

Relatório Técnico - Científico

Sistema de Gerenciamentos de Pedidos

Tia Lu Food Delivery

Grupo Pernambuco

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

Sistemas de informação – Centro Universitário de Excelência (UNEX)
Caixa Postal 45020-510 – Vitória 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 e 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 (*First In, 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 cardápio, exemplificando a flexibilidade da estrutura para gerenciar dados

```
30 # === MÓDULO 1: GERENCIAR MENU DE ITENS =====
31 if opcao_principal == '1':
32     print("\n" * 3)
33     print("--- Gerenciar Menu de Itens ---")
34     print("1. Cadastrar Item")
35     print("2. Consultar Item")
36     print("3. Atualizar Item")
37     print("4. Voltar ao Menu Principal")
38     opcao_itens = input("Escolha uma opção: ")
39
40 # --- Cadastrar Item ---
41 if opcao_itens == '1':
42     print("\n" * 3)
43     print("--- Cadastro de Novo Item ---")
44     nome = input("Nome do item: ")
45     descricao = input("Descrição do item: ")
46     try:
47         preco = float(input("Preço (ex: 45.50): "))
48         estoque = int(input("Quantidade em estoque: "))
49
50         # Cria o item
51         novo_item = {
52             'id': proximo_codigo_item,
53             'nome': nome,
54             'descricao': descricao,
55             'preco': preco,
56             'estoque': estoque
57         }
58         cardapio.append(novo_item)
59         proximo_codigo_item += 1
60         print(f"Item '{nome}' cadastrado com sucesso! id: {novo_item['id']}")
61     except ValueError:
62         print("\nERRO: Preço e estoque devem ser números. Operação cancelada.")
63     input("Pressione Enter para continuar...")
64
```

- 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.

```

121 # --- Criar Pedido ---
122 if opcao_pedido == '1':
123     print("\n" * 3)
124     print("Criação de Novo Pedido --")
125     if not cardapio:
126         print("Não é possível criar pedidos, pois o cardápio está vazio.")
127     elif not clientes:
128         print("Não é possível criar pedidos, pois não há clientes cadastrados.")
129     else:
130         print("--- Clientes Cadastrados ---")
131         for cliente in clientes:
132             print(f"ID: {cliente['id']} | Nome: {cliente['nome']}")
133
134         try:
135             cliente_id = int(input("Digite o ID do cliente para o pedido: "))
136             cliente_selecionado = None
137             for cliente in clientes:
138                 if cliente['id'] == cliente_id:
139                     cliente_selecionado = cliente
140                     break
141
142             if not cliente_selecionado:
143                 print("ID do cliente não encontrado. Operação cancelada.")
144             else:
145                 itens_do_pedido = []
146                 quantidade = 0.0
147
148                 while True:
149                     print("\n--- Cardápio ---")
150                     for item in cardapio:
151                         print(f"Código: {item['id']} | {item['nome']} | R${item['preco']:.2f} | Estoque: {item['estoque']}")
152
153                     codigo_str = input("Digite o ID do item para adicionar (ou 'F' para finalizar): ").upper()
154                     if codigo_str == 'F':
155                         break
156
157                     try:
158                         cod_item_pedido = int(codigo_str)
159                         item_selecionado = None
160                         for item in cardapio:
161                             if item['id'] == cod_item_pedido:
162                                 item_selecionado = item
163                                 break
164
165                         if not item_selecionado:
166                             print("Código de item não encontrado. Operação cancelada.")
167                     except ValueError:
168                         print("Código inválido. Operação cancelada.")
169
170                 if item_selecionado:
171                     itens_do_pedido.append(item_selecionado)
172                     quantidade += item_selecionado['estoque']
173
174                 print(f"Quantidade total: {quantidade}")
175                 decisao = input("Digite (A) para Aceitar ou (R) para Rejeitar o pedido: ").upper()
176
177                 if decisao == 'A':
178                     # Remove o item da fila de pendentes e adiciona na de aceitos
179                     pedido_aceito = fila_pedidos_pendentes.pop(0)
180                     pedido_aceito['status'] = 'ACEITO'
181                     fila_pedidos_aceitos.append(pedido_aceito)
182                     print("Pedido Aceito e movido para a fila de preparo.")
183                 elif decisao == 'R':
184                     pedido_rejeitado = fila_pedidos_pendentes.pop(0)
185                     pedido_rejeitado['status'] = 'REJEITADO'
186                     # Restituir o estoque
187                     for item_pedido in pedido_rejeitado['itens']:
188                         for item_cardapio in cardapio:
189                             if item_pedido['id'] == item_cardapio['id']:
190                                 item_cardapio['estoque'] += item_pedido['quantidade']
191                                 break
192                     print("Pedido Rejeitado. Estoque dos itens foi restaurado.")
193                 else:
194                     print("Decisão inválida. O pedido permanece na fila.")
195
196                 input("Pressione Enter para continuar...")

