

# API Fullstack Java - Gerenciamento de Usuários e Endereços

---

Este projeto é uma aplicação Fullstack completa desenvolvida em Java com Spring Boot para o backend e JSP com Bootstrap para o frontend. A aplicação implementa um CRUD completo de usuários e endereços, com sistema de autenticação e autorização via JWT, integração com a API externa ViaCEP e funcionalidades de paginação e ordenação.



## Funcionalidades Principais

### Backend (API REST)

- **CRUD Completo:** Gerenciamento de Usuários e seus respectivos Endereços.
- **Segurança:** Autenticação via JWT e autorização baseada em Roles (ADMIN, USER) com Spring Security.
- **Integração Externa:** Consulta de CEP e preenchimento automático de endereço utilizando a API do ViaCEP.
- **Paginação e Ordenação:** Listagem de usuários com suporte a paginação e ordenação por diversos campos.
- **Tratamento de Exceções:** Handler global para retornar mensagens de erro padronizadas e amigáveis.
- **Testes:** Exemplos de testes unitários (Mockito) e de integração (MockMvc).

### Frontend (JSP & Bootstrap)

- **Interface Responsiva:** Utilização do Bootstrap 5 para uma experiência de usuário agradável em diferentes dispositivos.
- **Fluxo de Autenticação:** Telas de Login e Registro que consomem a API REST.
- **Dashboard Interativo:**
  - Para **usuários comuns**, exibe e gerencia seus próprios endereços.
  - Para **administradores**, exibe uma lista de todos os usuários do sistema e permite gerenciar seus endereços.
- **CRUD de Endereços:** Interface com modal para adicionar, editar e excluir endereços.
- **Feedback Visual:** Loaders e mensagens de erro/sucesso para uma melhor experiência do usuário.



## Tecnologias Utilizadas

- **Backend:** Java 21, Spring Boot 3.2.5, Spring Security, Spring Data JPA, Hibernate, Maven.
- **Banco de Dados:** PostgreSQL.
- **Frontend:** JSP (Jakarta Server Pages), JavaScript (ES6+ com Fetch API), Bootstrap 5, Font Awesome.
- **Testes:** JUnit 5, Mockito.
- **API Externa:** ViaCEP.



## Aplicação Publicada para Teste

Uma versão de demonstração da aplicação está disponível online. Você pode acessá-la e testar as funcionalidades diretamente no navegador.

- **URL de Acesso:** <https://vpsw2882.publiccloud.com.br/frontend-angular/auth/login>

(este projeto esta em outro repositório: [https://github.com/mauandrade99/frontend\\_angular](https://github.com/mauandrade99/frontend_angular))

**Versão: JSP**

- **URL de Acesso:** <https://vpsw2882.publiccloud.com.br/fullstack/login>
- **Usuário Administrador:**
  - **Email:** `admin@admin`
  - **Senha:** `123456`

Com este usuário, você poderá testar todas as funcionalidades de administrador, como a listagem de todos os usuários.

## Guia de Instalação e Execução Local

Siga os passos abaixo para configurar e executar o projeto em seu ambiente de desenvolvimento.

### 1. Pré-requisitos

- **Java (JDK) 21** ou superior.
- **Maven 3.6** ou superior (ou utilize o Maven Wrapper incluído).
- **PostgreSQL** instalado e em execução.
- **Git** para clonar o repositório.

### 2. Clonar o Repositório

Abra um terminal e clone o projeto do GitHub:

```
git clone https://github.com/mauandrade99/fullstack.git
cd fullstackapi
```

### 3. Configurar o Banco de Dados PostgreSQL

Você precisa criar um banco de dados e um usuário para a aplicação. Conecte-se ao seu PostgreSQL (usando `psql` ou uma ferramenta como DBeaver/pgAdmin) e execute os seguintes comandos:

```
-- 1. Crie o banco de dados
CREATE DATABASE apifullstack;

-- 2. Crie um usuário (role) com senha
CREATE USER admin WITH PASSWORD 'admin';

-- 3. Dê ao usuário permissões de superusuário para criar tabelas e gerenciar o banco
ALTER USER admin WITH SUPERUSER;

-- 4. Defina o novo usuário como o dono do banco de dados
ALTER DATABASE apifullstack OWNER TO admin;
```

## 4. Configurar a Aplicação

O arquivo de configuração principal está em `src/main/resources/application.properties`. Por padrão, ele já está configurado para usar as credenciais que criamos acima.

