de ficheiro.

Veja também:

Formatos de ficheiros suportados

2.16.114 WEBLATE_FORMATS

Novo na versão 3.0.

Lista de formatos de ficheiro disponíveis para uso.

Nota: A lista predfinida já tem os formatos comuns.

Veja também:

Formatos de ficheiros suportados

2.16.115 WEBLATE_GPG_IDENTITY

Novo na versão 3.1.

Identidade usada pelo Weblate para assinar os commits Git, por exemplo:

```
WEBLATE_GPG_IDENTITY = 'Weblate <weblate@example.com>'
```

O chaveiro GPG do Weblate é pesquisado por uma chave correspondente (`home/.gnupg` em *DATA_DIR*). Se não for encontrado, uma chave é gerada. Consulte *Signing Git commits with GnuPG* para mais detalhes.

Veja também:

Signing Git commits with GnuPG

2.17 Sample configuration

The following example is shipped as `weblate/settings_example.py` with Weblate:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
```

(continues on next page)

(continuação da página anterior)

```
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#


import os
import platform
from logging.handlers import SysLogHandler

#
# Django settings for Weblate project.
#

DEBUG = True

ADMINS = (
    # ("Your Name", "your_email@example.com"),
)

MANAGERS = ADMINS

DATABASES = {
    "default": {
        # Use "postgresql" or "mysql".
        "ENGINE": "django.db.backends.postgresql",
        # Database name.
        "NAME": "weblate",
        # Database user.
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for authentication.
        # "ALTER_ROLE": "weblate",
        # Database password.
        "PASSWORD": "",
        # Set to empty string for localhost.
        "HOST": "127.0.0.1",
        # Set to empty string for default.
        "PORT": "",
        # Customizations for databases.
        "OPTIONS": {
            # In case of using an older MySQL server,
            # which has MyISAM as a default storage
            # "init_command": "SET storage_engine=INNODB",
            # Uncomment for MySQL older than 5.7:
            # "init_command": "SET sql_mode='STRICT_TRANS_TABLES'",
            # Set emoji capable charset for MySQL:
            # "charset": "utf8mb4",
            # Change connection timeout in case you get MySQL gone away error:
            # "connect_timeout": 28800,
        },
    }
}
```

(continues on next page)

(continuação da página anterior)

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Data directory
DATA_DIR = os.path.join(BASE_DIR, "data")

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# In a Windows environment this must be set to your system time zone.
TIME_ZONE = "UTC"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

LANGUAGES = (
    ("ar", "ةيبرعلا"),
    ("az", "Azərbaycan"),
    ("be", "Беларуская"),
    ("be@latin", "Biełaruskaja"),
    ("bg", "Български"),
    ("br", "Brezhoneg"),
    ("ca", "Català"),
    ("cs", "Čeština"),
    ("da", "Dansk"),
    ("de", "Deutsch"),
    ("en", "English"),
    ("el", "Ελληνικά"),
    ("en-gb", "English (United Kingdom)"),
    ("es", "Español"),
    ("fi", "Suomi"),
    ("fr", "Français"),
    ("gl", "Galego"),
    ("he", "עברית"),
    ("hu", "Magyar"),
    ("hr", "Hrvatski"),
    ("id", "Indonesia"),
    ("is", "Íslenska"),
    ("it", "Italiano"),
    ("ja", "□□□"),
    ("kab", "Taqbaylit"),
    ("kk", "Қазақ тілі"),
    ("ko", "□□□"),
    ("nb", "Norsk bokmål"),
    ("nl", "Nederlands"),
    ("pl", "Polski"),
    ("pt", "Português"),
    ("pt-br", "Português brasileiro"),
    ("ru", "Русский"),
    ("sk", "Slovenčina"),
    ("sl", "Slovenščina"),
    ("sq", "Shqip"),
    ("sr", "Српски"),
    ("sv", "Svenska"),
    ("tr", "Türkçe"),
    ("uk", "Українська"),
    ("zh-hans", "□□□"),
    ("zh-hant", "□□□"),
)
```

(continues on next page)

(continuação da página anterior)

```
SITE_ID = 1

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# URL prefix to use, please see documentation for more details
URL_PREFIX = ""

# Absolute filesystem path to the directory that will hold user-uploaded files.
MEDIA_ROOT = os.path.join(DATA_DIR, "media")

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash.
MEDIA_URL = f"{URL_PREFIX}/media/"

# Absolute path to the directory static files should be collected to.
# Don't put anything in this directory yourself; store your static files
# in apps' "static/" subdirectories and in STATICFILES_DIRS.
STATIC_ROOT = os.path.join(DATA_DIR, "static")

# URL prefix for static files.
STATIC_URL = f"{URL_PREFIX}/static/"

# Additional locations of static files
STATICFILES_DIRS = (
    # Put strings here, like "/home/html/static" or "C:/www/django/static".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
)

# List of finder classes that know how to find static files in
# various locations.
STATICFILES_FINDERS = (
    "django.contrib.staticfiles.finders.FileSystemFinder",
    "django.contrib.staticfiles.finders.AppDirectoriesFinder",
    "compressor.finders.CompressorFinder",
)

# Make this unique, and don't share it with anybody.
# You can generate it using weblate/examples/generate-secret-key
SECRET_KEY = ""

_TEMPLATE_LOADERS = [
    "django.template.loaders.filesystem.Loader",
    "django.template.loaders.app_directories.Loader",
]
if not DEBUG:
    _TEMPLATE_LOADERS = [("django.template.loaders.cached.Loader", _TEMPLATE_
↪LOADERS)]
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "OPTIONS": {
```

(continues on next page)

(continuação da página anterior)

```
            "context_processors": [
                "django.contrib.auth.context_processors.auth",
                "django.template.context_processors.debug",
                "django.template.context_processors.i18n",
                "django.template.context_processors.request",
                "django.template.context_processors.csrf",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": _TEMPLATE_LOADERS,
        },
    }
]


# GitHub username for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None
GITHUB_TOKEN = None

# GitLab username for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = None
GITLAB_TOKEN = None

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    # "social_core.backends.google.GoogleOAuth2",
    # "social_core.backends.github.GithubOAuth2",
    # "social_core.backends.bitbucket.BitbucketOAuth",
    # "social_core.backends.suse.OpenSUSEOpenId",
    # "social_core.backends.ubuntu.UbuntuOpenId",
    # "social_core.backends.fedora.FedoraOpenId",
    # "social_core.backends.facebook.FacebookOAuth2",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Custom user model
AUTH_USER_MODEL = "weblate_auth.User"

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = ""
SOCIAL_AUTH_GITHUB_SECRET = ""
SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]

SOCIAL_AUTH_BITBUCKET_KEY = ""
SOCIAL_AUTH_BITBUCKET_SECRET = ""
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True

SOCIAL_AUTH_FACEBOOK_KEY = ""
SOCIAL_AUTH_FACEBOOK_SECRET = ""
SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {"fields": "id,name,email"}

SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = ""
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ""

# Social auth settings
SOCIAL_AUTH_PIPELINE = (
    "social_core.pipeline.social_auth.social_details",
```

(continues on next page)

(continuação da página anterior)

```
    "social_core.pipeline.social_auth.social_uid",
    "social_core.pipeline.social_auth.auth_allowed",
    "social_core.pipeline.social_auth.social_user",
    "weblate.accounts.pipeline.store_params",
    "weblate.accounts.pipeline.verify_open",
    "social_core.pipeline.user.get_username",
    "weblate.accounts.pipeline.require_email",
    "social_core.pipeline.mail.mail_validation",
    "weblate.accounts.pipeline.revoke_mail_code",
    "weblate.accounts.pipeline.ensure_valid",
    "weblate.accounts.pipeline.remove_account",
    "social_core.pipeline.social_auth.associate_by_email",
    "weblate.accounts.pipeline.reauthenticate",
    "weblate.accounts.pipeline.verify_username",
    "social_core.pipeline.user.create_user",
    "social_core.pipeline.social_auth.associate_user",
    "social_core.pipeline.social_auth.load_extra_data",
    "weblate.accounts.pipeline.cleanup_next",
    "weblate.accounts.pipeline.user_full_name",
    "weblate.accounts.pipeline.store_email",
    "weblate.accounts.pipeline.notify_connect",
    "weblate.accounts.pipeline.password_reset",
)
SOCIAL_AUTH_DISCONNECT_PIPELINE = (
    "social_core.pipeline.disconnect.allowed_to_disconnect",
    "social_core.pipeline.disconnect.get_entries",
    "social_core.pipeline.disconnect.revoke_tokens",
    "weblate.accounts.pipeline.cycle_session",
    "weblate.accounts.pipeline.adjust_primary_mail",
    "weblate.accounts.pipeline.notify_disconnect",
    "social_core.pipeline.disconnect.disconnect",
    "weblate.accounts.pipeline.cleanup_next",
)

# Custom authentication strategy
SOCIAL_AUTH_STRATEGY = "weblate.accounts.strategy.WeblateStrategy"

# Raise exceptions so that we can handle them later
SOCIAL_AUTH_RAISE_EXCEPTIONS = True

SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = "weblate.accounts.pipeline.send_validation"
SOCIAL_AUTH_EMAIL_VALIDATION_URL = "{0}/accounts/email-sent/".format(URL_PREFIX)
SOCIAL_AUTH_LOGIN_ERROR_URL = "{0}/accounts/login/".format(URL_PREFIX)
SOCIAL_AUTH_EMAIL_FORM_URL = "{0}/accounts/email/".format(URL_PREFIX)
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = "{0}/accounts/profile/#account".format(
    URL_PREFIX
)
SOCIAL_AUTH_PROTECTED_USER_FIELDS = ("email",)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True
SOCIAL_AUTH_SLUGIFY_FUNCTION = "weblate.accounts.pipeline.slugify_username"

# Password validation configuration
AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.
↪UserAttributeSimilarityValidator"  # noqa: E501, pylint: disable=line-too-long
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
        "OPTIONS": {"min_length": 10},
    },
```

(continues on next page)

(continuação da página anterior)

```
    {"NAME": "django.contrib.auth.password_validation.CommonPasswordValidator"},
    {"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.CharsPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.PastPasswordsValidator"},
    # Optional password strength validation by django-zxcvbn-password
    # {
    #     "NAME": "zxcvbn_password.ZXCVBNValidator",
    #     "OPTIONS": {
    #         "min_score": 3,
    #         "user_attributes": ("username", "email", "full_name")
    #     }
    # },
]

# Allow new user registrations
REGISTRATION_OPEN = True

# Shortcut for login required setting
REQUIRE_LOGIN = False

# Middleware
MIDDLEWARE = [
    "weblate.middleware.RedirectMiddleware",
    "weblate.middleware.ProxyMiddleware",
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "weblate.accounts.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "social_django.middleware.SocialAuthExceptionMiddleware",
    "weblate.accounts.middleware.RequireLoginMiddleware",
    "weblate.api.middleware.ThrottlingMiddleware",
    "weblate.middleware.SecurityMiddleware",
]

ROOT_URLCONF = "weblate.urls"

# Django and Weblate apps
INSTALLED_APPS = [
    # Weblate apps on top to override Django locales and templates
    "weblate.addons",
    "weblate.auth",
    "weblate.checks",
    "weblate.formats",
    "weblate.glossary",
    "weblate.machinery",
    "weblate.trans",
    "weblate.lang",
    "weblate_language_data",
    "weblate.memory",
    "weblate.screenshots",
    "weblate.fonts",
    "weblate.accounts",
    "weblate.utils",
    "weblate.vcs",
    "weblate.wladmin",
    "weblate",
    # Optional: Git exporter
    "weblate.gitexport",
    # Standard Django modules
```

(continues on next page)

(continuação da página anterior)

```
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django.contrib.admin.apps.SimpleAdminConfig",
    "django.contrib.admindocs",
    "django.contrib.sitemaps",
    "django.contrib.humanize",
    # Third party Django modules
    "social_django",
    "crispy_forms",
    "compressor",
    "rest_framework",
    "rest_framework.authtoken",
    "django_filters",
]

# Custom exception reporter to include some details
DEFAULT_EXCEPTION_REPORTER_FILTER = "weblate.trans.debug.
↪WeblateExceptionReporterFilter"

# Default logging of Weblate messages
# - to syslog in production (if available)
# - otherwise to console
# - you can also choose "logfile" to log into separate file
#   after configuring it below

# Detect if we can connect to syslog
HAVE_SYSLOG = False
if platform.system() != "Windows":
    try:
        handler = SysLogHandler(address="/dev/log", facility=SysLogHandler.LOG_
↪LOCAL2)
        handler.close()
        HAVE_SYSLOG = True
    except IOError:
        HAVE_SYSLOG = False

if DEBUG or not HAVE_SYSLOG:
    DEFAULT_LOG = "console"
else:
    DEFAULT_LOG = "syslog"
DEFAULT_LOGLEVEL = "DEBUG" if DEBUG else "INFO"

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG=False.
# See http://docs.djangoproject.com/en/stable/topics/logging for
# more details on how to customize your logging configuration.
LOGGING = {
    "version": 1,
    "disable_existing_loggers": True,
    "filters": {"require_debug_false": {"()": "django.utils.log.RequireDebugFalse"}
↪},
    "formatters": {
        "syslog": {"format": "weblate[%(process)d]: %(levelname)s %(message)s"},
        "simple": {"format": "[%(asctime)s: %(levelname)s/%(process)s] %(message)s
↪"},
        "logfile": {"format": "%(asctime)s %(levelname)s %(message)s"},
        "django.server": {
```

(continues on next page)

(continuação da página anterior)

```
            "()": "django.utils.log.ServerFormatter",
            "format": "[%(server_time)s] %(message)s",
        },
    },
    "handlers": {
        "mail_admins": {
            "level": "ERROR",
            "filters": ["require_debug_false"],
            "class": "django.utils.log.AdminEmailHandler",
            "include_html": True,
        },
        "console": {
            "level": "DEBUG",
            "class": "logging.StreamHandler",
            "formatter": "simple",
        },
        "django.server": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "django.server",
        },
        "syslog": {
            "level": "DEBUG",
            "class": "logging.handlers.SysLogHandler",
            "formatter": "syslog",
            "address": "/dev/log",
            "facility": SysLogHandler.LOG_LOCAL2,
        },
        # Logging to a file
        # "logfile": {
        #     "level":"DEBUG",
        #     "class":"logging.handlers.RotatingFileHandler",
        #     "filename": "/var/log/weblate/weblate.log",
        #     "maxBytes": 100000,
        #     "backupCount": 3,
        #     "formatter": "logfile",
        # },
    },
    "loggers": {
        "django.request": {
            "handlers": ["mail_admins", DEFAULT_LOG],
            "level": "ERROR",
            "propagate": True,
        },
        "django.server": {
            "handlers": ["django.server"],
            "level": "INFO",
            "propagate": False,
        },
        # Logging database queries
        # "django.db.backends": {
        #     "handlers": [DEFAULT_LOG],
        #     "level": "DEBUG",
        # },
        "weblate": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Logging VCS operations
        "weblate.vcs": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Python Social Auth
        "social": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Django Authentication Using LDAP
        "django_auth_ldap": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
```

(continues on next page)

(continuação da página anterior)

```
        # SAML IdP
        "djangosaml2idp": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
    },
}

# Remove syslog setup if it's not present
if not HAVE_SYSLOG:
    del LOGGING["handlers"]["syslog"]

# List of machine translations
MT_SERVICES = (
    #     "weblate.machinery.apertium.ApertiumAPYTranslation",
    #     "weblate.machinery.baidu.BaiduTranslation",
    #     "weblate.machinery.deepl.DeepLTranslation",
    #     "weblate.machinery.glosbe.GlosbeTranslation",
    #     "weblate.machinery.google.GoogleTranslation",
    #     "weblate.machinery.googlev3.GoogleV3Translation",
    #     "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    #     "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    #     "weblate.machinery.modernmt.ModernMTTranslation",
    #     "weblate.machinery.mymemory.MyMemoryTranslation",
    #     "weblate.machinery.netease.NeteaseSightTranslation",
    #     "weblate.machinery.tmserver.AmagamaTranslation",
    #     "weblate.machinery.tmserver.TMServerTranslation",
    #     "weblate.machinery.yandex.YandexTranslation",
    #     "weblate.machinery.saptranslationhub.SAPTranslationHub",
    #     "weblate.machinery.youdao.YoudaoTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.memory.machine.WeblateMemory",
)

# Machine translation API keys

# URL of the Apertium APy server
MT_APERTIUM_APY = None

# DeepL API key
MT_DEEPL_KEY = None

# Microsoft Cognitive Services Translator API, register at
# https://portal.azure.com/
MT_MICROSOFT_COGNITIVE_KEY = None
MT_MICROSOFT_REGION = None

# ModernMT
MT_MODERNMT_KEY = None

# MyMemory identification email, see
# https://mymemory.translated.net/doc/spec.php
MT_MYMEMORY_EMAIL = None

# Optional MyMemory credentials to access private translation memory
MT_MYMEMORY_USER = None
MT_MYMEMORY_KEY = None

# Google API key for Google Translate API v2
MT_GOOGLE_KEY = None

# Google Translate API3 credentials and project id
MT_GOOGLE_CREDENTIALS = None
MT_GOOGLE_PROJECT = None
```

(continues on next page)

(continuação da página anterior)

```
# Baidu app key and secret
MT_BAIDU_ID = None
MT_BAIDU_SECRET = None

# Youdao Zhiyun app key and secret
MT_YOUDAO_ID = None
MT_YOUDAO_SECRET = None

# Netease Sight (Jianwai) app key and secret
MT_NETEASE_KEY = None
MT_NETEASE_SECRET = None

# API key for Yandex Translate API
MT_YANDEX_KEY = None

# tmserver URL
MT_TMSERVER = None

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

# Use HTTPS when creating redirect URLs for social authentication, see
# documentation for more details:
# https://python-social-auth-docs.readthedocs.io/en/latest/configuration/settings.
↪html#processing-redirects-and-urlopen
SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

# Make CSRF cookie HttpOnly, see documentation for more details:
# https://docs.djangoproject.com/en/1.11/ref/settings/#csrf-cookie-httponly
CSRF_COOKIE_HTTPONLY = True
CSRF_COOKIE_SECURE = ENABLE_HTTPS
# Store CSRF token in session
CSRF_USE_SESSIONS = True
# Customize CSRF failure view
CSRF_FAILURE_VIEW = "weblate.trans.views.error.csrf_failure"
SESSION_COOKIE_SECURE = ENABLE_HTTPS
SESSION_COOKIE_HTTPONLY = True
# SSL redirect
SECURE_SSL_REDIRECT = ENABLE_HTTPS
# Sent referrrer only for same origin links
SECURE_REFERRER_POLICY = "same-origin"
# SSL redirect URL exemption list
SECURE_REDIRECT_EXEMPT = (r"healthz/$",)  # Allowing HTTP access to health check
# Session cookie age (in seconds)
SESSION_COOKIE_AGE = 1000
SESSION_COOKIE_AGE_AUTHENTICATED = 1209600
# Increase allowed upload size
```

(continues on next page)

(continuação da página anterior)

```
DATA_UPLOAD_MAX_MEMORY_SIZE = 50000000

# Apply session coookie settings to language cookie as ewll
LANGUAGE_COOKIE_SECURE = SESSION_COOKIE_SECURE
LANGUAGE_COOKIE_HTTPONLY = SESSION_COOKIE_HTTPONLY
LANGUAGE_COOKIE_AGE = SESSION_COOKIE_AGE_AUTHENTICATED * 10

# Some security headers
SECURE_BROWSER_XSS_FILTER = True
X_FRAME_OPTIONS = "DENY"
SECURE_CONTENT_TYPE_NOSNIFF = True

# Optionally enable HSTS
SECURE_HSTS_SECONDS = 31536000 if ENABLE_HTTPS else 0
SECURE_HSTS_PRELOAD = ENABLE_HTTPS
SECURE_HSTS_INCLUDE_SUBDOMAINS = ENABLE_HTTPS

# HTTPS detection behind reverse proxy
SECURE_PROXY_SSL_HEADER = None

# URL of login
LOGIN_URL = "{0}/accounts/login/".format(URL_PREFIX)

# URL of logout
LOGOUT_URL = "{0}/accounts/logout/".format(URL_PREFIX)

# Default location for login
LOGIN_REDIRECT_URL = "{0}/".format(URL_PREFIX)

# Anonymous user name
ANONYMOUS_USER_NAME = "anonymous"

# Reverse proxy settings
IP_PROXY_HEADER = "HTTP_X_FORWARDED_FOR"
IP_BEHIND_REVERSE_PROXY = False
IP_PROXY_OFFSET = 0

# Sending HTML in mails
EMAIL_SEND_HTML = True

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = "[{0}] ".format(SITE_TITLE)

# Enable remote hooks
ENABLE_HOOKS = True

# By default the length of a given translation is limited to the length of
# the source string * 10 characters. Set this option to False to allow longer
# translations (up to 10.000 characters)
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True

# Use simple language codes for default language/country combinations
SIMPLIFY_LANGUAGES = True

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = "bootstrap3"

# List of quality checks
# CHECK_LIST = (
#     "weblate.checks.same.SameCheck",
#     "weblate.checks.chars.BeginNewlineCheck",
```

(continues on next page)

(continuação da página anterior)

```
#     "weblate.checks.chars.EndNewlineCheck",
#     "weblate.checks.chars.BeginSpaceCheck",
#     "weblate.checks.chars.EndSpaceCheck",
#     "weblate.checks.chars.DoubleSpaceCheck",
#     "weblate.checks.chars.EndStopCheck",
#     "weblate.checks.chars.EndColonCheck",
#     "weblate.checks.chars.EndQuestionCheck",
#     "weblate.checks.chars.EndExclamationCheck",
#     "weblate.checks.chars.EndEllipsisCheck",
#     "weblate.checks.chars.EndSemicolonCheck",
#     "weblate.checks.chars.MaxLengthCheck",
#     "weblate.checks.chars.KashidaCheck",
#     "weblate.checks.chars.PunctuationSpacingCheck",
#     "weblate.checks.format.PythonFormatCheck",
#     "weblate.checks.format.PythonBraceFormatCheck",
#     "weblate.checks.format.PHPFormatCheck",
#     "weblate.checks.format.CFormatCheck",
#     "weblate.checks.format.PerlFormatCheck",
#     "weblate.checks.format.JavaScriptFormatCheck",
#     "weblate.checks.format.CSharpFormatCheck",
#     "weblate.checks.format.JavaFormatCheck",
#     "weblate.checks.format.JavaMessageFormatCheck",
#     "weblate.checks.format.PercentPlaceholdersCheck",
#     "weblate.checks.format.VueFormattingCheck",
#     "weblate.checks.format.I18NextInterpolationCheck",
#     "weblate.checks.format.ESTemplateLiteralsCheck",
#     "weblate.checks.angularjs.AngularJSInterpolationCheck",
#     "weblate.checks.qt.QtFormatCheck",
#     "weblate.checks.qt.QtPluralCheck",
#     "weblate.checks.ruby.RubyFormatCheck",
#     "weblate.checks.consistency.PluralsCheck",
#     "weblate.checks.consistency.SamePluralsCheck",
#     "weblate.checks.consistency.ConsistencyCheck",
#     "weblate.checks.consistency.TranslatedCheck",
#     "weblate.checks.chars.EscapedNewlineCountingCheck",
#     "weblate.checks.chars.NewLineCountCheck",
#     "weblate.checks.markup.BBCodeCheck",
#     "weblate.checks.chars.ZeroWidthSpaceCheck",
#     "weblate.checks.render.MaxSizeCheck",
#     "weblate.checks.markup.XMLValidityCheck",
#     "weblate.checks.markup.XMLTagsCheck",
#     "weblate.checks.markup.MarkdownRefLinkCheck",
#     "weblate.checks.markup.MarkdownLinkCheck",
#     "weblate.checks.markup.MarkdownSyntaxCheck",
#     "weblate.checks.markup.URLCheck",
#     "weblate.checks.markup.SafeHTMLCheck",
#     "weblate.checks.placeholders.PlaceholderCheck",
#     "weblate.checks.placeholders.RegexCheck",
#     "weblate.checks.duplicate.DuplicateCheck",
#     "weblate.checks.source.OptionalPluralCheck",
#     "weblate.checks.source.EllipsisCheck",
#     "weblate.checks.source.MultipleFailingCheck",
#     "weblate.checks.source.LongUntranslatedCheck",
#     "weblate.checks.format.MultipleUnnamedFormatsCheck",
# )

# List of automatic fixups
# AUTOFIX_LIST = (
#     "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
#     "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
#     "weblate.trans.autofixes.chars.RemoveZeroSpace",
```

(continues on next page)

(continuação da página anterior)

```
#     "weblate.trans.autofixes.chars.RemoveControlChars",
# )

# List of enabled addons
# WEBLATE_ADDONS = (
#     "weblate.addons.gettext.GenerateMoAddon",
#     "weblate.addons.gettext.UpdateLinguasAddon",
#     "weblate.addons.gettext.UpdateConfigureAddon",
#     "weblate.addons.gettext.MsgmergeAddon",
#     "weblate.addons.gettext.GettextCustomizeAddon",
#     "weblate.addons.gettext.GettextAuthorComments",
#     "weblate.addons.cleanup.CleanupAddon",
#     "weblate.addons.consistency.LangaugeConsistencyAddon",
#     "weblate.addons.discovery.DiscoveryAddon",
#     "weblate.addons.flags.SourceEditAddon",
#     "weblate.addons.flags.TargetEditAddon",
#     "weblate.addons.flags.SameEditAddon",
#     "weblate.addons.flags.BulkEditAddon",
#     "weblate.addons.generate.GenerateFileAddon",
#     "weblate.addons.json.JSONCustomizeAddon",
#     "weblate.addons.properties.PropertiesSortAddon",
#     "weblate.addons.git.GitSquashAddon",
#     "weblate.addons.removal.RemoveComments",
#     "weblate.addons.removal.RemoveSuggestions",
#     "weblate.addons.resx.ResxUpdateAddon",
#     "weblate.addons.yaml.YAMLCustomizeAddon",
#     "weblate.addons.cdn.CDNJSAddon",
#     "weblate.addons.autotranslate.AutoTranslateAddon",
# )

# E-mail address that error messages come from.
SERVER_EMAIL = "noreply@example.com"

# Default email address to use for various automated correspondence from
# the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = "noreply@example.com"

# List of URLs your site is supposed to serve
ALLOWED_HOSTS = ["*"]

# Configuration for caching
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # "LOCATION": "unix:///var/run/redis/redis.sock?db=1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
            "PASSWORD": None,
            "CONNECTION_POOL_KWARGS": {},
        },
        "KEY_PREFIX": "weblate",
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 86400,
        "OPTIONS": {"MAX_ENTRIES": 1000},
```

(continues on next page)

(continuação da página anterior)

```
    },
}

# Store sessions in cache
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
# Store messages in session
MESSAGE_STORAGE = "django.contrib.messages.storage.session.SessionStorage"

# REST framework settings for API
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    "DEFAULT_PERMISSION_CLASSES": [
        # Require authentication for login required sites
        "rest_framework.permissions.IsAuthenticated"
        if REQUIRE_LOGIN
        else "rest_framework.permissions.IsAuthenticatedOrReadOnly"
    ],
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework.authentication.TokenAuthentication",
        "weblate.api.authentication.BearerAuthentication",
        "rest_framework.authentication.SessionAuthentication",
    ),
    "DEFAULT_THROTTLE_CLASSES": (
        "weblate.api.throttling.UserRateThrottle",
        "weblate.api.throttling.AnonRateThrottle",
    ),
    "DEFAULT_THROTTLE_RATES": {"anon": "100/day", "user": "5000/hour"},
    "DEFAULT_PAGINATION_CLASS": ("rest_framework.pagination.PageNumberPagination"),
    "PAGE_SIZE": 20,
    "VIEW_DESCRIPTION_FUNCTION": "weblate.api.views.get_view_description",
    "UNAUTHENTICATED_USER": "weblate.auth.models.get_anonymous",
}

# Fonts CDN URL
FONTS_CDN_URL = None

# Django compressor offline mode
COMPRESS_OFFLINE = False
COMPRESS_OFFLINE_CONTEXT = [
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI":␣
↪True},
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI":␣
↪False},
]

# Require login for all URLs
if REQUIRE_LOGIN:
    LOGIN_REQUIRED_URLS = (r"/(.*)$",)

# In such case you will want to include some of the exceptions
# LOGIN_REQUIRED_URLS_EXCEPTIONS = (
#     rf"{URL_PREFIX}/accounts/(.*)$",  # Required for login
#     rf"{URL_PREFIX}/admin/login/(.*)$",  # Required for admin login
#     rf"{URL_PREFIX}/static/(.*)$",  # Required for development mode
#     rf"{URL_PREFIX}/widgets/(.*)$",  # Allowing public access to widgets
#     rf"{URL_PREFIX}/data/(.*)$",  # Allowing public access to data exports
#     rf"{URL_PREFIX}/hooks/(.*)$",  # Allowing public access to notification hooks
#     rf"{URL_PREFIX}/healthz/$",  # Allowing public access to health check
#     rf"{URL_PREFIX}/api/(.*)$",  # Allowing access to API
#     rf"{URL_PREFIX}/js/i18n/$",  # JavaScript localization
```

(continues on next page)

(continuação da página anterior)

```
#     rf"{URL_PREFIX}/contact/$",  # Optional for contact form
#     rf"{URL_PREFIX}/legal/(.*)$",  # Optional for legal app
# )

# Silence some of the Django system checks
SILENCED_SYSTEM_CHECKS = [
    # We have modified django.contrib.auth.middleware.AuthenticationMiddleware
    # as weblate.accounts.middleware.AuthenticationMiddleware
    "admin.E408"
]

# Celery worker configuration for testing
# CELERY_TASK_ALWAYS_EAGER = True
# CELERY_BROKER_URL = "memory://"
# CELERY_TASK_EAGER_PROPAGATES = True
# Celery worker configuration for production
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL

# Celery settings, it is not recommended to change these
CELERY_WORKER_MAX_MEMORY_PER_CHILD = 200000
CELERY_BEAT_SCHEDULE_FILENAME = os.path.join(DATA_DIR, "celery", "beat-schedule")
CELERY_TASK_ROUTES = {
    "weblate.trans.tasks.auto_translate": {"queue": "translate"},
    "weblate.accounts.tasks.notify_*": {"queue": "notify"},
    "weblate.accounts.tasks.send_mails": {"queue": "notify"},
    "weblate.utils.tasks.settings_backup": {"queue": "backup"},
    "weblate.utils.tasks.database_backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup_service": {"queue": "backup"},
    "weblate.memory.tasks.*": {"queue": "memory"},
}

# Enable plain database backups
DATABASE_BACKUP = "plain"

# Enable auto updating
AUTO_UPDATE = False

# PGP commits signing
WEBLATE_GPG_IDENTITY = None

# Third party services integration
MATOMO_SITE_ID = None
MATOMO_URL = None
GOOGLE_ANALYTICS_ID = None
SENTRY_DSN = None
AKISMET_API_KEY = None
```

2.18 Management commands

Nota: Running management commands under a different user than the one running your webserver can result in files getting wrong permissions, please check *Permissões do sistema de ficheiros* for more details.

