

# 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.

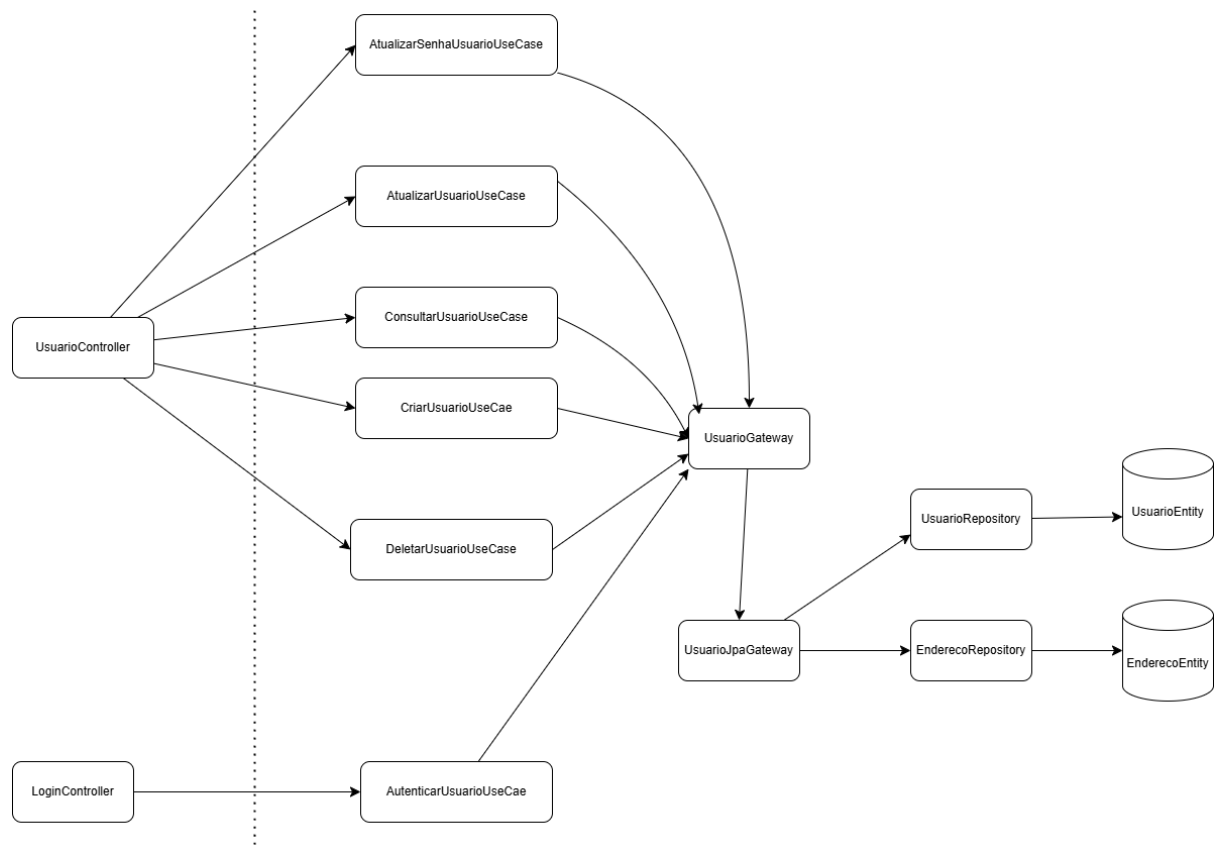
## 2. Arquitetura do Sistema

### 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": "987.654.321-77",
  "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: 201 OK

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

**POST:** localhost:8080/usuarios

Body:

```
{
  "cpf": "987.654.321-22",
  "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": "987.654.321-22",
  "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 inexistente:*

**GET**: localhost:8080/usuarios/100

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": "987.654.321-22",
  "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 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!"  
}
```

*Cenário - Deletar usuário passando um id de usuário utilizado como proprietário de um restaurante:*

**DELETE:** localhost:8080/usuarios/1

Resultado: 400 Bad Request

Body:

```
{
  "code": "usuario.usuarioUtilizado",
  "message": "Este usuário está vinculado a um ou mais restaurantes!"
}
```

Endpoint	Método	Descrição
/tipo_usuario	POST	Cadastro de novo tipo de usuário
/tipo_usuario/{id}	GET	Consulta de tipo de usuário por ID
/tipo_usuario /{id}	DELETE	Exclusão de tipo usuário
/tipo_usuario /{id}	PUT	Atualização de dados do tipo de usuário

### Exemplos de requisição e resposta

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

**POST:** localhost:8080/tipo\_usuario

Body:

```
{
  "nome": "Proprietário"
}
```

Resposta: 201 OK

*Cenário - Inclusão de tipo de usuários deixando campo obrigatório vazio:*

**POST:** localhost:8080/tipo\_usuario

Body:

```
{
  "nome": ""
}
```

Resposta: 400 Bad Request

Body

```
{
  "code": "argumentNotValid",
  "message": "nome:não deve estar em branco;"
}
```

```
}
```

