UNIVERSIDADE FEDERAL DE PERNAMBUCO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
ESTRUTURA DE DADOS ORIENTADA A OBJETOS

Letícia Uchôa Santos de Albuquerque Almeida
Lorena Cavalcanti Castelo Branco
Maria Eduarda Carvalho Braga

Sistema de Gerenciamento de Lanchonete

Projeto: Sistema de Informação com POO
Professor: Francisco Paulo Magalhães Simões

RECIFE
2025

Introdução

Diante da necessidade de aprimorar o funcionamento de uma lanchonete, desenvolvemos um sistema capaz de gerenciar e registrar seus pedidos e calcular seu faturamento, de forma automatizada e eficiente.
Aplicamos, no sistema, conceitos de POO como encapsulamento, herança e polimorfismo para, respectivamente, proteger e organizar as classes principais do código, reutilizar e hierarquizar essas classes entre si e aplicar métodos com funcionalidades diferentes para classes derivadas.

Metodologia

- Panorama geral dos requisitos para o sistema e funcionalidades necessárias;
- Organização através de diagramas (de Classe, Uso e Sequência);
- Implementação dos códigos em C++ com Code::Blocks, Qt e SQLite;
- Teste, validação e ajustes das funções implementadas.

Conceitos Utilizados

Abstração:

As classes Produto, Comida, Bebida e Ingrediente possuem apenas o necessário para funcionar no sistema, evitando as complexidades do mundo real. Como por exemplo a classe Produto que define uma abstração de produto genérico, e as classes derivadas (Comida, Bebida) especializam essa abstração, definindo atributos e comportamentos adicionais específicos para seus contextos

Encapsulamento:

A maior parte dos dados sensíveis, como o preço e o nome de um produto, são encapsulados, ou seja, só podem ser acessados e modificados por métodos específicos. Isso evita que esses atributos sejam alterados de forma indesejada fora da classe.

Herança:

A classe Produto (classe mãe) e suas classes filhas Comida e Bebida. A herança permite que as filhas compartilhem os mesmos comportamentos da classe mãe, mas ao mesmo tempo adicionem suas próprias características, isso evita repetição do código.

Polimorfismo:

O método detalhes() da classe base Produto é declarado como virtual, permitindo que ele seja sobrescrito nas classes derivadas (Comida e Bebida). Isso é um exemplo de polimorfismo, onde o método detalhes() pode ter comportamentos diferentes dependendo do tipo do objeto (produto genérico, comida ou bebida).

Construtores:

Cada classe possui um construtor que é chamado no momento da criação/inicialização do objeto.

Inicialização de Membros via Lista de Inicialização:

Em vez de inicializar os atributos dentro do corpo do construtor, usamos a lista de inicialização (como : disponibilidade(disponibilidade), preco(preco), nome(nome)). para inicializar os membros diretamente na criação do objeto, tornando o código mais limpo e eficiente.

Diagramas

Diagrama de Classes:

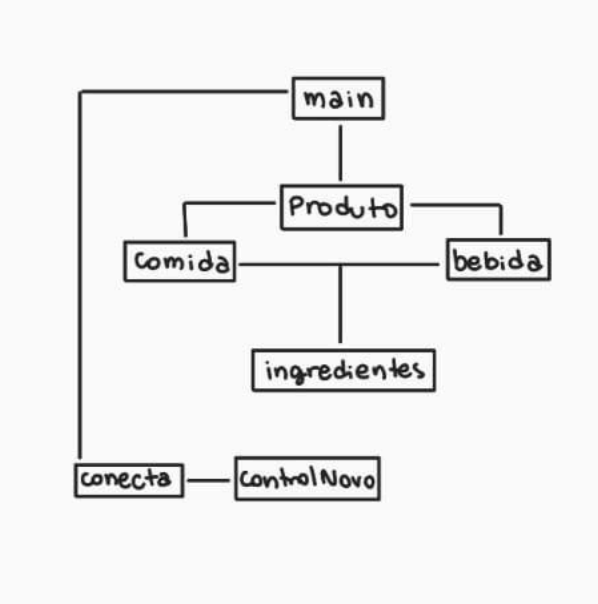
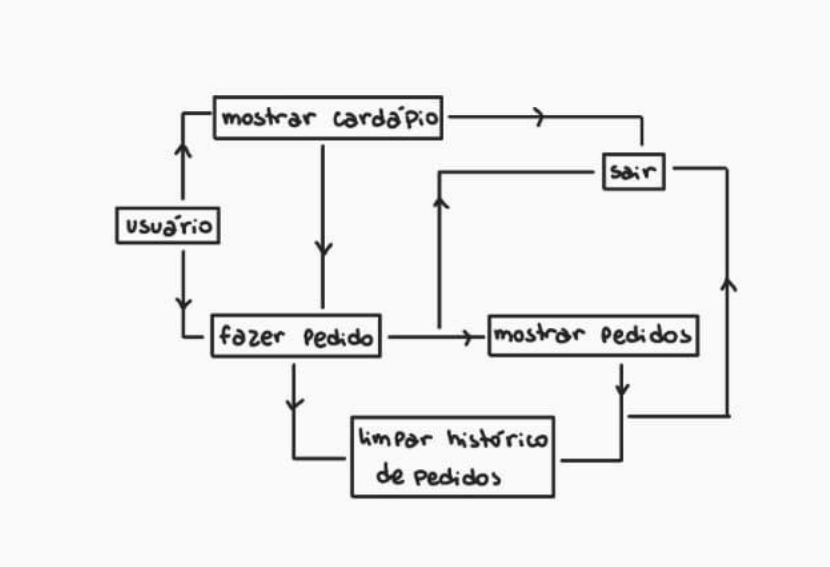