```

```

211 if not fila_pedidos_pendentes:
212     print("Não há pedidos pendentes para processar.")
213 else:
214     # Pega o primeiro pedido da fila (FIFO)
215     pedido_a_processar = fila_pedidos_pendentes[0]
216
217     print(f"Processando Pedido Cód: {pedido_a_processar['id']} | Cliente: {pedido_a_processar['cliente']}")
218     for item_p in pedido_a_processar['itens']:
219         print(f"Item p: {item_p['nome']} | Quantidade: {item_p['quantidade']}")
220     print(f"Total: R${pedido_a_processar['total']:.2f}")
221
222     decisao = input("Digite (A) para Aceitar ou (R) para Rejeitar o pedido: ").upper()
223
224     if decisao == 'A':
225         # Remove da fila de pendentes e adiciona na de aceitos
226         pedido_aceito = fila_pedidos_pendentes.pop(0)
227         pedido_aceito['status'] = 'ACEITO'
228         fila_pedidos_aceitos.append(pedido_aceito)
229         print("Pedido Aceito e movido para a fila de preparo.")
230     elif decisao == 'R':
231         pedido_rejeitado = fila_pedidos_pendentes.pop(0)
232         pedido_rejeitado['status'] = 'REJEITADO'
233         # Restituir o estoque
234         for item_pedido in pedido_rejeitado['itens']:
235             for item_cardapio in cardapio:
236                 if item_pedido['id'] == item_cardapio['id']:
237                     item_cardapio['estoque'] += item_pedido['quantidade']
238                     break
239             print("Pedido Rejeitado. Estoque dos itens foi restaurado.")
240     else:
241         print("Decisão inválida. O pedido permanece na fila.")
242
243     input("Pressione Enter para continuar...")

```

- 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.

```

211 # --- Processar Pedido Pendente ---
212 if opcao_pedido == '2':
213     print("\n" * 3)
214     print("Processamento de Pedidos Pendentes --")
215     if not fila_pedidos_pendentes:
216         print("Não há pedidos pendentes para processar.")
217     else:
218         pedido_a_processar = fila_pedidos_pendentes[0]
219
220         print(f"Processando Pedido Cód: {pedido_a_processar['id']} | Cliente: {pedido_a_processar['cliente']}")
221         for item_p in pedido_a_processar['itens']:
222             print(f"Item p: {item_p['nome']} | Quantidade: {item_p['quantidade']}")
223         print(f"Total: R${pedido_a_processar['total']:.2f}")
224
225         decisao = input("Digite (A) para Aceitar ou (R) para Rejeitar o pedido: ").upper()
226
227         if decisao == 'A':
228             pedido_aceito = fila_pedidos_pendentes.pop(0)
229             pedido_aceito['status'] = 'ACEITO'
230             fila_pedidos_aceitos.append(pedido_aceito)
231             print("Pedido Aceito e movido para a fila de preparo.")
232         elif decisao == 'R':
233             pedido_rejeitado = fila_pedidos_pendentes.pop(0)
234             pedido_rejeitado['status'] = 'REJEITADO'
235             for item_pedido in pedido_rejeitado['itens']:
236                 for item_cardapio in cardapio:
237                     if item_pedido['id'] == item_cardapio['id']:
238                         item_cardapio['estoque'] += item_pedido['quantidade']
239                         break
240             print("Pedido Rejeitado. Estoque dos itens foi restaurado.")
241         else:
242             print("Decisão inválida. O pedido permanece na fila.")
243
244         input("Pressione Enter para continuar...")