Para rodar a aplicação em uma porta diferente (ex: 8081), adicione a seguinte linha ao arquivo:

```
server.port=8081
```

## 5. Compilar e Executar a Aplicação

A forma mais fácil de rodar o projeto é utilizando o Maven Wrapper, que já está incluído.

No terminal, na raiz do projeto, execute o comando:

```
# No Windows
mvnw spring-boot:run

# No Linux ou macOS
./mvnw spring-boot:run
```

A aplicação iniciará e estará pronta quando você vir a mensagem `Tomcat started on port(s): 8080` (ou a porta que você configurou).

## 6. Acessar a Aplicação

Após a inicialização, acesse a aplicação no seu navegador:

- **URL Padrão:** <http://localhost:8080/fullstack/login>
- **Se configurou a porta 8081:** <http://localhost:8081/fullstack/login>

## 7. Tornando um Usuário Administrador (Opcional)

Após registrar um novo usuário através da interface, você pode promovê-lo a administrador diretamente no banco de dados.

1. Descubra o `id` do usuário que você deseja promover na tabela `users`.
2. Execute o seguinte comando SQL, substituindo `X` pelo ID do usuário:

```
UPDATE users SET role = 'ROLE_ADMIN' WHERE id = X;
```

## 8. Testando a API com o Postman

Para facilitar a verificação e o teste de todos os endpoints da API, uma collection do Postman (`FullStack.postman_collection.json`) está incluída na raiz deste repositório.

Siga os passos abaixo para importar e utilizar a collection.

## 8.1. Importação da Collection

1. Abra o Postman.
2. Clique no botão **Import** no canto superior esquerdo.
3. Selecione o arquivo `FullStack.postman_collection.json` que está neste projeto.
4. Após a importação, uma nova collection chamada "FullStack" aparecerá na sua lista.

## 8.2. Configuração das Variáveis

A collection utiliza variáveis para facilitar a troca entre o ambiente local e o de produção, e para gerenciar o token de autenticação.

1. Clique na collection "FullStack" e vá para a aba **"Variables"**.
2. **baseUrl**: Existem duas variáveis `baseUrl`.
  - **Para teste local**: Deixe a variável `http://localhost:8080/fullstack` **marcada** (habilitada).
  - **Para teste no servidor publicado**: Desmarque a variável local e **marque** a variável `https://vpsw2882.publiccloud.com.br/fullstack`.
3. **Token**: Esta variável armazenará o token JWT após o login. O próximo passo explica como preenchê-la automaticamente.

## 8.3. Fluxo de Teste Recomendado

Para testar a API de forma eficaz, siga esta sequência:

### Passo 1: Registrar Usuários

- **users register admin**: Execute esta requisição **POST** para criar um usuário com permissões de **ADMIN**.
- **users register**: Execute esta requisição **POST** para criar um usuário com permissões de **USER**.

### Passo 2: Autenticar e Obter o Token

Esta é a etapa mais importante. Vamos configurar o Postman para salvar o token automaticamente.

1. Abra a requisição **authenticate**.
2. No corpo (**Body**), insira as credenciais do usuário que deseja testar (o admin ou o usuário comum que você acabou de criar).
3. Clique em **"Send"**. Após uma autenticação bem-sucedida, o token será salvo automaticamente e usado em todas as outras requisições.

### Passo 3: Testar os Endpoints Protegidos

Agora que a variável `{{Token}}` está preenchida, você pode executar as outras requisições. Elas já estão configuradas para usar o Bearer Token.

- **Endpoints de Usuário (requerem token de ADMIN)**:
  - **users list**: Testa a listagem paginada de usuários.

- **users (GET, PUT, DELETE)**: Testam as operações em um usuário específico. Lembre-se de alterar o ID na URL para um ID de usuário válido.
- **Endpoints de Endereço (requerem qualquer token válido)**:
  - **addresses list**: Lista os endereços de um usuário. Altere o ID do usuário na URL.
  - **addresses (POST, PUT, DELETE)**: Criam, atualizam e excluem endereços. Altere o ID do usuário e do endereço na URL conforme necessário.

Seguindo este guia, será possível testar todas as funcionalidades da API de forma rápida e organizada.