Diagrama de Sequência/Casos de Uso:



Implementação

Para a criação dos códigos, dividimos o sistema em três partes principais:

- Gerenciamento de Cardápio: Adição, remoção e atualização de produtos. Foram criadas classes específicas para os determinados tipos de produtos oferecidos (Comida e Bebida) e cada um foi inicializado no banco de dados;

- Gerenciamento de Pedidos: Registro de novos pedidos, cálculo do valor total atualização do status. A classe Pedido tem métodos para gerenciar os valores e escolha de produtos a serem solicitados, através da interação com o teclado;

- Controle de Estoque: Monitoramento da quantidade de ingredientes disponíveis. Nosso banco de dados foi criado para fazer esse monitoramento, seguindo os atributos de cada produto implementado nos códigos base.

Explicação do Código

Classe Base: Produto

É a classe base para os tipos de produtos ofertados no sistema, nela os atributos são: **preço** (double), **nome** (string) e **disponibilidade** (bool), todos eles são inicializados no **construtor** da classe.
Os métodos são **getters** e **setters** para acessar e alterar preço e nome, e **detalhes()** que exibe o detalhamento do produto.

Classe Derivada: Comida

Essa Classe é uma especialização de Produto para produtos alimentícios. Seus atributos são: tempoPreparo(int) e tipo(string) que diz se é um acompanhamento ou principal, eles são inicializados pelo Construtor da classe.
Seus métodos são, além dos getters e setters, detalhes() sobrescrito da classe Produto e PrecoPorTipo() que categoriza os valores.

Classe Derivada: Bebida

Também herdada de Produto, essa classe modela as bebidas ofertadas, segue a mesma estrutura de Comida, mas seu atributo é tamanho (int) inicializado no Construtor.
Já os métodos são os getters e setters e detalhes().

Classe Ingredientes

Modela os ingredientes utilizados nos produtos alimentícios e usa como atributos: quantidadeI(int), nomeI(string), tipoI(string) e disponibilidadeI(bool), que são inicializados pelo Construtor e exibidos pelo método detalhesI(). Seu método NaoDisponivelI() que muda a disponibilidadeI se a quantidadeI for igual a 0.

Tecnologias Utilizadas

Linguagem: C++, Code::Blocks;
Interface Gráfica: Qt Creator;
Banco de Dados: SQLite.

Reuniões do grupo: Google Meet, Whatsapp, Discord;
Repositório: GitHub;

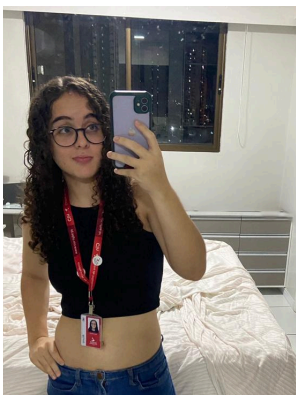
Divisão de Tarefas

Leticia Uchoa Santos de Albuquerque Almeida

Gerenciamento de Pedidos;
Implementação dos códigos base em C++ para o Qt Creator (Interface Gráfica).

Lorena Cavalcanti Castello Branco

Desenvolvimento das classes bases, estrutura e lógica em C++;
Gerenciamento dos Ingredientes e Produtos.



Maria Eduarda Carvalho Braga

Gerenciamento de Pedidos e Estoque;
Implementação do Banco de Dados e conexão com Qt Creator.