FoodDelivery com Listas e Filas

**Lara Nunes Silva, José Henrique Dias Moreira, Dênis Maciel,
Graciano Marôto e Douglas Pereira.**

***Centro Universitário de Excelência (Unex)* – Sistemas de informação – Vitória da Conquista – Bahia – Brasil**

**{nuneslarasilva@hotmail.com, jhenriquedm98@gmail.com, denisfilho398@gmail.com,
engdouglasps@gmail.com e gracianoml07@gmail.com.}**

Resumo

O trabalho analisa estruturas de dados, com foco em listas e filas. Listas permitem acesso direto a qualquer posição, enquanto filas seguem a lógica FIFO, útil para processos em sequência. Foi desenvolvido em Python um sistema de gerenciamento de pedidos e produtos, usando classes e menus no terminal. O sistema possibilita cadastrar, atualizar e cancelar pedidos, aplicar descontos e mudar status, armazenando tudo em listas. Os resultados foram positivos, apesar de desafios como a interface interativa e o tratamento de erros. Para o futuro, sugere-se integrar banco de dados, criar interface gráfica e adicionar relatórios e gestão de usuários.

Abstract

The work analyzes data structures, focusing on lists and queues. Lists allow direct access to any position, while queues follow the FIFO principle, useful for sequential processes. A Python-based system was developed for managing orders and products, using classes and terminal menus. The system enables registering, updating, and canceling orders, applying discounts, and changing statuses, with all data stored in lists. Results were positive, despite challenges such as building the interactive interface and handling errors. Future improvements include database integration, a graphical interface, and additional features like reports and user management.

1. Introdução

Estruturas de dados são formas organizadas de armazenar e gerenciar informações em um programa de computador. Elas são essenciais para otimizar operações como busca, inserção e exclusão de dados, influenciando diretamente a velocidade e o uso de memória. Uma escolha adequada melhora o desempenho e a escalabilidade dos sistemas. Além disso, permitem a solução eficiente de problemas complexos por meio de modelos como listas, pilhas, filas, árvores e grafos. Dominar estruturas de dados é fundamental para desenvolver softwares robustos e eficientes.

2. Fundamentação Teórica

Lista: Uma lista é uma coleção ordenada de elementos, onde cada elemento pode ser acessado diretamente por sua posição (índice). É como uma sequência organizada, em que cada item tem um "endereço" específico. As listas possuem ordem definida, ou seja, os elementos seguem sempre uma sequência. Elas permitem armazenar valores duplicados, oferecem acesso direto por meio de índices e possibilitam inserções em posições específicas. Além disso, podem crescer ou diminuir dinamicamente, dependendo da implementação. Entre as aplicações mais comuns, destacam-se o gerenciamento de tarefas em aplicativos de "to-do list", o armazenamento do histórico de navegação e o registro de dados em sistemas que precisam manter a ordem dos elementos.

Fila: Uma fila é uma coleção ordenada, mas funciona com base no princípio FIFO (*First In, First Out* "o primeiro a entrar é o primeiro a sair"). É como uma fila de pessoas em um banco: quem chega primeiro, é atendido primeiro. As filas têm sua ordem de processamento determinada pela ordem de inserção, em que os elementos são adicionados ao final (enfileiramento) e removidos do início (desenfileiramento). Elas não permitem acesso direto a elementos intermediários, sendo possível remover apenas o primeiro elemento diretamente. Entre suas aplicações práticas, destacam-se o gerenciamento de trabalhos em impressoras, a organização do atendimento em sistemas como call centers e suporte técnico, além do processamento de tarefas em sistemas operacionais. Quanto às similaridades entre listas e filas, ambas armazenam elementos em ordem, podem crescer ou diminuir

dinamicamente, são usadas para organizar dados de maneira estruturada e podem ser implementadas utilizando arrays ou listas encadeadas.

