Lanchonete do Bairro

versão 1.0.0-POC

Daniel Maria da Silva

Sumário

1. Introdução 2

Projeto2

* 1. Linguagem Ubíqua 2
  2. Fluxo de Funcionalidades (Representação Pictográfica) 3

2.2.1 *Preparação e entrega do pedido* 3

*2.2.2 Novo fluxo* 4

*2.2.3 Realização do pedido e seu pagamento* 4

*2.2.4 Novo fluxo* 4

2.3 Arquitetura hexagonal 5

2.4 OpenAPI5

2.5 Verificação de integridade (Health Check)6

1. Introdução

A Lanchonete do Bairro é um estabelecimento acolhedor e familiar localizado no coração da comunidade. Com um ambiente descontraído e amigável, é o lugar perfeito para os moradores e visitantes desfrutarem de deliciosas opções de lanches, acompanhamentos, bebidas e sobremesas.

1. Projeto

O projeto Lanchonete do Bairro tem como objetivo desenvolver um sistema de gerenciamento para uma lanchonete familiar. O sistema será desenvolvido utilizando a arquitetura hexagonal (*hexagonal architecture*) e seguirá os princípios do *Domain-Driven Design* (DDD).

Através desse sistema, os clientes terão acesso a uma interface intuitiva onde poderão realizar pedidos e efetuar pagamentos de forma prática. Será possível montar o combo de lanches com opções de lanche, acompanhamento e bebida. O sistema também permitirá que os clientes acompanhem o progresso do seu pedido, desde a confirmação até a entrega ou retirada.

Além das funcionalidades voltadas para os clientes, o sistema contará com um painel administrativo que permitirá o gerenciamento de clientes, produtos e categorias. O estabelecimento poderá cadastrar novos clientes, gerenciar campanhas promocionais, adicionar, editar e remover produtos, definindo nome, categoria, preço, descrição e imagens. Também será possível acompanhar os pedidos em andamento e verificar o tempo de espera de cada pedido.

O projeto será desenvolvido utilizando a linguagem de programação Java 16 e o framework Spring Boot. Será integrado ao banco de dados MariaDB para armazenar as informações dos clientes, produtos e pedidos. Além disso, o projeto incluirá a documentação do sistema utilizando a linguagem ubíqua (DDD) e a implementação de *endpoints* RESTful para as funcionalidades descritas.

Com o projeto Lanchonete do Bairro, pretendemos criar um sistema eficiente e intuitivo que facilite o processo de pedido e pagamento, proporcionando uma experiência agradável aos clientes e auxiliando o estabelecimento.

* 1. Linguagem Ubíqua

1. Lanchonete: Estabelecimento que oferece uma variedade de alimentos e bebidas.
2. Cliente: Pessoa que faz um pedido na lanchonete.
3. Pedido: Solicitação de alimentos e/ou bebidas feita por um cliente.
4. Produto: Produtos que compõem um pedido
5. Acompanhamento: Opção adicional selecionada pelo cliente para acompanhar seu lanche.
6. Lanche: Alimento principal do pedido, como hamburguês, pizza, etc.
7. Acompanhamento: Alimento secundário do pedido, como batata frita, salada, etc.
8. Bebida: Opção de bebida selecionada pelo cliente.
9. Sobremesa: Complemento da alimentação.
10. Pagamento: Processo de efetuar o pagamento do pedido.
11. Sistema de Pedido: Tela ou dispositivo no estabelecimento que mostra o status do pedido em diferentes etapas para os clientes e para a equipe da cozinha.
12. Equipe da cozinha: Funcionários responsáveis por preparar os pedidos.
13. Status do Pedido: Indicador do progresso do pedido, dividido em:

a. Recebido: Pedido registrado e aguardando preparação.

b. Em preparação: Pedido em processo de preparação na cozinha.

c. Pronto: Pedido concluído e pronto para retirada.

d. Finalizado: Pedido entregue e finalizado.

1. Entrega: Processo de notificar o cliente quando o pedido está pronto para retirada.
2. Acompanhamento de Pedidos: Funcionalidade que permite acompanhar o status dos pedidos em andamento e estimar o tempo de espera.
3. Balcão de recolha: Local físico onde os pedidos são entregas quando finalizado para a recolha pelo cliente.
   1. Fluxo de Funcionalidades (Representação Pictográfica)

Alguns dos fluxos que este Sistema se propõe a resolver são os de realização do pedido e seu pagamento e a preparação e entrega do pedido. Os fluxos foram mapeados como são feitos hoje, sem a implementação do sistema, e como se visualiza após a sua implementação.

