Relatório Desenvolvimento de sistema em Python

Manuella dos Santos Araujo - 2586101430

- **Objetivo**

Este relatório apresenta a documentação do sistema desenvolvido em Python para o Coffee Shops Tia Rosa ou Café da Tia Rosa. O sistema visa auxiliar na gestão de produtos, clientes e pedidos de forma simples e intuitiva, considerando que a equipe possui baixa familiaridade com tecnologia. Principais funcionalidades: - Cadastro de produtos - Cadastro de clientes - Registro de pedidos - Exibição de cardápio - Relatórios de vendas O sistema foi desenvolvido em Python utilizando conceitos como funções, listas e dicionários para armazenar dados temporariamente. Também foi utilizado tratamento de exceções para evitar erros durante a execução. A interface é baseada em menus de texto, facilitando o uso por colaboradores sem conhecimento em informática

Exemplo de execução do sistema:

```
=== DEMO: Coffee Shops Tia Rosa - Execução de Exemplo ===
Produtos cadastrados:
  - d2a2f4c3 | Café Expresso | R$ 4.50 | Estoque: 46
  - 50e5c5eb | Cappuccino | R$ 7.00 | Estoque: 28
  - ad0e1420 | Pão de Queijo | R$ 3.50 | Estoque: 34

Clientes cadastrados:
  - abe68d0e | Ana Silva | pontos: 32
  - abe31081 | João Pereira | pontos: 0

Pedidos realizados (resumo):
  - Pedido 2a70b6ee-2 | Cliente: abe68d0e | Total: R$ 16.00 | Itens: 2
  - Pedido a8e5abae-7 | Cliente: None | Total: R$ 10.50 | Itens: 1
  - Pedido 649abbaf-e | Cliente: abe68d0e | Total: R$ 16.00 | Itens: 2
  - Pedido f4d26eed-c | Cliente: None | Total: R$ 10.50 | Itens: 1

Vendas do dia 2025-08-11: total R$ 53.00 em 4 pedidos
```

1. Definição da classe Product

```python
# Classe que representa um produto no sistema
class Product:
    def __init__(self, name: str, price: float, stock: int, description: str = "", category: str = ""):
        self.id = str(uuid.uuid4())[:8]  # Gera um ID único curto para o produto
        self.name = name
        self.price = round(float(price), 2)  # Preço com duas casas decimais
        self.stock = int(stock)  # Quantidade em estoque
        self.description = description
        self.category = category
```

Explicação: Essa classe representa os produtos do café. Cada produto possui um ID único gerado automaticamente, nome, preço, quantidade em estoque, descrição e categoria. O uso de UUID garante que cada produto tenha um identificador exclusivo, facilitando o gerenciamento.

2. Função para adicionar produto no sistema

```python
# Adiciona um produto novo
def add_product(self, name, price, stock, description="", category=""):
    p = Product(name, price, stock, description, category)
    self.products[p.id] = p
    self.save_all()  # Salva as alterações
    return p
```

Explicação: Este método da classe `CoffeeSystem` permite adicionar um novo produto, criando uma instância da classe `Product`, armazenando no dicionário interno e salvando os dados no arquivo JSON para persistência.

3. Classe `Order` e método para adicionar itens

```python
# Classe que representa um pedido
class Order:
    def __init__(self, customer_id: str = None):
        self.id = str(uuid.uuid4())[:10]  # ID único para o pedido
        self.customer_id = customer_id  # ID do cliente (pode ser None para pedido avulso)
        self.items = []  # Lista de itens: cada item é dict com info do produto, quantidade, subtotal
        self.created_at = datetime.datetime.now().isoformat()  # Timestamp da criação

    # Adiciona um item ao pedido
    def add_item(self, product_id, name, unit_price, qty):
        self.items.append({
            "product_id": product_id,
            "name": name,
            "unit_price": round(unit_price,2),
            "quantity": int(qty),
            "subtotal": round(unit_price * qty, 2)
        })
```

Explicação: A classe `Order` representa os pedidos realizados pelos clientes. Permite associar o pedido a um cliente (ou ser avulso), armazenar os itens comprados com seus detalhes e calcular os valores de subtotal.

