LuFood: Um Sistema de Gerenciamento de Pedidos Baseado em Filas e Listas em Python

Levi Falcão de Queiroz, Gabriel Gomes Cruz Uzeda, Bruno Sampaio Silva, Claudio dos Santos Junior, Herick Marcio Matos Brito, Breno Matos Bastos

Centro Universitário de Excelência (UNEX) - Sistemas de Informação - Feira de Santana - Bahia - Brasil

{leviq.f@hotmail.com, gomes.uzeda@gmail.com, herickmarcio356@gmail.com, brunosilva.mee@gmail.com, brenomatos486@gmail.com, claudiosjrel@gmail.com}

Resumo. Este relatório detalha o desenvolvimento do "LuFood", um sistema de protótipo para gerenciamento de produtos e pedidos de um estabelecimento alimentício, implementado em Python via console. O projeto utiliza estruturas de dados fundamentais, com destaque para listas, para armazenar e gerenciar informações de itens e o ciclo de vida dos pedidos. A metodologia aborda a representação dos dados e a lógica processual que simula o comportamento de uma fila para o processamento de pedidos, desde sua criação até a entrega. O trabalho demonstra a aplicação prática de conceitos de estruturas de dados na resolução de um problema de gerenciamento de fluxo de trabalho.

Abstract. This report details the development of "LuFood," a prototype system for managing products and orders for a food establishment, implemented in Python via console. The project utilizes fundamental data structures, with an emphasis on lists, to store and manage information about items and the order lifecycle. The methodology addresses the data representation and the procedural logic that simulates queue behavior for order processing, from creation to delivery. The work demonstrates the practical application of data structure concepts in solving a workflow management problem.

1. Introdução

As estruturas de dados são pilares da Ciência da Computação, essenciais para a organização e manipulação eficiente da informação. A escolha correta de uma estrutura otimiza o desempenho e define a clareza da solução de software. Neste contexto, o projeto "LuFood" foi desenvolvido em Python para gerenciar o fluxo de pedidos de um restaurante, aplicando estruturas de dados essenciais, notadamente listas, para modelar e resolver um problema prático. O sistema gerencia o cadastro de produtos, a criação de pedidos e o acompanhamento de seu status, simulando uma linha de produção.

2. Fundamentação Teórica

Para a construção do sistema LuFood, foram aplicados conceitos de estruturas de dados lineares, que organizam seus elementos de forma sequencial, com destaque para listas e o conceito de filas.

Uma lista em Python é uma coleção de elementos ordenados e mutáveis, permitindo armazenar itens de tipos heterogêneos. Sua flexibilidade, com indexação, inserção, remoção e crescimento dinâmico, a torna ideal para armazenar coleções como um catálogo de produtos ou a base de dados de pedidos do projeto. Uma fila é uma estrutura que segue o princípio FIFO (First-In, First-Out), onde o primeiro elemento a entrar é o primeiro a sair, análogo a uma fila de atendimento. Filas são ideais para cenários onde a ordem de processamento é crucial, como o processamento de pedidos em um restaurante, garantindo que sejam preparados na ordem em que foram recebidos.

3. Metodologia e Implementação

O sistema LuFood foi desenvolvido em Python 3, sem bibliotecas externas, com interação via interface de linha de comando (CLI). A implementação se baseia na manipulação de listas para representar as entidades do sistema.

3.1. Representação de Dados

As entidades centrais do sistema são representadas por listas com uma ordem fixa de elementos:

Produto: Uma lista contendo [nome, codigo, preco, descricao, quantidade_estoque]. Todos os produtos são armazenados em uma lista principal chamada itens.

Pedido: Uma lista contendo [codigo, itens_do_pedido, valor_total, cupom, status]. O campo itens_do_pedido é uma lista de listas, onde cada sub-lista armazena [codigo_item, nome_item, quantidade, subtotal].

3.2. Arquitetura de Dados

O sistema implementa uma arquitetura híbrida com duas categorias de listas para gerenciar os pedidos:

Listas Principais: pedidos_ativos (pedidos não finalizados), pedidos_finalizados (arquivo morto) e todos_os_pedidos (histórico completo para consultas).

