Pizza Pizza Pizza

Documento de Arquitetura de Software

Versão 1.2

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 16/06/2021 | 1.0 | Versão da Documentação de Arquitetura de Software | Daniel Goor Junior |
| 17/06/2021 | 1.1 | Versão após finalização da implementação | Daniel Goor Junior |
| 18/06/2021 | 1.2 | Versão com atualização de seção | Daniel Goor Junior |
|  |  |  |  |

Índice Analítico

[1. Introdução 4](#_Toc74870934)

[1.1 Finalidade 4](#_Toc74870935)

[1.2 Escopo 4](#_Toc74870936)

[1.3 Definições, Acrônimos e Abreviações 4](#_Toc74870937)

[1.4 Visão Geral 4](#_Toc74870938)

[2. Representação Arquitetural 4](#_Toc74870939)

[3. Metas e Restrições da Arquitetura 5](#_Toc74870940)

[4. Visão de Casos de Uso 5](#_Toc74870941)

[4.1 Realizações de Casos de Uso 5](#_Toc74870942)

[*4.1.1* *Criar Novo Item no Cardápio* 5](#_Toc74870943)

[*4.1.2* *Ler Item do Cardápio.* 6](#_Toc74870944)

[*4.1.3* *Atualizar Item do Cardápio.* 6](#_Toc74870945)

[*4.1.4* *Remover Item do Cardápio* 6](#_Toc74870946)

[*4.1.5* *Emitir Relatório* 6](#_Toc74870947)

[5. Visão Lógica 7](#_Toc74870948)

[5.1 Classe Cardápio 7](#_Toc74870949)

[5.2 Classe Gerenciador de Pedido 7](#_Toc74870950)

[5.3 Classe Ingrediente 7](#_Toc74870951)

[5.4 Classe Pedido 7](#_Toc74870952)

[5.5 Classe Pizza 8](#_Toc74870953)

[5.6 Classe Relatorio 8](#_Toc74870954)

[6. Visão de Processos 8](#_Toc74870955)

[7. Visão de Implantação 9](#_Toc74870956)

[8. Visão da Implementação 9](#_Toc74870957)

[8.1 Visão Geral 9](#_Toc74870958)

[8.2 Camadas 9](#_Toc74870959)

[9. Tamanho e Desempenho 9](#_Toc74870960)

[10. Qualidade 10](#_Toc74870961)

Documento de Arquitetura de Software

# Introdução

## Finalidade

Este documento tem a finalidade de evidenciar as escolhas de arquitetura realizadas e justificar os motivos pelos quais tais escolhas foram feitas.

Ao mesmo tempo, este documento justifica as restrições feitas durante o desenvolvimento da aplicação baseado nas limitações impostas pelas arquiteturas escolhidas, bem como nos casos de uso definidos para esta solução.

Este documento é destinado aos desenvolvedores que futuramente poderão realizar manutenção no sistema, ampliando seu escopo de atuação, bem como corrigindo desvios de funcionamento do software.

## Escopo

Este documento tem por finalidade garantir que todos os aspectos necessários à manutenção do sistema e seu entendimento estarão presentes. Este documento é afetado pelos documentos de caso de uso e pelo documento de especificação de software. Isto porque o documento de casos de uso especifica o contexto de atuação do software para resolver uma necessidade do cliente. A partir disto, foram feitas análises para identificar as tecnologias que se adequariam a este desenvolvimento.

Com isto, uma vez definido a tecnologia e os requisitos do software, foi registrado no documento de especificação de software quais os requisitos do sistema a serem trabalhados. Dessa forma, pode-se delimitar os critérios de sucesso para a aplicação. Tais critérios de sucesso foram definidos, de acordo com cada usuário do sistema, no documento visão de negócio.

## Definições, Acrônimos e Abreviações

As definições de termos e acrônimos utilizados não somente neste documento, como em todos os documentos parte deste desenvolvimento, podem ser encontrados no documento Glossário do Projeto.

## Visão Geral

Este documento especifica as decisões tomadas no desenvolvimento desta aplicação. Para isto, tal documento faz referência aos documentos de casos de uso, visão de negócio e glossário de projeto, bem como ao documento de Especificação de Requisitos de Software.

Este documento especifica e justifica a arquitetura escolhida, relacionando com os casos de uso documentados e com o objetivo final a ser alcançado. Ao mesmo tempo, revisa os processos de negócio definidos no documento de Visão de Negócio como forma a relacionar o processo de negócio com as escolhas de desenvolvimento tomadas.

# Representação Arquitetural

Do ponto de vista de casos de uso, ficou claro que haviam estruturas que fariam parte do sistema que poderiam ser reutilizadas. Para isto, a abordagem orientada a objetos mostrou-se a mais indicada para este projeto.

