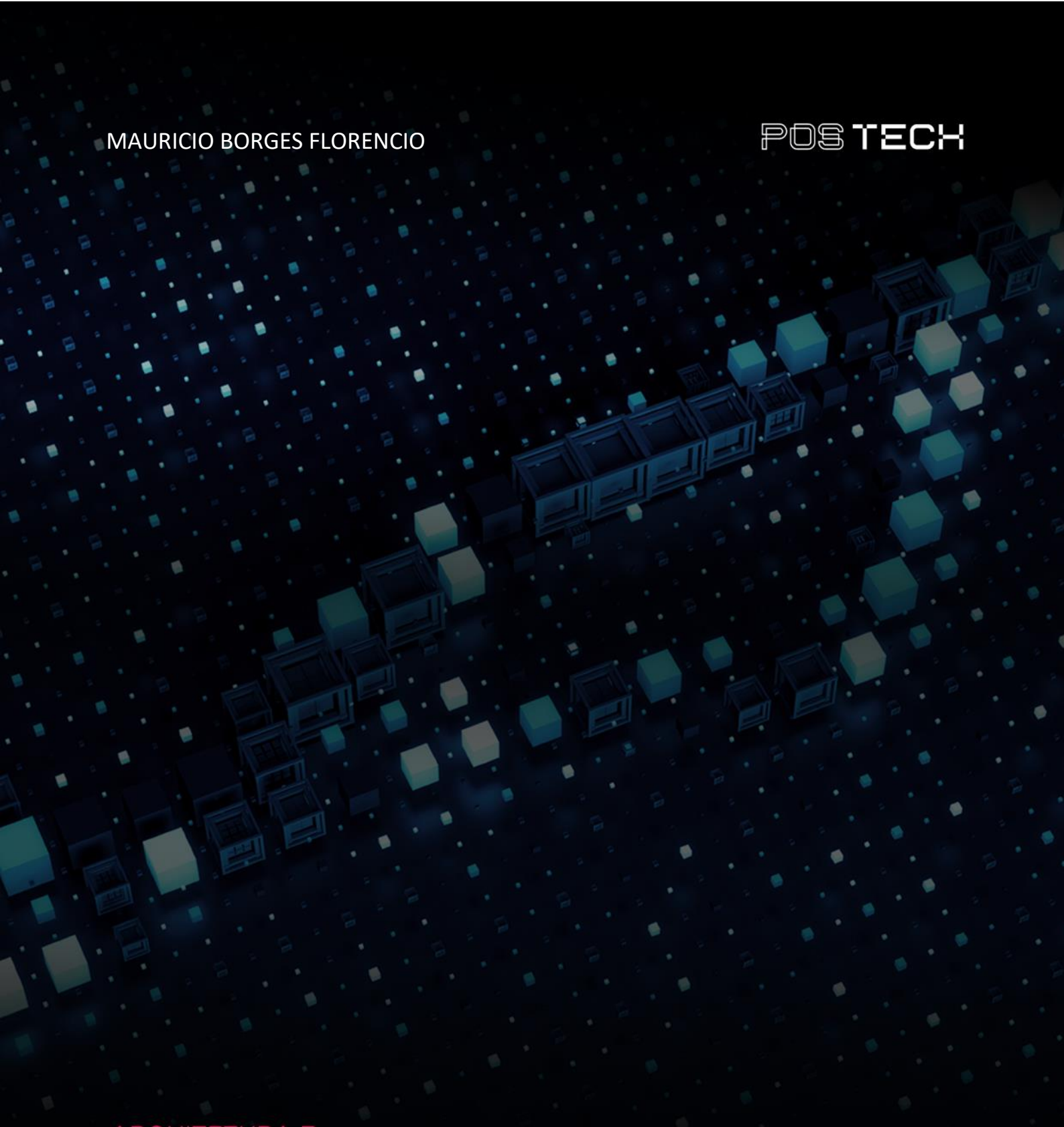
MAURICIO BORGES FLORENCIO

PÓS TECH



ARQUITETURA E
DESENVOLVIMENTO JAVA

TECH CHALLENGE

FASE 02

Tech Challenge

O Tech Challenge é o projeto 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 vale pontos na 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 pelos restaurantes e clientes.

Objetivo

Essa fase expande o sistema ao incluir a gestão dos tipos de usuários, cadastro de restaurantes e cardápios, reforçando práticas de desenvolvimento e estruturação de código limpo. Além disso, são incluídos requisitos técnicos para garantir que o sistema mantenha alta qualidade e organização, com suporte para documentação, testes automatizados e infraestrutura Docker para uma execução integrada.

Tipo de usuário

Implementar uma estrutura para distinguir entre usuários "Dono de Restaurante" e "Cliente", incluindo um CRUD para gerenciar tipos de usuário e associá-los a usuários existentes. Dependendo do banco de dados escolhido (SQL ou NoSQL), estratégias de associação específicas deverão ser consideradas. Os campos necessários para o cadastro de tipo usuário são:

Nome do Tipo.

Observação: será necessário criar uma forma de associar o usuário com o tipo de usuário.

Cadastro de restaurante

Criar um CRUD completo para o cadastro de restaurantes, incluindo campos para nome, endereço, tipo de cozinha, horário de funcionamento e dono do restaurante (associado a um usuário existente). Os campos necessários para um cadastro de restaurante são:

- Nome.
- Endereço.
- Tipo de cozinha.
- Horário de funcionamento.
- Dono do restaurante (é preciso atribuir um usuário como responsável por esse restaurante).

Cadastro dos itens do cardápio

Desenvolver um CRUD para os itens vendidos no restaurante, com detalhes como nome, descrição, preço, disponibilidade para consumo apenas no local e caminho de armazenamento da foto do prato. Os campos necessários para o cadastro de um item do cardápio são:

- Nome.
- Descrição.

- Preço.
- Disponibilidade para pedir apenas no Restaurante.
- Foto do prato (como estávamos fazendo um serviço de back-end e a foto não será utilizada, podemos apenas salvar um caminho de onde a foto estaria).

ENTREGÁVEIS E FATORES DE AVALIAÇÃO DA FASE 2

1. **Funcionalidade:**
    - Entregar as funcionalidades de Cadastro de Tipo de Usuário, Cadastro de Restaurante e Cadastro de item do Restaurante (itens que serão vendidos no cardápio).
    - Os endpoints funcionando conforme descrito.
2. **Qualidade do código:**
    - Uso adequado das práticas de desenvolvimento do Spring Boot.
    - Código devidamente organizado e documentado.
3. **Documentação do projeto:**
    - Descrição detalhada do projeto, incluindo a arquitetura, os endpoints da API e as instruções de configuração e execução.
4. **Collections para teste:**
    - Collections do Postman ou similar para testar os endpoints da API.
5. **Configuração Docker Compose:**
    - Arquivo docker-compose.yml configurado para subir a aplicação Java e o banco de dados.
6. **Repositório de código:**
    - Repositório de fontes **aberto** (GitHub, GitLab etc.) onde professores possam baixar o código-fonte do projeto.

7. **Clean Architecture:**
    - Organizar o código em camadas (Domain, Application, Infrastructure etc.) para garantir separação de responsabilidades e escalabilidade.
8. **Cobertura de teste:**
    - Testes unitários com cobertura de 80%.
    - Testes de integração para garantir que todos os componentes necessários para a aplicação estão funcionando.
9. **Vídeo:**
    - Vídeo de aproximadamente 5 minutos apresentando as funcionalidades solicitadas e o projeto executando e funcionando.

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!


POS TECH