Relatório Técnico - Científico
Sistema de Gerenciamentos de Pedidos
Tia Lu Food Delivery

Grupo Pernambuco

Jefte Martins, Pedro Silveira, Paulo Soares, David Cairo, Ian Neves

Sistemas de informação – Centro Universitário de Excelência (UNEX)
Caixa Postal 45020-510 – Vitoria da Conquista – BA – Brasil

Resumo. Este relatório descreve o desenvolvimento de um sistema de pedidos em Python para a **Tia Lu Food Delivery**, estruturado a partir do uso de listas e filas para organizar e processar dados. As listas foram utilizadas para o gerenciamento do cardápio e dos cadastros, enquanto as filas asseguraram o processamento dos pedidos em ordem cronológica, respeitando a disciplina FIFO. O trabalho evidenciou a versatilidade das listas e a importância das filas para manter a consistência do fluxo de informações. Além disso, reforçou a relevância do estudo das estruturas de dados para a construção de sistemas eficientes, capazes de atender a diferentes demandas práticas.

1. Introdução

A Ciência da Computação dedica-se fundamentalmente à resolução de problemas por meio de algoritmos e à manipulação eficiente de informações. Nesse cenário, as estruturas de dados surgem como um pilar essencial, oferecendo modelos lógicos para organizar, armazenar e acessar dados de forma otimizada. A escolha criteriosa de uma estrutura de dados é um fator determinante na eficiência de um software, impactando diretamente seu tempo de execução, consumo de memória e escalabilidade. Uma seleção inadequada pode tornar uma solução computacional inviável, enquanto a escolha correta é a base para sistemas robustos e performáticos.

Diante desse contexto, o presente trabalho tem como objetivo desenvolver um sistema de pedidos em Python, utilizando listas e filas como instrumentos de organização de dados. O trabalho aborda a construção dessas estruturas, a análise de seu comportamento operacional e uma comparação direta de suas características, vantagens e desvantagens. O objetivo é consolidar o entendimento teórico e demonstrar, na prática, como suas arquiteturas distintas as tornam adequadas para diferentes tipos de problemas.

2. Fundamentação Teórica

De acordo com Cormen et al. (2009), uma estrutura de dados pode ser definida como uma forma específica de armazenar e organizar informações em um computador, de modo que possam ser manipuladas de maneira eficiente. Entre as estruturas abordadas neste trabalho, destacam-se as listas e as filas, amplamente empregadas em diferentes contextos computacionais.

2.1 Lista

A lista é uma estrutura linear que armazena elementos em sequência, permitindo acesso direto por meio de índices. Ela possibilita operações como inserção, remoção e atualização em qualquer posição, conferindo grande flexibilidade. Segundo Lafore (2012), listas são especialmente úteis em situações que demandam manipulação frequente e variada dos elementos. As duas principais formas de implementação de listas determinam suas características de desempenho:

- **Listas Contíguas (Vetores):** Armazenam dados em memória contígua. Oferecem acesso por índice extremamente rápido (O (1)), mas possuem inserções e remoções lentas (O(n)) devido à necessidade de deslocar elementos.

- **Listas Encadeadas:** São compostas por nós ligados por ponteiros. Permitem inserções e remoções muito rápidas (O (1)) e tamanho flexível, ao custo de um acesso lento e sequencial (O(n)) para encontrar elementos.

2.2 Filas

Em contrapartida, a fila é uma estrutura de acesso restrito baseada na política FIFO (*First In, First Out*), na qual o primeiro elemento inserido é o primeiro a ser removido. Tal característica a torna adequada para cenários que exigem processamento em ordem cronológica, como sistemas de atendimento ao cliente ou gerenciamento de tarefas (TANENBAUM, 2015). As operações canônicas de uma fila são:

- **Enqueue (Enfileirar):** Adiciona um novo elemento ao final (cauda) da fila.
- **Dequeue (Desenfileirar):** Remove e retorna o elemento do início (cabeça) da fila.

2.3 Comparativo: Listas x Filas

A escolha entre lista e fila resume-se a um trade-off fundamental entre flexibilidade disciplina. Embora ambas sejam estruturas de dados lineares, suas políticas de acesso e aplicações são praticamente opostas.

A lista é uma estrutura de propósito geral. Ela permite total liberdade para acessar, inserir ou remover elementos em qualquer posição, sendo ideal para coleções que o usuário manipula constantemente, como uma lista de tarefas ou os itens em um carrinho de compras.

A fila, por outro lado, é uma estrutura especializada que impõe a disciplina FIFO (*FirstIn, First-Out*). O acesso é restrito: a inserção (`enqueue`) ocorre apenas no final e a remoção (`dequeue`) apenas no início. Essa rigidez é sua maior força, sendo essencial para garantir a ordem cronológica e a justiça no processamento de tarefas, como em uma fila de impressão ou no gerenciamento de requisições a um servidor.

Em suma, enquanto listas oferecem versatilidade para gerenciar dados, filas garantem a

ordem no fluxo de processos.

