VLibras Console

DEV Guide

fev/2021

I. Introdução

O VLibras Console é uma ferramenta para o monitoramento, configuração e gerenciamento dos serviços e ferramentas da Suíte VLibras. Este manual tem como objetivo apresentar as funcionalidades presentes no VLibras Console, mostrando fluxos e parte da arquitetura que as envolve, de modo mais técnico. Para a leitura deste documento, é recomendado ler o ReadMe do projeto para compreender a montagem do ambiente (back end e front end), como também o Manual de Uso, que pode ser encontrado ao clicar no botão do canto superior direito do VLibras Console.

II. Arquitetura e Fluxos

O VLibras Console possui vários escopos por ser um sistema de gerenciamento e acompanhamento. Abaixo, será apresentado o fluxo das funcionalidades que a aplicação possui e uma breve descrição sobre cada uma.

A. Usuários

No VLibras Console, a autenticação de usuários é feita através do Login Único, uma solução da Secretaria de Governo Digital do Ministério da Economia para unificar um método de autenticação de usuários para qualquer plataforma do governo. O roteiro de integração desta plataforma pode ser encontrado aqui.

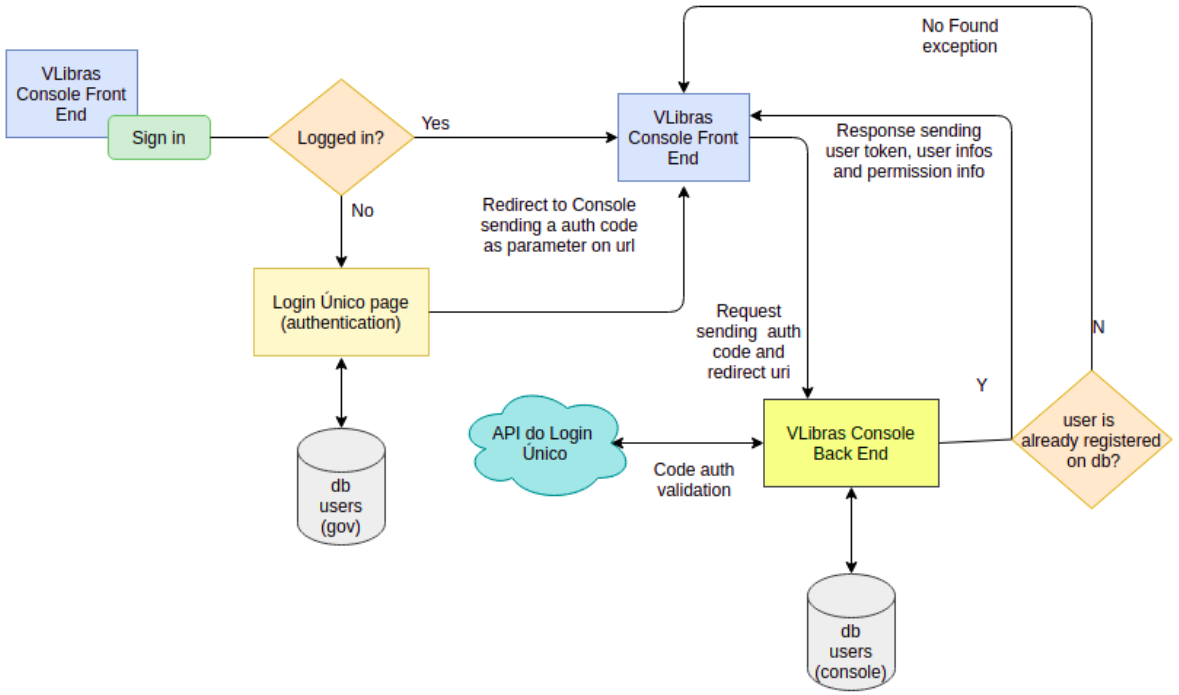

Figura 1 - Fluxo de autenticação e autorização do VLibras Console.