Dessa forma, podemos definir as classes que fariam parte do sistema e, com isto, definir como estas classes se relacionariam no sistema. A partir deste ponto, dentre as linguagens orientadas a objetos, escolheu-se a linhagem JAVA.

A partir da perspectiva lógica, Uma vez que as classes estavam definidas, foram estabelecidos os atributos e métodos que cada uma das classes deveria apresentar para o funcionamento de todo o sistema contendo suas funcionalidades.

Neste sentido, foi definido a classe Ingrediente, a qual deveria conter as informações de nome do ingrediente, quantidade calórica por porção e custo da porção. Ao conter essas informações, a classe Ingrediente pôde ser reutilizada em outros casos de uso dentro do sistema, tais como o caso de uso Pedido e Relatório.

# Metas e Restrições da Arquitetura

Como o principal objetivo deste software é garantir que o tempo de atendimento seja reduzido, restringiu-se em produzir uma solução para este fim. Desse modo, a integração da aplicação com os meios de pagamento não é o objetivo deste trabalho. Por isto, a conexão com gateways não é uma requisição do sistema, bem como desde o início foi uma restrição estabelecida pelo cliente. Isto porque o cliente é consiste em um empreendimento em início de operação, e pagar as taxas para uso destes gateways afetaria a margem da pizzaria.

# Visão de Casos de Uso

## Realizações de Casos de Uso

## *Criar Novo Item no Cardápio*

Esta atividade do caso de uso diz respeito a criar um novo item na lista denominada cardápio.

Para isto, o usuário gerente deverá cadastrar as seguintes informações: nome da pizza, código da pizza, ingrediente, caloria por porção e preço do ingrediente por porção.

Inicialmente, o usuário gerente deve selecionar a opção de incluir um novo item no cardápio.

Ao fazer isto, habilita-se um campo onde o usuário informa o código do novo item.

A seguir, um novo campo permite ao usuário preencher com o nome da pizza.

A próxima etapa consiste em incluir os ingredientes que compõem a pizza. Para isto, a adição é feita de maneira cíclica até que todos os ingredientes e suas informações sejam registradas.

Então, pensando na adição de um único ingrediente, inicia-se pela adição do nome do ingrediente, seguida pela quantidade utilizada deste ingrediente por pizza, a quantidade de caloria adicionada devido a este ingrediente e finalmente o custo da porção referente a adição deste ingrediente na pizza. Este passo se repete até que todos os ingredientes sejam adicionados.

Ao final deste processo, tem-se uma nova pizza adicionada ao cardápio, a qual contém a informação de código, nome da pizza, quantidade de calorias totais e preço final da pizza.

## *Ler Item do Cardápio.*

Ao realizar a venda por telefone, o atendente deve ser capaz de acessar rapidamente o item do cardápio e, com isso, tornar o processo de venda mais rápido.

Dessa forma, ao iniciar o processo, o atendente recebe a solicitação do cliente quanto ao sabor da pizza desejado. A partir desse momento, o atendente informa o código do sabor da pizza no sistema e clica em adicionar à lista de compras.

Dessa forma, o resultado final é que a pizza desejada é identificada no cardápio e adicionada ao carrinho de compras do cliente para o faturamento. Dessa maneira, o processo de atendimento do cliente torna-se muito mais rápido.

## *Atualizar Item do Cardápio.*

De uma maneira geral, os itens do cardápio podem sofrer alterações. Seja por mudanças de fornecedores, ou por alteração na receita da pizza.

Por isto, o gerente da pizzaria deve ser capaz de atualizar tais informações.

Para isto, o gerente deve inicialmente identificar qual a receita da pizza que ele deseja modificar.

Uma vez feito isto, ele informa ao sistema o código da pizza.

Em seguida ele seleciona o campo ingredientes, o qual contém todos os constituintes da formulação da pizza. Neste campo, ele faz a seleção do ingrediente que se deseja atualizar. Uma vez feito isto, ele deve alterar as informações de nome do ingrediente, quantidade em gramas utilizada por porção, calorias relativas ao ingrediente por porção além de atualizar o preço deste novo ingrediente utilizado na pizza.

O resultado final é a conservação do registro do nome da pizza, porém com a alteração de ingrediente realizada.

## *Remover Item do Cardápio*

Os itens de um cardápio podem ser removidos uma vez que o relatório de vendas mostre que determinado item não traz faturamento significativo à pizzaria. Isto é feito, pois manter itens sem grande volume de venda no cardápio gera grande complexidade na operação do negócio sem trazer retorno, ao mesmo tempo que podem propiciar o acúmulo e vencimento de ingredientes no estoque.