You will find basic management commands (available as `./manage.py` in the Django sources, or as an extended set in a script called **`weblate`** installable atop Weblate).

2.18.1 Invoking management commands

As mentioned before, invocation depends on how you installed Weblate.

If using virtualenv for Weblate, you can either specify the full path to **`weblate`**, or activate the virtualenv prior to invoking it:

```
# Direct invocation
~/weblate-env/bin/weblate

# Activating virtualenv adds it to search path
. ~/weblate-env/bin/activate
weblate
```

If you are using source code directly (either from a tarball or Git checkout), the management script is `./manage.py` available in the Weblate sources. To run it:

```
python ./manage.py list_versions
```

If you've installed Weblate using the pip or pip3 installer, or by using the `./setup.py` script, the **`weblate`** is installed to your path (or virtualenv path), from where you can use it to control Weblate:

```
weblate list_versions
```

For the Docker image, the script is installed like above, and you can run it using **`docker exec`**:

```
docker exec --user weblate <container> weblate list_versions
```

For **`docker-compose`** the process is similar, you just have to use **`docker-compose exec`**:

```
docker-compose exec --user weblate weblate weblate list_versions
```

In case you need to pass it a file, you can temporary add a volume:

```
docker-compose exec --user weblate /tmp:/tmp weblate weblate importusers /tmp/
↪users.json
```

Veja também:

Installing using Docker, *Installing on Debian and Ubuntu*, *Installing on SUSE and openSUSE*, *Installing on RedHat, Fedora and CentOS*

- *Installing from sources*, recomendado para o desenvolvimento.

2.18.2 add_suggestions

weblate add_suggestions <project> <component> <language> <file>

Novo na versão 2.5.

Imports a translation from the file to use as a suggestion for the given translation. It skips duplicated translations; only different ones are added.

--author USER@EXAMPLE.COM
: E-mail of author for the suggestions. This user has to exist prior to importing (you can create one in the admin interface if needed).

Exemplo:

```
weblate --author michal@cihar.com add_suggestions weblate application cs /tmp/
↪suggestions-cs.po
```

2.18.3 auto_translate

weblate auto_translate <project> <component> <language>

Novo na versão 2.5.

Performs automatic translation based on other component translations.

--source PROJECT/COMPONENT
: Specifies the component to use as source available for translation. If not specified all components in the project are used.

--user USERNAME
: Specify username listed as author of the translations. «Anonymous user» is used if not specified.

--overwrite
: Whether to overwrite existing translations.

--inconsistent
: Whether to overwrite existing translations that are inconsistent (see *Inconsistente*).

--add
: Automatically add language if a given translation does not exist.

--mt MT
: Use machine translation instead of other components as machine translations.

--threshold THRESHOLD
: Similarity threshold for machine translation, defaults to 80.

Exemplo:

```
weblate auto_translate --user nijel --inconsistent --source weblate/application␣
↪weblate website cs
```

Veja também:

Tradução automática

2.18.4 celery_queues

weblate celery_queues

Novo na versão 3.7.

Displays length of Celery task queues.

2.18.5 checkgit

weblate checkgit <project|project/component>

Prints current state of the back-end Git repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.6 commitgit

weblate commitgit <project|project/component>

Commits any possible pending changes to the back-end Git repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.7 commit_pending

weblate commit_pending <project|project/component>

Commits pending changes older than a given age.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

--age HOURS
: Age in hours for committing. If not specified the value configured in *Component configuration* is used.

Nota: This is automatically performed in the background by Weblate, so there no real need to invoke this manually, besides forcing an earlier commit than specified by *Component configuration*.

Veja também:

Executar tarefas de manutenção, *COMMIT_PENDING_HOURS*

2.18.8 cleanuptrans

weblate cleanuptrans

Cleans up orphaned checks and translation suggestions. There is normally no need to run this manually, as the cleanups happen automatically in the background.

Veja também:

Executar tarefas de manutenção

2.18.9 createadmin

weblate createadmin

Creates an `admin` account with a random password, unless it is specified.

--password PASSWORD
: Provides a password on the command-line, to not generate a random one.

--no-password
: Do not set password, this can be useful with *–update*.

--username USERNAME
: Use the given name instead of `admin`.

--email USER@EXAMPLE.COM
: Specify the admin e-mail address.

--name
: Specify the admin name (visible).

--update
: Update the existing user (you can use this to change passwords).

Alterado na versão 2.9: Added parameters `--username`, `--email`, `--name` and `--update`.

2.18.10 dump_memory

weblate dump_memory

Novo na versão 2.20.

Export a JSON file containing Weblate Translation Memory content.

Veja também:

Memória de Tradução, *Esquema de memória de tradução Weblate*

2.18.11 dumpuserdata

weblate dumpuserdata <file.json>

Dumps userdata to a file for later use by *importuserdata*

Dica: This comes in handy when migrating or merging Weblate instances.

2.18.12 import_demo

weblate import_demo

Novo na versão 4.1.

Creates a demo project with components based on <https://github.com/WeblateOrg/demo>.

This can be useful when developing Weblate.

2.18.13 import_json

weblate import_json <json-file>

Novo na versão 2.7.

Batch import of components based on JSON data.

The imported JSON file structure pretty much corresponds to the component object (see *GET /api/components/(string:project)/(string:component)/*). You have to include the `name` and `filemask` fields.

--project PROJECT
: Specifies where the components will be imported from.

--main-component COMPONENT
: Use the given VCS repository from this component for all of them.

--ignore
: Skip (already) imported components.

--update
: Update (already) imported components.

Alterado na versão 2.9: The parameters `--ignore` and `--update` are there to deal with already imported components.

Example of JSON file:

```
[
  {
    "slug": "po",
    "name": "Gettext PO",
    "file_format": "po",
    "filemask": "po/*.po",
    "new_lang": "none"
  },
  {
    "name": "Android",
    "filemask": "android/values-*/strings.xml",
    "template": "android/values/strings.xml",
    "repo": "weblate://test/test",
    "file_format": "aresource"
  }
]
```

Veja também:

import_memory

2.18.14 import_memory

weblate import_memory <file>

Novo na versão 2.20.

Imports a TMX or JSON file into the Weblate translation memory.

--language-map LANGMAP
: Allows mapping languages in the TMX to the Weblate translation memory. The language codes are mapped after normalization usually done by Weblate.

  `--language-map en_US:en` will for example import all `en_US` strings as `en` ones.

  This can be useful in case your TMX file locales happen not to match what you use in Weblate.

Veja também:

Memória de Tradução, *Esquema de memória de tradução Weblate*

2.18.15 import_project

weblate import_project <project> <gitrepo> <branch> <filemask>

Alterado na versão 3.0: The import_project command is now based on the *Descoberta de componentes* addon, leading to some changes in behavior and what parameters are accepted.

Batch imports components into project based on filemask.

<project> names an existing project, into which the components are to be imported.

The *<gitrepo>* defines the Git repository URL to use, and *<branch>* signifies the Git branch. To import additional translation components from an existing Weblate component, use a *weblate://<project>/<component>* URL for the *<gitrepo>*.

The *<filemask>* defines file discovery for the repository. It can be either be made simple using wildcards, or it can use the full power of regular expressions.

The simple matching uses `**` for component name and `*` for language, for example: `**/*.po`

The regular expression has to contain groups named *component* and *language*. For example: `(?P<language>[^/]*)/(?P<component>[^-/]*)\.po`

The import matches existing components based on files and adds the ones that do not exist. It does not change already existing ones.

--name-template TEMPLATE
: Customize the name of a component using Django template syntax.

  For example: `Documentation: {{ component }}`

--base-file-template TEMPLATE
: Customize the base file for monolingual translations.

  For example: `{{ component }}/res/values/string.xml`

--new-base-template TEMPLATE
: Customize the base file for addition of new translations.

  For example: `{{ component }}/ts/en.ts`

--file-format FORMAT
: You can also specify the file format to use (see *Formatos de ficheiros suportados*), the default is auto-detection.

--language-regex REGEX
: You can specify language filtering (see *Component configuration*) with this parameter. It has to be a valid regular expression.

--main-component
: You can specify which component will be chosen as the main one—the one actually containing the VCS repository.

--license NAME
: Specify the overall, project or component translation license.

--license-url URL
: Specify the URL where the translation license is to be found.

--vcs NAME
: In case you need to specify which version control system to use, you can do it here. The default version control is Git.

To give you some examples, let's try importing two projects.

First The Debian Handbook translations, where each language has separate a folder with the translations of each chapter:

```
weblate import_project \
    debian-handbook \
    git://anonscm.debian.org/debian-handbook/debian-handbook.git \
    squeeze/master \
    '*/**.po'
```

Then the Tanaguru tool, where the file format needs be specified, along with the base file template, and how all components and translations are located in single folder:

```
weblate import_project \
    --file-format=properties \
    --base-file-template=web-app/tgol-web-app/src/main/resources/i18n/%s-I18N.
↪properties \
    tanaguru \
    https://github.com/Tanaguru/Tanaguru \
    master \
    web-app/tgol-web-app/src/main/resources/i18n/**-I18N_*.properties
```

More complex example of parsing of filenames to get the correct component and language out of a filename like `src/security/Numerous_security_holes_in_0.10.1.de.po`:

```
weblate import_project \
    tails \
    git://git.tails.boum.org/tails master \
    'wiki/src/security/(?P<component>.*)\.(?P<language>[^.]*)\.po$'
```

Filtering only translations in a chosen language:

```
./manage import_project \
    --language-regex '^(cs|sk)$' \
    weblate \
    https://github.com/WeblateOrg/weblate.git \
    'weblate/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files:

```
$ weblate import_project --name-template 'Documentation: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files and directories:

```
$ weblate import_project --name-template 'Directory 1: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/dir1/**.po'
$ weblate import_project --name-template 'Directory 2: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/dir2/**.po'
```

Veja também:

More detailed examples can be found in the *Starting with internationalization* chapter, alternatively you might want to use *import_json*.

2.18.16 importuserdata

weblate importuserdata <file.json>

Imports user data from a file created by *dumpuserdata*

2.18.17 importusers

weblate importusers --check <file.json>

Imports users from JSON dump of the Django auth_users database.

--check
With this option it will just check whether a given file can be imported and report possible conflicts arising from usernames or e-mails.

You can dump users from the existing Django installation using:

```
weblate dumpdata auth.User > users.json
```

2.18.18 install_addon

Novo na versão 3.2.

weblate install_addon --addon ADDON <project|project/component>

Installs an addon to a set of components.

--addon ADDON
Name of the addon to install. For example `weblate.gettext.customize`.

--configuration CONFIG
JSON encoded configuration of an addon.

--update
Update the existing addon configuration.

You can either define which project or component to install the addon in (for example `weblate/application`), or use `--all` to include all existing components.

To install *Personalizar a saída gettext* for all components:

```
weblate install_addon --addon weblate.gettext.customize --config '{"width": -1}' --
↪update --all
```

Veja também:

Extensões

2.18.19 list_languages

weblate list_languages <locale>

Lists supported languages in MediaWiki markup - language codes, English names and localized names.

This is used to generate <https://wiki.l10n.cz/Slovn%C3%ADk_s_n%C3%A1zvy_jazyk%C5%AF>.

2.18.20 list_translators

weblate list_translators <project|project/component>

Lists translators by contributed language for the given project:

```
[French]
Jean Dupont <jean.dupont@example.com>
[English]
John Doe <jd@example.com>
```

--language-code
List names by language code instead of language name.

You can either define which project or component to use (for example `weblate/application`), or use `--all` to list translators from all existing components.

2.18.21 list_versions

weblate list_versions

Lists all Weblate dependencies and their versions.

2.18.22 loadpo

weblate loadpo <project|project/component>

Reloads translations from disk (for example in case you have done some updates in the VCS repository).

--force
Force update, even if the files should be up-to-date.

--lang LANGUAGE
Limit processing to a single language.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Nota: You seldom need to invoke this, Weblate will automatically load changed files for every VCS update. This is needed in case you manually changed an underlying Weblate VCS repository or in some special cases following an upgrade.

2.18.23 lock_translation

weblate lock_translation <project|project/component>

Prevents further translation of a component.

Dica: Useful in case you want to do some maintenance on the underlying repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Veja também:

unlock_translation

2.18.24 move_language

weblate move_language source target

Novo na versão 3.0.

Allows you to merge language content. This is useful when updating to a new version which contains aliases for previously unknown languages that have been created with the *(generated)* suffix. It moves all content from the *source* language to the *target* one.

Exemplo:

```
weblate move_language cze cs
```

After moving the content, you should check whether there is anything left (this is subject to race conditions when somebody updates the repository meanwhile) and remove the *(generated)* language.

2.18.25 pushgit

weblate pushgit <project|project/component>

Pushes committed changes to the upstream VCS repository.

--force-commit
: Force commits any pending changes, prior to pushing.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Nota: Weblate pushes changes automatically if *Push on commit* in *Component configuration* is turned on, which is the default.

2.18.26 unlock_translation

weblate unlock_translation <project|project/component>

Unlocks a given component, making it available for translation.

Dica: Useful in case you want to do some maintenance on the underlying repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Veja também:

lock_translation

2.18.27 setupgroups

weblate setupgroups

Configures default groups and optionally assigns all users to that default group.

--no-privs-update
: Turns off automatic updating of existing groups (only adds new ones).

--no-projects-update
: Prevents automatic updates of groups for existing projects. This allows adding newly added groups to existing projects, see *Controlo de acesso por projeto*.

Veja também:

Controlo de acesso

2.18.28 setuplang

weblate setuplang

Updates list of defined languages in Weblate.

--no-update
Turns off automatic updates of existing languages (only adds new ones).

2.18.29 updatechecks

weblate updatechecks <project|project/component>

Updates all checks for all strings.

Dica: Useful for upgrades which do major changes to checks.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.30 updategit

weblate updategit <project|project/component>

Fetches remote VCS repositories and updates the internal cache.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Nota: Usually it is better to configure hooks in the repository to trigger *Hooks de notificação*, instead of regular polling by `updategit`.

2.19 Anúncios

Alterado na versão 4.0: Em versões anteriores, esse recurso era chamado de mensagens de quadro de comunicações.

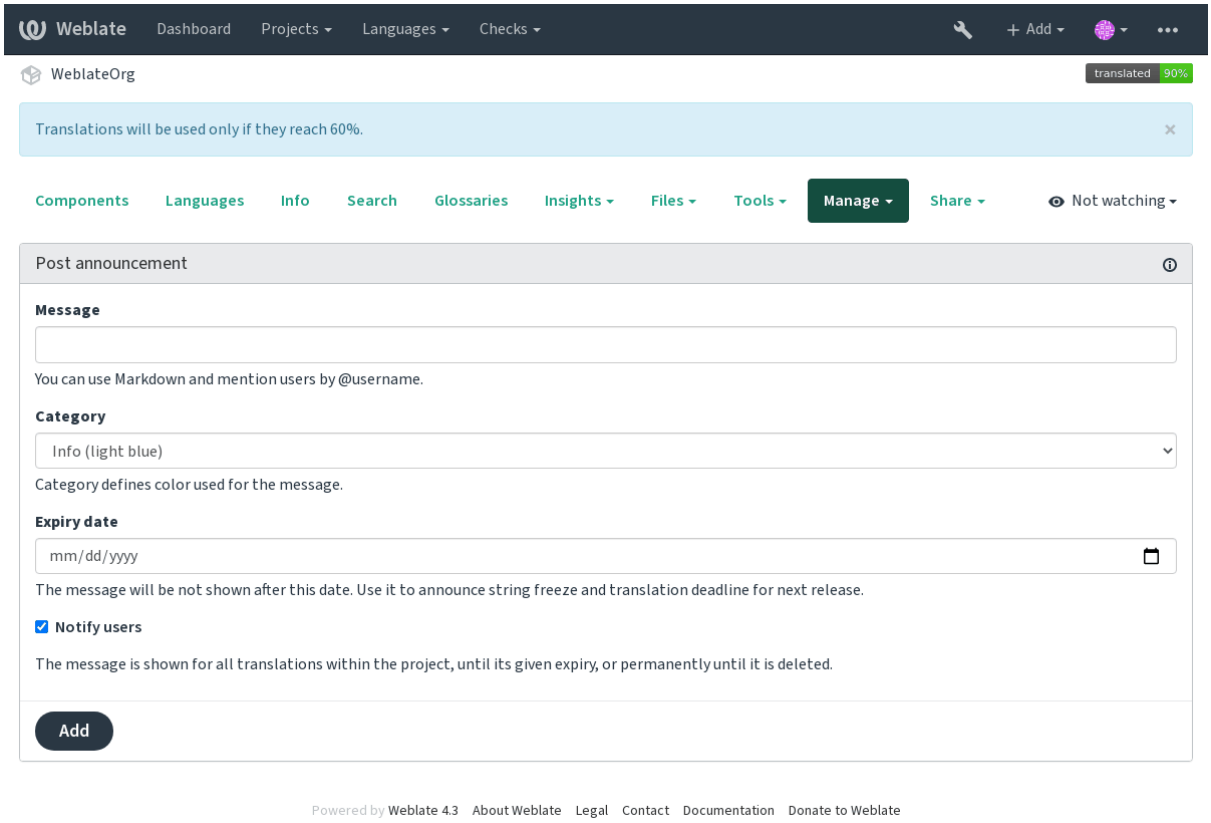
Forneça informações aos seus tradutores postando anúncios, em todo o site, por projeto, componente ou idioma.

Anuncie o propósito, prazos, estados ou especificar metas para tradução.

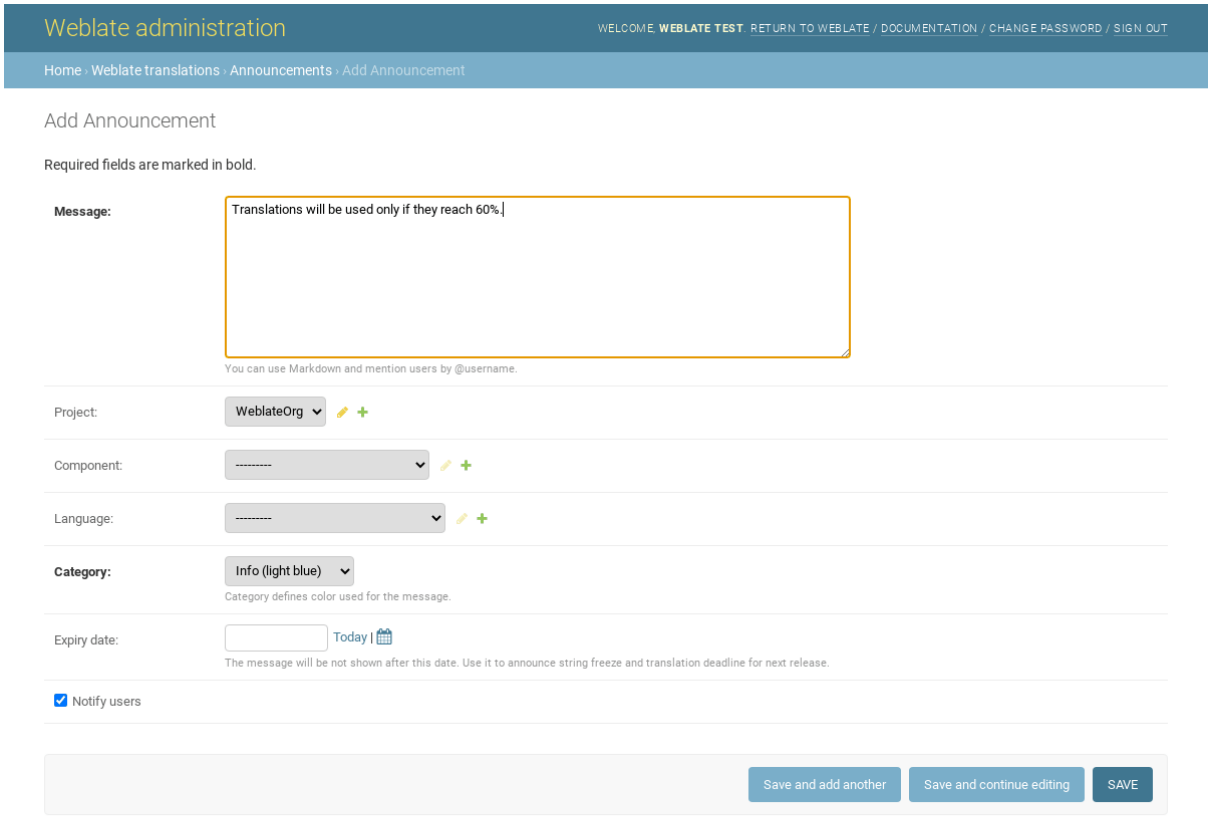
Os utilizadores receberão notificação sobre os anúncios de projetos assistidos (a menos que optem por não participar).

Isto pode ser útil para várias coisas, desde anunciar o propósito do site até especificar alvos para traduções.

Os anúncios podem ser publicados em cada nível no menu *Manage*, a usar :guilabel:"Publicar anúncio":



Ele também pode ser adicionado a usar a interface administrativa:



Os anúncios são então mostrados com base no seu contexto específico:

Nenhum contexto especificado

Mostrado no painel (página de chegada).

Projeto especificado

Mostrado dentro do projeto, incluindo todos os seus componentes e traduções.

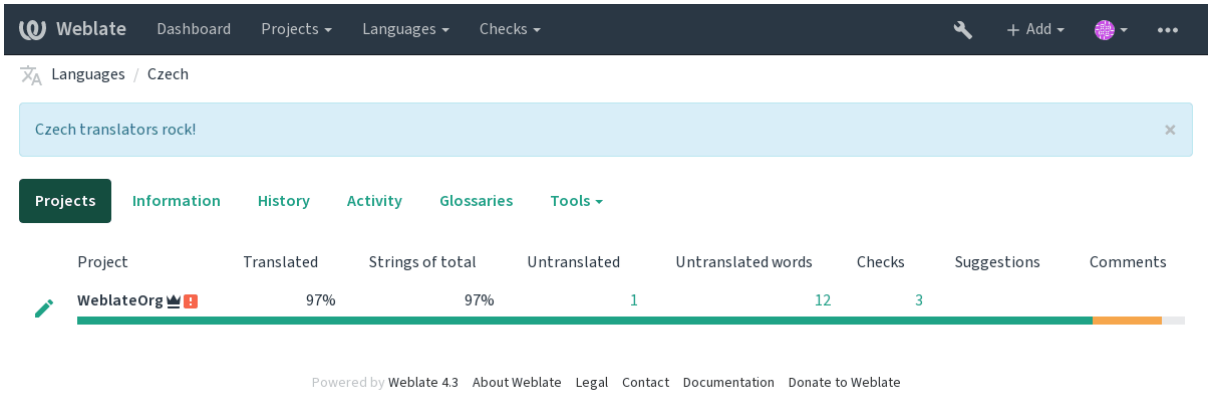
Componente especificado

Mostrado para um determinado componente e todas as traduções dele.

Idioma especificado

Mostrado na visão geral do idioma e todas as traduções nesse idioma.

Esta é a aparência na página de visão geral do idioma:



2.20 Lista de componentes

Especifique múltiplas listas de componentes para aparecer como opções no painel do utilizador, a partir do qual os utilizadores podem selecionar uma visualização como a visão predefinida. Veja *Painel* para saber mais.

Alterado na versão 2.20: Um estado vai ser apresentado para cada componente listado no painel.

Os nomes e conteúdos das listas de componentes podem ser especificados na interface administrativa, na secção *Component lists*. Cada lista de componentes deve ter um nome que é exibido ao utilizador e uma slug representando-a na URL.

Alterado na versão 2.13: Altera as configurações de painel para utilizadores anônimos da interface administrativa, a alterar qual painel é apresentado para utilizadores não autenticados.