Na Figura 1, observa-se o fluxo de autenticação de usuários, que pode ser descrito a seguir:

1. O usuário clica no botão de Entrar, na página inicial do VLibras Console.
2. Se o usuário não estiver logado, ele é redirecionado para página do Login Único do governo, a fim de fazer seu cadastro ou usar suas credenciais para logar no sistema.
    a. Se o usuário já estiver logado, com o token ativo, será feito um redirecionamento direto para a página de autorização do Console.
3. Ao logar, será feito outro redirecionamento, agora para própria url do Console, em uma rota de autorização, mandando como parâmetro na url um código de autenticação. Exemplo: https://console-dth.vlibras.gov.br/authorize?code=8GegVL.
4. Esse código de autenticação é enviado ao Console Backend via POST.
5. O Console valida o código de autenticação através de uma comunicação de uma API do Login Único. Com isso, é retornada as credenciais do usuário.
    a. Se o código não for autorizado, é retornada uma exceção na resposta do endpoint ao Console Front-End.
6. O Console valida se o usuário já está cadastrado em seu banco de dados. Essa etapa é importante pois não é qualquer pessoa que pode logar na aplicação, apenas pessoas pré-cadastradas por algum administrador. Ao serem cadastradas, elas recebem um tipo de permissão, que definirá o que poderá, ou não, ser visualizado no Console.
    a. Se o usuário não estiver pré-cadastrado, será retornada uma exceção na resposta do endpoint ao Console Front-End.
7. Com tudo validado, é enviado para o Front End as credenciais do usuário, como também sua permissão e os tokens que serão armazenados no cookie do navegador para guardar sua sessão.

B. Monitoramento

O monitoramento do Console permite a visualização de dados estatísticos dos serviços em produção. Em geral, esse monitoramento é feito via dashboards, e esses dashboards podem ser pré-definidos (carregados diretamente do Grafana) ou montados e personalizados pelo usuário. O Grafana fornece tabelas, gráficos e alertas para a Web quando conectado a fontes de dados suportadas.

Na Figura 2 observa-se o fluxo de hierarquia de monitoramento, que pode ser descrito a seguir:

1. É configurada previamente a plataforma Grafana. Para disponibilizar monitoramento através de dashboards, é necessário configurar data sources, que proverá dados para alimentar os painéis. Os dados são serviços da Suíte VLibras que estão em produção.
2. Com isso, cria-se dashboards, com os painéis referentes aos dados que deseja-se obter, podendo configurar a visualização do dado visualmente. Em alguns dashboards, é configurado para os dados mudarem conforme o namespace e o deployment, por exemplo.
3. Com os dashboards prontos no Grafana, o VLibras Console utiliza-se de iframes para carregar os dashboards completos via Front End, apenas usando a url do Grafana para fazer a exibição. Esses dashboards que são carregados por completo são denominados Dashboards Nativos. Atualmente, possuímos dashboards de Infraestrutura, Wikilibras, Tradução, Dicionário, Portal e VLibras Vídeo. O único monitoramento que não é feito via dashboard é o do Widget, que os dados são providos diretamente da API de estatística.
4. O Console também provê a opção de montar dashboards personalizados. Dessa maneira, em vez de usar os dashboards do Grafana, são usados os painéis. Esses painéis são carregados via iframe, passando sua url, e o Console Back End provê de uma estrutura que monta a lógica de carregamento dos painéis de acordo com sua categoria (tradução, dicionaŕio, wikilibras, etc.) e seu grupo de deployment (wikilibras-frontend, wikilibras-backend, etc).