Assim, para remover o item/sabor da pizza do cardápio, o usuário gerente deve selecionar o código da pizza a ser removido. Uma vez feito isto, o usuário gerente deve selecionar a opção remover. Ao fazer isto, todas as informações referentes ao item em questão são eliminadas.

Dessa forma, como resultado final, o tem-se no cardápio a eliminação de um sabor de pizza.

## *Emitir Relatório*

Neste caso de uso, o usuário gerente poderá ter uma visão do desempenho do seu negócio dentro de uma janela de tempo escolhida.

Inicialmente o sistema deve estar ligado e funcionando.

O gerente seleciona a opção de relatórios.

Nesta nova janela, o gerente deve selecionar o intervalo de tempo no qual ele/ela deseja fazer a avaliação do desempenho das vendas.

Para isso, inicialmente, deve-se selecionar a data inicial, ou seja, a data que marcará o início do período no qual o relatório será mostrado.

Em seguida, deve-se selecionar a data final desejada. Desse modo, a janela de tempo está estabelecida com as definições de data inicial e data final para o período de relatório.

Cumprida esta etapa, o gerente deverá clicar no botão relatório.

Dessa forma, as seguintes informações serão mostradas no relatório para o período determinado: quantidade de pedidos, total de pizzas vendidas, somatório de faturamento em reais, somatório dos valores relativos as taxas de entrega em reais, somatório do custo das pizzas vendidas em reais, lucro obtido em reais, as pizzas vendidas e suas respectivas quantidades.

# Visão Lógica

## Classe Cardápio

Tal classe faz uso das classes Pizza e Ingredientes, as quais são usadas na forma de variáveis como ArrayList.

Como funcionalidades, esta classe apresenta os métodos removePizza, alterarPizza, consultarPizza(por código ou por nome), addIngrediente, removeIngrediente, alterarIngrediente e consultarIngrediente.

## Classe Gerenciador de Pedido

Esta classe tem o objetivo de emitir o relatório final de venda do produto ao cliente. Para isto, tal classe apresenta relação com a classe pedido, uma vez que a classe pedido contem as informações relativas a classe pizza.

A classe pedido tem, portanto, a variável pedido que é do tipo Pedido e consiste em um ArrayList.

Os métodos da classe GerenciadorDePedido são: addPedido, removePedido, alterarPedido, consultarPedido, getTodosPedido e consultarUltimoPedido.

## Classe Ingrediente

Esta classe tem o objetivo único de armazenar as informações do ingrediente usado na pizza. Para isto, ela usa as seguintes variáveis para armazenar as informações: nome, caloriasPorPorcao, precoPorPorcao e quantidadePorPorcao.

Os métodos usados consistem em getters e setters dedicados a estabelecer e alterar os valores das variáveis usadas dentro da classe.

Tais métodos são: setNome, setPrecoPorPorcao, setCaloriasPorPorcao, setQuantidadePorPorcao, getNome, getCaloriaPorPorcao, getPrecoPorPorcao e getQuantidadePorPorcao.

## Classe Pedido

Esta classe tem o objetivo único de armazenar as informações do pedido feito pelo cliente. Para isto, ela usa as variáveis nome (do cliente), endereço (do cliente) e telefone(do cliente). Ademais ocorre o uso das seguintes variáveis para armazenar as informações do pedido em questão: tipoDePagamento, precoPizzas, taxaDeEntrega, precoTotal, data, além da variável do tipo Pizza que armazena os dados da pizza pedida.

Os métodos utilizados consistem em getters e setters além de métodos que visam mostrar as pizzas cadastradas até o momento no pedido.

## Classe Pizza

Esta classe tem o objetivo único de armazenar as informações da pizza pedida. Para isto, ela faz uso das seguintes variáveis: nome, codigo, preco, calorias e da variável do tipo Ingredientes para armazenar um ArrayList de valores referentes aos ingredientes usados na pizza em questão.

Os métodos desta classe consistem basicamente em getters e settters.

## Classe Relatorio

Esta classe tem o objetivo de mostrar o relatório de vendas da pizzaria em um determinado intervalo de tempo. Para isto, esta classe tem relacionamento com todas as outras classes, uma vez que compila todas as informações armazenadas nelas para poder gerar este relatório de informações.

# Visão de Processos

O negócio conta com um gerente proprietário, um recepcionista/atendente, um pizzaiolo, um ajudante de cozinha e dois motoboys. O gerente é o usuário administrador. Por isto, ele tem acesso aos módulos de gerar relatório e gerir os usuários do sistema.