Principais Diferenças: A diferença principal é que listas permitem acessar e alterar elementos em qualquer posição, enquanto filas seguem a ordem FIFO, permitindo inserir no fim e remover apenas do início. Listas são melhores para acesso aleatório; filas, para processamento sequencial.

3. Metodologia

O sistema foi desenvolvido em Python como uma aplicação interativa de terminal para gerenciamento de pedidos e produtos. A proposta foi criar uma solução funcional e de fácil uso, com menus que orientam o usuário em operações como cadastro de itens, atualização de estoque, criação e acompanhamento de pedidos. O desenvolvimento foi orientado à modularidade, com separação clara entre as funcionalidades de controle de itens e controle de pedidos. O menu de itens permite adicionar, editar e visualizar produtos disponíveis, enquanto o menu de pedidos possibilita a criação de novos pedidos, adição de produtos ao carrinho, aplicação de desconto e alteração do status conforme o andamento. Para armazenar os dados, o sistema utiliza listas, que permitem registrar e manipular as informações de forma dinâmica. A lógica do sistema é construída com estruturas de repetição e condicionais, que criam um fluxo de navegação simples e eficiente, garantindo que cada etapa da operação seja validada antes de prosseguir. A interação é toda feita via terminal, simulando um ambiente de uso real para pequenos comércios ou serviços de delivery, onde o controle de pedidos e produtos precisa ser ágil e direto.

4. Resultados e Discussões

Funcionamento Geral do Código: A ferramenta desenvolvida é um sistema de gerenciamento de pedidos e catálogo de produtos, implementado em Python, executado via terminal. Ele possui dois módulos principais: Gerenciamento de Itens e Gerenciamento de Pedidos.

O código utiliza **listas** como estrutura de dados principal para armazenar informações e **classes** para estruturar esses dados em objetos organizados. As classes definem atributos e comportamentos (métodos), permitindo manipular dados com mais organização.

4.1 - Uso das Classes (Item e Order)

4.1.1 Classe Item - Representa cada produto do catálogo e possui atributos como:

code: código único do item; **name:** nome do item; **description:** descrição do item; price: preço do item e **stock**: quantidade disponível em estoque.

Principais métodos:

- **update_stock(quantity):** adiciona ou retira quantidade do estoque, validando se há estoque suficiente.
- **update_name(), update_description(), update_price():** permitem alterar atributos do produto após confirmação do usuário.

4.1.2 Classe Order - Representa um pedido e possui atributos como:

code: código único do pedido; **costumer:** nome do cliente; **items_order:** lista de itens incluídos no pedido; **status:** estado do pedido ("Pending", "Accepted", etc.); **payment:** status do pagamento; **order_total_price:** preço final calculado.

Principais métodos:

- **total_price()**: calcula o preço total do pedido.
- **apply_discount()**: aplica desconto de 10% ao total.
- **finalize()**: confirma o total do pedido.

4.2 - Uso das Listas

O sistema utiliza principalmente **duas listas**:

4.2.1 Lista catalog

- Armazena objetos Item (produtos cadastrados).
- Funciona como um catálogo organizado por ordem de inserção.
- Permite acesso sequencial: o usuário navega pela lista para visualizar itens.
- É possível **remover ou alterar itens em qualquer posição**, desde que identificado pelo nome ou código.
- Não possui ordenação automática por preço ou estoque — mantém a ordem em que foram adicionados.

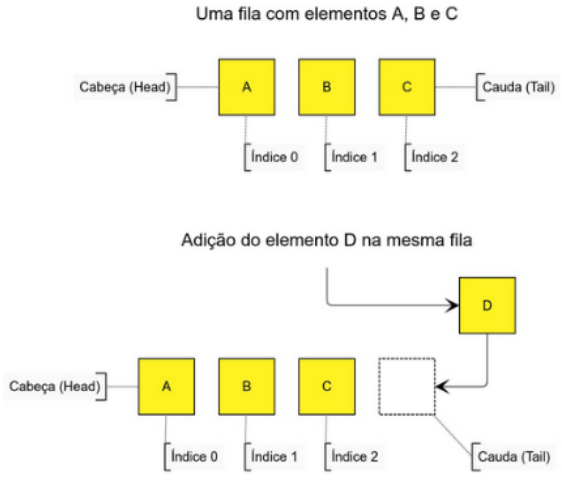
Operações principais:

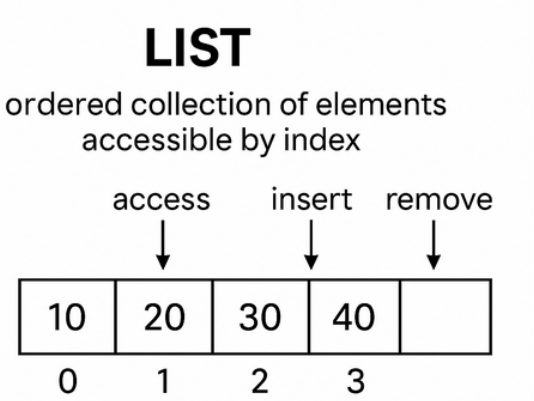
- Inserção de novos itens ao final da lista.
- Alteração de atributos de qualquer item existente.
- Visualização completa da lista de itens.

Exemplo: ao cadastrar um item, ele é adicionado ao final da lista. Ao atualizar, o sistema busca pelo nome e altera atributos diretamente na posição encontrada.

4.2.2 Lista all_orders

- Armazena objetos Order (pedidos realizados).
- Funciona como uma **fila** (FIFO — First In, First Out) para atendimento:
  - O primeiro pedido a ser criado será o primeiro listado como pendente.
  - Pedidos são atendidos na ordem em que foram inseridos.
  - Permite acesso sequencial para gerenciamento.





Operações principais:

- Inserção de novos pedidos ao final da lista.
- Alteração de status de pedidos e cancelamento de pedidos.

Exemplo: se dois pedidos são criados consecutivamente, o primeiro será o primeiro a aparecer em "Gerenciar pedidos pendentes".

4.3 - Detalhamento dos Módulos:

4.3.1 Gerenciamento de Itens:

Função: manage_menu_items(catalog)

- **Apresentar opções:** Adicionar item; Atualizar item; Visualizar itens; Retornar ao menu principal.
- **Adicionar item**: Usuário fornece nome, descrição, preço e quantidade; Um novo objeto Item é criado; Item é inserido no final da lista catalog.
- **Atualizar item**: Usuário busca pelo nome; Se encontrado, pode alterar nome, descrição, preço ou quantidade; Alterações são confirmadas antes de aplicar; Atualização afeta diretamente o objeto na lista.
- **Visualizar itens**: Percorre a lista catalog e imprime detalhes de cada item.

Observação: a lista catalog funciona como um registro persistente dentro da execução do programa, permitindo manipulação direta dos itens cadastrados.

4.3.2 Gerenciamento de Pedidos

Função: manage_orders(all_orders, catalog)

- **Apresenta opções:** Criar novo pedido; Gerenciar pedidos pendentes; Alterar status de pedidos; Cancelar pedido; Retornar ao menu principal.
- **Criar pedido**: Solicita nome do cliente; Cria objeto Order com lista vazia items_order; Usuário adiciona itens disponíveis no catalog ao pedido; O sistema verifica estoque antes de adicionar. Se estoque insuficiente → impede a adição; Ao adicionar, estoque do item é reduzido automaticamente; Pedido finalizado é inserido no final da lista all_orders com status "Pending".
- **Gerenciar pedidos pendentes**: Lista os pedidos na ordem em que foram criados (fila FIFO); Permite aceitar ou rejeitar pedidos.
- **Alterar status**: Lista todos os pedidos; Usuário seleciona um pedido e define um novo status.
- **Cancelar pedido**: Lista pedidos com status cancelável ("Pending" ou "Accepted"); Usuário seleciona pedido a cancelar.

Observação: a lista all_orders funciona como uma fila para gerenciamento de atendimento, respeitando a ordem de chegada.

