MAURICIO BORGES FLORENCIO

ARQUITETURA E DESENVOLVIMENTO JAVA

TECH CHALLENGE

O Tech Challenge é o projeto da fase que englobará os conhecimentos obtidos em todas as disciplinas da fase.Esta é uma atividade que, em princípio, deve ser desenvolvida em grupo.Importante atentar-se ao prazo de entrega, pois trata-se de uma atividade obrigatória, uma vez que valepontosna composição da nota final.

O problema

Na nossa região, um grupo de restaurantes decidiu contratar estudantes para construir um sistema de gestão para seus estabelecimentos. Essa decisão foi motivada pelo alto custo de sistemas individuais, o que levou os restaurantes a se unirem para desenvolver um sistema único e compartilhado. Esse sistema permitirá que os clientes escolham restaurantes com base na comida oferecida, em vez de se basearem na qualidade do sistema de gestão.

O objetivo é criar um sistema robusto que permita a todos os restaurantes gerenciar eficientemente suas operações, enquanto os clientes poderão consultar informações, deixar avaliações e fazer pedidos online. Devido à limitação de recursos financeiros, foi acordado que a entrega do sistema será realizada em fases, garantindo que cada etapa seja desenvolvida de forma cuidadosa e eficaz.

A divisão em fases possibilitará uma implementação gradual e controlada, permitindo ajustes e melhorias contínuas conforme o sistema for sendo utilizado e avaliado tanto pelos restaurantes quanto pelos clientes.

Objetivo

Desenvolver um **backend completo e robusto** utilizando **Spring Boot** e os princípios aprendidos na Fase 1 do curso.

O sistema deve permitir:

- Cadastro, atualização e exclusão de usuários;

- Troca de senha do usuário em endpoint separado;
- Atualização das demais informações do usuário em endpoint distinto do endpoint de senha;
- Registro da data da última alteração;
- Busca de usuários pelo nome;
- Garantia de que o e-mail cadastrado seja único;
- Validação de login obrigatória, por meio de um serviço que verifique se login e senha são válidos:
  - Não é obrigatório utilizar **Spring Security**;
  - Pode ser utilizada uma validação simples consultando os dados no banco.

A aplicação deverá ser **dockerizada**, utilizando **Docker Compose** para orquestração junto com um banco de dados **relacional (MySQL ou PostgreSQL)**.

Usuários

O sistema **deve obrigatoriamente contemplar dois tipos de usuário**:

- Dono de restaurante;
- Cliente.

Além desses, outros tipos de usuários poderão ser adicionados, caso o grupo considere necessário para enriquecer o modelo.

Campos obrigatórios para qualquer usuário:

- Nome (String);
- E-mail (String, único);
- Login (String);

- Senha (String);
- Data da última alteração (Date);
- Endereço (String ou objeto com atributos como rua, número, cidade, CEP).

Entregáveis e Critérios de Avaliação:

1. Funcionalidade

- O backend deve atender a todos os requisitos especificados;
- Os endpoints devem funcionar corretamente, com tratamento de erros adequado;
- Implementar estratégia de versionamento de API;
- Implementar o padrão ProblemDetail (RFC 7807) para padronizar as respostas de erro da aplicação;
- O sistema deve implementar **obrigatoriamente os dois tipos de usuário (dono de restaurante e cliente)**;
- Deve ser possível **buscar usuários por nome**;
- O sistema deve **garantir unicidade do e-mail** no cadastro de usuários;
- O sistema deve possuir um **serviço para validação de login do usuário (login e senha válidos)**;
- O sistema deve possuir um **endpoint separado para troca de senha**;
- O sistema deve possuir um **endpoint distinto para atualização das demais informações do usuário**.

2. Qualidade do Código

- Uso das boas práticas do **Spring Boot**, **SOLID** e **Orientação a Objetos**;

- Código organizado, testável e bem estruturado.

3. Documentação com Swagger

- Endpoints documentados com **Swagger/OpenAPI**;
- Exemplos de requisições e respostas de sucesso e erro.

4. Banco de Dados

- Utilização obrigatória de banco **relacional**;
- Bancos recomendados: **MySQL** ou **PostgreSQL**;
- O banco deve rodar em container Docker e estar configurado no docker-compose.yml.

5. Collections para Testes

- O aluno deve incluir no projeto a **coleção Postman em formato JSON**, cobrindo os principais cenários:
  - Cadastro de usuário válido;
  - Tentativa de cadastro inválido (ex.: e-mail duplicado, campos obrigatórios faltando);
  - Alteração de senha com sucesso e erro (endpoint exclusivo);
  - Atualização de dados do usuário com sucesso e erro (endpoint distinto);
  - Busca de usuários pelo nome;
  - **Validação de login (obrigatória)**.

6. Relatório Técnico (ÚNICO ENTREGÁVEL)

- O **único arquivo a ser entregue** será o **relatório em PDF**.
- Esse relatório deve conter, obrigatoriamente:
  - **Descrição detalhada da arquitetura** da aplicação;

- **Modelagem das entidades** e relacionamentos;
- **Descrição dos endpoints** disponíveis (com exemplos de uso);
- **Descrição da documentação Swagger** (prints ou trechos);
- **Descrição da coleção Postman** (com prints e exemplos);
- **Estrutura do banco de dados** (tabelas);
- **Passo a passo para executar a aplicação com Docker Compose** (variáveis de ambiente e exemplos).

7. Execução com Docker

- Arquivo docker-compose.yml para subir a aplicação e o banco.

8. Repositório de Código

- Repositório em GitHub/GitLab aberto;
- Deve conter o código-fonte, o README, a documentação Swagger e a coleção JSON do Postman;
- O **relatório em PDF deve ser submetido separadamente como entregável oficial**.

Opcional (Desafio Extra)

- Implementar autenticação utilizando **Spring Security com JWT** (não obrigatório nesta fase);
- Incluir testes unitários automatizados com **JUnit + Mockito**.

Tem alguma dúvida? Participe das nossas lives e acesse nossos grupos de estudos para falar com o(a) professor(a) que te ajudará nessa fase. Você também pode nos procurar no Discord!


PDS TECH