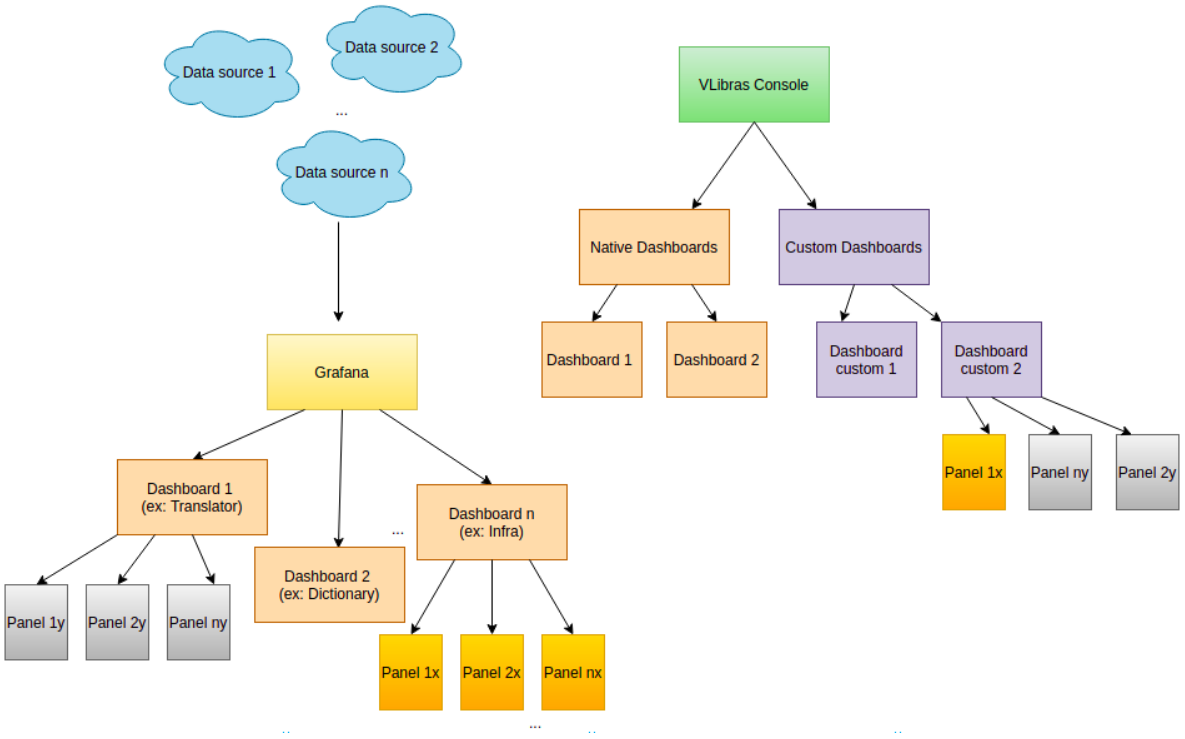

Figura 2 - Fluxo de autenticação e autorização do VLibras Console.

