THIAGO ADRIANO

PÓS TECH

SOFTWARE ARCHITECTURE

WELCOME TO ARQUITETURA DE MICROSSERVIÇOS

FASE 04

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 vale 90% da nota de todas as disciplinas da fase.

O problema

Há uma lanchonete de bairro que está expandindo devido seu grande sucesso. Porém, com a expansão e sem um sistema de controle de pedidos, o atendimento aos clientes pode ser caótico e confuso. Por exemplo, imagine que um cliente faça um pedido complexo, como um hambúrguer personalizado com ingredientes específicos, acompanhado de batatas fritas e uma bebida. O atendente pode anotar o pedido em um papel e entregá-lo à cozinha, mas não há garantia de que o pedido será preparado corretamente.

Sem um sistema de controle de pedidos, pode haver confusão entre os atendentes e a cozinha, resultando em atrasos na preparação e entrega dos pedidos. Os pedidos podem ser perdidos, mal interpretados ou esquecidos, levando a insatisfação dos clientes e a perda de negócios.

Em resumo, um sistema de controle de pedidos é essencial para garantir que a lanchonete possa atender os clientes de maneira eficiente e eficaz, gerenciando seus pedidos e estoques de forma adequada. Sem ele, expandir a lanchonete pode acabar, resultando em clientes insatisfeitos e impactando os negócios de forma negativa.

Para solucionar o problema, a lanchonete irá investir em um sistema de autoatendimento de fast food, que é composto por uma série de dispositivos e interfaces que permitem aos clientes selecionar e fazer pedidos sem precisar interagir com um atendente, com as seguintes funcionalidades:

Pedido: Os clientes são apresentados a uma interface de seleção na qual podem optar por se identificarem via CPF, se cadastrarem com nome, e-mail e CPF ou

não se identificar, podendo montar o combo na seguinte sequência, sendo todas elas opcionais:

1. Lanche;
2. Acompanhamento;
3. Bebida.

Em cada etapa é exibido o nome, descrição e preço de cada produto.

Pagamento: O sistema deverá possuir uma opção de pagamento integrada, no caso, para MVP, a forma de pagamento oferecida será via QRCode do Mercado Pago.

Acompanhamento: Uma vez que o pedido é confirmado e pago, ele é enviado para a cozinha para ser preparado. Simultaneamente deve aparecer em um monitor para o cliente acompanhar o progresso do seu pedido com as seguintes etapas:

- Recebido;
- Em preparação;
- Pronto;
- Finalizado.

Entrega: Quando o pedido estiver pronto, o sistema deverá notificar o cliente que ele está pronto para retirada. Ao ser retirado, o pedido deve ser atualizado para o status finalizado.

Além das etapas do cliente, o estabelecimento precisa de um acesso administrativo:

Gerenciar clientes: Com a identificação dos clientes o estabelecimento pode trabalhar em campanhas promocionais.

Gerenciar produtos e categorias: Os produtos dispostos para escolha do cliente serão gerenciados pelo estabelecimento, definindo nome, categoria, preço, descrição e imagens. Para esse sistema, teremos categorias fixas:

- Lanche;
- Acompanhamento;
- Bebida;

- Sobremesa.

Acompanhamento de pedidos: Deve ser possível acompanhar os pedidos em andamento e tempo de espera de cada pedido

As informações dispostas no sistema de pedidos precisarão ser gerenciadas pelo estabelecimento através de um painel administrativo.

Entregáveis FASE 4:

Nesta fase vamos continuar trabalhando no projeto existente, e dando continuidade ao desenvolvimento do software para a lanchonete, teremos as seguintes melhorias e alterações:

1. Refatore o projeto, separe-o em ao menos 3 (três) microsserviços. Alguns exemplos de serviços:
    a. **Pedido:** responsável por operacionalizar o processo de pedidos, registrando os pedidos, retornando as informações necessárias para montar um pedido, listando os pedidos registrados e em processo de produção (visão de cliente).
    b. **Pagamento:** responsável por operacionalizar a cobrança de um pedido, registrando a solicitação de pagamento, recebendo o retorno do processador de pagamento e atualizando o status do pedido.
    c. **Produção:** responsável por operacionalizar o processo de produção do pedido, acompanhando a fila de pedidos (visão da cozinha), atualização de status de cada passo do pedido.

Lembre-se de trabalhar com bancos de dados para cada aplicação. Use ao menos um banco de dados NoSQL e um SQL **(obrigatório)**; caso queira fazer com mais bancos, você pode decidir quais utilizar.

Os serviços devem se comunicar entre si, seja por chamada direta, mensagens em fila ou estratégias semelhantes. Um serviço não pode acessar o banco de dados de outro serviço, porque viola as regras de implementação de microsserviços.

2. Ao refatorar, os microsserviços devem conter testes unitários.

    a. Ao menos um dos caminhos de teste deve implementar BDD.
    b. Em todos os projetos, a cobertura de teste deve ser de 80%.

3. Seus repositórios devem ser separados para cada aplicação e devem respeitar as seguintes regras:

a. As branchs main/master devem ser protegidas, não permitindo commits diretamente.
b. Pull Request para branch main/master, que deve validar o build da aplicação, e a qualidade de código via sonarqube ou qualquer outro serviço semelhante, cobrindo 70% de coverage no mínimo.
c. No Merge, o deploy de todos seus microsserviços devem ser executados, isso significa que todos os repositórios devem estar com CI/CD criados, e executados corretamente.

Para a avaliação desta entrega, são esperados os seguintes artefatos:

1. Um vídeo demonstrando:

    a. O funcionamento da aplicação;
    b. As atualizações efetuadas na arquitetura;
    c. O processo de deploy de todos os microsserviços, mostrando o processo de teste funcionando como esperado;
    d. Não é necessário mostrar as pipelines em execução, somente os checks (verdes) indicando que todos os passos foram executados corretamente.

2. Os links para **todos** os repositórios usados com o código-fonte e a evidência de cobertura de testes. Essa evidência pode ser o link do serviço usado para testes ou a cópia da tela, mas deve estar no README do projeto, e a evidência deve ser por microsserviço.

    1. Para validação do código-fonte da aplicação, precisamos que vocês deixem o projeto como privado e adicionem o usuário **soat-architecture** para que possamos ver o projeto.

A entrega deve ser feita em um arquivo com o nome e a identificação no Discord de todos os(as) alunos e alunas do grupo e os links para os itens acima.


PDS TECH