# Projeto: Gestão de Restaurantes

Kelly Cristina Pereira dos Santos – RM361715

Jeferson Herculano de Matos – RM365011

Gustavo Guterres Leite - RM361802

Felipe Matos de Lima - RM362291

Michael Munhoz Pinheiro Barroso - RM363592

## 1. Introdução

### Descrição do problema

O desafio propõe o desenvolvimento de um sistema para que restaurantes possam gerenciar suas operações e os clientes possam consultar informações, deixar avaliações e realizar pedidos em uma aplicação de maneira centralizada, facilitando as ações e diminuindo os custos devido a utilização de sistemas individuais.

### Objetivo do projeto

Desenvolver um backend robusto utilizando Spring Boot para gerenciar usuários e atender aos requisitos definidos. As implementações serão feitas por etapas. Nessa primeira etapa será entregue funcionalidades com foco no usuário permitindo o cadastro, autenticação, atualização, deleção e consulta.

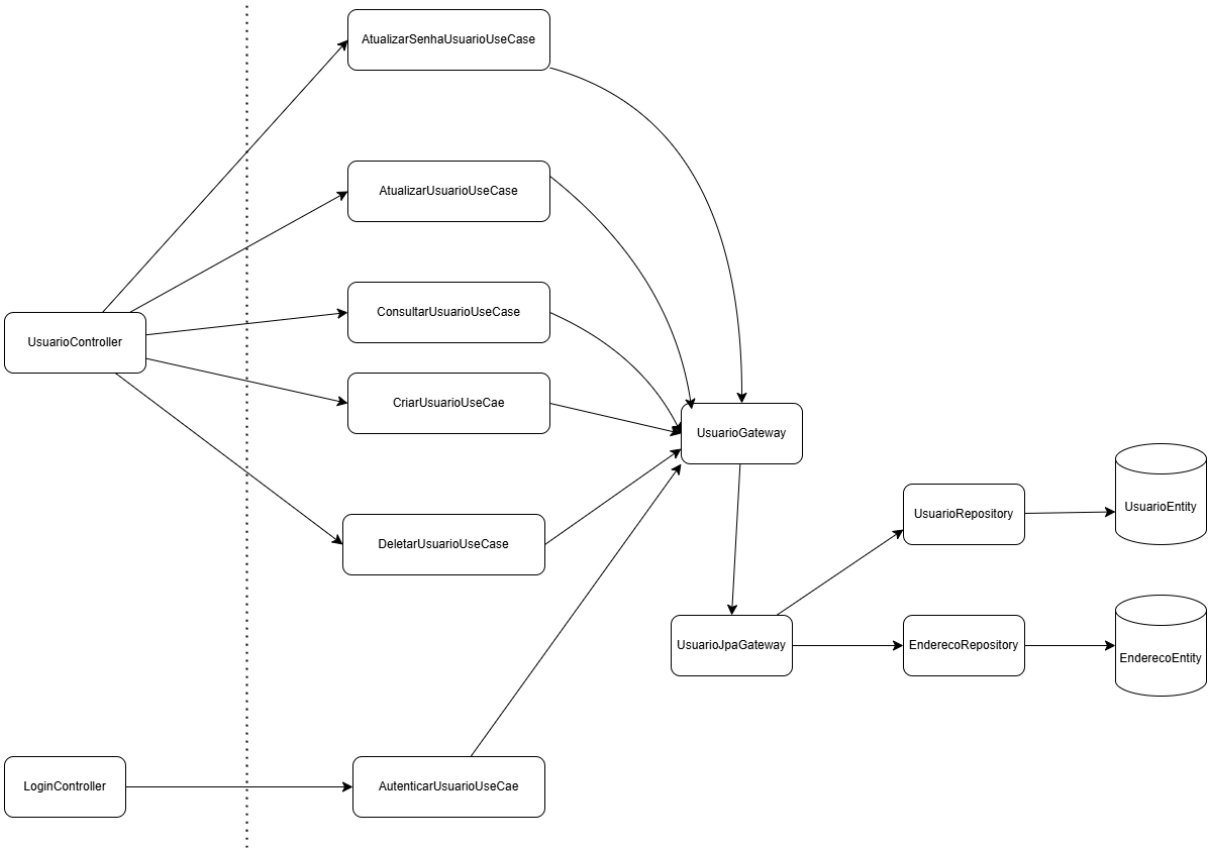
## 2. Arquitetura do Sistema

### Descrição da Arquitetura

O sistema segue uma arquitetura limpa, separando as responsabilidades nas seguintes camadas:

- **Controller:** Exposição dos endpoints REST;
- **Usecase:** Orquestração;
- **Gateway/Repository:** Acesso ao banco de dados PostgreSQL via Spring Data JPA;
- **Domain:** Entidades de domínio;
- **Exception:** Tratamento centralizado de exceções;
- **Banco de dados:** PostgreSQL;
- **Docker:** Utilizado para orquestrar a aplicação e o banco de dados.

### Diagrama da Arquitetura




## 3. Descrição dos Endpoints da API Tabela de Endpoints

| Endpoint | Método | Descrição |
|---|---|---|
| /usuarios | POST | Cadastro de novo usuário |
| /usuarios/{id} | GET | Consulta de usuário por ID |
| /usuarios/{id} | DELETE | Exclusão de usuário |
| /usuarios/{id} | PUT | Atualização de dados do usuário |
| /usuarios/{id}/senha | PATCH | Troca de senha do usuário |
| /login | POST | Autenticação de usuário |

### Exemplos de requisição e resposta

*Cenário - Inclusão de usuários preenchendo os dados corretamente:*

**POST:** localhost:8080/usuarios
Body:

```
{
	"cpf": "987654321",
	“tipoUsuario”: 0,
	"nome": "Maria Silva",
	"email": "maria.silva@email.com",
	"login": "mariasilva.br",
	"senha": "senhaSegura123!",
	"endereco":{
		"logradouro": "Avenida Brasil",
		"numero": "1234",
		"complemento": "apto 101",
		"bairro": "Centro",
		"cidade": "Rio de Janeiro",
		"estado": "RJ",
		"cep": "20000-000"
	}
}
```

Resposta: 200 OK

*Cenário - Inclusão de usuários preenchendo login já utilizado:*

**POST:** localhost:8080/usuarios
Body:

```
{
	"cpf": "987654321",
	“tipoUsuario”: 0,
	"nome": "Maria Silva",
	"email": "maria.silva@email.com",
	"login": "mariasilva.br",
	"senha": "senhaSegura123!",
	"endereco":{
		"logradouro": "Avenida Brasil",
		"numero": "1234",
		"complemento": "apto 101",
		"bairro": "Centro",
		"cidade": "Rio de Janeiro",
		"estado": "RJ",
		"cep": "20000-000"
	}
}
```

Resposta: 422 Unprocessable Entity
Body:

```
{
   "code": "usuario.usuarioJaExiste",
   "message": "Usuário já existe"
}
```


*Cenário - Consulta de usuário passando um código previamente cadastrado:*

**GET**: localhost:8080/usuarios/1

Resposta: 200 OK
Body:

```
{
   "id": 1,
   "cpf": "987654321",
   "tipoUsuario": 0,
   "nome": "Maria Silva",
   "email": "maria.silva@email.com",
   "login": "mariasilva.br",
   "senha": "senhaSegura123!",
   "dataUltimaAlteracao": "2025-06-01T20:57:26.549756",
   "endereco": {
      "id": 1,
      "logradouro": "Avenida Brasil",
      "numero": "1234",
      "complemento": "apto 101",
      "bairro": "Centro",
      "cidade": "Rio de Janeiro",
      "estado": "RJ",
      "cep": "20000-000"
   }
}
```


*Cenário - Consulta de usuário passando um código previamente cadastrado:*

**GET**: localhost:8080/usuarios/1

Resposta: 404 Not Found
Body:

```
{
   "code": "usuario.naoEncontrado",
```

```
   "message": "Usuário não encontrado!"
}
```