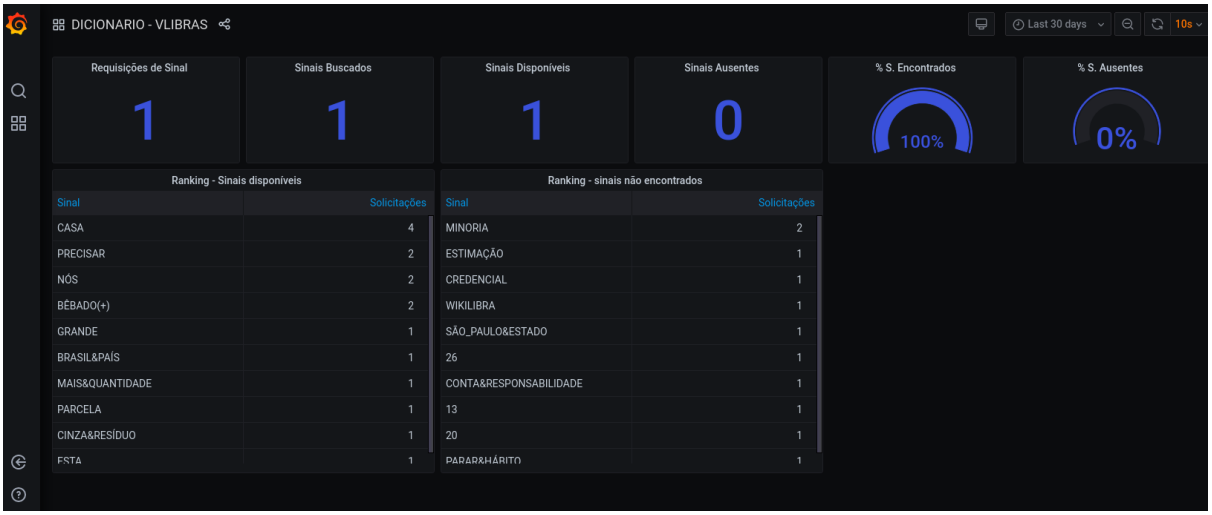

Figura 3 - Dashboard nativo do Grafana.

Figura 4 - Dashboard personalizado do Console.

C. Personalização de Avatar

A personalização de avatar é realizada no VLibras Widget, sendo passado um Json com informações relacionadas a cor de cabelo, roupa, pele, disposição de logo em camisa, posicionamento da logo, etc. Para a criação desse Json de personalização, foi criada uma tela que gera um arquivo e exibe essa personalização de forma gráfica ao usuário, para posteriormente ser usado no Widget.

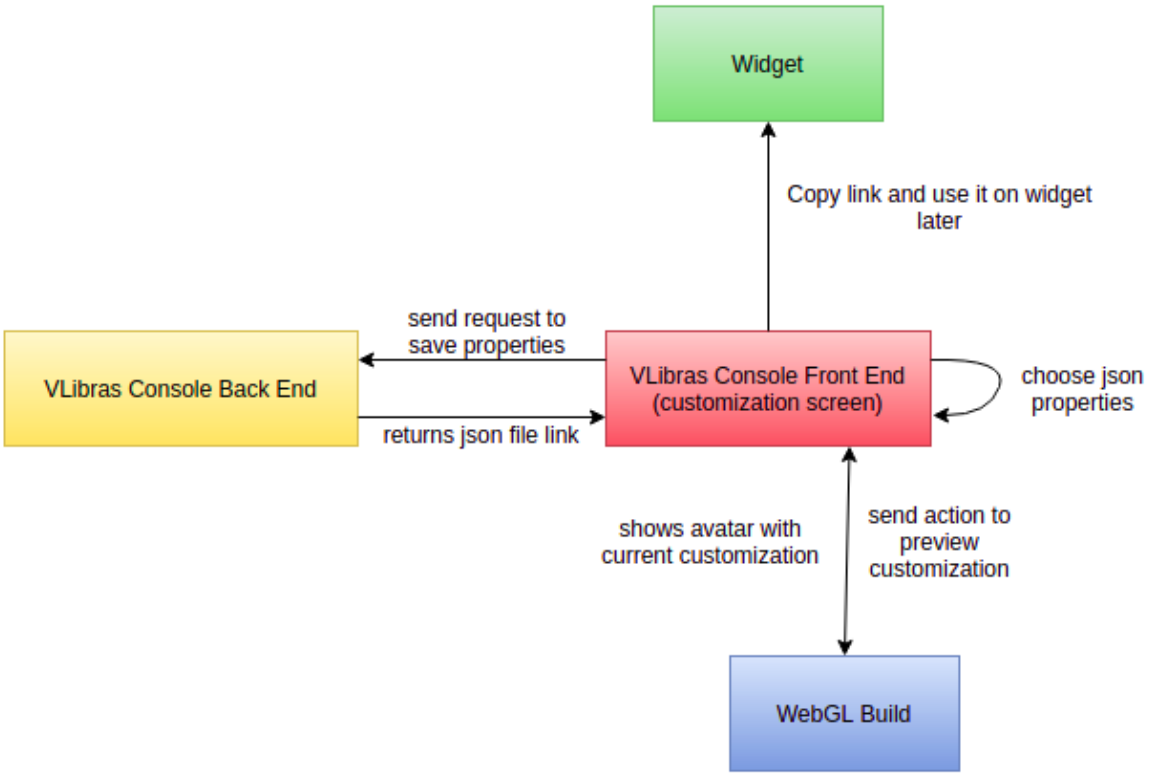

Figura 5 - Fluxo de criação de personalização no Console.