2.20.1 Listas de componentes automáticas

Novo na versão 2.13.

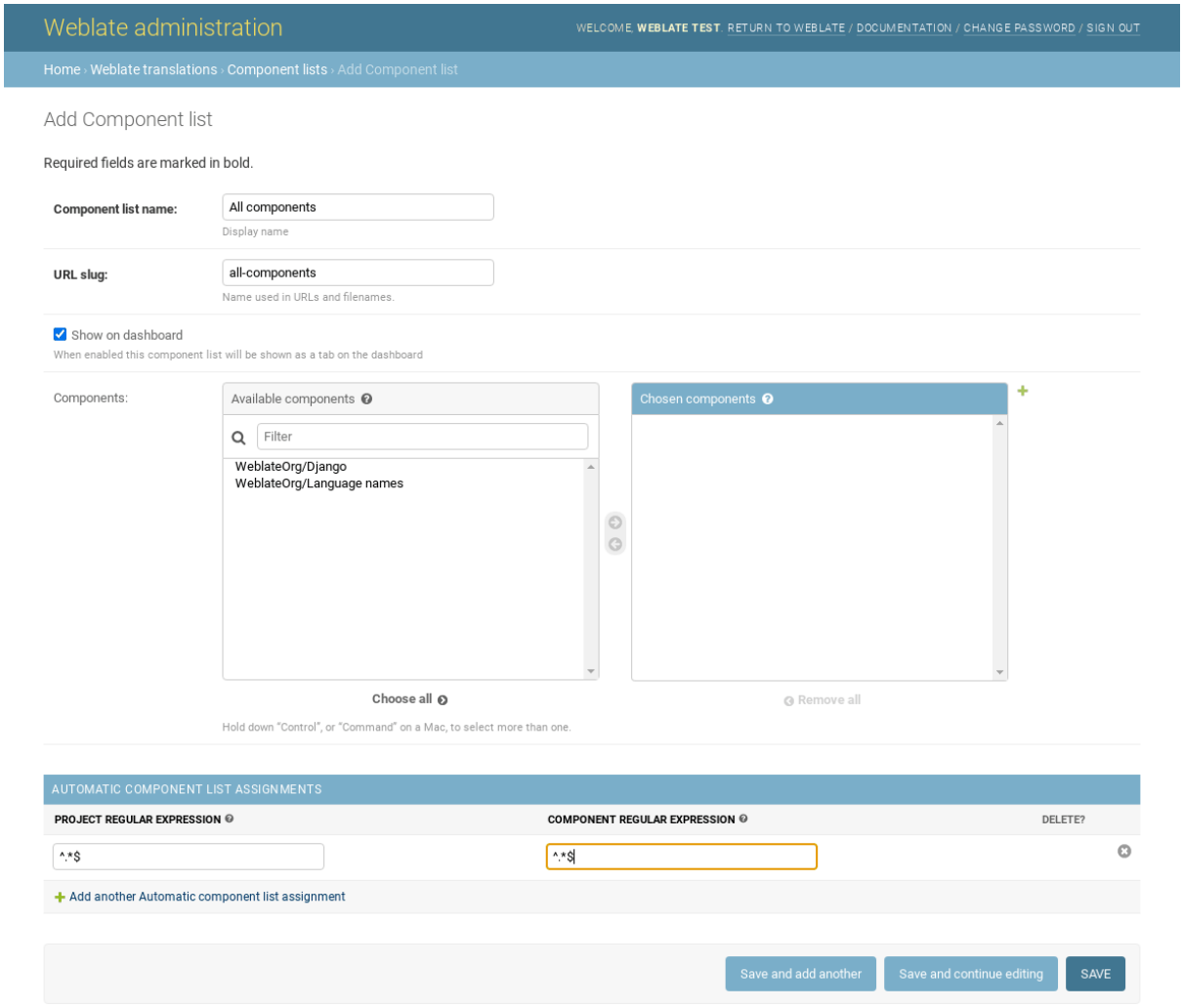
Adicione componentes à lista automaticamente com base nas suas slugs criando regras *Automatic component list assignment*.

- Útil para atualizar listas de componentes para grandes instalações, ou no caso de querer ter uma lista de componentes com todos os componentes na sua instalação do Weblate.

Dica: Faça uma lista de componentes contendo todos os componentes da sua instalação Weblate.

---

1. Define *Automatic component list assignment* with `^.*$` as regular expression in both the project and the component fields, as shown on this image:



2.21 Optional Weblate modules

Several optional modules are available for your setup.

2.21.1 Git exporter

Novo na versão 2.10.

Provides you read-only access to the underlying Git repository using HTTP(S).

Instalação

1. Add `weblate.gitexport` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.gitexport',
)
```

2. Export existing repositories by migrating your database after installation:

```
weblate migrate
```

Usage

The module automatically hooks into Weblate and sets the exported repository URL in the *Component configuration*. The repositories are accessible under the `/git/` part of the Weblate URL, for example `https://example.org/git/weblate/master/`:

```
git clone 'https://example.org/git/weblate/master/'
```

Repositories are available anonymously unless *Controlo de acesso por projeto* is turned on. This requires authenticate using your API token (it can be obtained in your *Perfil do utilizador*):

```
git clone 'https://user:KEY@example.org/git/weblate/master/'
```

2.21.2 Faturação

Novo na versão 2.4.

This is used on Hosted Weblate to define billing plans, track invoices and usage limits.

Instalação

1. Add `weblate.billing` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.billing',
)
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

Usage

After installation you can control billing in the admin interface. Users with billing enabled will get new *Billing* tab in their *Perfil do utilizador*.

The billing module additionally allows project admins to create new projects and components without being superusers (see *Adding translation projects and components*). This is possible when following conditions are met:

- The billing is in its configured limits (any overusage results in blocking of project/component creation) and paid (if its price is non zero)
- The user is admin of existing project with billing or user is owner of billing (the latter is necessary when creating new billing for users to be able to import new projects).

Upon project creation user is able to choose which billing should be charged for the project in case he has access to more of them.

2.21.3 Legal

Novo na versão 2.15.

This is used on Hosted Weblate to provide required legal documents. It comes provided with blank documents, and you are expected to fill out the following templates in the documents:

`legal/documents/tos.html` Terms of service document

`legal/documents/privacy.html` Privacy policy document

`legal/documents/summary.html` Short overview of the terms of service and privacy policy

---

Nota: Legal documents for the Hosted Weblate service is available in this Git repository <https://github.com/WeblateOrg/hosted/tree/master/wlhosted/legal/templates/legal/documents>.

Most likely these will not be directly usable to you, but might come in handy as a starting point if adjusted to meet your needs.

Instalação

1. Add `weblate.legal` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.legal',
)

# Optional:

# Social auth pipeline to confirm TOS upon registration/subsequent login
SOCIAL_AUTH_PIPELINE += (
    'weblate.legal.pipeline.tos_confirm',
)

# Middleware to enforce TOS confirmation of signed in users
MIDDLEWARE += [
    'weblate.legal.middleware.RequireTOSMiddleware',
]
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

3. Edit the legal documents in the `weblate/legal/templates/legal/` folder to match your service.

Usage

After installation and editing, the legal documents are shown in the Weblate UI.

2.21.4 Avatars

Avatars are downloaded and cached server-side to reduce information leaks to the sites serving them by default. The built-in support for fetching avatars from e-mails addresses configured for it can be turned off using *ENABLE_AVATARS*.

Weblate currently supports:

- Gravatar

Veja também:

Cache de avatares, *AVATAR_URL_PREFIX*, *ENABLE_AVATARS*

2.21.5 Spam protection

You can protect against suggestion spamming by unauthenticated users by using the akismet.com service.

1. Install the *akismet* Python module
2. Configure the Akismet API key.

Nota: This (among other things) relies on IP address of the client, please see *Executar por trás de um proxy reverso* for properly configuring that.

Veja também:

Executar por trás de um proxy reverso, *AKISMET_API_KEY*

2.21.6 Signing Git commits with GnuPG

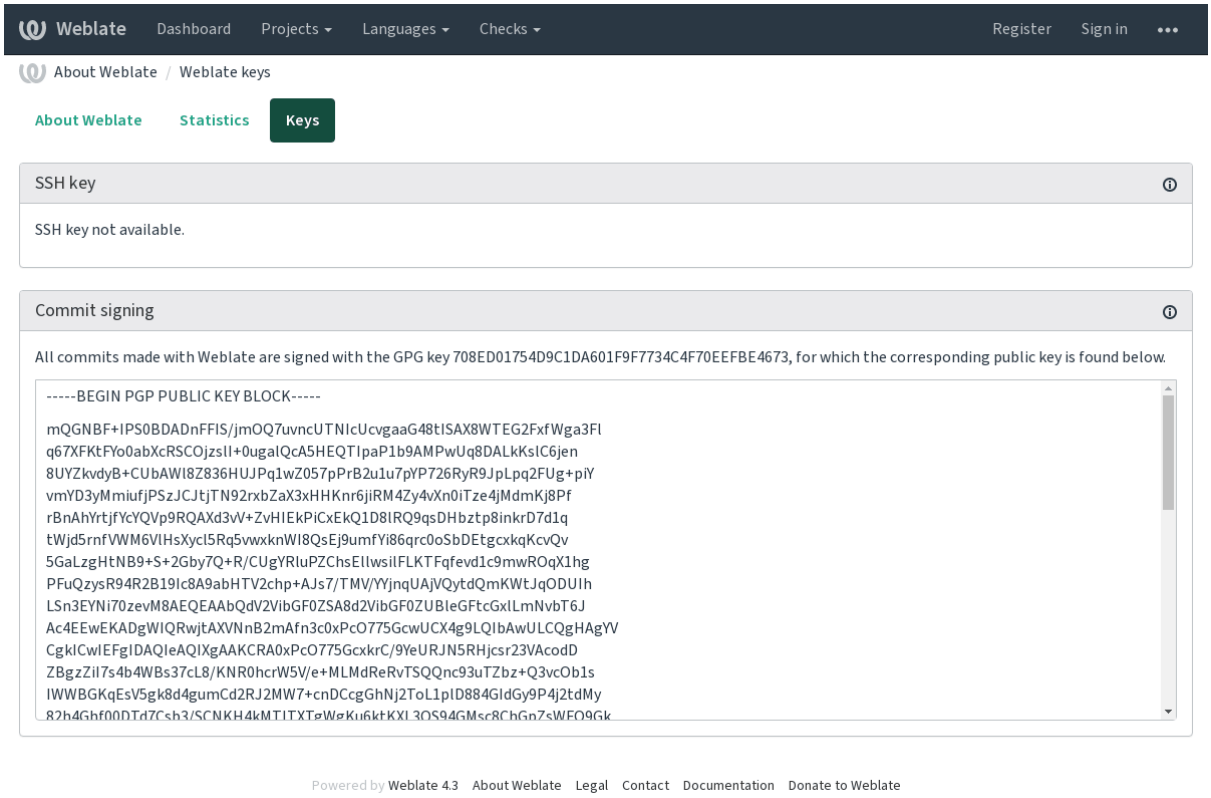
Novo na versão 3.1.

All commits can be signed by the GnuPG key of the Weblate instance.

1. Turn on *WEBLATE_GPG_IDENTITY*. (Weblate will generate a GnuPG key when needed and will use it to sign all translation commits.)

This feature needs GnuPG 2.1 or newer installed.

You can find the key in the *DATA_DIR* and the public key is shown on the «About» page:



2. Alternatively you can also import existing keys into Weblate, just set `HOME=$DATA_DIR/home` when invoking gpg.

Veja também:

WEBLATE_GPG_IDENTITY

2.21.7 Limitação de taxa

Alterado na versão 3.2: The rate limiting now accepts more fine-grained configuration.

Several operations in Weblate are rate limited. At most *RATELIMIT_ATTEMPTS* attempts are allowed within *RATELIMIT_WINDOW* seconds. The user is then blocked for *RATELIMIT_LOCKOUT*. There are also settings specific to scopes, for example `RATELIMIT_CONTACT_ATTEMPTS` or `RATELIMIT_TRANSLATE_ATTEMPTS`. The table below is a full list of available scopes.

The following operations are subject to rate limiting:

Nome	Âmbito	Allowed attempts	Ratelimit window	Lockout period
Registo	`REGISTRATION`	5	300	600
Sending message to admins	`MESSAGE`	5	300	600
Password authentication on login	`LOGIN`	5	300	600
Sitewide search	`SEARCH`	6	60	60
Traduzir	`TRANSLATE`	30	60	600
Adding to glossary	`GLOSSARY`	30	60	600

If a user fails to log in *AUTH_LOCK_ATTEMPTS* times, password authentication will be turned off on the account until having gone through the process of having its password reset.

Veja também:

Limitação de taxa, *Executar por trás de um proxy reverso*

2.22 Personalizar o Weblate

Amplie e personalize a usar Django e Python. Contribua as suas alterações para o upstream acima para que todos possam se beneficiar. Isso reduz os seus custos de manutenção; código no Weblate é cuidado ao alterar interfaces internas ou refatorar o código.

Aviso: Nem interfaces internas nem modelos são considerados uma API estável. Por favor, revise as suas próprias personalizações para cada atualização, as interfaces ou a semântica deles podem mudar sem aviso prévio.

Veja também:

Contribuir ao Weblate

2.22.1 Criar um módulo Python

Se não conheçe o Python, pode olhar para Python For Beginners, que explica o básico e aponta aos tutoriais adicionais.

Para escrever algum código Python personalizado (chamado de módulo), é necessário um lugar para armazená-lo, seja no caminho do sistema (geralmente algo como `/usr/lib/python3.7/site-packages/`) ou no diretório Weblate, que também é adicionado ao caminho de pesquisa do interpretador.

Melhor ainda, transforme a sua personalização num pacote Python adequado:

1. Crie uma pasta para o seu pacote (usaremos *weblate_customization*).
2. Dentro dele, crie um ficheiro `setup.py` para descrever o pacote:

```
from setuptools import setup

setup(
    name = "weblate_customization",
    version = "0.0.1",
    author = "Your name",
    author_email = "yourname@example.com",
    description = "Sample Custom check for Weblate.",
    license = "GPLv3+",
    keywords = "Weblate check example",
    packages=['weblate_customization'],
)
```

3. Crie uma pasta para o módulo Python (também chamado de `weblate_customization`) para o código de personalização.
4. Dentro dele, crie um ficheiro `__init__.py` para garantir que o Python possa importar o módulo.
5. Este pacote agora pode ser instalado a usar **`pip install -e`**. Mais informações a serem encontradas em "Editable" Installs.
6. Uma vez instalado, o módulo pode ser usado na configuração Weblate (por exemplo, `weblate_customization.checks.FooCheck`).

Sua estrutura de módulo deve ser assim:

```
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
```

(continues on next page)

(continuação da página anterior)

```
    ├── addons.py
    └── checks.py
```

Pode encontrar um exemplo de personalização do Weblate em <https://github.com/WeblateOrg/customize-example>, ele abrange todos os tópicos descritos abaixo.

2.22.2 Alterar o logotipo

1. Create a simple Django app containing the static files you want to overwrite (see *Criar um módulo Python*).

2. Adicione-o a `INSTALLED_APPS`:

```
INSTALLED_APPS = (
   # Add your customization as first
   'weblate_customization',

   # Weblate apps are here…
)
```

A marca aparece nos ficheiros seguintes:

`icons/weblate.svg` Logotipo mostrado na barra de navegação.

`logo-*.png` Ícones web dependendo da resolução do ecrã e do navegador web.

`favicon.ico` Ícone web usado por navegadores legados.

`weblate-*.png` Avatares para bots ou utilizadores anônimos. Alguns navegadores web usam-nos como ícones de atalho.

`email-logo.png` Usado em e-mails de notificações.

3. Execute `weblate collectstatic --noinput`, para coletar ficheiros estáticos servidos aos clientes.

Veja também:

Managing static files (e.g. images, JavaScript, CSS), *Servir ficheiros estáticos*

2.22.3 Verificações de qualidade personalizadas, extensões e correções automáticas

Para instalar o seu código para *Correções automáticas personalizadas*, *Escrever próprias verificações* ou *Escrever extensões* e no Weblate:

1. Ponha os ficheiros no módulo Python contendo a personalização ao Weblate (veja *Criar um módulo Python*).

2. Adicione o caminho totalmente qualificado dele à classe Python nas configurações dedicadas (*`WEBLATE_ADDONS`*, *`CHECK_LIST`* ou *`AUTOFIX_LIST`*):

```
# Checks
CHECK_LIST += (
    'weblate_customization.checks.FooCheck',
)

