Sistema de Cardápio e Pedidos para FoodDelivery

Denilson Santos da Silva, Gabriel Arlisson de Souza Santos Torres, João Arthur Mascarenhas Nascimento, Tony Cleriston Oliveira dos Santos Junior, Vinicius Cerqueira Oliveira

Sistemas de Informação – Centro Universitário de Excelência (UNIFTC) - BA

tonysonic1227@gmail.com, denilsonsant.ofc@gmail.com,
joaoarthurnasci@hotmail.com, gabriel.arlisson2017@gmail.com,
voliveira170@gmail.com

Abstract. *This work presents the development of an order management system in the user's own CLI. The project was structured into classes representing customers, menu items, and orders, including their interactions and business rules, such as the status flow. The system operates in a command-line environment, allowing registration, listing, and order tracking. The experience highlighted the importance of class modeling for organizing complex systems and contributes to scalability, clarity, and code reuse.*

Resumo. *Este trabalho apresenta o desenvolvimento de um sistema de gerenciamento de pedidos no próprio CLI do usuário. O projeto foi estruturado em classes que representam clientes, itens de cardápio e pedidos, incluindo suas interações e regras de negócio, como o fluxo de mudança de status. O sistema opera em linha de comando, permitindo cadastro, listagem e acompanhamento de pedidos. A experiência evidenciou a importância da modelagem de classes para organizar sistemas complexos e contribui para a escalabilidade, clareza e reutilização de código.*

1. Introdução

A Programação Orientada a Objetos representa um conceito fundamental na engenharia de software. Seu principal objetivo consiste em modelar sistemas complexos de forma mais intuitiva, espelhando entidades e interações do mundo real. Ao organizar o código em "objetos" que encapsulam dados (atributos) e comportamentos (métodos), a POO permite a reutilização de código bem como a escalabilidade das aplicações. Este projeto visa aplicar esses conceitos na prática para construir um sistema de gerenciamento de pedidos, demonstrando como a modelagem orientada a objetos pode solucionar problemas de negócio de forma eficaz.

2. Fundamentação Teórica

2.1. Conceito de Classes e Objetos

Uma classe é um molde ou uma planta que define os atributos que os objetos terão, já o objeto é uma instância concreta de uma classe.

2.2. Conceito de Atributos e Métodos

Atributos são as variáveis de uma classe, representando os dados ou o estado de um objeto, enquanto os métodos são as funções associadas a uma classe, definindo as ações que um objeto pode realizar.

2.3. Conceito de Construtores

Um construtor é um método especial que usamos para no momento da criação de um objeto inicializar os atributos do objeto, garantindo que ele passe a existir com os dados já estabelecidos.

2.4. Diagrama de Classes

É um formato visual usado para representar a estrutura de um sistema, ele exibe as classes, seus atributos, métodos e os relacionamentos entre eles, fornecendo um mapa claro da arquitetura do software.

3. Metodologia

A metodologia adotada neste trabalho consistiu no desenvolvimento de uma aplicação que funciona no ambiente de linha de comando e tem como objetivo principal o gerenciamento de pedidos de clientes. Ela tem como base três entidades principais: Cliente, ItemCardápio e Pedido, e uma entidade auxiliar, PedidoItemCardapio, para representar os itens de cada pedido.

3.1. Estrutura de Dados

Neste sistema primeiro foi criado o componente do Cliente, com seu identificador único, nome e telefone, depois o ItemCardapio com código, nome e valor, sendo duas classes atómicas. Posteriormente veio o Pedido contendo o código, cliente associado, status, data, e lista de itens vinculados, e por fim o PedidoItemCardapio que relaciona os itens do cardápio a um pedido específico com sua quantidade solicitada de cada item.

3.2. Funcionalidades

O sistema foi implementado para operar através de comandos diretos, que funcionam por meio de uma estrutura de dados responsável por armazenar os dados de cada entidade. Três elementos responsáveis pelo armazenamento existem na implementação: Array clientes (que armazena objetos do tipo Cliente), Array itens (que armazena objetos do tipo ItemCardapio) e Array pedidos (que armazena objetos do tipo Pedido). Para estes dados, foram implementadas funcionalidades de inclusão e listagem. O gerenciamento de dados foi implementado por meio da observação do status, que muda conforme o fluxo da entrega.

3.3. Processo de Desenvolvimento

No decorrer do desenvolvimento do projeto foi feita a modelagem visando a definição das entidades, atributos e relacionamentos, sendo possível a criação de comandos acessíveis ao usuário para a manipulação das entidades tratando suas entradas de dados.

Por fim realizou-se testes verificando o fluxo de pedidos, atualização de status, cadastro de clientes e inclusão de itens.