Na Figura 5, observa-se o fluxo de autenticação de usuários, que pode ser descrito a seguir:

1. O usuário escolhe as propriedades exibidas na tela, como cores, logo, posicionamento da logo, nome da personalização e descrição.
2. Para o usuário conseguir testar a personalização, clica-se no botão de **Preview**. Nele, é enviado uma ação para a build WebGL do Unity, construída exatamente para receber uma personalização e personalizar o avatar in *real time*. As propriedades escolhidas são enviadas como parâmetro e o avatar é atualizado.
3. Ao finalizar, basta apenas salvar a personalização. Com isso, O VLibras Console Back End vai salvar a personalização no banco e criar um arquivo

json num diretório específico para isso. Feito isso, a requisição para a API retorna o link do Json.

4. Para utilizá-lo, basta copiar e usar como parâmetro na instanciação do Widget.

D. Configuração de Serviços

Essa tela refere-se à configuração dos serviços do VLibras, em questões como pausa, iniciar, parar, atualizar, monitorar seu status e editar as variáveis de ambiente.

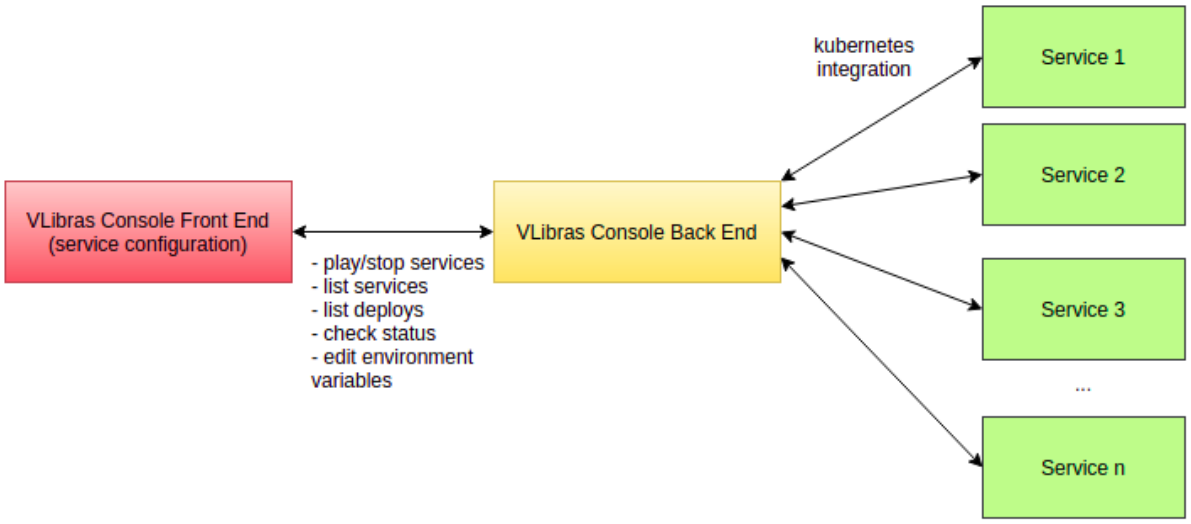

Figura 6 - Fluxo de integração com serviços

Na Figura 6, observa-se o fluxo de configuração de serviços, que pode ser descrito a seguir:

1. Temos a tela de listagem de stack de serviços. Essas stacks representam um conjunto de vários deploys. Exemplo: para a stack de serviços VLibras Dicionário, temos os deploys/serviços do VLibras Repositório e do VLibras Dicionário em si. Caso queiramos iniciar a stack de Dicionário, por exemplo, iremos iniciar todos os deploys ligados a ele.
2. Para cada stack, podemos editar as variáveis de ambiente. Porém, temos que escolher o deploy relacionado a stack, pois as variáveis de ambiente são únicas para cada serviço.
3. Para qualquer interação com os serviços, é feita uma integração do VLibras Console Backend com o Kubernetes, visto que os serviços ficam em *containers* apartados.