*Cenário - Atualizar usuário passando um código previamente cadastrado:*

**PUT**: localhost:8080/usuarios/1
Body:

```
{
        "cpf": "987654321",
        “tipoUsuario”: 1,
        "nome": "Maria Silva ATUALIZAR",
        "email": "maria.silva.atualizado@email.com",
        "login": "mariasilva.br",
        "senha": "senhaSegura1236!",
        "endereco":{
                "logradouro": "Avenida Brasil ATUALIZADO",
                "numero": "1234",
                "complemento": "apto 101",
                "bairro": "Centro",
                "cidade": "Rio de Janeiro",
                "estado": "RJ",
                "cep": "20000-000"
        }
}
```

Resposta: 204 No Content

*Cenário - Atualizar usuário passando um código previamente cadastrado:*

**PUT**: localhost:8080/usuarios/2
Body:

```
{
        "cpf": "987654321",
        “tipoUsuario”: 1,
        "nome": "Maria Silva ATUALIZAR",
        "email": "maria.silva.atualizado@email.com",
        "login": "mariasilva.br",
        "senha": "senhaSegura1236!",
        "endereco":{
                "logradouro": "Avenida Brasil ATUALIZADO",
                "numero": "1234",
                "complemento": "apto 101",
                "bairro": "Centro",
                "cidade": "Rio de Janeiro",
```

```
            "estado": "RJ",
            "cep": "20000-000"
        }
}
```

*Cenário - Atualizar usuário passando um código de usuário inexistente:*

Resposta: 404 Not Found
Body:

```
{
   "code": "usuario.naoEncontrado",
   "message": "Usuário não encontrado!"
}
```

*Cenário - Autenticar usuário passando as credenciais corretamente:*

**POST**: localhost:8080/login

```
{
        "login": "mariasilva.br",
        "senha": "senhaSegura1236!"
}
```

Resposta: 200 OK
Body:
Usuário autenticado com sucesso!

*Cenário - Autenticar usuário passando as credenciais incorretamente:*

**POST**: localhost:8080/login

```
{
        "login": "mariasilva.br",
        "senha": "senhaSegura12"
}
```

Resposta: 401 Unauthorized
Body:

```
{
   "code": "usuario.senhaIncorreta",
   "message": "A senha informada está incorreta!"
}
```

*Cenário - Atualizar senha do usuário passando os dados corretamente:*

**PATCH**: localhost:8080/usuarios/1/senha
Body:
```
{
        "novaSenha": "senhaSegura12!"
}
```

Resposta: 204 No Content

*Cenário - Deletar usuário passando um id previamente cadastrado:*

**DELETE**: localhost:8080/usuarios/1
Resultado: 204 No Content

*Cenário - Deletar usuário passando um id inexistente:*

**DELETE**: localhost:8080/usuarios/2
Resultado: 404 Not Found
Body:
```
{
   "code": "usuario.naoEncontrado",
   "message": "Usuário não encontrado!"
}
```

**4. Configuração do Projeto**

- Clone o repositório:

  ```
  git clone https://github.com/FIAP-7/TECH-CHALLENGE.git
  cd TECH-CHALLENGE
  ```

- Execute com o Docker Compose:

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

- A aplicação estará disponível em:

  http://localhost:8080

## 5. Qualidade do Código

### Boas Práticas Utilizadas

- Separação de camadas (Controller, Usecase, Gateway, Domain);
- Uso de interfaces e injeção de dependências (princípios SOLID);
- Validações centralizadas e tratamento de exceções customizadas;
- Convenções do Spring Boot e uso de Lombok;
- Organização dos DTOs e entidades;

## 6. Collections para Teste

**Link para a Collection do Postman:** /TECH-CHALLENGE/documentacao/postaman-collections/Tech-Challenge.postman_collection.json

### Descrição dos Testes Manuais

- Importe a collection no Postman.
- Execute as requisições de cadastro, login, consulta, atualização e deleção de usuário para validar os endpoints.
- Altere os dados conforme necessário para testar diferentes cenários (sugestões supracitadas).

## 7. Repositório do Código

**URL do Repositório:** https://github.com/FIAP-7/TECH-CHALLENGE