*Cenário - Inclusão de tipo de usuários preenchendo um nome já utilizado:*

**POST:** localhost:8080/tipo\_usuario

Body:

```
{
  "nome": "Proprietário"
}
```

Resposta: 400 Bad Request

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioMesmoNomeExistente",
  "message": "Já existe um tipo de usuário com este nome!"
}
```

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

**GET:** localhost:8080/tipo\_usuario/1

Resposta: 200 OK

Body:

```
{
  "id": 1,
  "nome": "Proprietário"
}
```

*Cenário - Consulta de tipo de usuário passando um código não existente:*

**GET:** localhost:8080/tipo\_usuario/100

Resposta: 404 Not Found

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioNaoEncontrado",
  "message": "Tipo de Usuário não encontrado!"
}
```

*Cenário - Atualizar um tipo de usuário passando um código previamente cadastrado e com dados corretos:*

**PUT:** localhost:8080/tipo\_usuario/1

Body:

```
{
  "nome": "Dono"
}
```



}

Resposta: 204 No Content

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

**PUT:** localhost:8080/tipo\_usuario/100

Resposta: 404 Not Found

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioNaoEncontrado",
  "message": "Tipo de Usuário não encontrado!"
}
```

*Cenário - Deletar um tipo de usuário utilizado no cadastro de um usuário:*

**DELETE:** localhost:8080/tipo\_usuario/1

Resultado: 400 Bad Request

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioUtilizado",
  "message": "Este tipo de usuário está vinculado a um ou mais usuários!"
}
```

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

**DELETE:** localhost:8080/tipo\_usuario/1

Resultado: 204 No Content

*Cenário - Deletar um tipo de usuário passando um id inexistente:*

**DELETE:** localhost:8080/tipo\_usuario/200

Resultado: 404 Not Found

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioNaoEncontrado",
  "message": "Tipo de Usuário não encontrado!"
}
```

Endpoint	Método	Descrição
/restaurante	POST	Cadastro de novo restaurante

/restaurante/{id}	GET	Consulta de um restaurante por ID
/restaurante/{id}	DELETE	Exclusão de restaurante
/restaurante/{id}	PUT	Atualização de dados do restaurante

## Exemplos de requisição e resposta

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

**POST:** localhost:8080/restaurante

Body:

```
{
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1
  },
  "endereco": {
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 201 OK

*Cenário - Inclusão de restaurante deixando campos obrigatórios vazios:*

**POST:** localhost:8080/restaurante

Body:

```
{
  "nome": "",
  "tipoCozinha": "",
  "horarioFuncionamento": "",
  "usuario": {
    "id": 1
  },
}
```

```
"endereço":{
"logradouro": "Avenida Ana Costa",
"numero": "1234",
"complemento": "",
"bairro": "Gonzaga",
"cidade": "Santos",
"estado": "SP",
"cep": "11086-100"
}
}
```

Resposta: 400 Bad Request

Body

```
{
  "code": "argumentNotValid",
  "message": "tipoCozinha:não deve estar em branco;nome:não deve estar em
branco;horarioFuncionamento:não deve estar em branco;"
}
```

*Cenário - Inclusão de restaurante preenchendo um nome já utilizado:*

**POST:** localhost:8080/restaurante

Body:

```
{
"nome": "MC Donalds",
"tipoCozinha": "Fast food",
"horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1
  },
"endereço":{
"logradouro": "Avenida Ana Costa",
"numero": "1234",
"complemento": "",
"bairro": "Gonzaga",
"cidade": "Santos",
"estado": "SP",
"cep": "11086-100"
}
}
```

Resposta: 422 Unprocessable Entity

Body:

```
{
  "code": "restaurante.restauranteJaExiste",
}
```

```
    "message": "Restaurante já existe"
}
```

*Cenário - Inclusão de restaurante preenchendo um código de usuário inexistente:*

**POST:** localhost:8080/restaurante

Body:

```
{
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 100
  },
  "endereco": {
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 404 Not Found

Body

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

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

**GET:** localhost:8080/restaurante/1

Resposta: 200 OK

Body:

```
{
  "id": 1,
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1,
    "nome": "Maria Silva"
  }
}
```

```
    },
    "endereço": {
      "id": 2,
      "logradouro": "Avenida Ana Costa",
      "numero": "1234",
      "complemento": "",
      "bairro": "Gonzaga",
      "cidade": "Santos",
      "estado": "SP",
      "cep": "11086-100"
    }
  }
}
```

*Cenário - Consulta de restaurante passando um código não existente:*

**GET**: localhost:8080/restaurante/100

Resposta: 404 Not Found

Body:

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante não encontrado!"
}
```

*Cenário - Atualizar um restaurante passando um código previamente cadastrado e com dados corretos:*

**PUT**: localhost:8080/restaurante/1

Body:

```
{
  "nome": "MC Donalds Atualizado",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1
  },
  "endereço": {
    "logradouro": "Avenida Paulista",
    "numero": "222",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 204 No Content

*Cenário - Atualizar um restaurante passando um código inexistente:*

**PUT:** localhost:8080/restaurante/100

Resposta: 404 Not Found

Body:

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante não encontrado!"
}
```

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

**DELETE:** localhost:8080/restaurante/1

Resultado: 204 No Content

*Cenário - Deletar um restaurante passando um id inexistente:*

**DELETE:** localhost:8080/restaurante/100

Resultado: 404 Not Found

Body:

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante 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>