Filas Conceituais por Status: O fluxo de trabalho é gerenciado movendo os pedidos entre diferentes estágios, como pedido_aguardando_aprovacao, pedido_aceito, pedido_fazendo, pedido_feito, entre outros, simulando um sistema de filas.

3.3. Módulos Funcionais

O sistema oferece seis módulos principais: Gestão de Produtos (cadastro, modificação, consulta), Criação de Pedidos com validação de estoque, Sistema de Cupons (DESCONTO10, DESCONTO20), Gerenciamento de Status para processamento sequencial do fluxo, Consulta de Pedidos com filtros por status e Cancelamento de Pedidos com devolução ao estoque.

4. Resultados e Discussões

O sistema implementado cumpre todos os requisitos funcionais propostos. A seguir, são apresentados exemplos de interação e um fluxograma que ilustra o comportamento do programa.

4.1. Cenário de Uso 1: Criação de um Pedido

O usuário seleciona a opção de criar um pedido. O sistema exibe apenas itens com estoque disponível, e o usuário adiciona produtos informando o código e a quantidade.

O sistema valida automaticamente a disponibilidade e reduz o estoque em tempo real. Ao final, pode aplicar cupons de desconto válidos

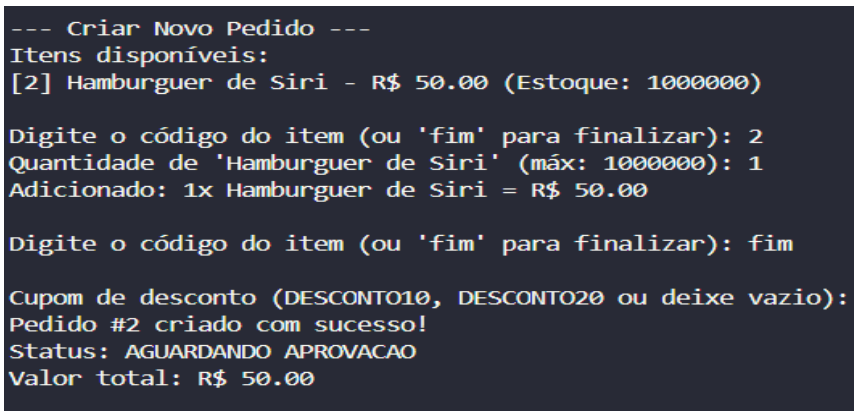

Figura 2: Tela de criação de um novo pedido.

4.2. Cenário de Uso 2: Gerenciamento de Status

O sistema implementa processamento sequencial baseado no princípio FIFO. Um funcionário do restaurante utiliza o menu "Processar Pedidos" para avançar pedidos no fluxo de produção sempre na ordem de chegada.

Processar Pedidos

O sistema processará sempre o pedido mais antigo de cada etapa.

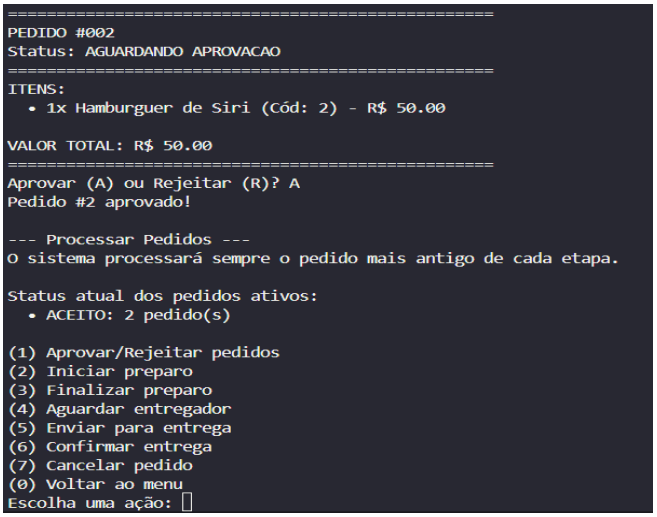

Figura 2: Tela de processamento de pedido, mostrando a aprovação e o resumo de status.