# Autofixes
AUTOFIX_LIST += (
  'weblate_customization.autofix.FooFixer',
)

# Addons
WEBLATE_ADDONS += (
```

(continues on next page)

(continuação da página anterior)

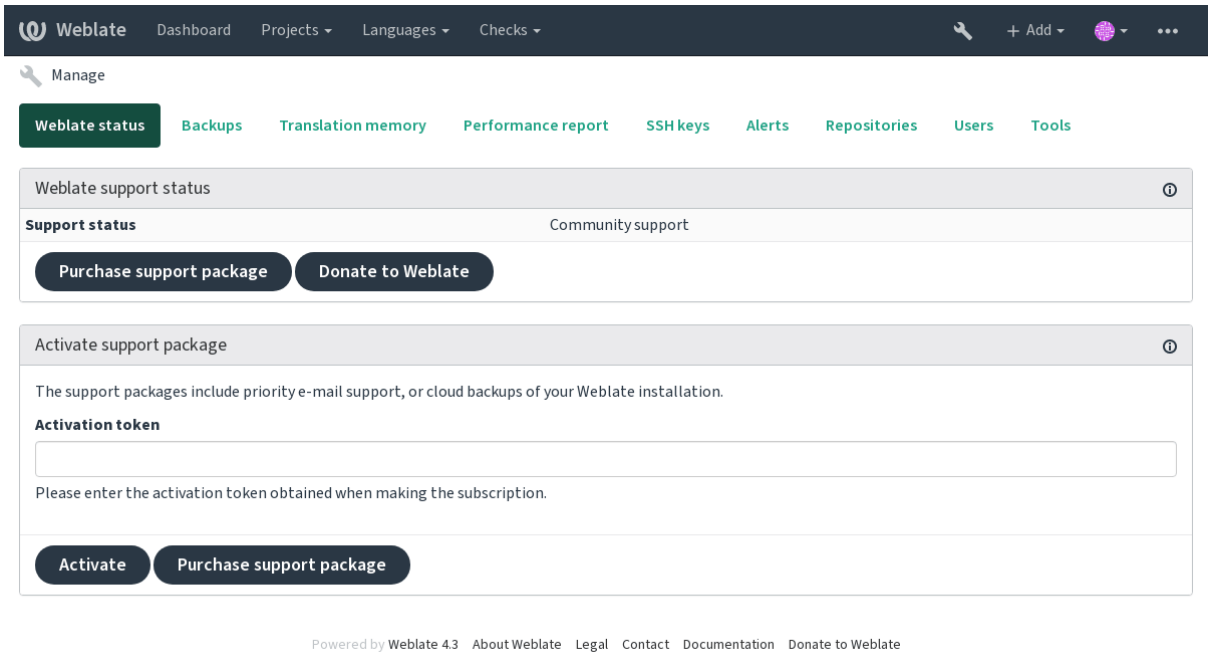
```
    'weblate_customization.addons.ExamplePreAddon',
)
```

Veja também:

Correções automáticas personalizadas, *Escrever próprias verificações*, *Escrever extensões*, *Executar scripts de extensões*

2.23 Interface de gestão

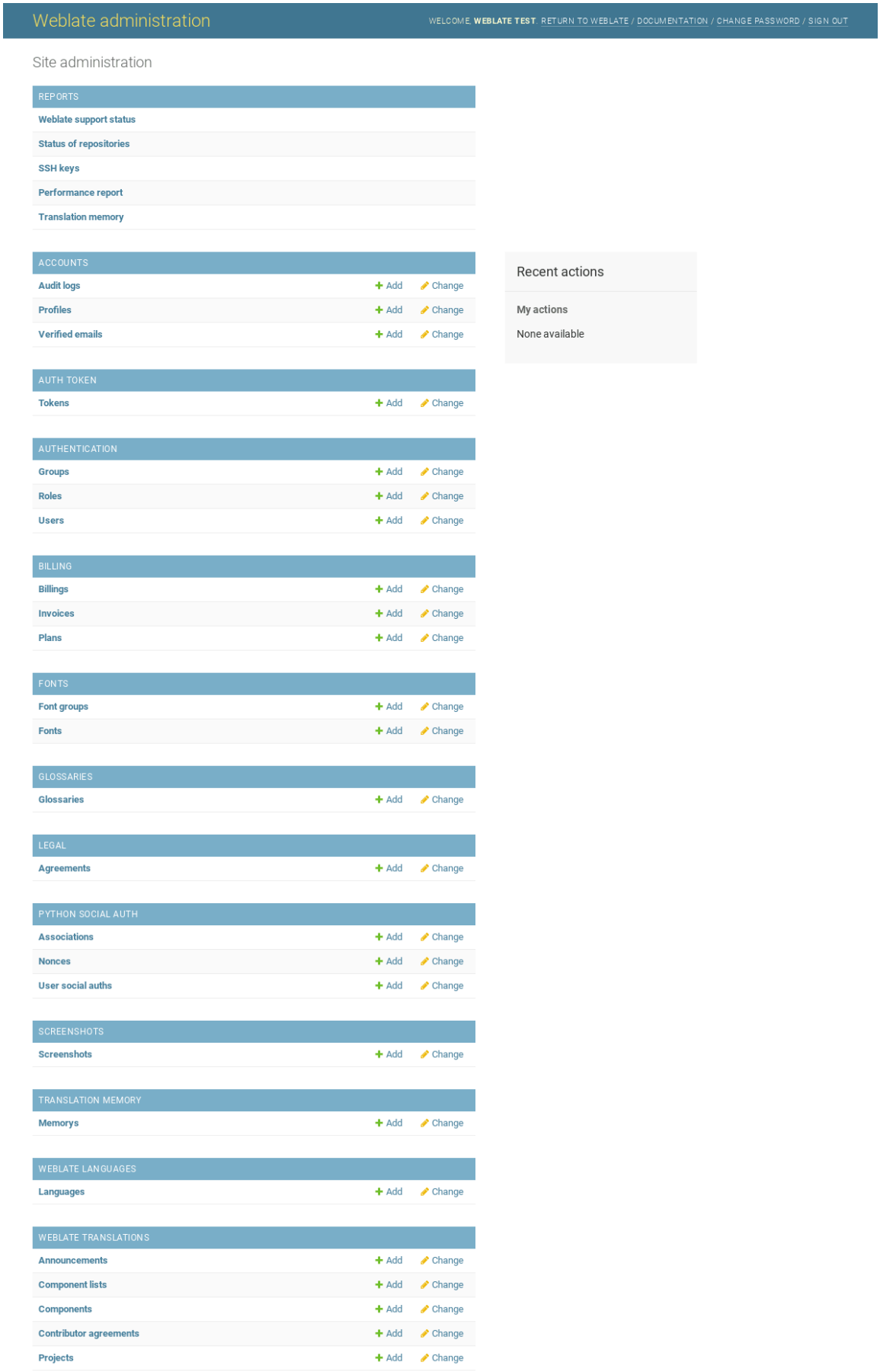
A interface de gestão oferece configurações de administração sob a URL `/management/`. Está disponível para utilizadores que se inscrevem com privilégios administrativos, acessíveis a usar o ícone da chave inglesa no canto superior direito:



2.23.1 A interface administrativa do Django

Aviso: Será removido no futuro, pois o uso dele é desencorajado — a maioria das funcionalidades pode ser geridas diretamente no Weblate.

Aqui pode gerir objetos armazenados no banco de dados, tais como utilizadores, traduções e outras configurações:


Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / SIGN OUT
Site administration
REPORTS
Weblate support status
Status of repositories
SSH keys
Performance report
Translation memory
ACCOUNTS
Audit logs + Add Change
Profiles + Add Change
Verified emails + Add Change
AUTH TOKEN
Tokens + Add Change
AUTHENTICATION
Groups + Add Change
Roles + Add Change
Users + Add Change
BILLING
Billings + Add Change
Invoices + Add Change
Plans + Add Change
FONTS
Font groups + Add Change
Fonts + Add Change
GLOSSARIES
Glossaries + Add Change
LEGAL
Agreements + Add Change
PYTHON SOCIAL AUTH
Associations + Add Change
Nonces + Add Change
User social auths + Add Change
SCREENSHOTS
Screenshots + Add Change
TRANSLATION MEMORY
Memorys + Add Change
WEBLATE LANGUAGES
Languages + Add Change
WEBLATE TRANSLATIONS
Announcements + Add Change
Component lists + Add Change
Components + Add Change
Contributor agreements + Add Change
Projects + Add Change
Recent actions
My actions
None available

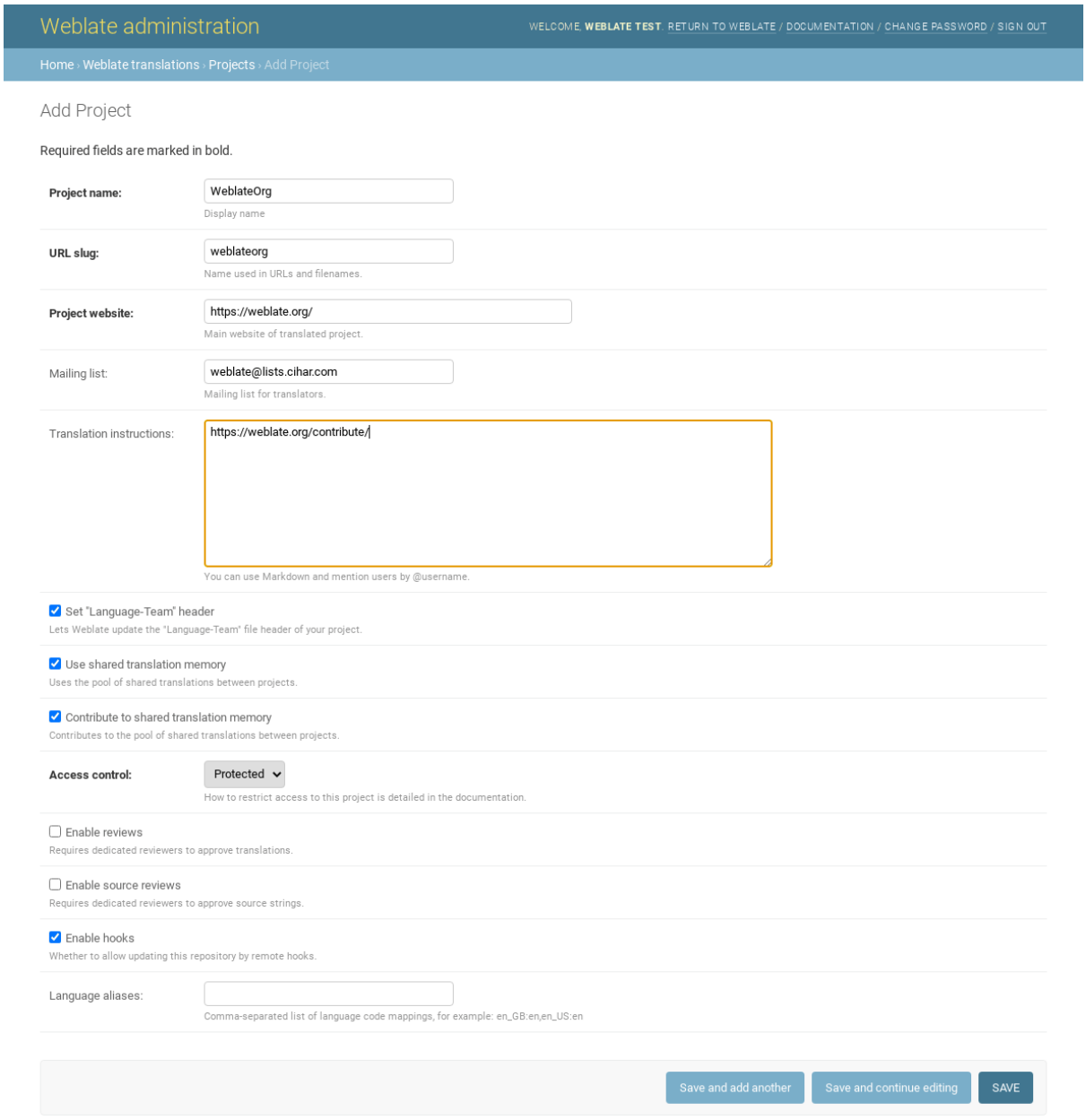
Na secção *Relatórios* pode verificar o estado do seu site, ajustá-lo para produção ou gerir chaves SSH usadas para acessar *Accessing repositories*.

Gerir objetos de banco de dados em qualquer uma das secções abaixo. A mais interessante é provavelmente *Traduções do Weblate*, onde pode gerir projetos traduzíveis, veja *Project configuration* e *Component configuration*.

Idiomas do Weblate detém as definições de idiomas, explicado melhor em *Language definitions*.

Adicionar um projeto

A adição de um projeto serve como contentor para todos os componentes. Normalmente cria um projeto para um software, ou livro (Veja *Project configuration* para informações sobre parâmetros individuais):

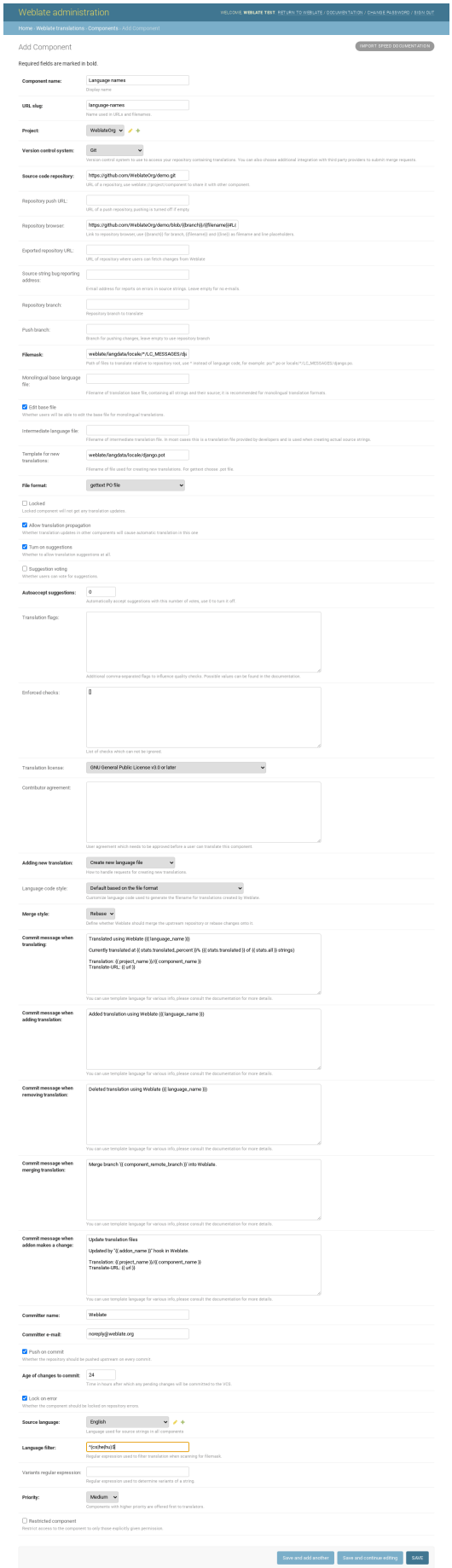


Veja também:

Project configuration

Componentes bilíngues

Uma vez que adicionou um projeto, os componentes de tradução podem ser adicionados-lo. (Ver *Component configuration* para obter informações sobre parâmetros individuais):

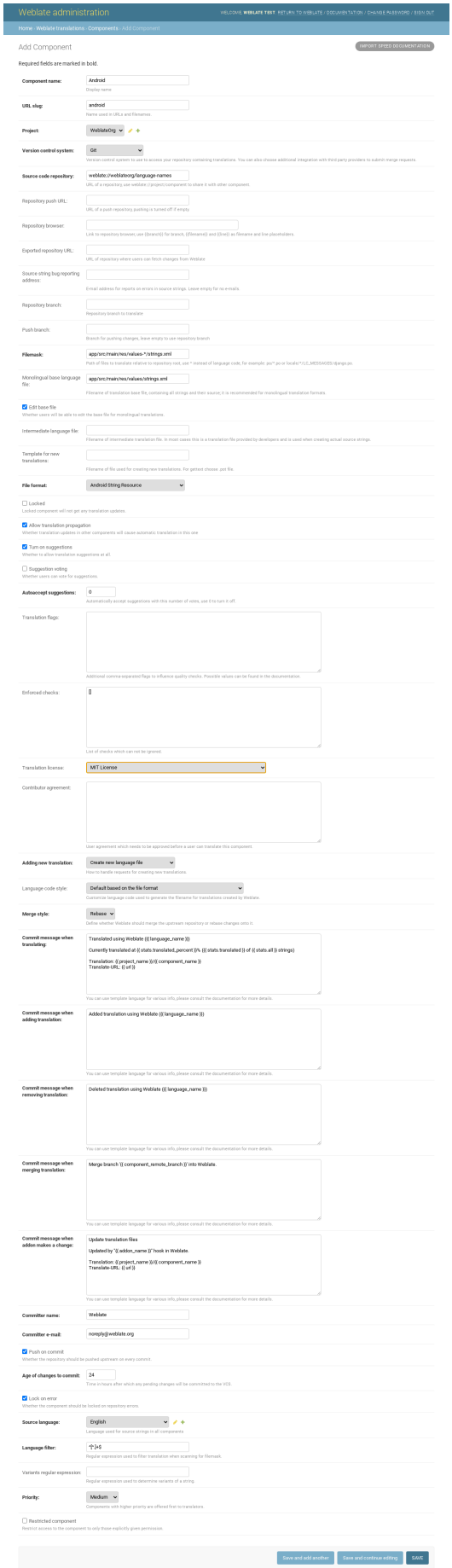

Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / SIGN OUT
Home › Weblate translations › Components › Add Component
Add Component
IMPORT SPEED DOCUMENTATION
Required fields are marked in bold.
Component name:
Language names
Display name
URL slug:
language-names
Name used in URLs and filenames.
Project:
WeblateOrg
Version control system:
Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.
Source code repository:
https://github.com/WeblateOrg/demo.git
URL of a repository, use weblate://project/component to share it with other component.
Repository push URL:
URL of a push repository, pushing is turned off if empty.
Repository browser:
https://github.com/WeblateOrg/demo/blob/{{branch}}/{{filename}}#L{
Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders.
Exported repository URL:
URL of repository where users can fetch changes from Weblate
Source string bug reporting address:
E-mail address for reports on errors in source strings. Leave empty for no e-mails.
Repository branch:
Repository branch to translate
Push branch:
Branch for pushing changes, leave empty to use repository branch
Filemask:
weblate/langdata/locale/^LC_MESSAGES/dj
Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/django.po.
Monolingual base language file:
Filename of translation base file, containing all strings and their source, it is recommended for monolingual translation formats.
Edit base file
Whether users will be able to edit the base file for monolingual translations.
Intermediate language file:
Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.
Template for new translations:
weblate/langdata/locale/django.pot
Filename of file used for creating new translations. For gettext choose .pot file.
File format:
gettext PO file
Locked
Locked component will not get any translation updates.
Allow translation propagation
Whether translation updates in other components will cause automatic translation in this one
Turn on suggestions
Whether to allow translation suggestions at all.
Suggestion voting
Whether users can vote for suggestions.
Autoaccept suggestions:
0
Automatically accept suggestions with this number of votes, use 0 to turn it off.
Translation flags:
Additional comma-separated flags to influence quality checks. Possible values can be found in the documentation.
Enforced checks:
[]
List of checks which can not be ignored.
Translation license:
GNU General Public License v3.0 or later
Contributor agreement:
User agreement which needs to be approved before a user can translate this component.
Adding new translation:
Create new language file
How to handle requests for creating new translations.
Language code style:
Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.
Merge style:
Rebase
Define whether Weblate should merge the upstream repository or rebase changes onto it.
Commit message when translating:
Translated using Weblate ({{ language_name }})
Currently translated at {{ stats.translated_percent }}% ({{ stats.translated }} of {{ stats.all }} strings)
Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}
You can use template language for various info, please consult the documentation for more details.
Commit message when adding translation:
Added translation using Weblate ({{ language_name }})
You can use template language for various info, please consult the documentation for more details.
Commit message when removing translation:
Deleted translation using Weblate ({{ language_name }})
You can use template language for various info, please consult the documentation for more details.
Commit message when merging translation:
Merge branch '{{ component_remote_branch }}' into Weblate.
You can use template language for various info, please consult the documentation for more details.
Commit message when addon makes a change:
Update translation files
Updated by "{{ addon_name }}" hook in Weblate.
Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}
You can use template language for various info, please consult the documentation for more details.
Committer name:
Weblate
Committer e-mail:
noreply@weblate.org
Push on commit
Whether the repository should be pushed upstream on every commit.
Age of changes to commit:
24
Time in hours after which any pending changes will be committed to the VCS.
Lock on error
Whether the component should be locked on repository errors.
Source language:
English
Language used for source strings in all components
Language filter:
^(cs|hu)$
Regular expression used to filter translation when scanning for filemask.
Variants regular expression:
Regular expression used to determine variants of a string.
Priority:
Medium
Components with higher priority are offered first to translators.
Restricted component
Restrict access to the component to only those explicitly given permission.
Save and add another
Save and continue editing
SAVE

Veja também:

Component configuration, *Bilingual and monolingual formats*

Componentes monolínguas

Para facilitar a tradução destes, forneça um ficheiro de modelo contendo o mapeamento de IDs de mensagem para respectivo idioma fonte dele (geralmente inglês). (Ver *Component configuration* para obter informações sobre parâmetros individuais):


Weblate administration
Add Component
Component name:
URL slug:
Project:
Version control system:
Source code repository:
Filemask:
File format:
Translation license:
Save and add another
Save and continue editing
SAVE

Veja também:

Component configuration, *Bilingual and monolingual formats*

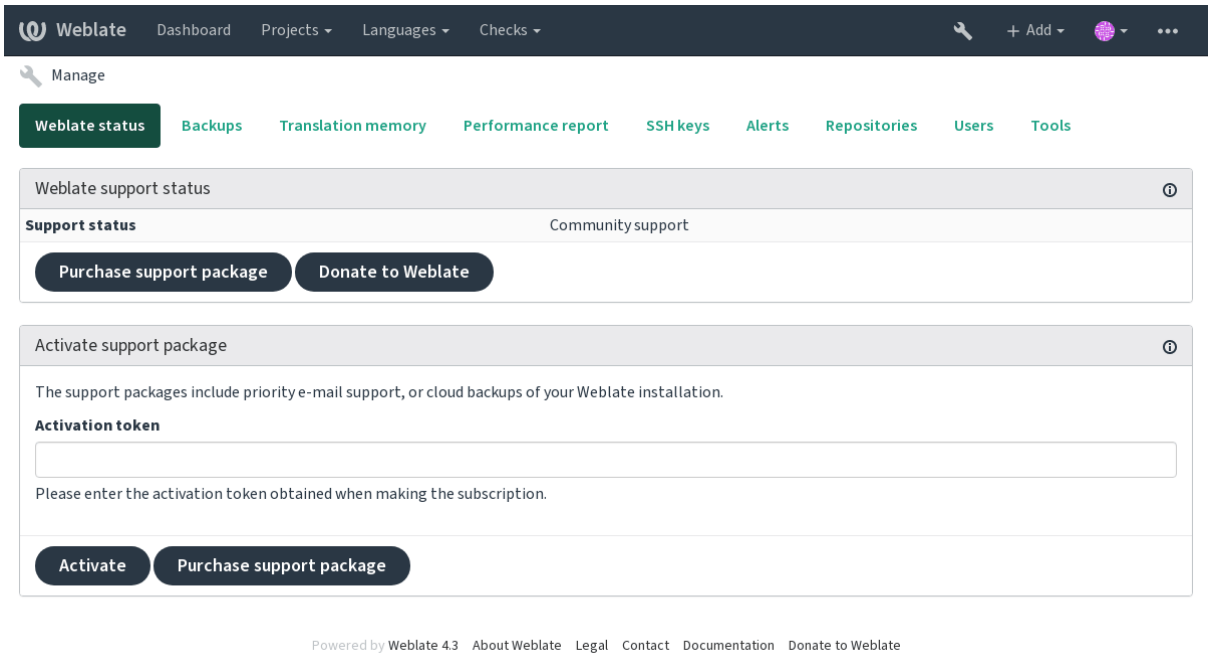
2.24 Obter suporte para o Weblate

Weblate é um software livre protegido por copyleft e com apoio comunitário. Os assinantes recebem apoio prioritário sem custo adicional. Pacotes de ajuda pré-pago estão disponíveis para todos. Pode encontrar mais informações sobre as ofertas de apoio atuais em <https://weblate.org/support/>.

2.24.1 Integrando o apoio

Novo na versão 3.8.

Os pacotes de apoio adquiridos podem ser integrados opcionalmente à sua gestão de assinatura do Weblate, de onde encontrará uma ligação a ele. Detalhes básicos da instância sobre a sua instalação também são relatados de volta ao Weblate desta forma.



2.24.2 Dados enviados ao Weblate

- URL onde a sua instância do Weblate está configurada
- Título do seu site
- A versão do Weblate que está a executar
- Contagem de alguns objetos no seu banco de dados Weblate (projetos, componentes, idiomas, cadeias fonte e utilizadores)
- A chave pública SSH da sua instância

Nenhum outro dado é enviado.

2.24.3 Serviços de integração

- Veja se o seu pacote de apoio ainda é válido
- *Armazenamento de backup provisionado do Weblate*

Dica: Os pacotes de apoio adquiridos já estão ativados no momento da compra e podem ser usados sem integrá-los.

2.25 Documentos legais

Nota: Aqui encontrará várias informações legais que pode precisar para operar Weblate em certas jurisdições legais. É fornecido como um meio de orientação, sem qualquer garantia de precisão ou correção. Em última análise, é a sua responsabilidade de garantir que o seu uso do Weblate esteja em conformidade com todas as leis e regulamentos aplicáveis.

2.25.1 ITAR e outros controles de exportação

O Weblate pode ser usado dentro do seu próprio datacenter ou nuvem privada virtual. Como tal, ele pode ser usado para armazenar informações ITAR ou outras controladas por exportação; no entanto, os utilizadores finais são responsáveis por garantir tal conformidade.

O serviço Hosted Weblate não foi auditado pela conformidade com ITAR ou outros controles de exportação e atualmente não oferece a capacidade de restringir traduções de acesso por país.

2.25.2 Controlos de criptografia dos EUA

O Weblate não contém nenhum código criptográfico, mas pode ser objeto de controles de exportação, pois usa componentes de terceiros utilizando criptografia para autenticação, integridade de dados e confidencialidade.

Provavelmente Weblate seria classificado como ECCN 5D002 ou 5D992 e, como software livre publicamente disponível, não deve ser sujeito ao EAR (veja «Itens de criptografia NÃO estão sujeitos a EAR <https://www.bis.doc.gov/index.php/policy-guidance/encryption/1-encryption-items-not-subject-to-the-ear>`_).

Componentes de software utilizados por Weblate (listando somente os componentes relacionados à função criptográfica):

Python Veja https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq#Is_Python_subject_to_export_laws.3F

GnuPG Opcionalmente usado pelo Weblate

Git Opcionalmente usado pelo Weblate

curl Usado pelo Git

OpenSSL Usado pelo Python e cURL

A força de chaves de criptografia depende da configuração do Weblate e os componentes de terceiros que interage com ele, mas em qualquer decente instalação, irá incluir todas as funções criptográficas com exportação restrita:

- Em excesso de 56 bits para um algoritmo simétrico
- Fatorização de inteiros acima de 512 bits para um algoritmo assimétrico
- Cálculo de logaritmos discretos num grupo multiplicativo de um campo finito de tamanho maior do que 512 bits para um algoritmo assimétrico
- Logaritmos discretos num grupo diferente do que acima de 112 bits para um algoritmo assimétrico

O Weblate não tem nenhum recurso de ativação criptográfica, mas pode ser configurado de maneira sem ter nenhum código de criptografia envolvido. Os recursos criptográficos incluem:

- Acessar servidores remotos a usar protocolos seguros (HTTPS)
- Gerar assinaturas para commits de código (PGP)

Veja também:

Controles de Exportação (EAR) em Software de Código Aberto *(inglês)*

CAPÍTULO 3

Documentação de colaborador

3.1 Contribuir ao Weblate

Há dezenas de maneiras de contribuir no Weblate. Qualquer ajuda é bem-vinda, seja codificação, design gráfico, documentação ou patrocínio:

- *Reporting issues in Weblate*
- *Starting contributing code to Weblate*
- *Traduzir o Weblate*
- *Financiar o desenvolvimento do Weblate*

3.1.1 Traduzir o Weblate

O Weblate está a ser traduzido pelo próprio Weblate. Sinta-se à vontade para participar do esforço de disponibilizar o Weblate na maior quantidade possível de idiomas humanos.

3.1.2 Financiar o desenvolvimento do Weblate

Pode financiar mais desenvolvimento do Weblate na página de doação. Os fundos coletados são lá usados para financiar a hospedagem grátis para projetos de software livre e o desenvolvimento adicional do Weblate. Por favor, verifique a *página de doação* para obter detalhes, como metas de financiamento e recompensas que pode obter por ser um financiador.

Apoiadores que financiaram o Weblate

Lista de apoiadores do Weblate:

- Yashiro Ccs
- Cheng-Chia Tseng
- Timon Reinhard
- Cassidy James
- Loic Dachary
- Marozed
- https://freedombox.org/
- GNU Solidario (GNU Health)

Gostaria de estar na lista? Veja as opções em Doar ao Weblate.

3.2 Starting contributing code to Weblate

To understand Weblate source code, please first look into *Código-fonte do Weblate*, *Weblate frontend* and *Weblate internals*.

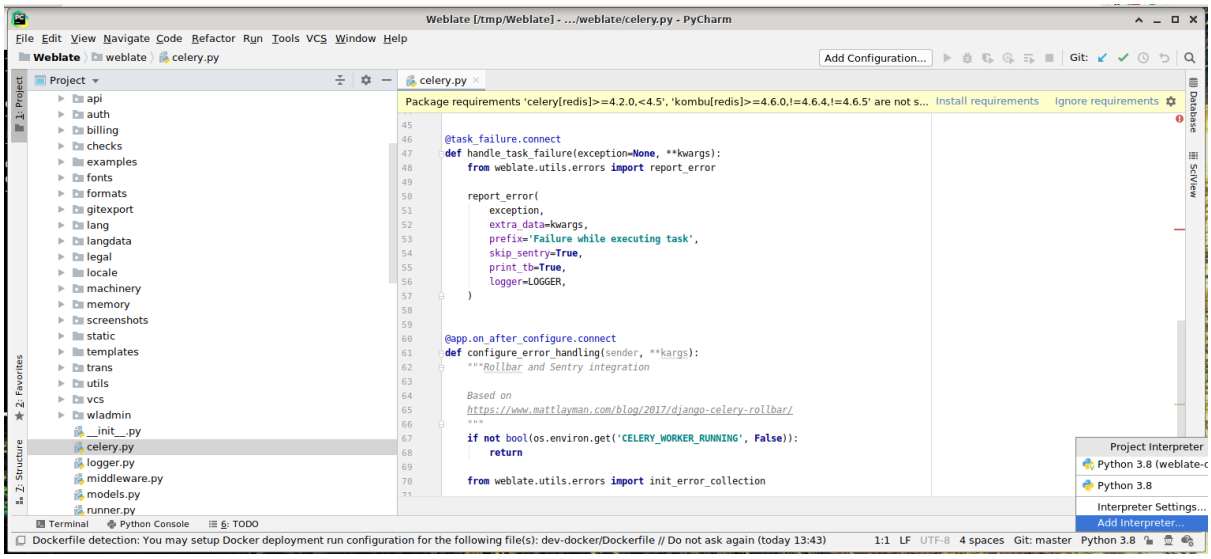
3.2.1 Starting with our codebase

If looking for some bugs to familiarize yourself with the Weblate codebase, look for ones labelled good first issue.

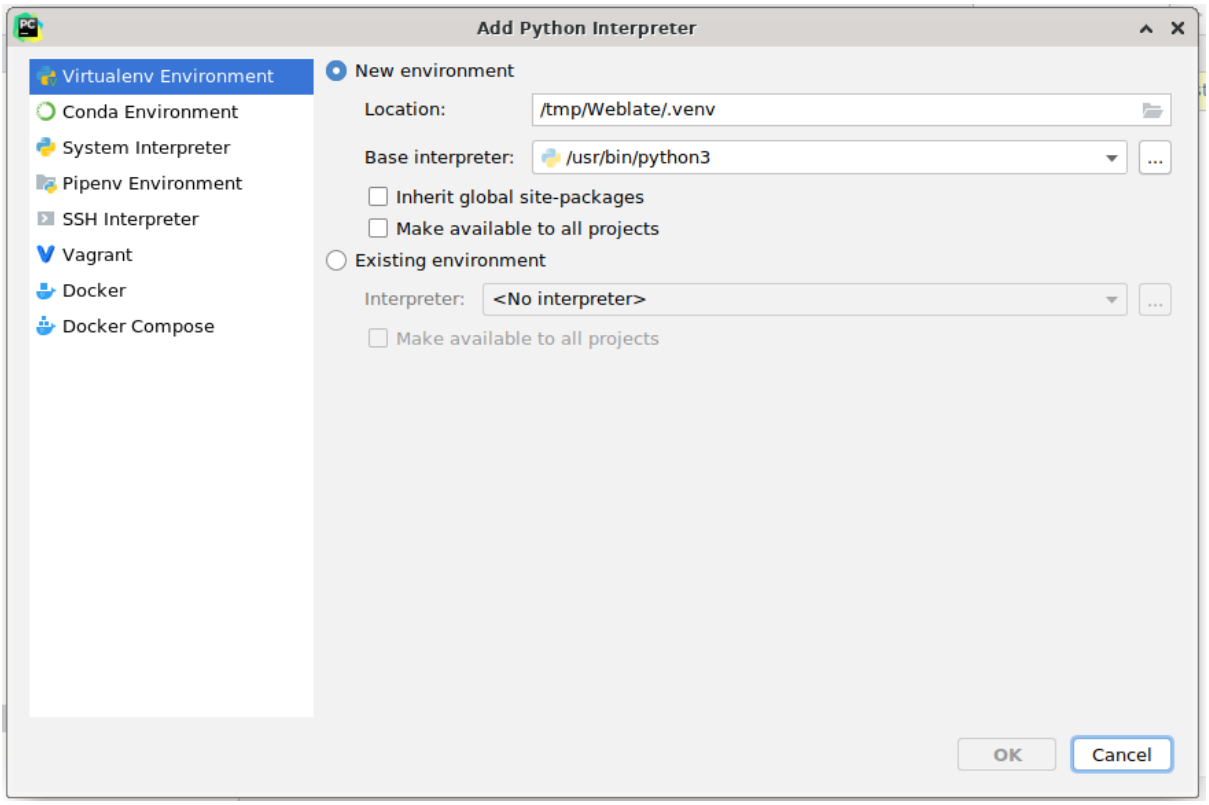
3.2.2 Coding Weblate with PyCharm

PyCharm is a known IDE for Python, here's some guidelines to help you setup Weblate project in it.

Considering you have just cloned the Github repository, just open the folder in which you cloned it in PyCharm. Once the IDE is open, the first step is to specify the interpreter you want:

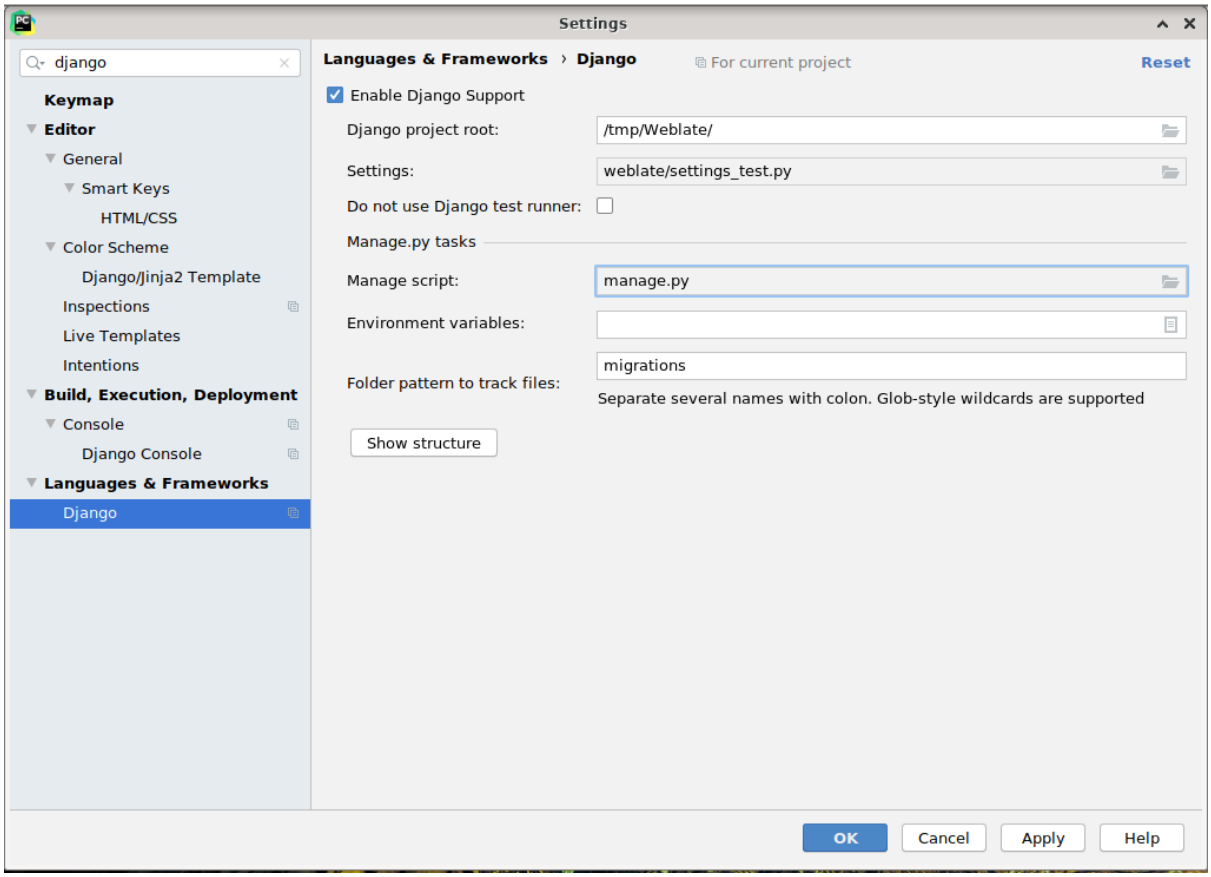


You can either choose to let PyCharm create the virtualenv for you, or select an already existing one:



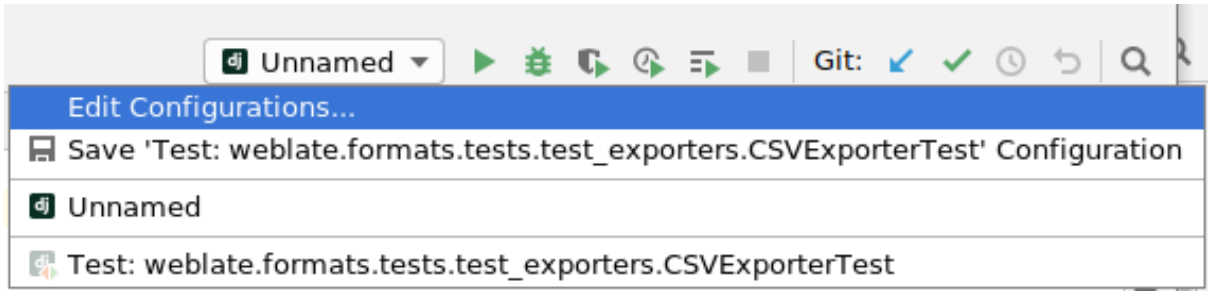
Don't forget to install the dependencies once the interpreter is set: you can do it, either through the console (the console from the IDE will directly use your virtualenv by default), or through the interface when you get a warning about missing dependencies.

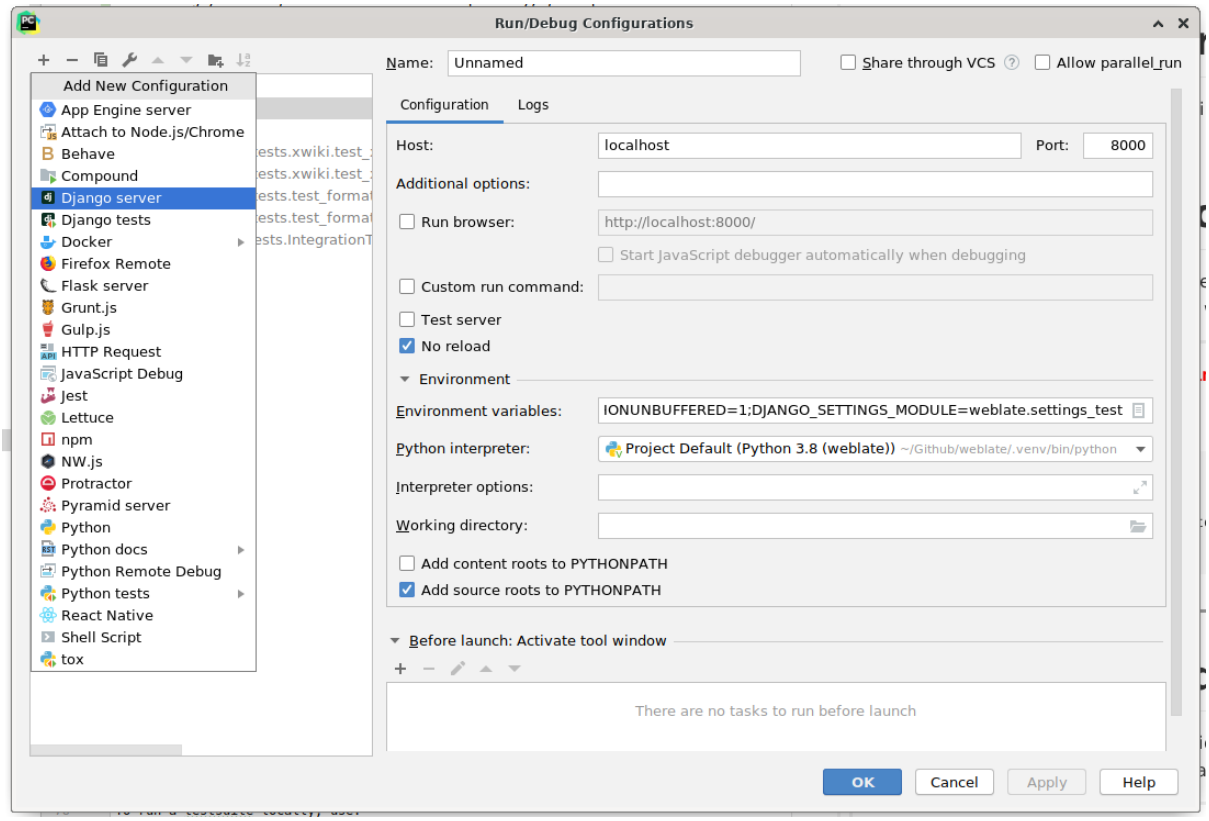
The second step is to set the right information to use natively Django inside PyCharm: the idea is to be able to immediately trigger the unit tests in the IDE. For that you need to specify the root path of Django and the path of one setting:



Be careful, the *Django project root* is the root of the repository, not the weblate sub-directory. About the settings, I personally use the *settings_test* from the repository, but you could create your own setting and set it there.

Last step is to be able to run the server and to put breakpoints on the code to be able to debug it. This is done by creating a new *Django Server* configuration:





Be careful to properly checked «No reload»: you won't get anymore the server live reload if you modify some files, but the debugger will be stopped on the breakpoint you set.

3.2.3 Running Weblate locally

The most comfortable approach to get started with Weblate development is to follow *Installing from sources*. It will get you a virtual env with editable Weblate sources.

To install all dependencies useful for development, do:

```
pip install -r requirements-dev.txt
```

To start a development server run:

```
weblate runserver
```

Depending on your configuration you might also want to start Celery workers:

```
./weblate/examples/celery start
```

Running Weblate locally in Docker

If you have Docker and docker-compose installed, you can spin up the development environment simply by running:

```
./rundev.sh
```

It will create development Docker image and start it. Weblate is running on <http://127.0.0.1:8080/> and you can sign in with `admin` user and `admin` password. The new installation is empty, so you might want to continue with *Adding translation projects and components*.

The `Dockerfile` and `docker-compose.yml` for this are located in `dev-docker` directory.

The script also accepts some parameters, to execute tests run it with `test` parameter and then specify any `test` parameters, for example:

```
./rundev.sh test --failfast weblate.trans
```

Be careful that your Docker containers are up and running before running the tests. You can check that by running the `docker ps` command.

To stop the background containers run:

```
./rundev.sh stop
```

Running the script without args will recreate Docker container and restart it.

Nota: This is not suitable setup for production, it includes several hacks which are insecure, but make development easier.

3.2.4 Bootstraping your devel instance

You might want to use *`import_demo`* to create demo translations and *`createadmin`* to create admin user.

3.3 Código-fonte do Weblate

O Weblate é desenvolvido no GitHub. É bem-vindo para criar um fork do código e abrir pull requests. Patches em qualquer outra forma também são bem-vindos.

Veja também:

Confira *Weblate internals* para ver como o Weblate se parece por dentro.

3.3.1 Princípios de Segurança por Design

Qualquer código para Weblate deve ser escrito com *Princípios de Segurança por Design* (inglês) em mente.

3.3.2 Padrão de codificação

O código deve seguir as diretrizes de codificação PEP-8 e deve ser formatado a usar o formatador de código **`black`**.

Para verificar a qualidade do código, pode usar o **:programa:`flake8`**, os plugins recomendados estão listados em `.pre-commit-config.yaml` e a configuração dele está em `setup.cfg`.

A abordagem mais fácil para impor tudo isso é instalar pre-commit. O repositório do Weblate contém a configuração para verificar se os ficheiros do commit estão sãos. Depois de instalá-lo (ele já está incluído no `requirements-lint.txt`), ative-o executando `pré-commit install` na sua cópia do Weblate. Desta forma, todas as suas alterações serão verificadas automaticamente.

Também pode acionar a verificação manualmente, para verificar todos os ficheiros execute:

```
pre-commit run --all
```

3.4 Debugging Weblate

Bugs can behave as application crashes or as misbehavior. You are welcome to collect info on any such issue and submit it to the issue tracker.

3.4.1 Modo de depuração

Turning on debug mode will make the exceptions show in the browser. This is useful to debug issues in the web interface, but not suitable for production environment as it has performance consequences and might leak private data.

Veja também:

Desativar o modo de depuração

3.4.2 Weblate logs

Weblate can produce detailed logs of what is going in the background. In the default configuration it uses syslog and that makes the log appear either in `/var/log/messages` or `/var/log/syslog` (depending on your syslog daemon configuration).

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using `docker-compose logs`.

Veja também:

Sample configuration contains `LOGGING` configuration.

3.4.3 Analyzing application crashes

In case the application crashes, it is useful to collect as much info about the crash as possible. The easiest way to achieve this is by using third-party services which can collect such info automatically. You can find info on how to set this up in *Collecting error reports*.

3.4.4 Silent failures

Lots of tasks are offloaded to Celery for background processing. Failures are not shown in the user interface, but appear in the Celery logs. Configuring *Collecting error reports* helps you to notice such failures easier.

3.4.5 Performance issues

In case Weblate performs badly in some situation, please collect the relevant logs showing the issue, and anything that might help figuring out where the code might be improved.

In case some requests take too long without any indication, you might want to install *dogslow <https://pypi.org/project/dogslow/>* along with *Collecting error reports* and get pinpointed and detailed tracebacks in the error collection tool.

3.5 Weblate internals

Nota: This chapter will give you basic overview of Weblate internals.

Weblate derives most of its code structure from, and is based on Django.

3.5.1 Directory structure

Quick overview of directory structure of Weblate main repository:

docs Source code for this documentation, built using Sphinx.

dev-docker Docker code to run development server, see *Running Weblate locally in Docker*.

weblate Source code of Weblate as a Django application, see *Weblate internals*.

weblate/static Client files (CSS, Javascript and images), see *Weblate frontend*.

3.5.2 Modules

Weblate consists of several Django applications (some optional, see *Optional Weblate modules*):

`accounts`
: User account, profiles and notifications.

`addons`
: Addons to tweak Weblate behavior, see *Extensões*.

`api`
: API based on Django REST framework.

`auth`
: Authentication and permissions.

`billing`
: The optional *Faturação* module.

`checks`
: Translation string *Verificações de qualidade* module.

`fonts`
: Font rendering checks module.

`formats`
: File format abstraction layer based on translate-toolkit.

`gitexport`
: The optional *Git exporter* module.

`lang`
: Module defining language and plural models.

`legal`
: The optional *Legal* module.

`machinery`

Integration of machine translation services.

`memory`

Built in translation memory, see *Memória de Tradução*.

`screenshots`

Screenshots management and OCR module.

`trans`

Main module handling translations.

`utils`

Various helper utilities.

`vcs`

Version control system abstraction.

`wladmin`

Django admin interface customization.

3.6 Weblate frontend

The frontend is currently built using Bootstrap, jQuery and few third party libraries.

3.6.1 Dependency management

The yarn package manager is used to update third party libraries. The configuration lives in `scripts/yarn` and there is a wrapper script `scripts/yarn-update` to upgrade the libraries, build them and copy to correct locations in `weblate/static/vendor`, where all third partly frontend code is located.

3.6.2 Coding style

Weblate relies on Prettier for the code formatting for both JavaScript and CSS files.

We also use ESLint to check the JavaScript code.

3.6.3 Tradução

Should you need any user visible text in the frontend code, it should be localizable. In most cases all you need is to wrap your text inside `gettext` function, but there are more complex features available:

```
document.write(gettext('this is to be translated'));

var object_count = 1 // or 0, or 2, or 3, ...
s = ngettext('literal for the singular case',
        'literal for the plural case', object_count);

fmts = ngettext('There is %s object. Remaining: %s',
        'There are %s objects. Remaining: %s', 11);
s = interpolate(fmts, [11, 20]);
// s is 'There are 11 objects. Remaining: 20'
```

Veja também:

Translation topic in the Django documentation

3.6.4 Icons

Weblate currently uses material design icons, in case you are looking for new one, check <https://materialdesignicons.com/>.

Additionally, there is `scripts/optimize-svg` to reduce size of the SVG as most of the icons are embedded inside the HTML to allow styling of the paths.

3.7 Reporting issues in Weblate

Our issue tracker is hosted at GitHub:

Feel welcome to report any issues with, or suggest improvement of Weblate there. If what you have found is a security issue in Weblate, please consult the «Security issues» section below.

3.7.1 Problemas de segurança

In order to give the community time to respond and upgrade you are strongly urged to report all security issues privately. HackerOne is used to handle security issues, and can be reported directly at HackerOne.

Alternatively, report to security@weblate.org, which ends up on HackerOne as well.

If you don't want to use HackerOne, for whatever reason, you can send the report by e-mail to michal@cihar.com. You can choose to encrypt it using this PGP key *3CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D*. You can also get the PGP key from Keybase.

Nota: Weblate depends on third party components for many things. In case you find a vulnerability affecting one of those components in general, please report it directly to the respective project.

Some of these are:

- Django
- Django REST framework
- Python Social Auth

3.8 Weblate testsuite and continuous integration

Testsuites exist for most of the current code, increase coverage by adding testcases for any new functionality, and verify that it works.

3.8.1 Continuous integration

Current test results can be found on GitHub Actions and coverage is reported on Codecov.

There are several jobs to verify different aspects:

- Unit tests
- Documentation build and external links
- Migration testing from all supported releases
- Code linting
- Setup verification (ensures that generated dist files do not miss anything and can be tested)

The configuration for the CI is in `.github/workflows` directory. It heavily uses helper scripts stored in `ci` directory. The scripts can be also executed manually, but they require several environment variables, mostly defining Django settings file to use and database connection. The example definition of that is in `scripts/test-database`:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

The simple execution can look like:

```
. scripts/test-database
./ci/run-migrate
./ci/run-test
./ci/run-docs
./ci/run-setup
```

3.8.2 Local testing

To run a testsuite locally, use:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

Dica: You will need a database (PostgreSQL) server to be used for tests. By default Django creates separate database to run tests with `test_` prefix, so in case your settings is configured to use `weblate`, the tests will use `test_weblate` database. See *Configuração de banco de dados para o Weblate* for setup instructions.

The `weblate/settings_test.py` is used in CI environment as well (see *Continuous integration*) and can be tuned using environment variables:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

Prior to running tests you should collect static files as some tests rely on them being present:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py collectstatic
```