* + 1. *Preparação e entrega do pedido*

A picture containing text, font, white, screenshot

Description automatically generated

Fluxo 1. Fluxo antigo onde o pedido é recebido pela equipe de cozinha por uma anotação em papel e prepara todos os produtos, sem informar seu status a ninguém.

* + 1. *Novo fluxo de preparação e entrega do pedido*

A picture containing line, receipt, diagram, text

Description automatically generated

Fluxo 2. Novo fluxo proposto onde a equipe de cozinha pode visualizar os pedidos em um sistema de pedidos e à medida que os produtos são feitos, seu status é alterado e o cliente pode acompanhar este status.

* + 1. *Realização do pedido e seu pagamento*

A picture containing line, diagram, screenshot, circle

Description automatically generated

Fluxo 3. Fluxo antigo onde o cliente depende de um funcionário para realizar o pedido e pagamento. Além disso o pedido só é enviado para a equipe da cozinha por uma ação do funcionário.

* + 1. *Novo fluxo de realização do pedido e seu pagamento*

A picture containing screenshot, diagram, line, font

Description automatically generated

Fluxo 4. Fluxo atualizado com nova proposta. Cliente pode interagir diretamente com a interface de seleção de produtos e realizar ele mesmo o pagamento. Além disso, o pedido vai diretamente para o sistema de pedidos assim que o pagamento é realizado.

* 1. Arquitetura hexagonal

A arquitetura hexagonal foi utilizada visto que permite criar sistemas flexíveis, testáveis e independentes de tecnologias externas.

A principal razão para utilizar a arquitetura hexagonal foi sua capacidade de isolar o núcleo do sistema, onde estão as regras de negócio essenciais, das dependências externas, como bancos de dados e frameworks. Essa separação possibilita o desenvolvimento e teste independentes do núcleo, facilitando a manutenção do código.

Além disso, também proporciona flexibilidade e adaptabilidade ao sistema. Ao definir as interfaces externas como portas de entrada e saída, é possível substituir ou adicionar adaptadores para conectar-se a diferentes tecnologias ou serviços externos, sem afetar o núcleo da aplicação. Isso torna o sistema mais preparado para lidar com mudanças de requisitos e tecnologias.

* 1. OpenAPI

O OpenAPI é uma especificação que define um padrão para descrever APIs RESTful de maneira padronizada, independente de linguagem e de plataforma. Com o OpenAPI é possível documentar de forma estruturada todos os *endpoints*, parâmetros, respostas e demais detalhes de uma API, facilitando a compreensão e a utilização por parte de desenvolvedores e consumidores. Já o Swagger é uma ferramenta que utiliza a especificação OpenAPI para gerar automaticamente a documentação interativa da API. Através do Swagger UI é possível acessar a documentação da API em um formato amigável, que permite explorar os *endpoints*, testar as requisições e visualizar as respostas esperadas. O uso do OpenAPI e do Swagger traz diversos benefícios, como a padronização da documentação, a facilidade de integração entre sistemas e a promoção de uma documentação atualizada e de qualidade. Para acessar a documentação gerada pelo Swagger, basta acessar a URL específica fornecida pelo projeto que é /swagger-ui/index.html#/

* 1. Verificação de integridade (Health Check)

O Health Check é um mecanismo que permite verificar o estado de saúde de um sistema ou serviço em tempo real. É utilizado para monitorar e validar se o sistema está funcionando corretamente, fornecendo informações sobre sua disponibilidade e desempenho.

No contexto deste projeto, o Health Check foi implementado para garantir a integridade e a estabilidade do sistema, permitindo que os administradores ou ferramentas de monitoramento verifiquem a saúde da aplicação de forma automatizada. O projeto utiliza o Health Check para fornecer um endpoint específico, '/actuator/health', que retorna informações sobre o estado do sistema, incluindo o status atual. Isso é útil para identificar problemas e tomar ações corretivas, garantindo a disponibilidade e a qualidade do serviço. Ao utilizar o Health Check, o projeto busca facilitar o monitoramento e o diagnóstico de problemas, contribuindo para uma melhor experiência do usuário e um ambiente mais confiável.

Alguns endpoints úteis:

/actuator/health: Informações de saúde da aplicação, incluindo o status do banco de dados.

/actuator/info: Informações gerais sobre a aplicação, incluindo informações sobre o banco de dados.

/actuator/metrics: Métricas da aplicação, como tempo de execução de consultas no banco de dados.