

Teste para desenvolvedor backend Documento de Software

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
27/09/2020	1.0	Documentação Final do projeto	Vitor Machado Cid

Documento de Software

1. Introdução

O projeto abordado nesse documento, se trata de uma simulação de um software de gerenciamento de redes varejistas, onde os usuários podem cadastrar novas redes e visitantes, bem como registrar a entrada e saída de visitantes.

1.1 Finalidade

Esse documento tem como principal objetivo elucidar as estratégias para resolução do teste fornecido pela empresa Mob2con. Além disso, tem-se a finalidade de fornecer informações a respeito das tecnologias e bibliotecas utilizadas, instruções para utilização e informações importante, como as rotas de acesso às APIs.







1.2 Escopo

O presente projeto, se refere apenas a uma resolução de um teste para o processo seletivo da empresa Mob2con. Ele não possui uma aplicação real e não deve ser utilizado com esse propósito.

1.3 Visão Geral

Nas sessões seguintes, serão abordadas as tecnologias utilizadas para o desenvolvimento do software, instruções e requisitos de instalação, descrição das rotas para as APIs e detalhes específicos das estratégias para a resolução do teste.

2. Tecnologias utilizadas

2.1 Ruby on Rails	
	Para o desenvolvimento das APIs, utilizou-se o framework Ruby on Rails com a configuração API-only, na versão 5.2.4 e com o ruby configurado na versão 2.5.5.
2.2 PostgreSQL	
	Como banco de dados relacional, utilizou-se o PostgreSQL na versão 11.7.
2.3 Jwt / bcrypt	
	Para a autenticação do sistema utilizou-se o padrão Jwt (Json Web Token) com o auxílio das gems Jwt e bcrypt para codificação e decodificação das chaves.
2.4 Pundit	
	Devido a necessidade de restrição de acesso dependendo do tipo de usuário, utilizou-se a gem Pundit para fazer esse gerenciamento.
2.5 Rspec	
	Para o desenvolvimento de testes automatizados, utilizou-se a ferramenta Rspec, integrada com o Rails.
2.6 ElasticSearch	
	Em algumas buscas, em especial as que necessitam filtragem pelo nome do registro, utilizou-se o mecanismo elasticSearch, com auxílio das gems elasticsearch-model e elasticsearch-rails.

3. Instruções de Instalação

Para que a aplicação seja executada em uma aplicação local, é necessário que o usuário tenha as seguintes aplicações instaladas:

- Ruby na versão 2.5.5
- PostgreSQL na versão 11.7
- ElasticSearch na versão 7.9.12

Para iniciar a aplicação, siga os seguintes passos no diretório do projeto:

1. Bundle install
2. rake db:create
3. rake db:migrate
4. rake db:seed
5. Certifique-se que o serviço do ElasticSearch esteja ativo
6. Rails server

4. Rotas de acesso

4.1 Login **Post**

Rota: localhost:3000/auth/login

Parâmetros: username :string, password :string

4.2 Logout **Post**

Rota : localhost:3000/auth/logout

Autorização: Token gerado no login :string

4.3 Users Create **Post**

Rota: localhost:3000/users

Parâmetros: name :string, username :string, password :string, password confirmation: string, featured_image :imagem

4.4 Users All **Get**

Rota: localhost:3000/users

Autorização: Token gerado no login :string

4.5 Users one **Get**

Rota: localhost:3000/users/{username}

Autorização: Token gerado no login :string

4.6 Users update **Put**

Rota: localhost:3000/users/{username}

Autorização: Token gerado no login :string

Parâmetros(body): name :string, username :string, password :string, password_confirmation :string, featured_image :imagem