You can also specify individual tests to run:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test weblate.gitexport
```

Dica: The tests can also be executed inside developer docker container, see *Running Weblate locally in Docker*.

Veja também:

See Testing in Django for more info on running and writing tests for Django.

3.9 Data schemas

Weblate uses JSON Schema to define layout of external JSON files.

3.9.1 Esquema de memória de tradução Weblate

https://weblate.org/schemas/weblate-memory.schema.json			
tipo	*array*		
items	*O item de memória de tradução*		
	tipo	*objeto*	
	propriedades		
	• **categoria**	*A categoria de cadeia*	
		1 is global, 2 is shared, 10000000+ are project specific, 20000000+ are user specific	
		tipo	*integer*
		exemplos	1
		minimum	0
		predefinido	1
	• **origem**	*The String Origin*	
		Nome do ficheiro ou componente	
		tipo	*cadeia*
		exemplos	test
		pattern	^(.*)$
		predefinido	
	• **source**	*A cadeia fonte*	
		tipo	*cadeia*
		exemplos	Olá
		pattern	^(.+)$
		predefinido	
	• **idioma_ fonte**	**O idioma fonte**	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		tipo	*cadeia*
		exemplos	en
		pattern	^([^]+)$
		predefinido	
	• **target**	*A cadeia de destino*	
		tipo	*cadeia*
		exemplos	Ahoj
		pattern	^(.+)$
		predefinido	
	• **idioma_de_destino**	*O idioma de destino*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		tipo	*cadeia*
		exemplos	cs

continues on next page

Table 1 – continuação da página anterior

		pattern	^([^]+)$
		predefinido	
	additionalProperties	False	
definições			

Veja também:

Memória de Tradução, `dump_memory`, `import_memory`

3.9.2 Exportação de dados de utilizadores do Weblate

https://weblate.org/schemas/weblate-userdata.schema.json			
tipo	*objeto*		
propriedades			
• **basic**	*Básico*		
	tipo	*objeto*	
	propriedades		
	• **nome do utilizador**	*Nome do utilizador*	
		tipo	*cadeia*
		exemplos	administrador
		pattern	^.*$
		predefinido	
	• **nome_completo**	*Nome completo*	
		tipo	*cadeia*
		exemplos	Admin Weblate
		pattern	^.*$
		predefinido	
	• **email**	*E-mail*	
		tipo	*cadeia*
		exemplos	noreply@example.com
		pattern	^.*$
		predefinido	
	• **data_de_adesão**	*Data de adesão*	
		tipo	*cadeia*
		exemplos	2019-11-18T18:53:54.862Z
		pattern	^.*$
		predefinido	
• **perfil**	*Perfil*		
	tipo	*objeto*	
	propriedades		
	• **idioma**	*Idioma*	
		tipo	*cadeia*
		exemplos	cs
		pattern	^.*$
		predefinido	
	• **sugerido**	*Quantidade de cadeias fonte*	
		tipo	*integer*
		exemplos	1
		predefinido	0
	• **traduzido**	*Quantidade de cadeias traduzidas*	
		tipo	*integer*
		exemplos	24
		predefinido	0
	• **enviado**	*Quantidade de capturas de ecrã enviadas*	
		tipo	*integer*

continues on next page

Table 2 – continuação da página anterior

		exemplos	1	
		predefinido	0	
	• **hide_completed**	*Ocultar as traduções completas no painel*		
		tipo	*boolean*	
		exemplos	False	
		predefinido	True	
	• **secondary_in_zen**	*Mostrar traduções secundárias no modo Zen*		
		tipo	*boolean*	
		exemplos	True	
		predefinido	True	
	• **hide_source_secondary**	*Ocultar a fonte se existir uma tradução secundária*		
		tipo	*boolean*	
		exemplos	False	
		predefinido	True	
	• **hiperligação_do_editor**	*Hiperligação_do_editor*		
		tipo	*cadeia*	
		exemplos		
		pattern	^.*$	
		predefinido		
	• **modo_de_tradução**	*Modo do editor de tradução*		
		tipo	*integer*	
		exemplos	0	
		predefinido	0	
	• **zen_mode**	*Modo de editor Zen*		
		tipo	*integer*	
		exemplos	0	
		predefinido	0	
	• **carateres_especiais**	*Carateres especiais*		
		tipo	*cadeia*	
		exemplos		
		pattern	^.*$	
		predefinido		
	• **vista_do_painel**	*Vista predefinida do painel*		
		tipo	*integer*	
		exemplos	1	
		predefinido	0	
	• **dashboard_component_list**	*Lista predefinida de componentes*		
		predefinido	Nenhum	
		anyOf	tipo	*null*
			tipo	*integer*
	• **idiomas**	*Idiomas traduzidos*		
		tipo	*array*	
		predefinido		
		items	*Código do idioma*	
			tipo	*cadeia*
			exemplos	cs
			pattern	^.*$
			predefinido	
	• **idiomas_secundários**	*Idiomas secundários*		
		tipo	*array*	
		predefinido		
		items	*Código do idioma*	
			tipo	*cadeia*
			exemplos	sk
			pattern	^.*$
			predefinido	

continues on next page

Table 2 – continuação da página anterior

	• **observado**	*Projetos vigiados*		
		tipo	*array*	
		predefinido		
		items	*Slug do projeto*	
			tipo	*cadeia*
			exemplos	weblate
			pattern	^.*$
			predefinido	
• **registo de auditoria**	*Registo de auditoria*			
	tipo	*array*		
	predefinido			
	items	*Items*		
		tipo	*objeto*	
		propriedades		
		• **endereço**	*Endereço IP*	
			tipo	*cadeia*
			exemplos	127.0.0.1
			pattern	^.*$
			predefinido	
		• **agente_do_utilizador**	*Agente do utilizador*	
			tipo	*cadeia*
			exemplos	PC / Linux / Firefox 70.0
			pattern	^.*$
			predefinido	
		• **timestamp**	*Timestamp*	
			tipo	*cadeia*
			exemplos	2019-11-18T18:58:30.845Z
			pattern	^.*$
			predefinido	
		• **atividade**	*Atividade*	
			tipo	*cadeia*
			exemplos	sessão
			pattern	^.*$
			predefinido	
definições				

Veja também:

Perfil do utilizador, `dumpuserdata`

3.10 Releasing Weblate

Things to check prior to release:

1. Check newly translated languages by **`./scripts/list-translated-languages`**.
2. Set final version by **`./scripts/prepare-release`**.
3. Make sure screenshots are up to date **`make -C docs update-screenshots`**

Perform the release:

4. Create a release **`./scripts/create-release --tag`** (see below for requirements)

Post release manual steps:

5. Update Docker image.

6. Close GitHub milestone.
7. Once the Docker image is tested, add a tag and push it.
8. Update Helm chart to new version.
9. Include new version in `.github/workflows/migrations.yml` to cover it in migration testing.
10. Increase version in the repository by **`./scripts/set-version`**.

To create tags using the **`./scripts/create-release`** script you will need following:

- GnuPG with private key used to sign the release
- Push access to Weblate git repositories (it pushes tags)
- Configured **`hub`** tool and access to create releases on the Weblate repo
- SSH access to Weblate download server (the Website downloads are copied there)

3.11 Sobre o Weblate

3.11.1 Objetivos do projeto

Ferramenta de localização contínua baseada na web com *Integração de controlo de versões* suportando uma ampla gama de *Formatos de ficheiros suportados*, facilitando a contribuição dos tradutores.

3.11.2 Nome do projeto

«Weblate» é uma palavra-valise das palavras «web» e «translate».

3.11.3 Site da Web do Projeto

A página inicial é <https://weblate.org/> e um serviço hospedado na nuvem em <https://hosted.weblate.org/>. Esta documentação encontra-se em <https://docs.weblate.org/>.

3.11.4 Logotipos do projeto

Os logotipos do projeto e outros gráficos estão disponíveis no repositório <https://github.com/WeblateOrg/graphics/>.

3.11.5 Liderança

Este projeto é mantido por Michal Čihař <michal@cihar.com>.

3.11.6 Autores

Weblate foi iniciado por Michal Čihař <michal@cihar.com>. Desde a criação em 2012, milhares de pessoas contribuíram.

3.12 Licença

Copyright (C) 2012 - 2020 Michal Čihař <michal@cihar.com>

Este programa é um software livre: pode redistribuí-lo e/ou modificá-lo sob os termos da Licença Pública Geral GNU, conforme publicado pela Free Software Foundation, seja a versão 3 da Licença, ou (ao seu critério) qualquer versão posterior.

Este programa é distribuído na esperança de que ele seja útil, mas sem qualquer garantia; sem sequer a garantia implícita de COMERCIALIZAÇÃO ou ADEQUAÇÃO PARA UM PROPÓSITO ESPECÍFICO. Consulte a Licença Pública Geral GNU para obter mais detalhes.

Deve ter recebido uma cópia da Licença Pública Geral GNU junto com este programa. Caso contrário, veja <https://www.gnu.org/licenses/>.

CAPÍTULO 4

Histórico de alterações

4.1 Weblate 4.3.2

Released on November 4th 2020.

- Fixed crash on certain component filemasks.
- Melhoria da precisão da verificação das palavras duplicadas consecutivas.
- Suporte adicional para solicitações de Pagure.
- Improved error messages on failed registraiton.
- Reverted rendering developer comments as markdown.
- Simplified setup of Git repositories with different default branch than master.
- Newly created internal repositories now use main as default branch.
- Reduced false positives rate of unchanged translation while translating reStructuredText.
- Fixed CodeMirror display issues in some situations.
- Renamed Template group to Sources to clarify its meaning.
- Fixed GitLab pull requests on repos with longer path.

4.2 Weblate 4.3.1

Released on October 21st 2020.

- Melhoria do desempenho da tradução automática.
- Expiração da sessão para utilizadores autenticados corrigida.
- Suporte para ocultar informações da versão adicionado.
- Improve hooks compatibility with Bitbucket Server.
- Desempenho de atualizações da memória de tradução melhorada.
- Reduced memory usage.
- Melhor desempenho da visão matriz.

- Confirmação antes de remover o utilizador de um projeto adicionada.

4.3 Weblate 4.3

Released on October 15th 2020.

- Include user stats in the API.
- Fixed component ordering on paginated pages.
- Define source language for a glossary.
- Rewritten support for GitHub and GitLab pull requests.
- Contagens de estatísticas corrigidas após a remoção da sugestão.
- Perfil do utilizador público estendido.
- Fixed configuration of enforced checks.
- Improve documentation about built-in backups.
- Moved source language attribute from project to a component.
- Adicionar a verificação de formatação Vue I18n.
- Generic placeholders check now supports regular expressions.
- Improved look of matrix mode.
- A maquinaria é agora chamada sugestões automáticas.
- Added support for interacting with multiple GitLab or GitHub instances.
- Extended API to cover project updates, unit updates and removals and glossaries.
- Unit API now properly handles plural strings.
- Component creation can now handle ZIP file or document upload.
- Consolidated API response status codes.
- Suportar markdown no acordo do contribuinte.
- Rastreamento de cadeias de origem melhorado.
- Improved JSON, YAML and CSV formats compatibility.
- Suporte adicional para remover cadeias.
- Improved performance of file downloads.
- Improved repository management view.
- Automatically enable java-format for Android.
- Suporte adicional para capturas de ecrã localizadas.
- Suporte adicional para Python 3.9.
- Fixed translating HTML files under certain conditions.

4.4 Weblate 4.2.2

Released on September 2nd 2020.

- Correspondência de textos fonte para formatos JSON corrigido.
- Fixed login redirect for some authentication configurations.
- Autenticação LDAP corrigida com sincronização de grupo.
- Falha na comunicação do progresso da tradução automática corrigida.
- Fixed Git commit squashing with trailers enabled.
- Fixed creating local VCS components using API.

4.5 Weblate 4.2.1

Released on August 21st 2020.

- Fixed saving plurals for some locales in Android resources.
- Fixed crash in the cleanup addon for some XLIFF files.
- Allow to configure localization CDN in Docker image.

4.6 Weblate 4.2

Released on August 18th 2020.

- Improved user pages and added listing of users.
- Dropped support for migrating from 3.x releases, migrate through 4.1 or 4.0.
- Added exports into several monolingual formats.
- Improved activity charts.
- Number of displayed nearby strings can be configured.
- Suporte adicionado para bloquear componentes com erros no repositório.
- Simplified main navigation (replaced buttons with icons).
- Improved language code handling in Google Translate integration.
- The Git squash addon can generate `Co-authored-by:` trailers.
- Improved query search parser.
- Improved user feedback from format strings checks.
- Improved performance of bulk state changes.
- Added compatibility redirects after project or component renaming.
- Added notifications for strings approval, component locking and license change.
- Added support for ModernMT.
- Allow to avoid overwriting approved translations on file upload.
- Dropped support for some compatibility URL redirects.
- Verificação para literais de modelo de ECMAScript adicionada.
- A opção para observar um componente foi adicionada.

- Removed leading dot from JSON unit keys.
- Fila separada de Celery para memória de tradução foi removida.
- Permitir a tradução de todos os componentes de uma só vez.
- Allow to configure `Content-Security-Policy` HTTP headers.
- Added support for aliasing languages at project level.
- New addon to help with HTML or JavaScript localization, see *CDN de localização JavaScript*.
- The Weblate domain is now configured in the settings, see *SITE_DOMAIN*.
- Adicionar suporte para a pesquisa por componente e projeto.

4.7 Weblate 4.1.1

Released on June 19th 2020.

- Fixed changing autofix or addons configuration in Docker.
- Fixed possible crash in «About» page.
- Improved installation of byte-compiled locale files.
- Fixed adding words to glossary.
- Fixed keyboard shortcuts for machinery.
- Removed debugging output causing discarding log events in some setups.
- Fixed lock indication on project listing.
- Fixed listing GPG keys in some setups.
- Added option for which DeepL API version to use.
- Added support for acting as SAML Service Provider, see *Autenticação por SAML*.

4.8 Weblate 4.1

Released on June 15th 2020.

- Added support for creating new translations with included country code.
- Added support for searching source strings with screenshot.
- Extended info available in the stats insights.
- Improved search editing on «Translate» pages.
- Improve handling of concurrent repository updates.
- Include source language in project creation form.
- Include changes count in credits.
- Fixed UI language selection in some cases.
- Allow to whitelist registration methods with registrations closed.
- Improved lookup of related terms in glossary.
- Improved translation memory matches.
- Group same machinery results.
- Add direct link to edit screenshot from translate page.

- Improved removal confirmation dialog.
- Include templates in ZIP download.
- Add support for Markdown and notification configuration in announcements.
- Extended details in check listings.
- Added support for new file formats: *Cadeias de PHP Laravel*, *HTML files*, *OpenDocument Format*, *IDML Format*, *Windows RC files*, *INI translations*, *Traduções Inno Setup INI*, *GWT properties*, *go-i18n JSON files*, *ARB File*.
- Consistently use dismissed as state of dismissed checks.
- Add support for configuring default addons to enable.
- Fixed editor keyboard shortcut to dismiss checks.
- Improved machine translation of strings with placeholders.
- Show ghost translation for user languages to ease starting them.
- Improved language code parsing.
- Show translations in user language first in the list.
- Renamed shapings to more generic name variants.
- Added new quality checks: *Várias variáveis sem nome*, *Não traduzido há muito tempo*, *Palavras consecutivas duplicadas*.
- Reintroduced support for wiping translation memory.
- Fixed option to ignore source checks.
- Added support for configuring different branch for pushing changes.
- API now reports rate limiting status in the HTTP headers.
- Added support for Google Translate V3 API (Advanced).
- Added ability to restrict access on component level.
- Added support for whitespace and other special chars in translation flags, see *Personalizar o comportamento*.
- Always show rendered text check if enabled.
- API now supports filtering of changes.
- Added support for sharing glossaries between projects.

4.9 Weblate 4.0.4

Released on May 07th 2020.

- Fixed testsuite execution on some Python 3.8 environments.
- Typo fixes in the documentation.
- Fixed creating components using API in some cases.
- Fixed JavaScript errors breaking mobile navigation.
- Fixed crash on displaying some checks.
- Fixed screenshots listing.
- Fixed monthly digest notifications.
- Fixed intermediate translation behavior with units non existing in translation.

4.10 Weblate 4.0.3

Released on May 02nd 2020.

- Fixed possible crash in reports.
- User mentions in comments are now case insensitive.
- Fixed PostgreSQL migration for non superusers.
- Fixed changing the repository URL while creating component.
- Fixed crash when upstream repository is gone.

4.11 Weblate 4.0.2

Released on April 27th 2020.

- Improved performance of translation stats.
- Improved performance of changing labels.
- Improved bulk edit performance.
- Improved translation memory performance.
- Fixed possible crash on component deletion.
- Fixed displaying of translation changes in some corner cases.
- Improved warning about too long celery queue.
- Fixed possible false positives in the consistency check.
- Fixed deadlock when changing linked component repository.
- Included edit distance in changes listing and CSV and reports.
- Avoid false positives of punctuation spacing check for Canadian French.
- Fixed XLIFF export with placeholders.
- Fixed false positive with zero width check.
- Improved reporting of configuration errors.
- Fixed bilingual source upload.
- Automatically detect supported languages for DeepL machine translation.
- Fixed progress bar display in some corner cases.
- Fixed some checks triggering on non translated strings.

4.12 Weblate 4.0.1

Released on April 16th 2020.

- Fixed package installation from PyPI.

4.13 Weblate 4.0

Released on April 16th 2020.

- Weblate now requires Python 3.6 or newer.
- Added management overview of component alerts.
- Added component alert for broken repository browser URLs.
- Improved sign in and registration pages.
- Project access control and workflow configuration integrated to project settings.
- Added check and highlighter for i18next interpolation and nesting.
- Added check and highlighter for percent placeholders.
- Mostrar falhas nas verificações de sugestões.
- Record source string changes in history.
- Upgraded Microsoft Translator to version 3 API.
- Reimplemented translation memory backend.
- Added support for several `is:` lookups in *Searching*.
- Allow to make *Tradução inalterada* avoid internal blacklist.
- Improved comments extraction from monolingual po files.
- Renamed whiteboard messages to announcements.
- Fixed occasional problems with registration mails.
- Improved LINGUAS update addon to handle more syntax variants.
- Fixed editing monolingual XLIFF source file.
- Added support for exact matching in *Searching*.
- Extended API to cover screenshots, users, groups, componentlists and extended creating projects.
- Add support for source upload on bilingual translations.
- Added support for intermediate language from developers.
- Added support for source strings review.
- Extended download options for platform wide translation memory.

4.14 Weblate 3.x series

4.14.1 Weblate 3.11.3

Released on March 11th 2020.

- Fixed searching for fields with certain priority.
- Fixed predefined query for recently added strings.
- Fixed searching returning duplicate matches.
- Fixed notifications rendering in Gmail.
- Fixed reverting changes from the history.
- Added links to events in digest notifications.
- Fixed email for account removal confirmation.

- Added support for Slack authentication in Docker container.
- Avoid sending notifications for not subscribed languages.
- Include Celery queues in performance overview.
- Fixed documentation links for addons.
- Reduced false negatives for unchanged translation check.
- Raised bleach dependency to address CVE-2020-6802.
- Fixed listing project level changes in history.
- Fixed stats invalidation in some corner cases.
- Fixed searching for certain string states.
- Improved format string checks behavior on missing percent.
- Fixed authentication using some third party providers.

4.14.2 Weblate 3.11.2

Released on February 22nd 2020.

- Fixed rendering of suggestions.
- Fixed some strings wrongly reported as having no words.

4.14.3 Weblate 3.11.1

Released on February 20th 2020.

- Documented Celery setup changes.
- Improved filename validation on component creation.
- Fixed minimal versions of some dependencies.
- Fixed adding groups with certain Django versions.
- Fixed manual pushing to upstream repository.
- Improved glossary matching.

4.14.4 Weblate 3.11

Released on February 17th 2020.

- Allow using VCS push URL during component creation via API.
- Rendered width check now shows image with the render.
- Fixed links in notifications e-mails.
- Improved look of plaintext e-mails.
- Display ignored checks and allow to make them active again.
- Display nearby keys on monolingual translations.
- Suporte adicionado para agrupar formas de cadeias.
- Recommend upgrade to new Weblate versions in the system checks.
- Provide more detailed analysis for duplicate language alert.
- Include more detailed license info on the project pages.

- Automatically unshallow local copies if needed.
- Fixed download of strings needing action.
- New alert to warn about using the same filemask twice.
- Improve XML placeables extraction.
- The *SINGLE_PROJECT* can now enforce redirection to chosen project.
- Added option to resolve comments.
- Added bulk editing of flags.
- Added support for *String labels*.
- Added bulk edit addon.
- Added option for *Forçar verificações*.
- Increased default validity of confirmation links.
- Improved Matomo integration.
- Fixed *Foi traduzido* to correctly handle source string change.
- Extended automatic updates configuration by *AUTO_UPDATE*.
- LINGUAS addons now do full sync of translations in Weblate.

4.14.5 Weblate 3.10.3

Released on January 18th 2020.

- Support for translate-toolkit 2.5.0.

4.14.6 Weblate 3.10.2

Released on January 18th 2020.

- Add lock indication to projects.
- Fixed CSS bug causing flickering in some web browsers.
- Fixed searching on systems with non-English locales.
- Improved repository matching for GitHub and Bitbucket hooks.
- Fixed data migration on some Python 2.7 installations.
- Allow configuration of Git shallow cloning.
- Improved background notification processing.
- Fixed broken form submission when navigating back in web browser.
- New addon to configure YAML formatting.
- Fixed same plurals check to not fire on single plural form languages.
- Fixed regex search on some fields.

4.14.7 Weblate 3.10.1

Released on January 9th 2020.

- Extended API with translation creation.
- Fixed several corner cases in data migrations.
- Compatibility with Django 3.0.
- Improved data cleanup performance.
- Added support for customizable security.txt.
- Improved breadcrumbs in changelog.
- Improved translations listing on dashboard.
- Improved HTTP responses for webhooks.
- Added support for GitLab merge requests in Docker container.

4.14.8 Weblate 3.10

Released on December 20th 2019.

- Improved application user interface.
- Added doublespace check.
- Fixed creating new languages.
- Avoid sending auditlog notifications to deleted e-mails.
- Added support for read only strings.
- Added support for Markdown in comments.
- Allow placing translation instruction text in project info.
- Add copy to clipboard for secondary languages.
- Improved support for Mercurial.
- Improved Git repository fetching performance.
- Add search lookup for age of string.
- Show source language for all translations.
- Show context for nearby strings.
- Added support for notifications on repository operations.
- Improved translation listings.
- Extended search capabilities.
- Added support for automatic translation strings marked for editing.
- Avoid sending duplicate notifications for linked component alerts.
- Improve default merge request message.
- Better indicate string state in Zen mode.
- Added support for more languages in Yandex Translate.
- Improved look of notification e-mails.
- Provide choice for translation license.

4.14.9 Weblate 3.9.1

Released on October 28th 2019.

- Remove some unneeded files from backups.
- Fixed potential crash in reports.
- Fixed cross database migration failure.
- Added support for force pushing Git repositories.
- Reduced risk of registration token invalidation.
- Fixed account removal hitting rate limiter.
- Added search based on priority.
- Fixed possible crash on adding strings to JSON file.
- Safe HTML check and fixup now honor source string markup.
- Avoid sending notifications to invited and deleted users.
- Fix SSL connection to redis in Celery in Docker container.

4.14.10 Weblate 3.9

Released on October 15th 2019.

- Include Weblate metadata in downloaded files.
- Improved UI for failing checks.
- Indicate missing strings in format checks.
- Separate check for French punctuation spacing.
- Add support for fixing some of quality checks errors.
- Add separate permission to create new projects.
- Extend stats for char counts.
- Improve support for Java style language codes.
- Added new generic check for placeholders.
- Added support for WebExtension JSON placeholders.
- Added support for flat XML format.
- Extended API with project, component and translation removal and creation.
- Added support for Gitea and Gitee webhooks.
- Added new custom regex based check.
- Allow to configure contributing to shared translation memory.
- Added ZIP download for more translation files.
- Make XLIFF standard compliant parsing of maxwidth and font.
- Added new check and fixer for safe HTML markup for translating web applications.
- Add component alert on unsupported configuration.
- Added automatic translation addon to bootstrap translations.
- Extend automatic translation to add suggestions.
- Display addon parameters on overview.

- Sentry is now supported through modern Sentry SDK instead of Raven.
- Changed example settings to be better fit for production environment.
- Added automated backups using BorgBackup.
- Split cleanup addon for RESX to avoid unwanted file updates.
- Added advanced search capabilities.
- Allow users to download their own reports.
- Added localization guide to help configuring components.
- Added support for GitLab merge requests.
- Improved display of repository status.
- Perform automated translation in the background.

4.14.11 Weblate 3.8

Released on August 15th 2019.

- Added support for simplified creating of similar components.
- Added support for parsing translation flags from the XML based file formats.
- Log exceptions into Celery log.
- Improve performance of repository scoped addons.
- Improved look of notification e-mails.
- Fixed password reset behavior.
- Improved performance on most of translation pages.
- Fixed listing of languages not known to Weblate.
- Add support for cloning addons to discovered components.
- Add support for replacing file content with uploaded.
- Add support for translating non VCS based content.
- Added OpenGraph widget image to use on social networks.
- Added support for animated screenshots.
- Improved handling of monolingual XLIFF files.
- Avoid sending multiple notifications for single event.
- Add support for filtering changes.
- Extended predefined periods for reporting.
- Added webhook support for Azure Repos.
- New opt-in notifications on pending suggestions or untranslated strings.
- Add one click unsubscribe link to notification e-mails.
- Fixed false positives with Has been translated check.
- New management interface for admins.
- String priority can now be specified using flags.
- Added language management views.
- Add checks for Qt library and Ruby format strings.
- Added configuration to better fit single project installations.

- Notify about new string on source string change on monolingual translations.
- Added separate view for translation memory with search capability.

4.14.12 Weblate 3.7.1

Released on June 28th 2019.

- Documentation updates.
- Fixed some requirements constraints.
- Updated language database.
- Localization updates.
- Various user interface tweaks.
- Improved handling of unsupported but discovered translation files.
- More verbosely report missing file format requirements.

4.14.13 Weblate 3.7

Released on June 21st 2019.

- Added separate Celery queue for notifications.
- Use consistent look with application for API browsing.
- Include approved stats in the reports.
- Report progress when updating translation component.
- Allow to abort running background component update.
- Extend template language for filename manipulations.
- Use templates for editor link and repository browser URL.
- Indicate max length and current characters count when editing translation.
- Improved handling of abbreviations in unchanged translation check.
- Refreshed landing page for new contributors.
- Add support for configuring msgmerge addon.
- Delay opening SMTP connection when sending notifications.
- Improved error logging.
- Allow custom location in MO generating addon.
- Added addons to cleanup old suggestions or comments.
- Added option to enable horizontal mode in the Zen editor.
- Improved import performance with many linked components.
- Fixed examples installation in some cases.
- Improved rendering of alerts in changes.
- Added new horizontal stats widget.
- Improved format strings check on plurals.
- Added font management tool.
- New check for rendered text dimensions.

- Added support for subtitle formats.
- Include overall completion stats for languages.
- Added reporting at project and global scope.
- Improved user interface when showing translation status.
- New Weblate logo and color scheme.
- New look of bitmap badges.

4.14.14 Weblate 3.6.1

Released on April 26th 2019.

- Improved handling of monolingual XLIFF files.
- Fixed digest notifications in some corner cases.
- Fixed addon script error alert.
- Fixed generating MO file for monolingual PO files.
- Fixed display of uninstalled checks.
- Indicate administered projects on project listing.
- Allow update to recover from missing VCS repository.

4.14.15 Weblate 3.6

Released on April 20th 2019.

- Add support for downloading user data.
- Addons are now automatically triggered upon installation.
- Improved instructions for resolving merge conflicts.
- Cleanup addon is now compatible with app store metadata translations.
- Configurable language code syntax when adding new translations.
- Warn about using Python 2 with planned termination of support in April 2020.
- Extract special characters from the source string for visual keyboard.
- Extended contributor stats to reflect both source and target counts.
- Admins and consistency addons can now add translations even if disabled for users.
- Fixed description of toggle disabling `Language-Team` header manipulation.
- Notify users mentioned in comments.
- Removed file format autodetection from component setup.
- Fixed generating MO file for monolingual PO files.
- Added digest notifications.
- Added support for muting component notifications.
- Added notifications for new alerts, whiteboard messages or components.
- Notifications for administered projects can now be configured.
- Improved handling of three letter language codes.

4.14.16 Weblate 3.5.1

Released on March 10th 2019.

- Fixed Celery systemd unit example.
- Fixed notifications from HTTP repositories with login.
- Fixed race condition in editing source string for monolingual translations.
- Include output of failed addon execution in the logs.
- Improved validation of choices for adding new language.
- Allow to edit file format in component settings.
- Update installation instructions to prefer Python 3.
- Performance and consistency improvements for loading translations.
- Make Microsoft Terminology service compatible with current Zeep releases.
- Localization updates.

4.14.17 Weblate 3.5

Released on March 3rd 2019.

- Improved performance of built-in translation memory.
- Added interface to manage global translation memory.
- Improved alerting on bad component state.
- Added user interface to manage whiteboard messages.
- Addon commit message now can be configured.