4.3. Fluxograma do Ciclo de Vida de um Pedido

A Figura 3 ilustra visualmente todas as etapas e transições de status possíveis para um pedido dentro do sistema LuFood, desde sua criação até a finalização.

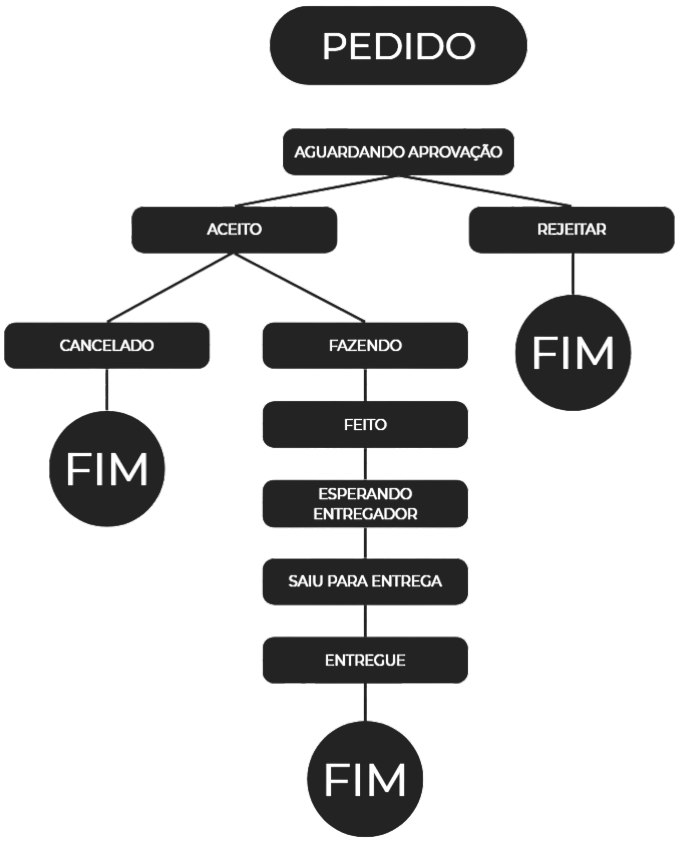

Figura 3: Fluxograma de Status de Pedidos

4.4. Discussão

Os resultados demonstram que a utilização de listas foi suficiente para modelar a complexidade do problema. A implementação híbrida, combinando listas principais com filas conceituais por status, provou ser eficaz para manter a organização e permitir o processamento FIFO. A função buscar_proximo_pedido() garante que sempre o pedido mais antigo de cada status seja processado primeiro, simulando o comportamento de uma fila real.

O sistema de validação implementado previne inconsistências, como tentar processar pedidos inexistentes ou em status inadequados. A separação entre pedidos_ativos e pedidos_finalizados otimiza as operações de gerenciamento, que precisam iterar apenas sobre os pedidos relevantes.

A funcionalidade de cancelamento com devolução automática ao estoque demonstra a importância do controle de estado consistente, evitando perda de produtos quando pedidos são cancelados.

5.Conclusão

O desenvolvimento do projeto LuFood permitiu a aplicação prática de conceitos de estruturas de dados em um cenário tangível. O principal desafio foi gerenciar a consistência dos dados utilizando apenas listas, e a simulação de filas para manter o princípio FIFO foi uma solução eficaz. A validação de entradas e o tratamento de casos como estoque insuficiente foram cruciais para a robustez do sistema.

Como trabalhos futuros, diversas melhorias poderiam ser implementadas, como a persistência de dados em arquivos (CSV, JSON) para evitar a perda de informações ao encerrar o sistema; o uso de estruturas mais adequadas, como dicionários, para tornar o código mais legível; o desenvolvimento de uma interface gráfica (GUI) para melhorar a experiência do usuário; e a implementação de um sistema de relatórios de vendas. Conclui-se que o projeto foi bem-sucedido em atingir seus objetivos educacionais, demonstrando como estruturas simples podem ser combinadas para resolver problemas complexos.

Referências

PYTHON SOFTWARE FOUNDATION. 5. Estruturas de dados — documentação Python 3.7.17.