```

```

30 # --- MÓDULO 1: GERENCIAR MENU DE ITENS ---
31 if opcao_principal == '1':
32     print("\n" * 3)
33     print("Gerenciar Menu de Itens --")
34     print("1. Cadastrar Item")
35     print("2. Consultar Itens")
36     print("3. Atualizar Item")
37     print("4. Voltar ao Menu Principal")
38     opcao_itens = input("Escolha uma opção: ")
39
40     # --- Cadastrar Item ---
41     if opcao_itens == '1':
42         print("\n" * 3)
43         print("Cadastro de Novo Item --")
44         nome = input("Nome do item: ")
45         descricao = input("Descrição do item: ")
46
47         try:
48             preco = float(input("Preço (ex: 45.50): "))
49             estoque = int(input("Quantidade em estoque: "))
50
51             # Cria o item
52             novo_item = {
53                 'id': proximo_codigo_item,
54                 'nome': nome,
55                 'descricao': descricao,
56                 'preco': preco,
57                 'estoque': estoque
58             }
59             cardapio.append(novo_item)
60             proximo_codigo_item += 1
61             print(f"Item '{nome}' cadastrado com sucesso! ID: {novo_item['id']}")
62         except ValueError:
63             print("Erro: Preço e estoque devem ser números. Operação cancelada.")
64
65         input("Pressione Enter para continuar...")

```

- 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.

```

347 # --- MÓDULO 3: CONSULTAS E RELATORIOS ---
348 if opcao_principal == '3':
349     print("\n" * 3)
350     print("Consultas e Relatórios --")
351     print("1. Exibir todos os pedidos")
352     print("2. Filtrar pedidos por status")
353     print("3. Voltar ao Menu Principal")
354     opcao_consultas = input("Escolha uma opção: ")
355
356     # --- Exibir Todos os Pedidos ---
357     if opcao_consultas == '1':
358         print("\n" * 3)
359         print("Todos os Pedidos Registrados --")
360         if not todos_os_pedidos:
361             print("Nenhum pedido foi criado ainda.")
362         else:
363             for p in todos_os_pedidos:
364                 print(f"Cód: {p['id']} | Cliente: {p['cliente']} | Total: R${p['total']:.2f} | Status: {p['status']}")
365             input("Pressione Enter para continuar...")
366
367     # --- Filtrar por Status ---
368     elif opcao_consultas == '2':
369         print("\n" * 3)
370         status_filtro = input("Digite o status para filtrar (ex: FAZENDO, ACEITO): ").upper()
371         print(f"--- Pedidos com Status: {status_filtro} ---")
372         encontrados = False
373         for p in todos_os_pedidos:
374             if p['status'] == status_filtro:
375                 print(f"Cód: {p['id']} | Cliente: {p['cliente']} | Total: R${p['total']:.2f}")
376                 encontrados = True
377         if not encontrados:
378             print("Nenhum pedido encontrado com o status '{status_filtro}'.")
379         input("Pressione Enter para continuar...")
380
381     # --- Voltar ---
382     elif opcao_consultas == '3':
383         pass
384     else:
385         input("Opção inválida. Pressione Enter para continuar...")
386

```

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