- Reduce number of commits when updating upstream repository.
- Fixed possible metadata loss when moving component between projects.
- Improved navigation in the Zen mode.
- Added several new quality checks (Markdown related and URL).
- Added support for app store metadata files.
- Added support for toggling GitHub or Gerrit integration.
- Added check for Kashida letters.
- Added option to squash commits based on authors.
- Improved support for XLSX file format.
- Compatibility with Tesseract 4.0.
- Billing addon now removes projects for unpaid billings after 45 days.

4.14.18 Weblate 3.4

Released on January 22nd 2019.

- Added support for XLIFF placeholders.
- Celery can now utilize multiple task queues.
- Added support for renaming and moving projects and components.
- Include characters counts in reports.
- Added guided adding of translation components with automatic detection of translation files.
- Customizable merge commit messages for Git.
- Added visual indication of component alerts in navigation.
- Improved performance of loading translation files.
- New addon to squash commits prior to push.
- Improved displaying of translation changes.
- Changed default merge style to rebase and made that configurable.
- Better handle private use subtags in language code.
- Improved performance of fulltext index updates.
- Extended file upload API to support more parameters.

4.14.19 Weblate 3.3

Released on November 30th 2018.

- Added support for component and project removal.
- Improved performance for some monolingual translations.
- Added translation component alerts to highlight problems with a translation.
- Expose XLIFF string resname as context when available.
- Added support for XLIFF states.
- Added check for non writable files in DATA_DIR.
- Improved CSV export for changes.

4.14.20 Weblate 3.2.2

Released on October 20th 2018.

- Remove no longer needed Babel dependency.
- Updated language definitions.
- Improve documentation for addons, LDAP and Celery.
- Fixed enabling new dos-eol and auto-java-messageformat flags.
- Fixed running setup.py test from PyPI package.
- Improved plurals handling.
- Fixed translation upload API failure in some corner cases.
- Fixed updating Git configuration in case it was changed manually.

4.14.21 Weblate 3.2.1

Released on October 10th 2018.

- Document dependency on backports.csv on Python 2.7.
- Fix running tests under root.
- Improved error handling in gitexport module.
- Fixed progress reporting for newly added languages.
- Correctly report Celery worker errors to Sentry.
- Fixed creating new translations with Qt Linguist.
- Fixed occasional fulltext index update failures.
- Improved validation when creating new components.
- Added support for cleanup of old suggestions.

4.14.22 Weblate 3.2

Released on October 6th 2018.

- Add install_addon management command for automated addon installation.
- Allow more fine grained ratelimit settings.
- Added support for export and import of Excel files.
- Improve component cleanup in case of multiple component discovery addons.
- Rewritten Microsoft Terminology machine translation backend.
- Weblate now uses Celery to offload some processing.
- Improved search capabilities and added regular expression search.
- Added support for Youdao Zhiyun API machine translation.
- Added support for Baidu API machine translation.
- Integrated maintenance and cleanup tasks using Celery.
- Improved performance of loading translations by almost 25%.
- Removed support for merging headers on upload.
- Removed support for custom commit messages.
- Configurable editing mode (zen/full).
- Added support for error reporting to Sentry.
- Added support for automated daily update of repositories.
- Added support for creating projects and components by users.
- Built in translation memory now automatically stores translations done.
- Users and projects can import their existing translation memories.
- Better management of related strings for screenshots.
- Added support for checking Java MessageFormat.

See 3.2 milestone on GitHub for detailed list of addressed issues.

4.14.23 Weblate 3.1.1

Released on July 27th 2018.

- Fix testsuite failure on some setups.

4.14.24 Weblate 3.1

Released on July 27th 2018.

- Upgrades from older version than 3.0.1 are not supported.
- Allow to override default commit messages from settings.
- Improve webhooks compatibility with self hosted environments.
- Added support for Amazon Translate.
- Compatibility with Django 2.1.
- Django system checks are now used to diagnose problems with installation.
- Removed support for soon shutdown libravatar service.
- New addon to mark unchanged translations as needing edit.
- Add support for jumping to specific location while translating.
- Downloaded translations can now be customized.
- Improved calculation of string similarity in translation memory matches.
- Added support by signing Git commits by GnuPG.

4.14.25 Weblate 3.0.1

Released on June 10th 2018.

- Fixed possible migration issue from 2.20.
- Localization updates.
- Removed obsolete hook examples.
- Improved caching documentation.
- Fixed displaying of admin documentation.
- Improved handling of long language names.

4.14.26 Weblate 3.0

Released on June 1st 2018.

- Rewritten access control.
- Several code cleanups that lead to moved and renamed modules.
- New addon for automatic component discovery.
- The import_project management command has now slightly different parameters.
- Added basic support for Windows RC files.
- New addon to store contributor names in PO file headers.
- The per component hook scripts are removed, use addons instead.
- Add support for collecting contributor agreements.

- Access control changes are now tracked in history.
- New addon to ensure all components in a project have same translations.
- Support for more variables in commit message templates.
- Add support for providing additional textual context.

4.15 Weblate 2.x series

4.15.1 Weblate 2.20

Released on April 4th 2018.

- Improved speed of cloning subversion repositories.
- Changed repository locking to use third party library.
- Added support for downloading only strings needing action.
- Added support for searching in several languages at once.
- New addon to configure gettext output wrapping.
- New addon to configure JSON formatting.
- Added support for authentication in API using RFC 6750 compatible Bearer authentication.
- Added support for automatic translation using machine translation services.
- Added support for HTML markup in whiteboard messages.
- Added support for mass changing state of strings.
- Translate-toolkit at least 2.3.0 is now required, older versions are no longer supported.
- Added built in translation memory.
- Added componentlists overview to dashboard and per component list overview pages.
- Added support for DeepL machine translation service.
- Machine translation results are now cached inside Weblate.
- Adicionado suporte para reordenar alterações de commits feitos.

4.15.2 Weblate 2.19.1

Released on February 20th 2018.

- Fixed migration issue on upgrade from 2.18.
- Improved file upload API validation.

4.15.3 Weblate 2.19

Released on February 15th 2018.

- Fixed imports across some file formats.
- Display human friendly browser information in audit log.
- Added TMX exporter for files.
- Various performance improvements for loading translation files.
- Added option to disable access management in Weblate in favor of Django one.

- Improved glossary lookup speed for large strings.
- Compatibility with django_auth_ldap 1.3.0.
- Configuration errors are now stored and reported persistently.
- Honor ignore flags in whitespace autofixer.
- Improved compatibility with some Subversion setups.
- Improved built in machine translation service.
- Added support for SAP Translation Hub service.
- Added support for Microsoft Terminology service.
- Removed support for advertisement in notification e-mails.
- Improved translation progress reporting at language level.
- Improved support for different plural formulas.
- Added support for Subversion repositories not using stdlayout.
- Added addons to customize translation workflows.

4.15.4 Weblate 2.18

Released on December 15th 2017.

- Extended contributor stats.
- Improved configuration of special characters virtual keyboard.
- Added support for DTD file format.
- Changed keyboard shortcuts to less likely collide with browser/system ones.
- Improved support for approved flag in XLIFF files.
- Added support for not wrapping long strings in gettext PO files.
- Added button to copy permalink for current translation.
- Dropped support for Django 1.10 and added support for Django 2.0.
- Removed locking of translations while translating.
- Added support for adding new strings to monolingual translations.
- Added support for translation workflows with dedicated reviewers.

4.15.5 Weblate 2.17.1

Released on October 13th 2017.

- Fixed running testsuite in some specific situations.
- Locales updates.

4.15.6 Weblate 2.17

Released on October 13th 2017.

- Weblate by default does shallow Git clones now.
- Improved performance when updating large translation files.
- Added support for blocking certain e-mails from registration.
- Users can now delete their own comments.
- Added preview step to search and replace feature.
- Client side persistence of settings in search and upload forms.
- Extended search capabilities.
- More fine grained per project ACL configuration.
- Default value of BASE_DIR has been changed.
- Added two step account removal to prevent accidental removal.
- Project access control settings is now editable.
- Added optional spam protection for suggestions using Akismet.

4.15.7 Weblate 2.16

Released on August 11th 2017.

- Various performance improvements.
- Added support for nested JSON format.
- Added support for WebExtension JSON format.
- Fixed git exporter authentication.
- Improved CSV import in certain situations.
- Improved look of Other translations widget.
- The max-length checks is now enforcing length of text in form.
- Make the commit_pending age configurable per component.
- Various user interface cleanups.
- Fixed component/project/site wide search for translations.

4.15.8 Weblate 2.15

Released on June 30th 2017.

- Show more related translations in other translations.
- Add option to see translations of current string to other languages.
- Use 4 plural forms for Lithuanian by default.
- Fixed upload for monolingual files of different format.
- Improved error messages on failed authentication.
- Keep page state when removing word from glossary.
- Added direct link to edit secondary language translation.
- Added Perl format quality check.

- Added support for rejecting reused passwords.
- Extended toolbar for editing RTL languages.

4.15.9 Weblate 2.14.1

Released on May 24th 2017.

- Fixed possible error when paginating search results.
- Fixed migrations from older versions in some corner cases.
- Fixed possible CSRF on project watch and unwatch.
- The password reset no longer authenticates user.
- Fixed possible CAPTCHA bypass on forgotten password.

4.15.10 Weblate 2.14

Released on May 17th 2017.

- Add glossary entries using AJAX.
- The logout now uses POST to avoid CSRF.
- The API key token reset now uses POST to avoid CSRF.
- Weblate sets Content-Security-Policy by default.
- The local editor URL is validated to avoid self-XSS.
- The password is now validated against common flaws by default.
- Notify users about important activity with their account such as password change.
- The CSV exports now escape potential formulas.
- Various minor improvements in security.
- The authentication attempts are now rate limited.
- Suggestion content is stored in the history.
- Store important account activity in audit log.
- Ask for password confirmation when removing account or adding new associations.
- Show time when suggestion has been made.
- There is new quality check for trailing semicolon.
- Ensure that search links can be shared.
- Included source string information and screenshots in the API.
- Allow to overwrite translations through API upload.

4.15.11 Weblate 2.13.1

Released on Apr 12th 2017.

- Fixed listing of managed projects in profile.
- Fixed migration issue where some permissions were missing.
- Fixed listing of current file format in translation download.
- Return HTTP 404 when trying to access project where user lacks privileges.

4.15.12 Weblate 2.13

Released on Apr 12th 2017.

- Fixed quality checks on translation templates.
- Added quality check to trigger on losing translation.
- Add option to view pending suggestions from user.
- Add option to automatically build component lists.
- Default dashboard for unauthenticated users can be configured.
- Add option to browse 25 random strings for review.
- History now indicates string change.
- Better error reporting when adding new translation.
- Added per language search within project.
- Group ACLs can now be limited to certain permissions.
- The per project ALCs are now implemented using Group ACL.
- Added more fine grained privileges control.
- Various minor UI improvements.

4.15.13 Weblate 2.12

Released on Mar 3rd 2017.

- Improved admin interface for groups.
- Added support for Yandex Translate API.
- Improved speed of site wide search.
- Added project and component wide search.
- Added project and component wide search and replace.
- Improved rendering of inconsistent translations.
- Added support for opening source files in local editor.
- Added support for configuring visual keyboard with special characters.
- Improved screenshot management with OCR support for matching source strings.
- Default commit message now includes translation information and URL.
- Added support for Joomla translation format.
- Improved reliability of import across file formats.

4.15.14 Weblate 2.11

Released on Jan 31st 2017.

- Include language detailed information on language page.
- Mercurial backend improvements.
- Added option to specify translation component priority.
- More consistent usage of Group ACL even with less used permissions.
- Added WL_BRANCH variable to hook scripts.
- Improved developer documentation.
- Better compatibility with various Git versions in Git exporter addon.
- Included per project and component stats.
- Added language code mapping for better support of Microsoft Translate API.
- Moved fulltext cleanup to background job to make translation removal faster.
- Fixed displaying of plural source for languages with single plural form.
- Improved error handling in import_project.
- Various performance improvements.

4.15.15 Weblate 2.10.1

Released on Jan 20th 2017.

- Do not leak account existence on password reset form (CVE-2017-5537).

4.15.16 Weblate 2.10

Released on Dec 15th 2016.

- Added quality check to check whether plurals are translated differently.
- Fixed GitHub hooks for repositories with authentication.
- Added optional Git exporter module.
- Support for Microsoft Cognitive Services Translator API.
- Simplified project and component user interface.
- Added automatic fix to remove control characters.
- Added per language overview to project.
- Added support for CSV export.
- Added CSV download for stats.
- Added matrix view for quick overview of all translations
- Added basic API for changes and strings.
- Added support for Apertium APy server for machine translations.

4.15.17 Weblate 2.9

Released on Nov 4th 2016.

- Extended parameters for createadmin management command.
- Extended import_json to be able to handle with existing components.
- Added support for YAML files.
- Project owners can now configure translation component and project details.
- Use «Watched» instead of «Subscribed» projects.
- Projects can be watched directly from project page.
- Added multi language status widget.
- Highlight secondary language if not showing source.
- Record suggestion deletion in history.
- Improved UX of languages selection in profile.
- Fixed showing whiteboard messages for component.
- Keep preferences tab selected after saving.
- Show source string comment more prominently.
- Automatically install Gettext PO merge driver for Git repositories.
- Added search and replace feature.
- Added support for uploading visual context (screenshots) for translations.

4.15.18 Weblate 2.8

Released on Aug 31st 2016.

- Documentation improvements.
- Translations.
- Updated bundled javascript libraries.
- Added list_translators management command.
- Django 1.8 is no longer supported.
- Fixed compatibility with Django 1.10.
- Added Subversion support.
- Separated XML validity check from XML mismatched tags.
- Fixed API to honor HIDE_REPO_CREDENTIALS settings.
- Show source change in Zen mode.
- Alt+PageUp/PageDown/Home/End now works in Zen mode as well.
- Add tooltip showing exact time of changes.
- Add option to select filters and search from translation page.
- Added UI for translation removal.
- Improved behavior when inserting placeables.
- Fixed auto locking issues in Zen mode.

4.15.19 Weblate 2.7

Released on Jul 10th 2016.

- Removed Google web translate machine translation.
- Improved commit message when adding translation.
- Fixed Google Translate API for Hebrew language.
- Compatibility with Mercurial 3.8.
- Added import_json management command.
- Correct ordering of listed translations.
- Show full suggestion text, not only a diff.
- Extend API (detailed repository status, statistics, …).
- Testsuite no longer requires network access to test repositories.

4.15.20 Weblate 2.6

Released on Apr 28th 2016.

- Fixed validation of components with language filter.
- Improved support for XLIFF files.
- Fixed machine translation for non English sources.
- Added REST API.
- Django 1.10 compatibility.
- Added categories to whiteboard messages.

4.15.21 Weblate 2.5

Released on Mar 10th 2016.

- Fixed automatic translation for project owners.
- Improved performance of commit and push operations.
- New management command to add suggestions from command line.
- Added support for merging comments on file upload.
- Added support for some GNU extensions to C printf format.
- Documentation improvements.
- Added support for generating translator credits.
- Added support for generating contributor stats.
- Site wide search can search only in one language.
- Improve quality checks for Armenian.
- Support for starting translation components without existing translations.
- Support for adding new translations in Qt TS.
- Improved support for translating PHP files.
- Performance improvements for quality checks.
- Pesquisa corrigida para todo o site por verificações com falha.

- Added option to specify source language.
- Improved support for XLIFF files.
- Extended list of options for import_project.
- Improved targeting for whiteboard messages.
- Support for automatic translation across projects.
- Optimized fulltext search index.
- Added management command for auto translation.
- Added placeables highlighting.
- Added keyboard shortcuts for placeables, checks and machine translations.
- Improved translation locking.
- Added quality check for AngularJS interpolation.
- Added extensive group based ACLs.
- Clarified terminology on strings needing review (formerly fuzzy).
- Clarified terminology on strings needing action and not translated strings.
- Support for Python 3.
- Dropped support for Django 1.7.
- Dropped dependency on msginit for creating new gettext PO files.
- Added configurable dashboard views.
- Improved notifications on parse errors.
- Added option to import components with duplicate name to import_project.
- Improved support for translating PHP files
- Added XLIFF export for dictionary.
- Added XLIFF and gettext PO export for all translations.
- Documentation improvements.
- Added support for configurable automatic group assignments.
- Improved adding of new translations.

4.15.22 Weblate 2.4

Released on Sep 20th 2015.

- Improved support for PHP files.
- Ability to add ACL to anonymous user.
- Improved configurability of import_project command.
- Added CSV dump of history.
- Avoid copy/paste errors with whitespace characters.
- Added support for Bitbucket webhooks.
- Tighter control on fuzzy strings on translation upload.
- Several URLs have changed, you might have to update your bookmarks.
- Hook scripts are executed with VCS root as current directory.
- Hook scripts are executed with environment variables describing current component.

- Add management command to optimize fulltext index.
- Added support for error reporting to Rollbar.
- Projects now can have multiple owners.
- Project owners can manage themselves.
- Added support for `javascript-format` used in gettext PO.
- Support for adding new translations in XLIFF.
- Improved file format autodetection.
- Extended keyboard shortcuts.
- Improved dictionary matching for several languages.
- Improved layout of most of pages.
- Support for adding words to dictionary while translating.
- Added support for filtering languages to be managed by Weblate.
- Added support for translating and importing CSV files.
- Rewritten handling of static files.
- Direct login/registration links to third-party service if that's the only one.
- Commit pending changes on account removal.
- Add management command to change site name.
- Add option to configure default committer.
- Add hook after adding new translation.
- Add option to specify multiple files to add to commit.

4.15.23 Weblate 2.3

Released on May 22nd 2015.

- Dropped support for Django 1.6 and South migrations.
- Support for adding new translations when using Java Property files
- Allow to accept suggestion without editing.
- Improved support for Google OAuth 2.0
- Added support for Microsoft .resx files.
- Tuned default robots.txt to disallow big crawling of translations.
- Simplified workflow for accepting suggestions.
- Added project owners who always receive important notifications.
- Allow to disable editing of monolingual template.
- More detailed repository status view.
- Direct link for editing template when changing translation.
- Allow to add more permissions to project owners.
- Allow to show secondary language in Zen mode.
- Support for hiding source string in favor of secondary language.

4.15.24 Weblate 2.2

Released on Feb 19th 2015.

- Performance improvements.
- Fulltext search on location and comments fields.
- New SVG/javascript based activity charts.
- Support for Django 1.8.
- Support for deleting comments.
- Added own SVG badge.
- Added support for Google Analytics.
- Improved handling of translation filenames.
- Added support for monolingual JSON translations.
- Record component locking in a history.
- Support for editing source (template) language for monolingual translations.
- Added basic support for Gerrit.

4.15.25 Weblate 2.1

Released on Dec 5th 2014.

- Added support for Mercurial repositories.
- Replaced Glyphicon font by Awesome.
- Added icons for social authentication services.
- Better consistency of button colors and icons.
- Documentation improvements.
- Various bugfixes.
- Automatic hiding of columns in translation listing for small screens.
- Changed configuration of filesystem paths.
- Improved SSH keys handling and storage.
- Improved repository locking.
- Customizable quality checks per source string.
- Allow to hide completed translations from dashboard.

4.15.26 Weblate 2.0

Released on Nov 6th 2014.

- New responsive UI using Bootstrap.
- Rewritten VCS backend.
- Documentation improvements.
- Added whiteboard for site wide messages.
- Configurable strings priority.
- Added support for JSON file format.

- Fixed generating mo files in certain cases.
- Added support for GitLab notifications.
- Added support for disabling translation suggestions.
- Django 1.7 support.
- ACL projects now have user management.
- Extended search possibilities.
- Give more hints to translators about plurals.
- Fixed Git repository locking.
- Compatibility with older Git versions.
- Improved ACL support.
- Added buttons for per language quotes and other special characters.
- Support for exporting stats as JSONP.

4.16 Weblate 1.x series

4.16.1 Weblate 1.9

Released on May 6th 2014.

- Django 1.6 compatibility.
- No longer maintained compatibility with Django 1.4.
- Management commands for locking/unlocking translations.
- Improved support for Qt TS files.
- Users can now delete their account.
- Avatars can be disabled.
- Merged first and last name attributes.
- Avatars are now fetched and cached server side.
- Added support for shields.io badge.

4.16.2 Weblate 1.8

Released on November 7th 2013.

- Please check manual for upgrade instructions.
- Nicer listing of project summary.
- Better visible options for sharing.
- More control over anonymous users privileges.
- Supports login using third party services, check manual for more details.
- Users can login by e-mail instead of username.
- Documentation improvements.
- Improved source strings review.
- Searching across all strings.

- Better tracking of source strings.
- Captcha protection for registration.

4.16.3 Weblate 1.7

Released on October 7th 2013.

- Please check manual for upgrade instructions.
- Support for checking Python brace format string.
- Per component customization of quality checks.
- Detailed per translation stats.
- Changed way of linking suggestions, checks and comments to strings.
- Users can now add text to commit message.
- Support for subscribing on new language requests.
- Support for adding new translations.
- Widgets and charts are now rendered using Pillow instead of Pango + Cairo.
- Add status badge widget.
- Dropped invalid text direction check.
- Changes in dictionary are now logged in history.
- Performance improvements for translating view.

4.16.4 Weblate 1.6

Released on July 25th 2013.

- Nicer error handling on registration.
- Browsing of changes.
- Fixed sorting of machine translation suggestions.
- Improved support for MyMemory machine translation.
- Added support for Amagama machine translation.
- Various optimizations on frequently used pages.
- Highlights searched phrase in search results.
- Support for automatic fixups while saving the message.
- Tracking of translation history and option to revert it.
- Added support for Google Translate API.
- Added support for managing SSH host keys.
- Various form validation improvements.
- Various quality checks improvements.
- Performance improvements for import.
- Added support for voting on suggestions.
- Cleanup of admin interface.

4.16.5 Weblate 1.5

Released on April 16th 2013.

- Please check manual for upgrade instructions.
- Added public user pages.
- Better naming of plural forms.
- Added support for TBX export of glossary.
- Added support for Bitbucket notifications.
- Activity charts are now available for each translation, language or user.
- Extended options of import_project admin command.
- Compatible with Django 1.5.
- Avatars are now shown using libravatar.
- Added possibility to pretty print JSON export.
- Various performance improvements.
- Indicate failing checks or fuzzy strings in progress bars for projects or languages as well.
- Added support for custom pre-commit hooks and committing additional files.
- Rewritten search for better performance and user experience.
- New interface for machine translations.
- Added support for monolingual po files.
- Extend amount of cached metadata to improve speed of various searches.
- Now shows word counts as well.

4.16.6 Weblate 1.4

Released on January 23rd 2013.

- Fixed deleting of checks/comments on string deletion.
- Added option to disable automatic propagation of translations.
- Added option to subscribe for merge failures.
- Correctly import on projects which needs custom ttkit loader.
- Added sitemaps to allow easier access by crawlers.
- Provide direct links to string in notification e-mails or feeds.
- Various improvements to admin interface.
- Provide hints for production setup in admin interface.
- Added per language widgets and engage page.
- Improved translation locking handling.
- Show code snippets for widgets in more variants.
- Indicate failing checks or fuzzy strings in progress bars.
- More options for formatting commit message.
- Fixed error handling with machine translation services.
- Improved automatic translation locking behaviour.

- Support for showing changes from previous source string.
- Added support for substring search.
- Various quality checks improvements.
- Support for per project ACL.
- Basic string tests coverage.

4.16.7 Weblate 1.3

Released on November 16th 2012.

- Compatibility with PostgreSQL database backend.
- Removes languages removed in upstream git repository.
- Improved quality checks processing.
- Added new checks (BB code, XML markup and newlines).
- Support for optional rebasing instead of merge.
- Possibility to relocate Weblate (for example to run it under /weblate path).
- Support for manually choosing file type in case autodetection fails.
- Better support for Android resources.
- Support for generating SSH key from web interface.
- More visible data exports.
- New buttons to enter some special characters.
- Support for exporting dictionary.
- Support for locking down whole Weblate installation.
- Checks for source strings and support for source strings review.
- Support for user comments for both translations and source strings.
- Better changes log tracking.
- Changes can now be monitored using RSS.
- Improved support for RTL languages.

4.16.8 Weblate 1.2

Released on August 14th 2012.

- Weblate now uses South for database migration, please check upgrade instructions if you are upgrading.
- Fixed minor issues with linked git repos.
- New introduction page for engaging people with translating using Weblate.
- Added widgets which can be used for promoting translation projects.
- Added option to reset repository to origin (for privileged users).
- Project or component can now be locked for translations.
- Possibility to disable some translations.
- Configurable options for adding new translations.
- Configuration of git commits per project.

- Simple antispam protection.
- Better layout of main page.
- Support for automatically pushing changes on every commit.
- Support for e-mail notifications of translators.
- List only used languages in preferences.
- Improved handling of not known languages when importing project.
- Support for locking translation by translator.
- Optionally maintain `Language-Team` header in po file.
- Include some statistics in about page.
- Supports (and requires) django-registration 0.8.
- Caching of counted strings with failing checks.
- Checking of requirements during setup.
- Documentation improvements.

4.16.9 Weblate 1.1

Released on July 4th 2012.

- Improved several translations.
- Better validation while creating component.
- Added support for shared git repositories across components.
- Do not necessary commit on every attempt to pull remote repo.
- Added support for offloading indexing.

4.16.10 Weblate 1.0

Released on May 10th 2012.

- Improved validation while adding/saving component.
- Experimental support for Android component files (needs patched ttkit).
- Updates from hooks are run in background.
- Improved installation instructions.
- Improved navigation in dictionary.

4.17 Weblate 0.x series

4.17.1 Weblate 0.9

Released on April 18th 2012.

- Fixed import of unknown languages.
- Improved listing of nearby messages.
- Improved several checks.
- Documentation updates.

- Added definition for several more languages.
- Various code cleanups.
- Documentation improvements.
- Changed file layout.
- Update helper scripts to Django 1.4.
- Improved navigation while translating.
- Better handling of po file renames.
- Better validation while creating component.
- Integrated full setup into syncdb.
- Added list of recent changes to all translation pages.
- Check for not translated strings ignores format string only messages.