4. Método para processar e salvar pedidos

```python
# Processa o pedido: verifica estoque, atualiza estoque, salva pedido, adiciona pontos de fidelidade
def place_order(self, order: Order):
    # Verifica se há estoque suficiente para cada item
    for item in order.items:
        pid = item["product_id"]
        qty = item["quantity"]
        if pid not in self.products:
            raise KeyError(f"Produto {pid} não encontrado")
        if self.products[pid].stock < qty:
            raise ValueError(f"Estoque insuficiente para {self.products[pid].name}")
    # Deduz o estoque dos produtos vendidos
    for item in order.items:
        pid = item["product_id"]
        qty = item["quantity"]
        self.products[pid].stock -= qty

    # Salva o pedido
    self.orders[order.id] = order.to_dict()

    # Atualiza pontos de fidelidade (1 ponto por unidade monetária gasta)
    if order.customer_id and order.customer_id in self.customers:
        pts = int(order.total())
        self.customers[order.customer_id].points += pts

    self.save_all()
    return order
```

Explicação: Este método verifica se os produtos do pedido possuem estoque suficiente, atualiza o estoque, salva o pedido e ainda adiciona pontos no programa de fidelidade do cliente, caso ele esteja cadastrado.

5. Exemplo do menu interativo

```
        menu = """
Coffee Shops Tia Rosa - Sistema
1) Listar produtos
2) Adicionar produto
3) Editar produto (por id)
4) Listar clientes
5) Adicionar cliente
6) Realizar pedido (simples)
7) Ver vendas do dia
0) Sair
"""
```

Explicação: O sistema conta com um menu interativo simples no terminal que permite ao usuário navegar pelas principais funcionalidades, como gerenciar produtos, clientes, realizar pedidos e visualizar vendas do dia, garantindo uma interface acessível para colaboradores com pouca familiaridade tecnológica.

- **Resumo do funcionamento geral**
  1. O sistema armazena produtos, clientes e pedidos em arquivos JSON, garantindo persistência.
  2. Cada produto, cliente e pedido recebe um ID único curto (gerado com uuid.uuid4()).
  3. O menu interativo permite gerenciar produtos e clientes, além de criar pedidos.
  4. Ao criar um pedido, o sistema verifica estoque disponível, atualiza estoque, registra o pedido e adiciona pontos de fidelidade ao cliente, se houver.
  5. Há uma função de demonstração que cria dados fictícios e realiza operações para mostrar o fluxo do sistema.
  6. O sistema é todo rodado pela linha de comando, simples e direto, pensando em usabilidade para a equipe da cafeteria.

- **Conclusão e aprendizado**

Durante o desenvolvimento do sistema para o Coffee Shops Tia Rosa ou Café da Tia Rosa, pude aplicar e consolidar diversos conceitos estudados ao longo das aulas do curso. A criação desse projeto permitiu uma visão prática do uso da linguagem Python para resolver problemas reais, principalmente no contexto de sistemas de gestão simples e eficientes. Desde as primeiras aulas, com a introdução aos algoritmos e ao raciocínio lógico, aprendi a estruturar a lógica necessária para modelar as funcionalidades do

sistema, como o cadastro e gerenciamento de produtos, clientes e pedidos. A familiarização com o ambiente de desenvolvimento Python e a instalação de bibliotecas foi fundamental para organizar o projeto e trabalhar com recursos externos, como a manipulação de arquivos JSON para persistência dos dados.

Nos módulos dedicados a tipos de dados, variáveis e operadores, foi possível entender a importância de representar corretamente as informações, garantindo que valores como preços, quantidades e identificadores fossem tratados de forma apropriada. A utilização dos comandos condicionais `if` e os laços `while` e `for` possibilitaram criar menus interativos e iterar sobre coleções de dados, como listas de produtos e pedidos, de maneira dinâmica e eficiente. A aplicação dos conceitos de funções e classes foi essencial para a organização do código, promovendo reutilização, modularidade e clareza. As classes `Product`, `Customer` e `Order` ilustram como encapsular dados e comportamentos relacionados, enquanto a classe `CoffeeSystem` gerencia a integração entre eles, facilitando a manutenção e evolução do sistema.

Portanto, os conhecimentos sobre gravação e leitura de arquivos foram aplicados para implementar a persistência dos dados, permitindo que as informações cadastradas fossem armazenadas e recuperadas entre execuções do programa, um requisito fundamental para qualquer sistema de gestão. Com esse projeto, além de reforçar conceitos teóricos, desenvolvi uma solução simples e funcional que atende às necessidades do Coffee Shops Tia Rosa, demonstrando como a programação pode ser uma ferramenta poderosa para melhorar processos do dia a dia, mesmo para usuários com pouca familiaridade tecnológica.

Link do repositório no GitHub: https://github.com/mwnu-ctrl/tia_rosa_caf-