3. Metodologia

A implementação do sistema foi conduzida em ambiente de console, com o objetivo de atender às restrições acadêmicas quanto ao uso de estruturas de dados. Assim, listas foram utilizadas como elemento central, tanto para o gerenciamento do cardápio quanto para o controle dos pedidos.

3.1 Estrutura do Sistema

O sistema foi organizado em módulos independentes, de modo a facilitar a implementação e a manutenção do código.

- **Gerenciamento de Itens:** responsável por cadastrar, consultar e atualizar os produtos disponíveis. Os dados são armazenados em listas, que simulam um vetor dinâmico.
- **Gerenciamento de Pedidos:** os pedidos são vinculados a clientes e organizados em três listas que funcionam como filas (pendentes, aceitos e prontos), respeitando a disciplina FIFO.
- **Consultas e Relatórios:** permite a visualização do histórico de pedidos e a filtragem por status de processamento.

Essa organização modular favoreceu a clareza na implementação e o controle sobre o fluxo de dados.

4. Resultados e Discussões

Durante a execução prática do sistema, observou-se o funcionamento adequado das listas que representam o cardápio e das filas que controlam o fluxo de pedidos. Os principais resultados obtidos foram:

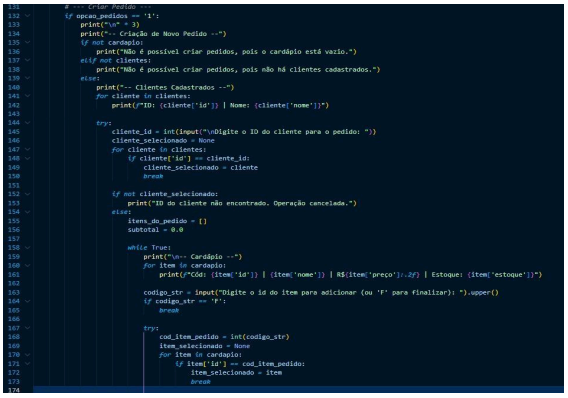
- Cadastro e edição de itens no cardápio:

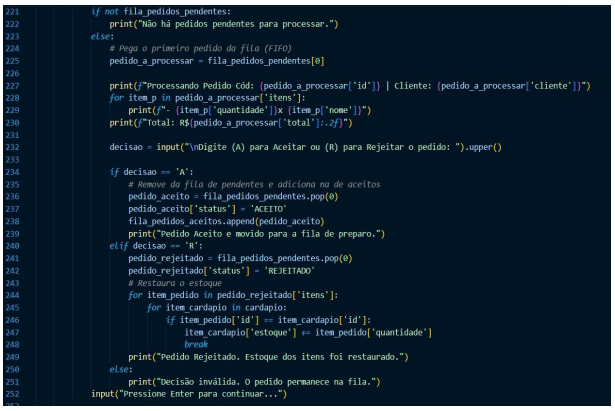
Este código demonstra como um novo item é adicionado à lista cardapio, exemplificando a flexibilidade da estrutura para gerenciar dados

```python
    # === MÓDULO 1: GERENCIAR MENU DE ITENS =========
    if opcao_principal == '1':
        print("\n" * 3)
        print("--- Gerenciar Menu de Itens ---")
        print("1. Cadastrar Item")
        print("2. Consultar Itens")
        print("3. Atualizar Item")
        print("4. Voltar ao Menu Principal")
        opcao_itens = input("Escolha uma opção: ")

        # --- Cadastrar Item ---
        if opcao_itens == '1':
            print("\n" * 3)
            print("-- Cadastro de Novo Item --")
            nome = input("Nome do item: ")
            descricao = input("Descrição do item: ")
            try:
                preco = float(input("Preço (ex: 45.50): "))
                estoque = int(input("Quantidade em estoque: "))

                # Cria o item
                novo_item = {
                    'id': proximo_codigo_item,
                    'nome': nome,
                    'descrição': descricao,
                    'preço': preco,
                    'estoque': estoque
                }
                cardapio.append(novo_item)
                proximo_codigo_item += 1
                print(f"\nItem '{nome}' cadastrado com sucesso! id: {novo_item['id']}")
            except ValueError:
                print("\nERRO: Preço e estoque devem ser números. Operação cancelada.")
            input("Pressione Enter para continuar...")
```

- Criação de pedidos associados a clientes, respeitando a disciplina FIFO;

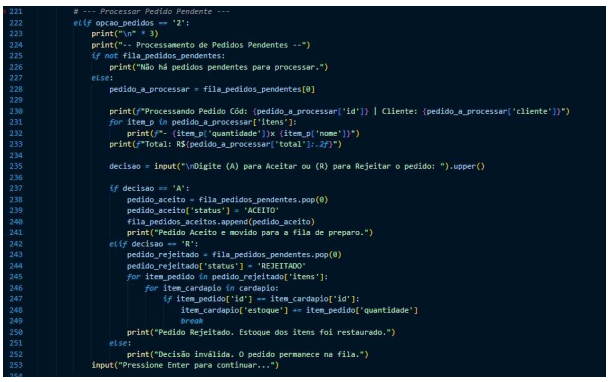
Este bloco é o mais importante, pois mostra as duas operações principais de uma fila: adicionar no final (append) e remover do início (pop(0)), o que garante a disciplina FIFO.