O recepcionista/atendente anota os pedidos dos clientes, faz o cálculo dos valores a serem pagos e recebe o pagamento, caso o cliente faça o pedido presencialmente. Para realizar esta tarefa utiliza-se a classe Cardápio para ter acesso aos itens do menu. A partir disto, a classe Pizza é acionada para trazer consigo as informações de nome, código, preço, e ingredientes da pizza. A informação ingredientes consiste em uma classe particular na qual são armazenadas as informações de nome do ingrediente, preço por porção de ingrediente usado e quantidade da porção utilizada.

Ademais, caso o cliente não tenha cadastro no sistema, o atendente deverá fazer o registro dessas informações para que, no próximo pedido deste cliente, tais informações possam ser reutilizadas no sentido de reduzir o tempo de ligação. Para isto, o sistema faz uso da classe Pedido. Nesta classe são armazenadas as informações do cliente tais como: nome, endereço e telefone.

Após anotar o pedido, o recepcionista/atendente repassa os itens a serem preparados para a cozinha.

O pizzaiolo e o ajudante de cozinha são responsáveis por preparar as massas e organizar a cozinha e ingredientes de modo a atender às solicitações dos clientes em tempo hábil e de acordo com a descrição do menu. Tudo isso deve ser realizado em consonância com os protocolos de Vigilância Sanitária para manipulação de alimentos.

Após o preparo, o recepcionista/atendente é quem realiza a entrega da pizza ao cliente ou ao motoboy que fará a entrega. Neste caso, o motoboy deve receber o endereço que deve localizar e o valor do pagamento a ser recolhido no momento da entrega. Nesta etapa, o atendente é responsável por emitir o recibo do cliente. Para isto, o sistema faz uso da classe Gerenciador de Pedido, a qual emiti um relatório com as informações armazenadas dentro das variáveis dessa classe. Tais variáveis armazenam as seguintes informações: nome do cliente (proveniente da classe pedido), endereço completo do cliente(proveniente da classe pedido), telefone do cliente(proveniente da classe pedido), o pedido, a forma de pagamento, o valor correspondente à taxa de entrega e o valor total da compra.

Os motoboys realizam as entregas aos clientes que optaram por este serviço e recebem o pagamento.

Os pagamentos podem ser realizados em dinheiro, cartão (débito e crédito) e pix.

O proprietário da pizzaria é quem realiza as compras, faz atendimentos em horários em que o número de clientes é maior e cuida das tratativas de reclamações do negócio.

# Visão de Implantação

A configuração mínima para executar este software consiste em um hardware com um processador no mínimo Intel i3, com 4 GB de memória ram.

O software MySQL deve estar instalado de modo a armazenar o banco de dados utilizado pelo sistema. Ademais, o driver mysqlconnector deve estar instalado para promover a interface de comunicação entre a aplicação JAVA e o banco de dados.

# Visão da Implementação

## Visão Geral

Devido ao fato de a aplicação estar dividida em duas camadas, temos regras especificas que determinam cada camada.

Dessa forma, na primeira camada estão todas as interfaces de interação do software com o usuário. Assim, a parte do código que liga com as regras de negócio e os módulos estabelecidos nos casos de uso devem ser alocados nesta camada.

Na segunda camada estão localizados todos os itens relativos à camada de dados, isto é, itens relativos ao banco de dados necessários para armazenamento dos dados dos clientes e suas compras.

## Camadas

O software é dividido em duas camadas: uma camada de interação e uma camada de banco de dados.

A camada de interação é composta pelas classes criadas em JAVA. Tais classes são responsáveis por armazenar os valores das variáveis usadas durante a utilização do software. Ademais, as classes contêm os métodos que deverão realizar as funções necessárias para o correto funcionamento da aplicação.

A segunda camada é composta pelo banco de dados. Para isto, existe um script que cria o banco de dados contendo as colunas necessárias para armazenar os valores recebidos através das variáveis da interface JAVA.

# Tamanho e Desempenho

Este software é dedicado a tornar o processo de receber o pedido do cliente mais ágil. Ao mesmo tempo, visa a reduzir o tempo total para o cliente receber o seu pedido. Dessa forma, não há integração entre o software e meios de pagamento.

Dessa maneira, este software não requer conexão com a internet para faer a troca de informações com o gateway de pagamento, uma vez que todo o pagamento é feito através de pix, cartão de débito ou crédito, ou mesmo dinheiro.

# Qualidade

A arquitetura do software preza pela eficiência. Dessa forma, o desenvolvimento em duas camadas busca diferenciar claramente a divisão funcional de cada uma delas.

Com isto, a primeira camada é dedicada exclusivamente a interação com o usuário e por manter as regras de negócio ativas.

Enquanto isso, a segunda camada preza por fazer o armazenamento dos dados em um banco de dados MySQL. Para esta interação entre as camadas acontecer, é necessário que um driver mysqlconnector esteja ativo e instalado na máquina na qual o software irá ser utilizado.