Aluno: Marco Ezequiel Cedro Barros Borges
Curso: Pós Tech - Java
Instituição: FIAP

📘 Documentação da API - Spring Boot com JWT

📄 Visão Geral

Este projeto é uma API RESTful desenvolvida em Java com Spring Boot, que gerencia usuários e utiliza JWT (JSON Web Token) para autenticação.
A aplicação é protegida por um filtro customizado que valida o token em cada requisição, garantindo que apenas usuários autenticados possam acessar endpoints restritos.

🏛️ Arquitetura

O projeto adota uma arquitetura em camadas:

- **Controller:** Recebe as requisições HTTP e encaminha para a camada de serviço.
- **Service:** Contém a lógica de negócio da aplicação.
- **Model/Entity:** Representa as entidades que serão persistidas no banco de dados.
- **Repository:** Interface com o banco de dados utilizando Spring Data JPA.
- **Security:** Configuração do filtro JWT e controle de acesso via Spring Security.
- **DTOs:** Usados para transferência de dados entre cliente e servidor de forma segura.
- **Exception Handler:** Captura erros da aplicação e retorna respostas padronizadas.

🔐 Autenticação

A autenticação é feita via JWT:

1. O usuário faz login com login e senha.
2. Um token JWT é retornado.
3. Esse token deve ser incluído nas requisições protegidas, usando o header:

```
Authorization: Bearer {seu_token}
```

🔁 Endpoints da API

Método	URL	Descrição	Autenticação
POST	/api/usuarios/cadastrar	Cadastra um novo usuário	❌ Não
PUT	/api/usuarios/atualizar/{id}	Atualiza dados de um usuário	✅ Sim
DELETE	/api/usuarios/excluir/{id}	Exclui um usuário	✅ Sim
GET	/api/usuarios/{id}	Busca usuário por ID	✅ Sim
POST	/api/usuarios/login	Faz login e retorna token JWT	❌ Não
POST	/api/usuarios/trocar-senha/{id}	Troca a senha do usuário	✅ Sim

⚙️ Pré-requisitos

- Java 21
- Maven 3.8+
- Banco de dados Postgres
- IDE (STS4)

🧪 Configuração Local

1. Clonar o repositório:

```
git clone https://github.com/diowmac/tech-challenge-backend.git
```

🐳 Executando com Docker Compose

📂 Estrutura esperada

Na pasta terá os seguinte arquivos na **raiz do projeto**:

```
tech-challenge-backend/
├── app.jar              ✅ Gerado via Maven
├── Dockerfile           ✅ Define a imagem da aplicação
└── docker-compose.yml   ✅ Orquestra o container
```

📄 Dockerfile

```
Dockerfile
CopiarEditar
FROM eclipse-temurin:21-jdk-alpine
VOLUME /tmp
COPY app.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

📄 docker-compose.yml

```
yaml
CopiarEditar
services:
  app:
    build: .
    container_name: tech_challenge_app
    ports:
      - "8080:8080"
```

▶️ Passos para executar

Dentro da pasta app execute o power shell e rode o comando a baixo.

```
docker-compose up --build
```

1. Acesse a API:

   ```
   http://localhost:9095
   ```

🔐 Fluxo de Autenticação

1. O usuário envia login e senha para `/api/usuarios/login`.
2. A API valida as credenciais e retorna um token JWT.
3. As requisições seguintes devem incluir o token no cabeçalho:

```
Authorization: Bearer eyJhbGciOiJIUzI1...
```

📡 Exemplo de Requisição com Token

```
PUT /api/usuarios/atualizar/1 HTTP/1.1
Host: localhost:9095
Authorization: Bearer eyJhbGciOiJIUzI1...
Content-Type: application/json

{
  "nome": "Marco Borges",
  "email": "novo@email.com",
  "login": "marcob",
  "senha": "novaSenha123"
   ...
}
```