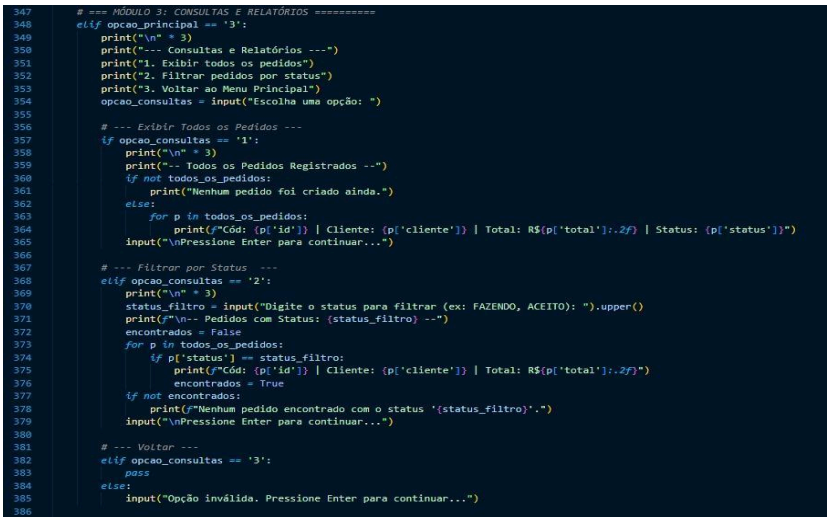
- Evolução consistente dos status dos pedidos (pendente, aceito, em preparo, finalizado);

Este código ilustra como um pedido transita entre diferentes filas, o que garante a consistência do fluxo e evita transições de status incorretas.



- Realização de consultas gerais e filtradas conforme a necessidade do usuário.

Este trecho final ilustra a lógica de consulta de dados na lista todos_os_pedidos. O código permite gerar relatórios filtrando os pedidos por qualquer um dos status definidos no sistema (como ACEITO, FAZENDO ou REJEITADO). Essa funcionalidade é a base para uma interface de relatórios mais complexa.



4.1 Desafios Encontrados

O principal desafio consistiu em manter a consistência no fluxo dos pedidos, garantindo que não houvesse transições incorretas entre os status. Esse aspecto exigiu atenção especial na lógica de verificação.

4.2 Avaliação da Abordagem

A experiência demonstrou que listas são estruturas eficientes para simular diferentes componentes dentro de um mesmo sistema. O maior desafio encontrado foi garantir a integridade do fluxo de estados dos pedidos, evitando transições incorretas entre as filas.

5. Considerações finais

O desenvolvimento deste projeto proporcionou uma experiência enriquecedora, tanto no aspecto técnico quanto no de aprendizado coletivo. Um dos pontos mais desafiadores foi garantir a consistência no fluxo dos pedidos, evitando transições incorretas entre os diferentes status. Esse desafio exigiu uma atenção especial à lógica do sistema e ao controle das estruturas de dados, reforçando a importância da disciplina FIFO no uso das filas.

Por outro lado, foi interessante observar como listas, quando bem aplicadas, podem assumir diferentes papéis dentro de um mesmo sistema — desde o gerenciamento de itens do cardápio até o controle do fluxo de pedidos. Essa versatilidade demonstrou, na prática, a relevância do estudo das estruturas de dados para resolver problemas computacionais do dia a dia.

Se tivéssemos mais tempo disponível para o desenvolvimento, implementaríamos funcionalidades adicionais que tornariam o sistema mais completo e próximo de um software real, Entre elas, destacam-se: o cadastro detalhado de clientes, relatórios de vendas automatizados e, eventualmente, uma interface gráfica que tornaria a utilização mais intuitiva e acessível.

Em suma, o projeto cumpriu seu propósito de consolidar o entendimento sobre listas e filas, mas também abriu caminhos para reflexões sobre como sistemas simples podem evoluir em direção a soluções mais complexas e robustas.

6. Referências

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to Algorithms. 3. ed. Cambridge, MA: MIT Press, 2009.

(Título em português: Algoritmos: Teoria e Prática)

LAFORE, R. Data Structures & Algorithms in Java. 2. ed. Indianapolis, IN: Sams Publishing, 2012.

TANENBAUM, A. S.; BOS, H. Modern Operating Systems. 4. ed. Upper Saddle River, NJ: Pearson Education, 2015.

(Título em português: Sistemas Operacionais Modernos)

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: Teoria e
Prática. 3.

Ed. Rio de Janeiro: Elsevier, 2012.

WIRTH, N. Algoritmos + Estruturas de Dados = Programas. 1. Ed. Rio de Janeiro: LTC, 1985.

SEDGEWICK, R.; WAYNE, K. Algorithms. 4. Ed. Boston:

Addison-Wesley, 2011. TANENBAUM, A. S. Estruturas de Dados Usando C. São

Paulo: Pearson, 2009.

LAUDON, K. C.; LAUDON, J. P. Sistemas de Informação Gerenciais. 12. Ed. São Paulo: Pearson, 201