4. Resultados e Discussões

A classe Pedido encapsula a lógica de transição de status. O método avancarStatus() contém a regra que permite a mudança apenas para o próximo estado válido no ciclo de vida, lançando uma exceção caso a transição seja inválida.

Foram criados métodos específicos para gerar relatórios simplificados e detalhados, esses métodos iteram sobre a lista de pedidos, calculando os totais e formatando a saída conforme solicitado.

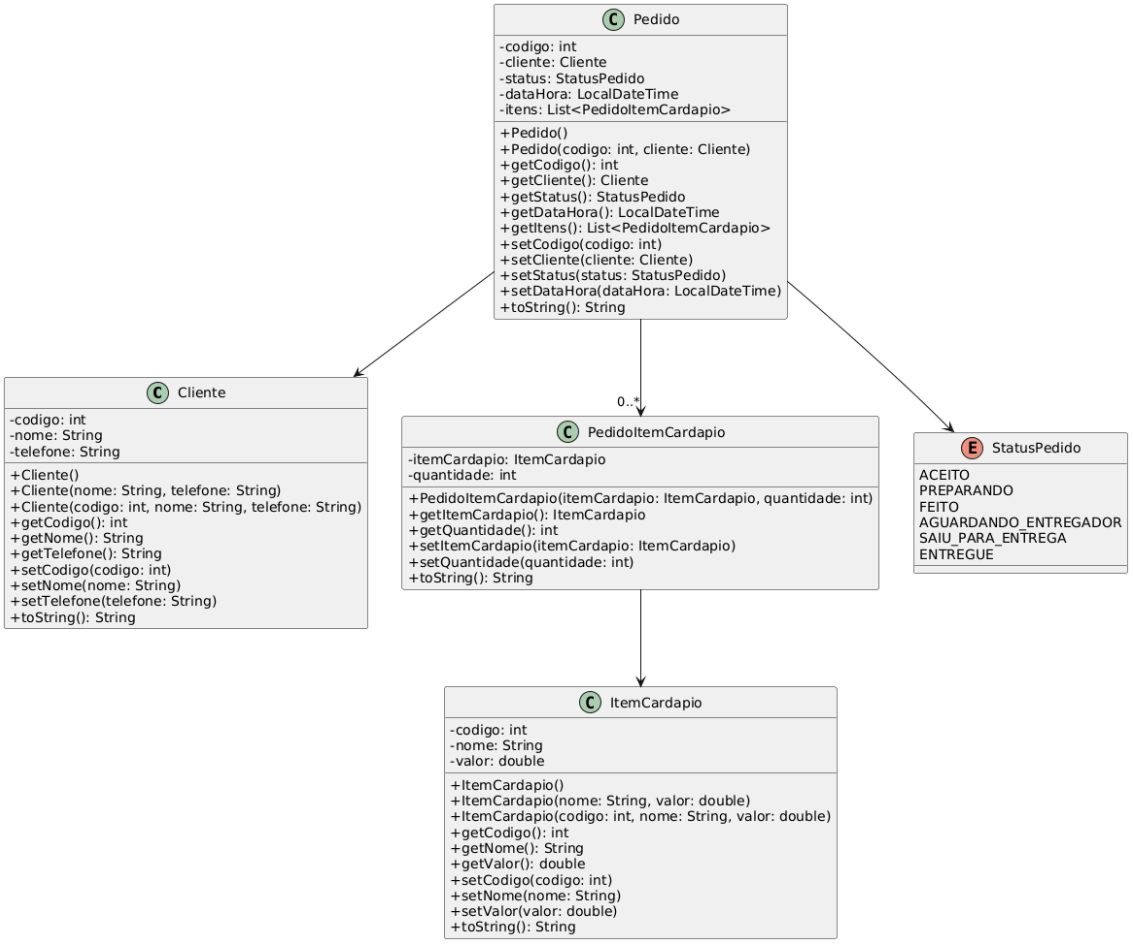

Figura 1. Diagrama de classes final da aplicação CLI

5. Considerações finais

O momento mais desafiador foi realizar a interpretação da regra de negócio e transformá-la em uma peça de código tentando encontrar a solução que mais se encaixasse no desafio proposto.

A parte mais interessante do desenvolvimento da aplicação foi pensar sobre como implementar a lógica do fluxo de entrega do pedido e como programar a mudança de

status do mesmo.

Caso houvesse mais tempo disponível, uma refatoração poderia ter sido planejada de forma mais eficiente para melhor manutenibilidade do código e maior legibilidade. No entanto, o resultado alcançado foi satisfatório para os integrantes da equipe.

Referências

Booch, G.; Rumbaugh, J.; Jacobson, I. UML: Guia do Usuário. 2. ed. Rio de Janeiro: Campus, 2005.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000.