E. Treinamento de IA

Essa seção usa uma retaguarda diferente das outras abas. Basicamente, com "Treinamento de IA" é possível treinar, testar e publicar modelos do tradutor VLibras. Os reducers, actions e sagas utilizados nesta seção são: *Automation*, *Training* e *Corpus*.

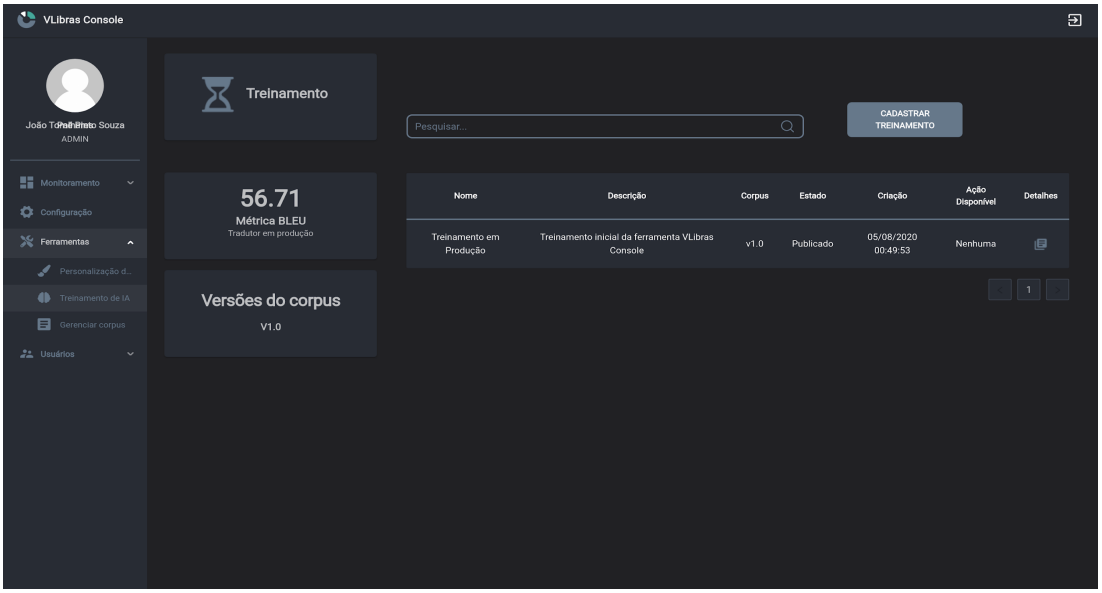

Figura 7 - Tela principal do Treinamento IA

Na figura 7, todas as requisições são feitas para uma API (rota) diferente das demais seções. No arquivo *axios/base.js* é possível verificar a instância utilizada nos *sagas* citados anteriormentes.

F. Gerenciar corpus

Essa seção apresenta apenas o *corpus* de teste do tradutor VLibras. É realizado uma simples rota (com paginação) para API do *Automation* (mesma utilizada na seção anterior). O *reducer* e o conjunto de *actions* utilizada nesta pagina pode ser encontrado em modules/Corpus.

