# Documentação da Aplicação User Management

* **Link repositório:** <https://github.com/leandropeloso/Postech---ADJ---Fase-1---Tech-Challenge.git>   
    
  [leandropeloso/Postech---ADJ---Fase-2---Tech-Challenge](https://github.com/leandropeloso/Postech---ADJ---Fase-2---Tech-Challenge)

## 1. Introdução

Esta documentação detalha a aplicação User Management, desenvolvida em Java utilizando o framework Spring Boot e o banco de dados H2.

A aplicação permite o gerenciamento de usuários com operações como criação, listagem, validação de login e exclusão. Além disso, a aplicação está configurada para ser executada em containers Docker.

## 2. Requisitos

- Docker

- Docker Compose

- Maven para build da aplicação

- Java 17

## 3. Como Rodar o Projeto

1. Certifique-se de que o Docker e o Docker Compose estão instalados em sua máquina.  
2. Navegue até o diretório raiz do projeto onde os arquivos `Dockerfile` e `docker-compose.yml` estão localizados.

3. Execute o comando abaixo para construir e iniciar o projeto:

docker-compose up –build

4. Acesse a aplicação no navegador ou ferramenta de testes (Postman) utilizando o endereço:  
 http://localhost:8080  
5. Para encerrar a aplicação, pressione Ctrl + C no terminal e execute:  
 docker-compose down

## 4. Endpoints da API

### 4.1 Criar Usuário

URL: /users

Método: POST

Descrição: Cria um novo usuário no sistema.

Exemplo de Requisição (Body):

{  
 "name": "John Doe",

"email": "john.doe@example.com",

"login": "johndoe",

"password": "password123",

"address": "123 Main St",

"userType": "CUSTOMER"

}

Resposta Esperada: 201 Created

### 4.2 Listar Todos os Usuários

URL: /users

Método: GET

Descrição: Retorna a lista de todos os usuários cadastrados.

Resposta Esperada: 200 OK

### 4.3 Validar Login

URL: /users/validate

Método: POST

Descrição: Verifica se o login e a senha fornecidos são válidos.

Exemplo de Requisição (Body):

{  
 "login": "johndoe",

"password": "password123"

}

Resposta Esperada: 200 OK: 'Usuário logado' ou 401 Unauthorized: 'Usuário ou senha incorretos'

### 4.4 Atualizar Usuário

URL: /users/{id}

Método: PUT

Descrição: Atualiza as informações de um usuário existente.

Exemplo de Requisição (Body):

{  
 "name": "Jane Doe",

"email": "jane.doe@example.com",

"login": "janedoe",

"password": "newpassword123",

"address": "456 Elm St",

"userType": "RESTAURANT\_OWNER"

}

Resposta Esperada: 200 OK

### 4.5 Deletar Usuário

URL: /users/{id}

Método: DELETE

Descrição: Remove um usuário do sistema pelo ID.

Resposta Esperada: 204 No Content

## 5. Estrutura do Banco de Dados

A tabela `app\_user` contém os seguintes campos:

- id: Identificador único do usuário (Long).

- name: Nome do usuário (String).

- email: Email do usuário (String).

- login: Login do usuário (String).

- password: Senha do usuário (String).

- address: Endereço do usuário (String).

- userType: Tipo do usuário (ENUM: CUSTOMER, RESTAURANT\_OWNER).

- lastModified: Data da última modificação (Date).

## 6. Observações

- O projeto utiliza o banco de dados H2 em memória, acessível em:

http://localhost:8080/h2-console  
 - JDBC URL: jdbc:h2:mem:userdb

- Username: sa

- Password: password

- As portas podem ser ajustadas no arquivo docker-compose.yml caso necessário.

# Arquitetura do Projeto User Management