4.17.2 Weblate 0.8

Released on April 3rd 2012.

- Replaced own full text search with Whoosh.
- Various fixes and improvements to checks.
- New command updatechecks.
- Lot of translation updates.
- Added dictionary for storing most frequently used terms.
- Added /admin/report/ for overview of repositories status.
- Machine translation services no longer block page loading.
- Management interface now contains also useful actions to update data.
- Records log of changes made by users.
- Ability to postpone commit to Git to generate less commits from single user.
- Possibility to browse failing checks.
- Automatic translation using already translated strings.
- New about page showing used versions.
- Django 1.4 compatibility.
- Ability to push changes to remote repo from web interface.
- Added review of translations done by others.

4.17.3 Weblate 0.7

Released on February 16th 2012.

- Direct support for GitHub notifications.
- Added support for cleaning up orphaned checks and translations.
- Displays nearby strings while translating.
- Displays similar strings while translating.
- Improved searching for string.

4.17.4 Weblate 0.6

Released on February 14th 2012.

- Added various checks for translated messages.
- Tunable access control.
- Improved handling of translations with new lines.
- Added client side sorting of tables.
- Please check upgrading instructions in case you are upgrading.

4.17.5 Weblate 0.5

Released on February 12th 2012.

- **Support for machine translation using following online services:**
  - Apertium
  - Microsoft Translator
  - MyMemory
- Several new translations.
- Improved merging of upstream changes.
- Better handle concurrent git pull and translation.
- Propagating works for fuzzy changes as well.
- Propagating works also for file upload.
- Fixed file downloads while using FastCGI (and possibly others).

4.17.6 Weblate 0.4

Released on February 8th 2012.

- Added usage guide to documentation.
- Fixed API hooks not to require CSRF protection.

4.17.7 Weblate 0.3

Released on February 8th 2012.

- Better display of source for plural translations.
- New documentation in Sphinx format.
- Displays secondary languages while translating.
- Improved error page to give list of existing projects.
- New per language stats.

4.17.8 Weblate 0.2

Released on February 7th 2012.

- Improved validation of several forms.
- Warn users on profile upgrade.
- Lembre-se de URL para fazer o login.
- Naming of text areas while entering plural forms.
- Automatic expanding of translation area.

4.17.9 Weblate 0.1

Released on February 6th 2012.

- Initial release.

Índice de Módulos do Python

W

wlc, 146
wlc.config, 147
wlc.main, 147

HTTP Routing Table

/

ANY /, 98

/api

GET /api/, 100

/api/changes

GET /api/changes/, 130
GET /api/changes/(int:id)/, 131

/api/component-lists

GET /api/component-lists/, 133
GET /api/component-lists/(str:slug)/, 133
POST /api/component-lists/(str:slug)/components/, 134
PUT /api/component-lists/(str:slug)/, 134
DELETE /api/component-lists/(str:slug)/, 134
DELETE /api/component-lists/(str:slug)/components/(str:component_slug), 134
PATCH /api/component-lists/(str:slug)/, 134

/api/components

GET /api/components/, 115
GET /api/components/(string:project)/(string:component)/, 115
GET /api/components/(string:project)/(string:component)/changes/, 119
GET /api/components/(string:project)/(string:component)/lock/, 119
GET /api/components/(string:project)/(string:component)/monolingual/base/, 121
GET /api/components/(string:project)/(string:component)/new_template/, 121
GET /api/components/(string:project)/(string:component)/repository/, 120
GET /api/components/(string:project)/(string:component)/screenshots/, 119
GET /api/components/(string:project)/(string:component)/statistics/, 123
GET /api/components/(string:project)/(string:component)/translations/, 121
POST /api/components/(string:project)/(string:component)/changes/, 119
POST /api/components/(string:project)/(string:component)/lock/, 120
POST /api/components/(string:project)/(string:component)/monolingual/base/, 122
PUT /api/components/(string:project)/(string:component)/, 118
DELETE /api/components/(string:project)/(string:component)/, 118
PATCH /api/components/(string:project)/(string:component)/, 117

/api/glossary

GET /api/glossary/, 135
GET /api/glossary/(int:id)/, 135
GET /api/glossary/(int:id)/projects/, 136
GET /api/glossary/(int:id)/terms/, 137
GET /api/glossary/(int:id)/terms/(int:term_id)/, 137
POST /api/glossary/(int:id)/projects/, 137
POST /api/glossary/(int:id)/terms/, 137
PUT /api/glossary/(int:id)/, 136
PUT /api/glossary/(int:id)/terms/(int:term_id)/, 137
DELETE /api/glossary/(int:id)/, 136
DELETE /api/glossary/(int:id)/projects/, 137
DELETE /api/glossary/(int:id)/terms/(int:term_id)/, 138
PATCH /api/glossary/(int:id)/, 136
PATCH /api/glossary/(int:id)/terms/(int:term_id)/, 138

/api/groups

GET /api/groups/, 104
GET /api/groups/(int:id)/, 104
POST /api/groups/, 104

POST /api/groups/(int:id)/componentlists/, 106
POST /api/groups/(int:id)/components/, 105
POST /api/groups/(int:id)/languages/, 106
POST /api/groups/(int:id)/projects/, 106
POST /api/groups/(int:id)/roles/, 105
PUT /api/groups/(int:id)/, 105
DELETE /api/groups/(int:id)/, 105
DELETE /api/groups/(int:id)/componentlists/(int:componentlist_id), 106
DELETE /api/groups/(int:id)/components/(int:component_id), 106
DELETE /api/groups/(int:id)/languages/(string:language_code), 106
DELETE /api/groups/(int:id)/projects/(int:project_id), 106
PATCH /api/groups/(int:id)/, 105

/api/languages

GET /api/languages/, 108
GET /api/languages/(string:language)/, 108
GET /api/languages/(string:language)/statistics/, 109
POST /api/languages/, 108
PUT /api/languages/(string:language)/, 108
DELETE /api/languages/(string:language)/, 109
PATCH /api/languages/(string:language)/, 109

/api/projects

GET /api/projects/, 110
GET /api/projects/(string:project)/, 110
GET /api/projects/(string:project)/changes/, 111
GET /api/projects/(string:project)/components/, 112
GET /api/projects/(string:project)/languages/, 114
GET /api/projects/(string:project)/repository/, 111
GET /api/projects/(string:project)/statistics/, 114
POST /api/projects/, 110
POST /api/projects/(string:project)/components/, 112
POST /api/projects/(string:project)/repository/, 111
PUT /api/projects/(string:project)/, 110
DELETE /api/projects/(string:project)/, 111
PATCH /api/projects/(string:project)/, 110

/api/roles

GET /api/roles/, 107
GET /api/roles/(int:id)/, 107
POST /api/roles/, 107
PUT /api/roles/(int:id)/, 107
DELETE /api/roles/(int:id)/, 107
PATCH /api/roles/(int:id)/, 107

/api/screenshots

GET /api/screenshots/, 131
GET /api/screenshots/(int:id)/, 131
GET /api/screenshots/(int:id)/file/, 131
POST /api/screenshots/, 132
POST /api/screenshots/(int:id)/file/, 132
POST /api/screenshots/(int:id)/units/, 132
PUT /api/screenshots/(int:id)/, 133
DELETE /api/screenshots/(int:id)/, 133
DELETE /api/screenshots/(int:id)/units/(int:unit_
PATCH /api/screenshots/(int:id)/, 133

/api/translations

GET /api/translations/, 123
GET /api/translations/(string:project)/(string:co
123
GET /api/translations/(string:project)/(string:co
125
GET /api/translations/(string:project)/(string:co
126
GET /api/translations/(string:project)/(string:co
127
GET /api/translations/(string:project)/(string:co
128
GET /api/translations/(string:project)/(string:co
126
POST /api/translations/(string:project)/(string:c
126
POST /api/translations/(string:project)/(string:c
127
POST /api/translations/(string:project)/(string:c
128
POST /api/translations/(string:project)/(string:c
126
DELETE /api/translations/(string:project)/(string
125

/api/units

GET /api/units/, 129
GET /api/units/(int:id)/, 129
PUT /api/units/(int:id)/, 130
DELETE /api/units/(int:id)/, 130
PATCH /api/units/(int:id)/, 129

/api/users

GET /api/users/, 101
GET /api/users/(str:username)/, 101
GET /api/users/(str:username)/notifications/,
103
GET /api/users/(str:username)/notifications/(int:subscription_id)/,
103
GET /api/users/(str:username)/statistics/,
102
POST /api/users/, 101
POST /api/users/(str:username)/groups/,
102
POST /api/users/(str:username)/notifications/,
103
PUT /api/users/(str:username)/, 102
PUT /api/users/(str:username)/notifications/(int:subscription_id)/,
103
DELETE /api/users/(str:username)/, 102
DELETE /api/users/(str:username)/notifications/(int:subscription_id)/,
104
PATCH /api/users/(str:username)/, 102
PATCH /api/users/(str:username)/notifications/(int:subscription_id)/,
103

/exports

GET /exports/rss/, 141
GET /exports/rss/(string:project)/, 141
GET /exports/rss/(string:project)/(string:component)/,
141
GET /exports/rss/(string:project)/(string:component)/(string:language)/,
141
GET /exports/rss/language/(string:language)/,
141
GET /exports/stats/(string:project)/(string:component)/,
140

/hooks

GET /hooks/update/(string:project)/,
138
GET /hooks/update/(string:project)/(string:component)/,
138
POST /hooks/azure/, 139
POST /hooks/bitbucket/, 139
POST /hooks/gitea/, 139
POST /hooks/gitee/, 139
POST /hooks/github/, 138
POST /hooks/gitlab/, 138
POST /hooks/pagure/, 139

Índice

Símbolos

.XML resource file
    file format, 85
--add
    auto_translate command line option, 335
--addon ADDON
    install_addon command line option, 341
--age HOURS
    commit_pending command line option, 336
--author USER@EXAMPLE.COM
    add_suggestions command line option, 335
--base-file-template TEMPLATE
    import_project command line option, 339
--check
    importusers command line option, 341
--config PATH
    wlc command line option, 143
--config-section SECTION
    wlc command line option, 143
--configuration CONFIG
    install_addon command line option, 341
--convert
    wlc command line option, 144
--email USER@EXAMPLE.COM
    createadmin command line option, 337
--file-format FORMAT
    import_project command line option, 339
--force
    loadpo command line option, 342
--force-commit
    pushgit command line option, 343
--format {csv,json,text,html}
    wlc command line option, 143
--ignore
    import_json command line option, 338
--inconsistent
    auto_translate command line option, 335
--input
    wlc command line option, 144
--key KEY
    wlc command line option, 143
--lang LANGUAGE
    loadpo command line option, 342
--language-code
    list_translators command line option, 342
--language-map LANGMAP
    import_memory command line option, 338
--language-regex REGEX
    import_project command line option, 339
--license NAME
    import_project command line option, 339
--license-url URL
    import_project command line option, 339
--main-component
    import_project command line option, 339
--main-component COMPONENT
    import_json command line option, 338
--mt MT
    auto_translate command line option, 335
--name
    createadmin command line option, 337
--name-template TEMPLATE
    import_project command line option, 339
--new-base-template TEMPLATE
    import_project command line option, 339
--no-password
    createadmin command line option, 337
--no-privs-update
    setupgroups command line option, 343
--no-projects-update

setupgroups command line option, 343
--no-update
setuplang command line option, 344
--output
wlc command line option, 144
--overwrite
auto_translate command line option, 335
wlc command line option, 144
--password PASSWORD
createadmin command line option, 337
--project PROJECT
import_json command line option, 338
--source PROJECT/COMPONENT
auto_translate command line option, 335
--threshold THRESHOLD
auto_translate command line option, 335
--update
createadmin command line option, 337
import_json command line option, 338
install_addon command line option, 341
--url URL
wlc command line option, 143
--user USERNAME
auto_translate command line option, 335
--username USERNAME
createadmin command line option, 337
--vcs NAME
import_project command line option, 339

A

add_suggestions
weblate admin command, 335
add_suggestions command line option
--author USER@EXAMPLE.COM, 335
ADMINS
setting, 186
AKISMET_API_KEY
setting, 292
ALLOWED_HOSTS
setting, 187
Android
file format, 80
ANONYMOUS_USER_NAME
setting, 292
API, 98, 141, 146
Apple strings
file format, 81
ARB
file format, 84
AUDITLOG_EXPIRY
setting, 293
AUTH_LOCK_ATTEMPTS
setting, 293
AUTH_TOKEN_VALID
setting, 294
auto_translate
weblate admin command, 335
auto_translate command line option
--add, 335
--inconsistent, 335
--mt MT, 335
--overwrite, 335
--source PROJECT/COMPONENT, 335
--threshold THRESHOLD, 335
--user USERNAME, 335
AUTO_UPDATE
setting, 293
AUTOFIX_LIST
setting, 294
AVATAR_URL_PREFIX
setting, 293

B

BASE_DIR
setting, 295
bilingual
translation, 73

C

celery_queues
weblate admin command, 336
changes
wlc command line option, 144
CHECK_LIST
setting, 295
checkgit
weblate admin command, 336
cleanup
wlc command line option, 144
cleanuptrans
weblate admin command, 336
Comma separated values
file format, 86
Command (*classe em wlc.main*), 148
COMMENT_CLEANUP_DAYS
setting, 296
commit
wlc command line option, 143
commit_pending
weblate admin command, 336
commit_pending command line option
--age HOURS, 336
COMMIT_PENDING_HOURS
setting, 296
commitgit
weblate admin command, 336
createadmin
weblate admin command, 337
createadmin command line option
--email USER@EXAMPLE.COM, 337

--name, 337
--no-password, 337
--password PASSWORD, 337
--update, 337
--username USERNAME, 337
CSP_CONNECT_SRC
setting, 295
CSP_FONT_SRC
setting, 295
CSP_IMG_SRC
setting, 295
CSP_SCRIPT_SRC
setting, 295
CSP_STYLE_SRC
setting, 295
CSV
file format, 86

D

DATA_DIR
setting, 296
DATABASE_BACKUP
setting, 296
DATABASES
setting, 187
DEBUG
setting, 187
DEFAULT_ACCESS_CONTROL
setting, 297
DEFAULT_ADD_MESSAGE
setting, 297
DEFAULT_ADDON_MESSAGE
setting, 297
DEFAULT_ADDONS
setting, 297
DEFAULT_COMMIT_MESSAGE
setting, 297
DEFAULT_COMMITER_EMAIL
setting, 298
DEFAULT_COMMITER_NAME
setting, 298
DEFAULT_DELETE_MESSAGE
setting, 297
DEFAULT_FROM_EMAIL
setting, 187
DEFAULT_LANGUAGE
setting, 298
DEFAULT_MERGE_MESSAGE
setting, 297
DEFAULT_MERGE_STYLE
setting, 298
DEFAULT_PULL_MESSAGE
setting, 299
DEFAULT_RESTRICTED_COMPONENT
setting, 297
DEFAULT_TRANSLATION_PROPAGATION
setting, 299
download
wlc command line option, 144
DTD
file format, 87
dump_memory
weblate admin command, 337
dumpuserdata
weblate admin command, 337

E

ENABLE_AVATARS
setting, 299
ENABLE_HOOKS
setting, 299
ENABLE_HTTPS
setting, 299
ENABLE_SHARING
setting, 299

F

file format
.XML resource file, 85
Android, 80
Apple strings, 81
ARB, 84
Comma separated values, 86
CSV, 86
DTD, 87
gettext, 75
go-i18n, 84
GWT properties, 78
i18next, 83
INI translations, 79
Java properties, 78
Joomla translations, 79
JSON, 82
PHP strings, 81
PO, 75
Qt, 80
RC, 88
RESX, 85
Ruby YAML, 87
Ruby YAML Ain't Markup Language, 87
string resources, 80
TS, 80
XLIFF, 76
XML, 87
YAML, 86
YAML Ain't Markup Language, 86

G

get() (*método wlc.Weblate*), 147
gettext
file format, 75
GITHUB_CREDENTIALS
setting, 300
GITHUB_TOKEN
setting, 301
GITHUB_USERNAME

setting, 301
GITLAB_CREDENTIALS
setting, 300
GITLAB_TOKEN
setting, 300
GITLAB_USERNAME
setting, 300
go-i18n
file format, 84
GOOGLE_ANALYTICS_ID
setting, 301
GWT properties
file format, 78

H

HIDE_REPO_CREDENTIALS
setting, 301
HIDE_VERSION
setting, 301

I

i18next
file format, 83
import_demo
weblate admin command, 337
import_json
weblate admin command, 338
import_json command line option
--ignore, 338
--main-component COMPONENT, 338
--project PROJECT, 338
--update, 338
import_memory
weblate admin command, 338
import_memory command line option
--language-map LANGMAP, 338
import_project
weblate admin command, 339
import_project command line option
--base-file-template TEMPLATE, 339
--file-format FORMAT, 339
--language-regex REGEX, 339
--license NAME, 339
--license-url URL, 339
--main-component, 339
--name-template TEMPLATE, 339
--new-base-template TEMPLATE, 339
--vcs NAME, 339
importuserdata
weblate admin command, 341
importusers
weblate admin command, 341
importusers command line option
--check, 341
INI translations
file format, 79
install_addon
weblate admin command, 341
install_addon command line option
--addon ADDON, 341
--configuration CONFIG, 341
--update, 341
IP_BEHIND_REVERSE_PROXY
setting, 302
IP_PROXY_HEADER
setting, 302
IP_PROXY_OFFSET
setting, 302
iPad
translation, 81
iPhone
translation, 81

J

Java properties
file format, 78
Joomla translations
file format, 79
JSON
file format, 82

L

LEGAL_URL
setting, 302
LICENSE_EXTRA
setting, 303
LICENSE_FILTER
setting, 303
LICENSE_REQUIRED
setting, 303
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH
setting, 304
list_languages
weblate admin command, 341
list_translators
weblate admin command, 342
list_translators command line option
--language-code, 342
list_versions
weblate admin command, 342
list-components
wlc command line option, 143
list-languages
wlc command line option, 143
list-projects
wlc command line option, 143
list-translations
wlc command line option, 143
load() (*método wlc.config.WeblateConfig*), 147
loadpo
weblate admin command, 342
loadpo command line option
--force, 342
--lang LANGUAGE, 342
LOCALIZE_CDN_PATH
setting, 304

LOCALIZE_CDN_URL
    setting, 304
lock
    wlc command line option, 144
lock_translation
    weblate admin command, 342
lock-status
    wlc command line option, 144
LOGIN_REQUIRED_URLS
    setting, 304
LOGIN_REQUIRED_URLS_EXCEPTIONS
    setting, 304
ls
    wlc command line option, 143

M

MACHINE_TRANSLATION_SERVICES
    setting, 305
main() (*no módulo wlc.main*), 147
MATOMO_SITE_ID
    setting, 305
MATOMO_URL
    setting, 305
monolingual
    translation, 73
move_language
    weblate admin command, 343
MT_APERTIUM_APY
    setting, 306
MT_AWS_ACCESS_KEY_ID
    setting, 306
MT_AWS_REGION
    setting, 306
MT_AWS_SECRET_ACCESS_KEY
    setting, 306
MT_BAIDU_ID
    setting, 306
MT_BAIDU_SECRET
    setting, 307
MT_DEEPL_API_VERSION
    setting, 307
MT_DEEPL_KEY
    setting, 307
MT_GOOGLE_CREDENTIALS
    setting, 307
MT_GOOGLE_KEY
    setting, 307
MT_GOOGLE_LOCATION
    setting, 308
MT_GOOGLE_PROJECT
    setting, 308
MT_MICROSOFT_BASE_URL
    setting, 308
MT_MICROSOFT_COGNITIVE_KEY
    setting, 308
MT_MICROSOFT_ENDPOINT_URL
    setting, 308
MT_MICROSOFT_REGION
    setting, 308
MT_MODERNMT_KEY
    setting, 308
MT_MODERNMT_URL
    setting, 309
MT_MYMEMORY_EMAIL
    setting, 309
MT_MYMEMORY_KEY
    setting, 309
MT_MYMEMORY_USER
    setting, 309
MT_NETEASE_KEY
    setting, 309
MT_NETEASE_SECRET
    setting, 309
MT_SAP_BASE_URL
    setting, 310
MT_SAP_PASSWORD
    setting, 310
MT_SAP_SANDBOX_APIKEY
    setting, 310
MT_SAP_USE_MT
    setting, 311
MT_SAP_USERNAME
    setting, 310
MT_SERVICES
    setting, 305
MT_TMSERVER
    setting, 309
MT_YANDEX_KEY
    setting, 310
MT_YOUDAO_ID
    setting, 310
MT_YOUDAO_SECRET
    setting, 310
módulo
    wlc, 146
    wlc.config, 147
    wlc.main, 147

N

NEARBY_MESSAGES
    setting, 311

P

PAGURE_CREDENTIALS
    setting, 311
PAGURE_TOKEN
    setting, 311
PAGURE_USERNAME
    setting, 311
PHP strings
    file format, 81
PIWIK_SITE_ID
    setting, 305
PIWIK_URL
    setting, 305
PO

file format, 75
post() (*método wlc.Weblate*), 147
pull
wlc command line option, 143
push
wlc command line option, 144
pushgit
weblate admin command, 343
pushgit command line option
--force-commit, 343
Python, 146

Q

Qt
file format, 80

R

RATELIMIT_ATTEMPTS
setting, 312
RATELIMIT_LOCKOUT
setting, 312
RATELIMIT_WINDOW
setting, 312
RC
file format, 88
register_command() (*no módulo wlc.main*), 147
REGISTRATION_ALLOW_BACKENDS
setting, 312
REGISTRATION_CAPTCHA
setting, 313
REGISTRATION_EMAIL_MATCH
setting, 313
REGISTRATION_OPEN
setting, 313
repo
wlc command line option, 144
REPOSITORY_ALERT_THRESHOLD
setting, 313
REQUIRE_LOGIN
setting, 314
reset
wlc command line option, 144
REST, 98
RESX
file format, 85
RFC
RFC 4646, 73
Ruby YAML
file format, 87
Ruby YAML Ain't Markup Language
file format, 87

S

SECRET_KEY
setting, 187
SENTRY_DSN
setting, 314
SERVER_EMAIL
setting, 187
SESSION_COOKIE_AGE_AUTHENTICATED
setting, 314
SESSION_ENGINE
setting, 187
setting
ADMINS, 186
AKISMET_API_KEY, 292
ALLOWED_HOSTS, 187
ANONYMOUS_USER_NAME, 292
AUDITLOG_EXPIRY, 293
AUTH_LOCK_ATTEMPTS, 293
AUTH_TOKEN_VALID, 294
AUTO_UPDATE, 293
AUTOFIX_LIST, 294
AVATAR_URL_PREFIX, 293
BASE_DIR, 295
CHECK_LIST, 295
COMMENT_CLEANUP_DAYS, 296
COMMIT_PENDING_HOURS, 296
CSP_CONNECT_SRC, 295
CSP_FONT_SRC, 295
CSP_IMG_SRC, 295
CSP_SCRIPT_SRC, 295
CSP_STYLE_SRC, 295
DATA_DIR, 296
DATABASE_BACKUP, 296
DATABASES, 187
DEBUG, 187
DEFAULT_ACCESS_CONTROL, 297
DEFAULT_ADD_MESSAGE, 297
DEFAULT_ADDON_MESSAGE, 297
DEFAULT_ADDONS, 297
DEFAULT_COMMIT_MESSAGE, 297
DEFAULT_COMMITER_EMAIL, 298
DEFAULT_COMMITER_NAME, 298
DEFAULT_DELETE_MESSAGE, 297
DEFAULT_FROM_EMAIL, 187
DEFAULT_LANGUAGE, 298
DEFAULT_MERGE_MESSAGE, 297
DEFAULT_MERGE_STYLE, 298
DEFAULT_PULL_MESSAGE, 299
DEFAULT_RESTRICTED_COMPONENT, 297
DEFAULT_TRANSLATION_PROPAGATION, 299
ENABLE_AVATARS, 299
ENABLE_HOOKS, 299
ENABLE_HTTPS, 299
ENABLE_SHARING, 299
GITHUB_CREDENTIALS, 300
GITHUB_TOKEN, 301
GITHUB_USERNAME, 301
GITLAB_CREDENTIALS, 300
GITLAB_TOKEN, 300
GITLAB_USERNAME, 300
GOOGLE_ANALYTICS_ID, 301
HIDE_REPO_CREDENTIALS, 301
HIDE_VERSION, 301

IP_BEHIND_REVERSE_PROXY, 302
IP_PROXY_HEADER, 302
IP_PROXY_OFFSET, 302
LEGAL_URL, 302
LICENSE_EXTRA, 303
LICENSE_FILTER, 303
LICENSE_REQUIRED, 303
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH, 304
LOCALIZE_CDN_PATH, 304
LOCALIZE_CDN_URL, 304
LOGIN_REQUIRED_URLS, 304
LOGIN_REQUIRED_URLS_EXCEPTIONS, 304
MACHINE_TRANSLATION_SERVICES, 305
MATOMO_SITE_ID, 305
MATOMO_URL, 305
MT_APERTIUM_APY, 306
MT_AWS_ACCESS_KEY_ID, 306
MT_AWS_REGION, 306
MT_AWS_SECRET_ACCESS_KEY, 306
MT_BAIDU_ID, 306
MT_BAIDU_SECRET, 307
MT_DEEPL_API_VERSION, 307
MT_DEEPL_KEY, 307
MT_GOOGLE_CREDENTIALS, 307
MT_GOOGLE_KEY, 307
MT_GOOGLE_LOCATION, 308
MT_GOOGLE_PROJECT, 308
MT_MICROSOFT_BASE_URL, 308
MT_MICROSOFT_COGNITIVE_KEY, 308
MT_MICROSOFT_ENDPOINT_URL, 308
MT_MICROSOFT_REGION, 308
MT_MODERNMT_KEY, 308
MT_MODERNMT_URL, 309
MT_MYMEMORY_EMAIL, 309
MT_MYMEMORY_KEY, 309
MT_MYMEMORY_USER, 309
MT_NETEASE_KEY, 309
MT_NETEASE_SECRET, 309
MT_SAP_BASE_URL, 310
MT_SAP_PASSWORD, 310
MT_SAP_SANDBOX_APIKEY, 310
MT_SAP_USE_MT, 311
MT_SAP_USERNAME, 310
MT_SERVICES, 305
MT_TMSERVER, 309
MT_YANDEX_KEY, 310
MT_YOUDAO_ID, 310
MT_YOUDAO_SECRET, 310
NEARBY_MESSAGES, 311
PAGURE_CREDENTIALS, 311
PAGURE_TOKEN, 311
PAGURE_USERNAME, 311
PIWIK_SITE_ID, 305
PIWIK_URL, 305
RATELIMIT_ATTEMPTS, 312
RATELIMIT_LOCKOUT, 312
RATELIMIT_WINDOW, 312
REGISTRATION_ALLOW_BACKENDS, 312
REGISTRATION_CAPTCHA, 313
REGISTRATION_EMAIL_MATCH, 313
REGISTRATION_OPEN, 313
REPOSITORY_ALERT_THRESHOLD, 313
REQUIRE_LOGIN, 314
SECRET_KEY, 187
SENTRY_DSN, 314
SERVER_EMAIL, 187
SESSION_COOKIE_AGE_AUTHENTICATED, 314
SESSION_ENGINE, 187
SIMPLIFY_LANGUAGES, 314
SINGLE_PROJECT, 315
SITE_DOMAIN, 314
SITE_TITLE, 315
SPECIAL_CHARS, 315
STATUS_URL, 315
SUGGESTION_CLEANUP_DAYS, 316
UPDATE_LANGUAGES, 316
URL_PREFIX, 316
VCS_BACKENDS, 316
VCS_CLONE_DEPTH, 317
WEBLATE_ADDONS, 317
WEBLATE_EXPORTERS, 318
WEBLATE_FORMATS, 318
WEBLATE_GPG_IDENTITY, 318
setupgroups
    weblate admin command, 343
setupgroups command line option
    --no-privs-update, 343
    --no-projects-update, 343
setuplang
    weblate admin command, 344
setuplang command line option
    --no-update, 344
show
    wlc command line option, 143
SIMPLIFY_LANGUAGES
    setting, 314
SINGLE_PROJECT
    setting, 315
SITE_DOMAIN
    setting, 314
SITE_TITLE
    setting, 315
SPECIAL_CHARS
    setting, 315
statistics
    wlc command line option, 144
STATUS_URL
    setting, 315
string resources
    file format, 80
SUGGESTION_CLEANUP_DAYS
    setting, 316

T

translation
    bilingual, 73
    iPad, 81
    iPhone, 81
    monolingual, 73
TS
    file format, 80

U

unlock
    wlc command line option, 144
unlock_translation
    weblate admin command, 343
UPDATE_LANGUAGES
    setting, 316
updatechecks
    weblate admin command, 344
updategit
    weblate admin command, 344
upload
    wlc command line option, 144
URL_PREFIX
    setting, 316

V

variável de ambiente
    CELERY_BACKUP_OPTIONS, 165
    CELERY_BEAT_OPTIONS, 165
    CELERY_MAIN_OPTIONS, 165
    CELERY_MEMORY_OPTIONS, 165
    CELERY_NOTIFY_OPTIONS, 165
    CELERY_TRANSLATE_OPTIONS, 165
    POSTGRES_ALTER_ROLE, 162
    POSTGRES_DATABASE, 162
    POSTGRES_HOST, 162
    POSTGRES_PASSWORD, 162
    POSTGRES_PORT, 162
    POSTGRES_SSL_MODE, 162
    POSTGRES_USER, 162
    REDIS_DB, 162
    REDIS_HOST, 162
    REDIS_PASSWORD, 162
    REDIS_PORT, 162
    REDIS_TLS, 163
    REDIS_VERIFY_SSL, 163
    ROLLBAR_ENVIRONMENT, 164
    ROLLBAR_KEY, 164
    SENTRY_DSN, 164
    SENTRY_ENVIRONMENT, 164
    SOCIAL_AUTH_SLACK_SECRET, 161
    UWSGI_WORKERS, 165
    WEBLATE_ADD_ADDONS, 165
    WEBLATE_ADD_APPS, 165
    WEBLATE_ADD_AUTOFIX, 165
    WEBLATE_ADD_CHECK, 165
    WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS, 156
    WEBLATE_ADMIN_EMAIL, 153, 154, 158
    WEBLATE_ADMIN_NAME, 153, 154
    WEBLATE_ADMIN_PASSWORD, 150, 153, 154
    WEBLATE_AKISMET_API_KEY, 157
    WEBLATE_ALLOWED_HOSTS, 153, 154, 187, 191, 315
    WEBLATE_AUTH_LDAP_BIND_DN, 159
    WEBLATE_AUTH_LDAP_BIND_PASSWORD, 159
    WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS, 159
    WEBLATE_AUTH_LDAP_SERVER_URI, 159
    WEBLATE_AUTH_LDAP_USER_ATTR_MAP, 159
    WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE, 159
    WEBLATE_AUTH_LDAP_USER_SEARCH, 159
    WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER, 159
    WEBLATE_AUTH_LDAP_USER_SEARCH_UNION, 159
    WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER, 159
    WEBLATE_CSP_CONNECT_SRC, 157
    WEBLATE_CSP_FONT_SRC, 157
    WEBLATE_CSP_IMG_SRC, 157
    WEBLATE_CSP_SCRIPT_SRC, 157
    WEBLATE_CSP_STYLE_SRC, 157
    WEBLATE_DATABASE_BACKUP, 162
    WEBLATE_DEBUG, 153
    WEBLATE_DEFAULT_ACCESS_CONTROL, 156
    WEBLATE_DEFAULT_COMMITER_EMAIL, 156
    WEBLATE_DEFAULT_COMMITER_NAME, 156
    WEBLATE_DEFAULT_FROM_EMAIL, 154
    WEBLATE_DEFAULT_RESTRICTED_COMPONENT, 156
    WEBLATE_DEFAULT_TRANSLATION_PROPAGATION, 156
    WEBLATE_EMAIL_BACKEND, 164
    WEBLATE_EMAIL_HOST, 163
    WEBLATE_EMAIL_HOST_PASSWORD, 163
    WEBLATE_EMAIL_HOST_USER, 163
    WEBLATE_EMAIL_PORT, 163, 164
    WEBLATE_EMAIL_USE_SSL, 163, 164
    WEBLATE_EMAIL_USE_TLS, 163, 164
    WEBLATE_ENABLE_HTTPS, 155
    WEBLATE_GITHUB_TOKEN, 156
    WEBLATE_GITHUB_USERNAME, 156
    WEBLATE_GITLAB_TOKEN, 156
    WEBLATE_GITLAB_USERNAME, 156
    WEBLATE_GOOGLE_ANALYTICS_ID, 156
    WEBLATE_GPG_IDENTITY, 157
    WEBLATE_HIDE_VERSION, 157
    WEBLATE_IP_PROXY_HEADER, 155
    WEBLATE_LICENSE_FILTER, 157
    WEBLATE_LOCALIZE_CDN_PATH, 164
    WEBLATE_LOCALIZE_CDN_URL, 164

WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS, 155
WEBLATE_LOGLEVEL, 153
WEBLATE_MT_APERTIUM_APY, 157
WEBLATE_MT_AWS_ACCESS_KEY_ID, 157
WEBLATE_MT_AWS_REGION, 157
WEBLATE_MT_AWS_SECRET_ACCESS_KEY, 157
WEBLATE_MT_DEEPL_API_VERSION, 157
WEBLATE_MT_DEEPL_KEY, 157
WEBLATE_MT_GLOSBE_ENABLED, 158
WEBLATE_MT_GOOGLE_KEY, 157
WEBLATE_MT_MICROSOFT_BASE_URL, 158
WEBLATE_MT_MICROSOFT_COGNITIVE_KEY, 157
WEBLATE_MT_MICROSOFT_ENDPOINT_URL, 158
WEBLATE_MT_MICROSOFT_REGION, 158
WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED, 158
WEBLATE_MT_MODERNMT_KEY, 158
WEBLATE_MT_MYMEMORY_ENABLED, 158
WEBLATE_MT_SAP_BASE_URL, 158
WEBLATE_MT_SAP_PASSWORD, 158
WEBLATE_MT_SAP_SANDBOX_APIKEY, 158
WEBLATE_MT_SAP_USE_MT, 158
WEBLATE_MT_SAP_USERNAME, 158
WEBLATE_NO_EMAIL_AUTH, 162
WEBLATE_PAGURE_TOKEN, 156
WEBLATE_PAGURE_USERNAME, 156
WEBLATE_REGISTRATION_ALLOW_BACKENDS, 154
WEBLATE_REGISTRATION_OPEN, 154
WEBLATE_REMOVE_ADDONS, 165
WEBLATE_REMOVE_APPS, 165
WEBLATE_REMOVE_AUTOFIX, 165
WEBLATE_REMOVE_CHECK, 165
WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS, 156
WEBLATE_REQUIRE_LOGIN, 155, 314
WEBLATE_SAML_IDP_ENTITY_ID, 161
WEBLATE_SAML_IDP_URL, 161
WEBLATE_SAML_IDP_X509CERT, 161
WEBLATE_SECURE_PROXY_SSL_HEADER, 155
WEBLATE_SERVER_EMAIL, 154
WEBLATE_SILENCED_SYSTEM_CHECKS, 157, 212
WEBLATE_SIMPLIFY_LANGUAGES, 156
WEBLATE_SITE_DOMAIN, 153, 189, 206, 315
WEBLATE_SITE_TITLE, 153
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY, 160
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET, 160
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY, 161
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET, 161
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID, 161
WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY, 160
WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET, 160
WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY, 160
WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET, 160
WEBLATE_SOCIAL_AUTH_FEDORA, 161
WEBLATE_SOCIAL_AUTH_GITHUB_KEY, 160
WEBLATE_SOCIAL_AUTH_GITHUB_SECRET, 160
WEBLATE_SOCIAL_AUTH_GITLAB_API_URL, 160
WEBLATE_SOCIAL_AUTH_GITLAB_KEY, 160
WEBLATE_SOCIAL_AUTH_GITLAB_SECRET, 160
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY, 160
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET, 160
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS, 160
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS, 160
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL, 161
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM, 161
WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URL, 161
WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY, 161
WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY, 161
WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET, 161
WEBLATE_SOCIAL_AUTH_OPENSUSE, 161
WEBLATE_SOCIAL_AUTH_SLACK_KEY, 161
WEBLATE_SOCIAL_AUTH_UBUNTU, 161
WEBLATE_TIME_ZONE, 154
WEBLATE_URL_PREFIX, 157
WL_BRANCH, 289
WL_COMPONENT_NAME, 289
WL_COMPONENT_SLUG, 289
WL_COMPONENT_URL, 289
WL_ENGAGE_URL, 289
WL_FILE_FORMAT, 289
WL_FILEMASK, 289
WL_LANGUAGE, 289
WL_NEW_BASE, 289
WL_PATH, 289
WL_PREVIOUS_HEAD, 289
WL_PROJECT_NAME, 289

    WL_PROJECT_SLUG, 289
    WL_REPO, 289
    WL_TEMPLATE, 289
    WL_VCS, 289
VCS_BACKENDS
    setting, 316
VCS_CLONE_DEPTH
    setting, 317
version
    wlc command line option, 143

W

Weblate (*classe em wlc*), 147
weblate admin command
    add_suggestions, 335
    auto_translate, 335
    celery_queues, 336
    checkgit, 336
    cleanuptrans, 336
    commit_pending, 336
    commitgit, 336
    createadmin, 337
    dump_memory, 337
    dumpuserdata, 337
    import_demo, 337
    import_json, 338
    import_memory, 338
    import_project, 339
    importuserdata, 341
    importusers, 341
    install_addon, 341
    list_languages, 341
    list_translators, 342
    list_versions, 342
    loadpo, 342
    lock_translation, 342
    move_language, 343
    pushgit, 343
    setupgroups, 343
    setuplang, 344
    unlock_translation, 343
    updatechecks, 344
    updategit, 344
WEBLATE_ADDONS
    setting, 317
WEBLATE_ADMIN_EMAIL, 153, 154, 158
WEBLATE_ADMIN_NAME, 153, 154
WEBLATE_ADMIN_PASSWORD, 150, 153, 154
WEBLATE_ALLOWED_HOSTS, 153, 187, 191, 315
WEBLATE_EMAIL_PORT, 163, 164
WEBLATE_EMAIL_USE_SSL, 163, 164
WEBLATE_EMAIL_USE_TLS, 163, 164
WEBLATE_EXPORTERS
    setting, 318
WEBLATE_FORMATS
    setting, 318
WEBLATE_GPG_IDENTITY
    setting, 318
WEBLATE_LOCALIZE_CDN_PATH, 164
WEBLATE_REQUIRE_LOGIN, 314
WEBLATE_SILENCED_SYSTEM_CHECKS, 212
WEBLATE_SITE_DOMAIN, 189, 206, 315
WeblateConfig (*classe em wlc.config*), 147
WeblateException, 146
wlc, 141
    módulo, 146
wlc command line option
    --config PATH, 143
    --config-section SECTION, 143
    --convert, 144
    --format {csv,json,text,html}, 143
    --input, 144
    --key KEY, 143
    --output, 144
    --overwrite, 144
    --url URL, 143
    changes, 144
    cleanup, 144
    commit, 143
    download, 144
    list-components, 143
    list-languages, 143
    list-projects, 143
    list-translations, 143
    lock, 144
    lock-status, 144
    ls, 143
    pull, 143
    push, 144
    repo, 144
    reset, 144
    show, 143
    statistics, 144
    unlock, 144
    upload, 144
    version, 143
wlc.config
    módulo, 147
wlc.main
    módulo, 147

X

XLIFF
    file format, 76
XML
    file format, 87

Y

YAML
    file format, 86
YAML Ain't Markup Language
    file format, 86