4.7 Users delete **Del**

Rota: localhost:3000/users/{username}

Autorização: Token gerado no login :string

4.8 Chains Create **Post**

Rota: localhost:3000/chains

Parâmetros: name :string, cnpj :string

Autorização: Token gerado no login :string

4.9 Chains All **Get**

Rota: localhost:3000/chains

Autorização: Token gerado no login :string

4.10 Chains one **Get**

Rota: localhost:3000/chains/{name}

Autorização: Token gerado no login :string

- 4.11 Chains update Put**
Rota: localhost:3000/chains/{name}
Autorização: Token gerado no login :string
Parâmetros(body): name :string, cnpj:string
- 4.12 Chains delete Del**
Rota: localhost:3000/chains/{name}
Autorização: Token gerado no login :string
- 4.13 Chains total Get**
Rota: localhost:3000/chains/total
- 4.14 Visitors Create Post**
Rota: localhost:3000/visitors
Parâmetros: name :string, chain_id :integer, featured_image :imagem
Autorização: Token gerado no login :string
- 4.15 Visitors All Get**
Rota: localhost:3000/visitors
Autorização: Token gerado no login :string
- 4.16 Visitors one Get**
Rota: localhost:3000/visitors/{name}
Autorização: Token gerado no login :string
- 4.17 Visitors update Put**
Rota: localhost:3000/visitors/{name}
Autorização: Token gerado no login :string
Parâmetros(body): name :string, chain_id:integer, featured_image :imagem
- 4.18 Visitors delete Del**
Rota: localhost:3000/visitors/{name}
Autorização: Token gerado no login :string
- 4.19 Visitors register_incidence Post**
Rota: localhost:3000/visitors/{name}/register_incidence
Autorização: Token gerado no login :string
Parâmetros(body): kind: string, datetime: datetime

5. Instruções de uso

Uso geral

Para ter acesso ao sistema, o usuário deverá seguir os seguintes passos:

- Gerar as roles user e admin. Esse passo já deve estar cumprido caso o usuário tenha seguido as instruções de instalação corretamente. Comando rake db:seed.
- Criar um usuário através da rota 4.3
- Fazer Login através da rota 4.1
- Utilizar as demais rotas com o token gerado através do login.
- Imagens podem ser enviadas através das rotas de criação e edição de usuários e visitantes.

Restrição de acesso

As seguintes rotas estão disponíveis para usuários sem identificação no sistema:

- 4.1, 4.3 e 4.13

Os usuários de rede têm acesso as rotas:

- 4.1, 4.2, 4.3, 4.13 e 4.19

Os administradores têm acesso a todas as rotas.

6. Estratégias de implementação

Roles e restrições de acesso

O projeto abordado nesse documento, aborda uma série de CRUDS, onde a principal decisão de implementação é referente a construção das roles e das restrições de acesso ao sistema. Para isso, optou-se por utilizar uma única tabela para identificação dos usuários. Essa tabela foi nomeada como users. O tipo do usuário é definido através das roles, uma tabela auxiliar que está relacionada de forma n-n com a tabela users. As roles disponíveis são user e admin. Quando um usuário é criado através da rota disponibilizada, a role “user” é relacionada ao seu registro automaticamente. Ou seja, não é possível o cadastro de um administrador sem acesso interno ao sistema.

Com as roles definidas, restringe-se o acesso às rotas diretamente nos controllers, com o auxílio da gem Pundit.

Logout

O logout do sistema no backend utilizando a arquitetura JWT não é comum, já que normalmente essa operação é feita através da remoção do token de acesso no frontend. Porém, a fim de cumprir os requisitos do teste, utilizou-se a seguinte estratégia:

Criou-se uma coluna na tabela users chamada logout :boolean. Quando o usuário é logado, essa coluna é preenchida com o atributo false e quando a operação de logout é feita, ela recebe true. Com isso, o bloqueio do token de autorização é feito mesmo que ele ainda não tenha sido expirado.

Envio de fotos

Optou-se por não criar rotas extras para essa operação, já que o envio de fotos pode ser feito na criação/edição de visitantes. Como extra, adicionou-se a possibilidade da inclusão de fotos também para a entidade usuários.

Elastic Search

Como sugerido na especificação do teste, utilizou-se a busca com elastic search em algumas rotas do sistema. Optou-se por utilizar essa estratégia apenas nos métodos show, onde o registro é buscado pelo nome e filtrado para o usuário.

Rspec

Os testes foram feitos através da ferramenta Rspec e visam garantir a integridade dos modelos da aplicação. Portanto, eles abrangem todas as validações e relacionamentos definidos em cada entidade.