4.4 - Visão geral Fluxo do Sistema

Iniciar o sistema e visualizar o menu principal com as opções:

- Gerenciar Itens
- Gerenciar Pedidos
- Consultas e Relatórios
- Sair

Caso escolha **Gerenciar Itens**:

- Adicionar novo item ao cardápio (informando nome, descrição, preço e estoque).
- Atualizar item existente (alterar nome, descrição, preço ou estoque).
- Visualizar todos os itens cadastrados.
- Voltar ao menu principal.

Caso escolha **Gerenciar Pedidos**:

- Criar novo pedido (informando nome e telefone do cliente, escolhendo itens pelo código, podendo aplicar desconto de 10%).
- Gerenciar pedidos pendentes (aceitar ou rejeitar).

- Atualizar status de pedidos (Accepted → Making → Ready → Waiting Delivery → Delivering → Delivered).
- Cancelar pedidos (apenas se estiverem *Pending* ou *Accepted*).
- Voltar ao menu principal.

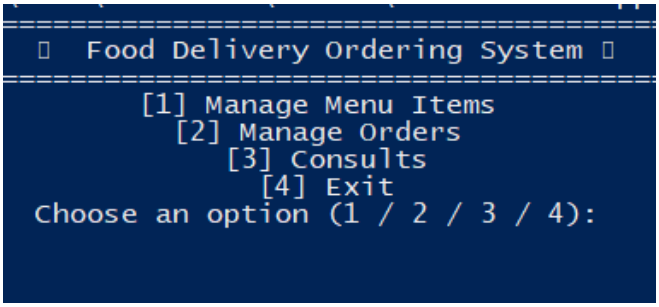
Caso escolha **Consultas e Relatórios**:

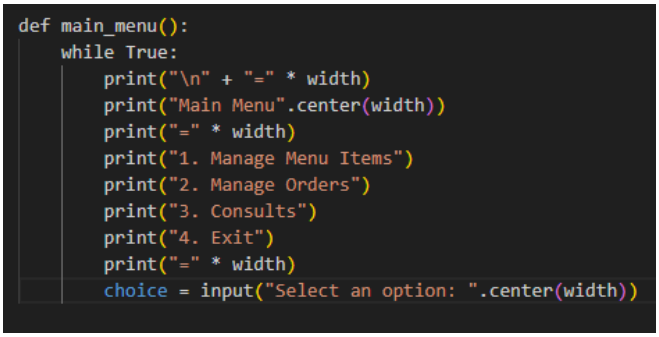
- Ver todos os pedidos.
- Filtrar pedidos por status.
- Gerar relatório de vendas (todos os registros ou apenas entregues).
- Visualizar todos os clientes cadastrados.
- Voltar ao menu principal.

Caso escolha **Sair**, o sistema encerra a execução.

4.5 Prints das Telas

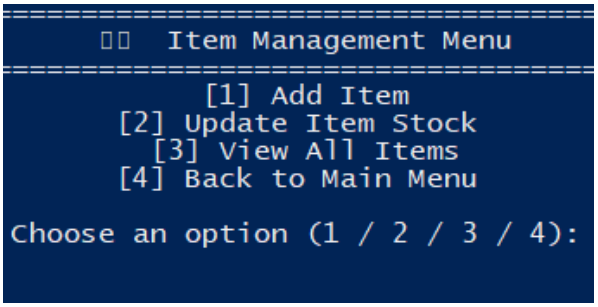
1 - Visualizar o menu principal com as opções:

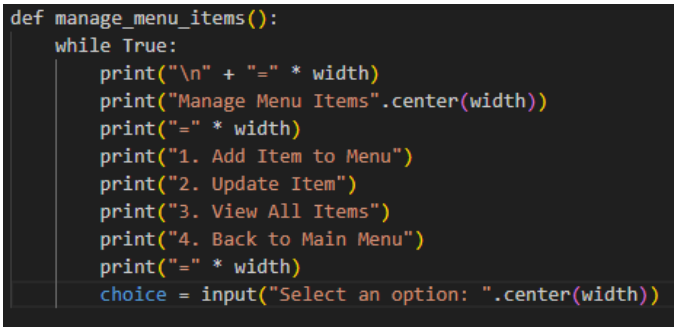




```
def main_menu():
    while True:
        print("\n" + "=" * width)
        print("Main Menu".center(width))
        print("=" * width)
        print("1. Manage Menu Items")
        print("2. Manage Orders")
        print("3. Consults")
        print("4. Exit")
        print("=" * width)
        choice = input("Select an option: ".center(width))
```

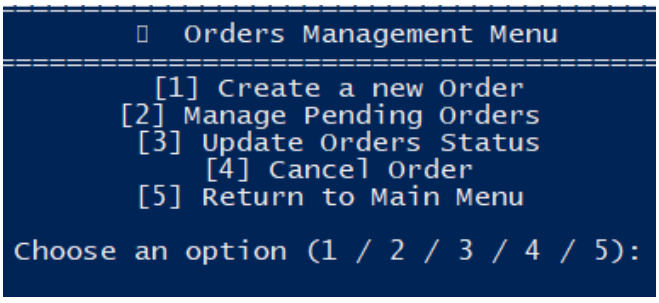
2- **Gerenciar Itens**:



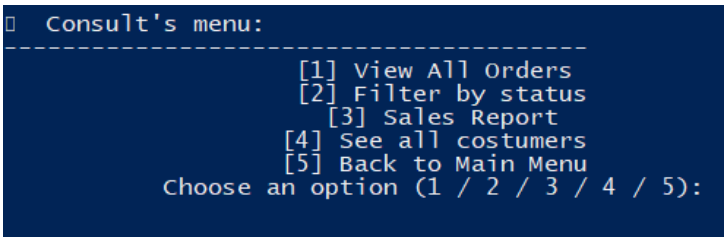


```
def manage_menu_items():
    while True:
        print("\n" + "=" * width)
        print("Manage Menu Items".center(width))
        print("=" * width)
        print("1. Add Item to Menu")
        print("2. Update Item")
        print("3. View All Items")
        print("4. Back to Main Menu")
        print("=" * width)
        choice = input("Select an option: ".center(width))
```

3 - **Gerenciar Pedidos**:



4 - Relatórios:



5. Considerações Finais

Durante o desenvolvimento deste projeto, encontramos desafios interessantes que tornaram a experiência enriquecedora. Um dos maiores foi criar uma interface interativa no terminal, garantindo que o sistema respondesse de forma clara mesmo quando o usuário inseria dados incorretos. Também foi um desafio pensar em como tratar situações específicas, como estoque insuficiente ou pedidos sem itens, para que o programa funcionasse de maneira estável.

Algo que se mostrou muito gratificante foi a construção de uma estrutura organizada, usando classes para representar itens e pedidos. Isso não apenas deixou o código mais limpo, mas também facilitou a manutenção. Integrar essa estrutura ao menu interativo foi uma parte divertida, porque permitiu criar uma experiência próxima de um sistema real de gerenciamento.

Se tivéssemos mais tempo, certamente iríamos ampliar bastante o projeto. Entre as ideias, estão integrar o sistema a um banco de dados para guardar informações permanentemente e criar uma interface gráfica para torná-lo mais intuitivo. Também seria interessante adicionar funções como relatórios de vendas, histórico de pedidos e gestão de usuários, transformando a aplicação em uma solução completa para negócios.

6. Referências

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. *Algoritmos: teoria e prática*. 3. ed. Rio de Janeiro: LTC, 2014.

GOODRICH, Michael T.; TAMASSIA, Roberto. *Estruturas de dados e algoritmos em Python*. São Paulo: Bookman, 2016.

PYTHON SOFTWARE FOUNDATION. *Documentação Python*. Disponível em: https://docs.python.org/3/