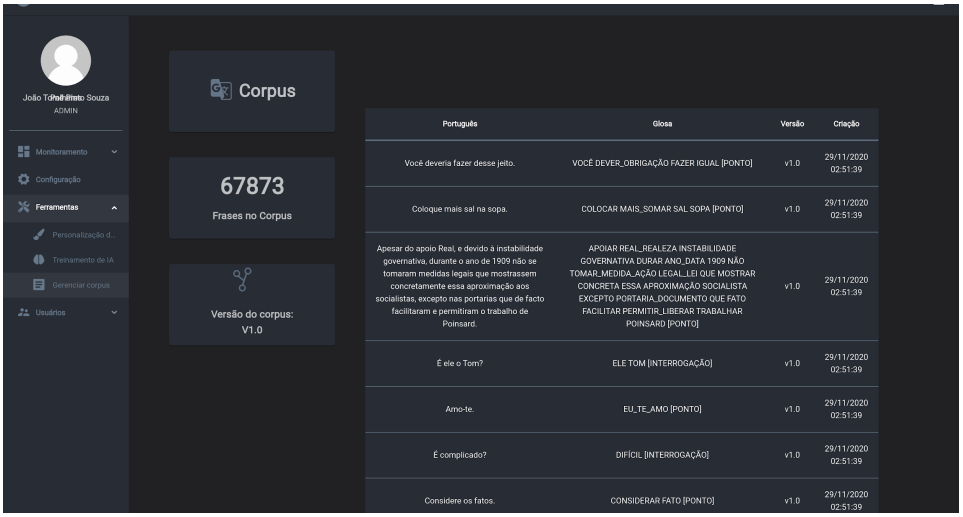

Figura 8 - Gerenciamento de Corpus

IV - Instalação

- Iniciando

Essas instruções fornecerão uma cópia do projeto instalado e funcionando em sua máquina local para fins de desenvolvimento e teste. Consulte a implantação para obter notas sobre como implantar o projeto em um sistema ativo.

- Requisitos de sistema
    - OS: Ubuntu 18.04.3 LTS (Bionic Beaver)
- Pré Requisitos

O que você precisa para instalar o software e como instalá-los.

Em primeiro lugar, siga estas três etapas para continuar a instalação:

1. Na pasta "scripts", arquivo "DevelopmentEnvironment.js", substitua ambas as linhas comentadas:

```
MONGO_CONNECTION_STRING="mongodb://127.0.0.1:27017/vlibras-console

mongo $MONGO_CONNECTION_STRING --eval "db.dropDatabase()
```

2. Em seguida, em *src/environment/environment.js*, descomente as linhas abaixo:

```
if (process.env.NODE_ENV === 'dev') {
    const dotEnvFile = path.join(__dirname, `.env.${process.env.NODE_ENV}`);
    return dotenv.config({ path: dotEnvFile }).parsed;
  }
```

3. O último pré-passo é alterar três variáveis de diretórios, de acordo com o caminho do console em seu computador. Caminho do arquivo src / environment / .env.dev e como deve ser, por exemplo:

```
TMP_FOLDER=/home/test_user/lavid/vlibras-console/vlibras-console
-backend/tmp
SRC_FOLDER=/home/test_user/lavid/vlibras-console/vlibras-console
-backend/src
SERVICES_SCHEMAS_FOLDER=/home/test_user/lavid/vlibras-console/vl
ibras-console-backend/ServicesSchemas
```

Lembre-se de preservar as três últimas "paths" dos documentos "tmp", "src" e "ServicesSchemas".

- Node.js

  Adicione o repositório NodeSource.

  ```
  curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash
  ```

  Instale Node.js.

  ```
  sudo apt-get install -y nodejs
  ```

- MongoDB

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo
apt-key add -
```

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu
bionic/mongodb-org/4.4 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.4.list
```

```
sudo apt-get update
```

```
sudo apt-get install -y mongodb-org
```

```
sudo service mongod start
```

- Instalando

  Após instalação de todos os pré-requisitos, instale o projeto executando o comando abaixo:

  ```
  npm install
  ```

  Execute o script para popular o banco de dados.

  ```
  npm run restart-db
  ```

  Execute o projeto

  ```
  npm run start:dev
  ```