Sistema de Estoque: Fase Final

Read more →

Contextualização

Na fase inicial tivemos nossa primeira reunião com nossa cliente para saber sobre seu negócio e como funciona, onde a cliente comentou sobre a necessidade de um sistema de gerenciamento de estoque, após isso, já com a ideia de projeto em mãos juntamente a aprovação e vontade da cliente, tivemos uma segunda reunião para discutirmos seus requisitos e suas exigências sobre o sistema para prosseguir com o projeto.

Contextualização

Na fase final, nossa equipe trabalhou na implementação do sistema, codificando todas as funcionalidades conforme os requisitos definidos pela cliente. Durante esse período, também criamos diagramas adicionais para representar melhor a estrutura e o fluxo do sistema. Além disso, realizamos várias atualizações no código, otimizando as funcionalidades e corrigindo pequenos problemas que surgiram durante os testes.

Requisitos do Sistema

- ADICIONAR PRODUTOS
- EDITAR PRODUTOS
- BUSCAR PRODUTO
- CADASTRAR CLIENTES
- FAZER PEDIDOS
- CONSULTAR ESTOQUE

Modelo Conceitual

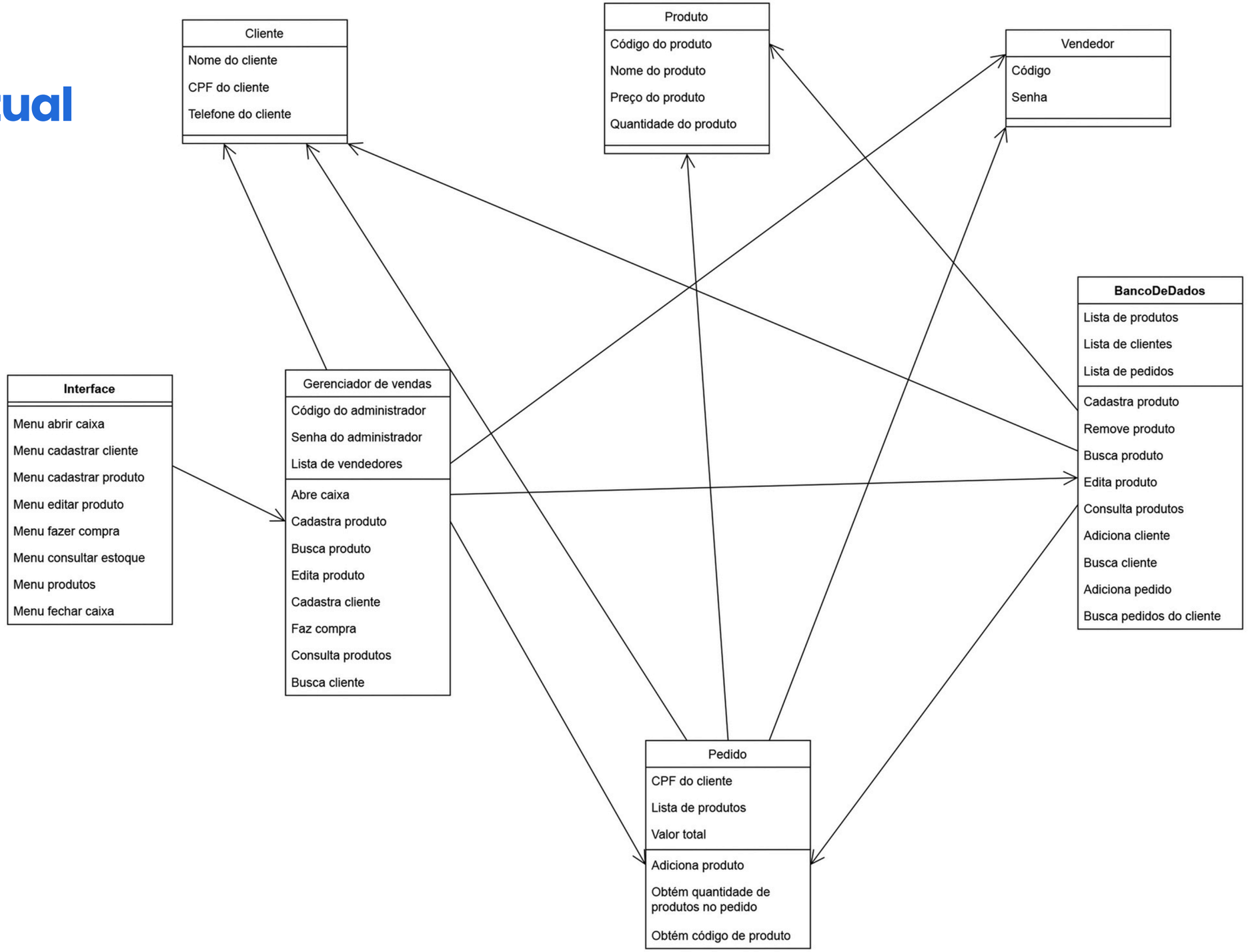


Diagrama de Classes

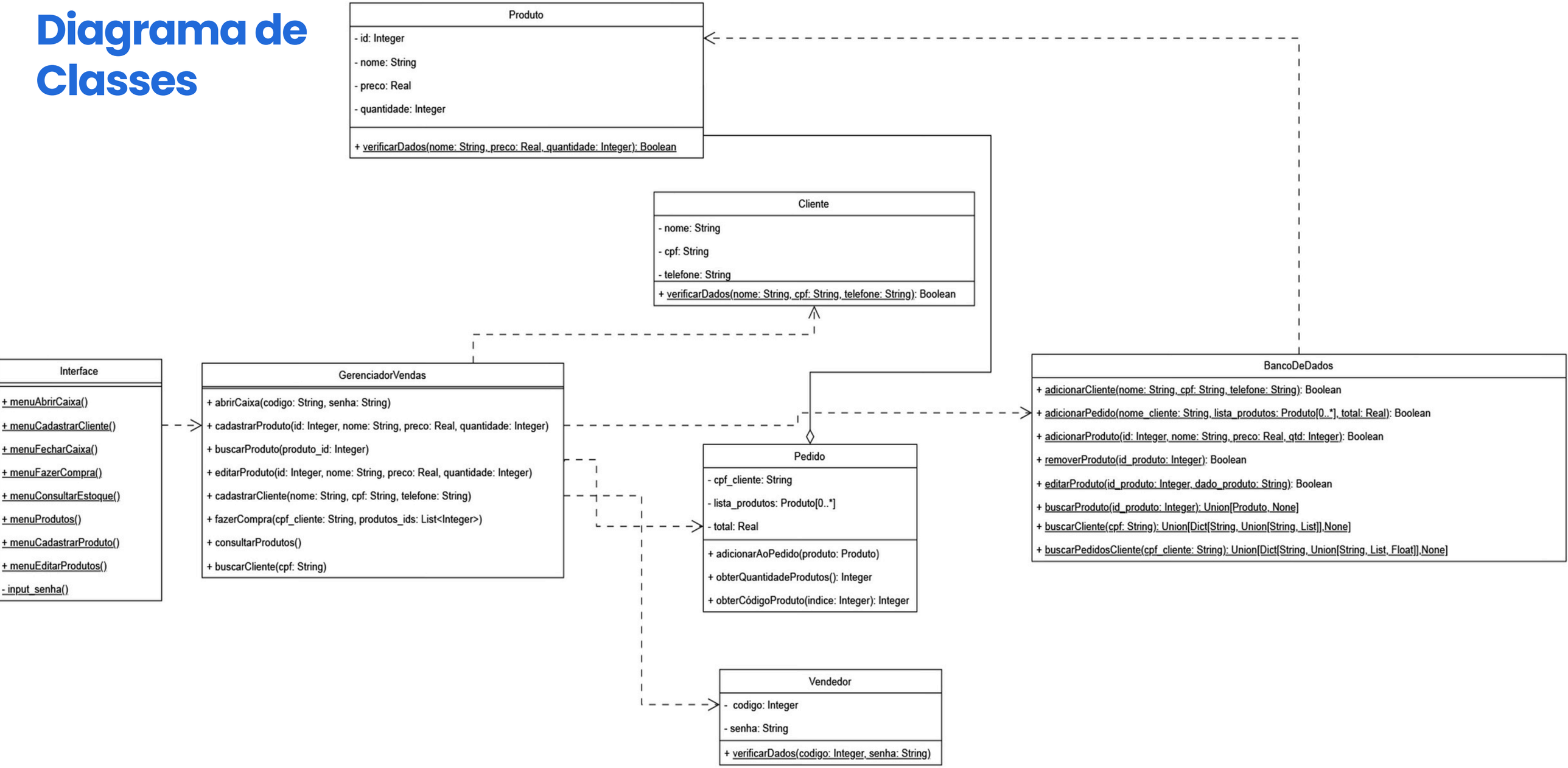


Diagrama de Sequência

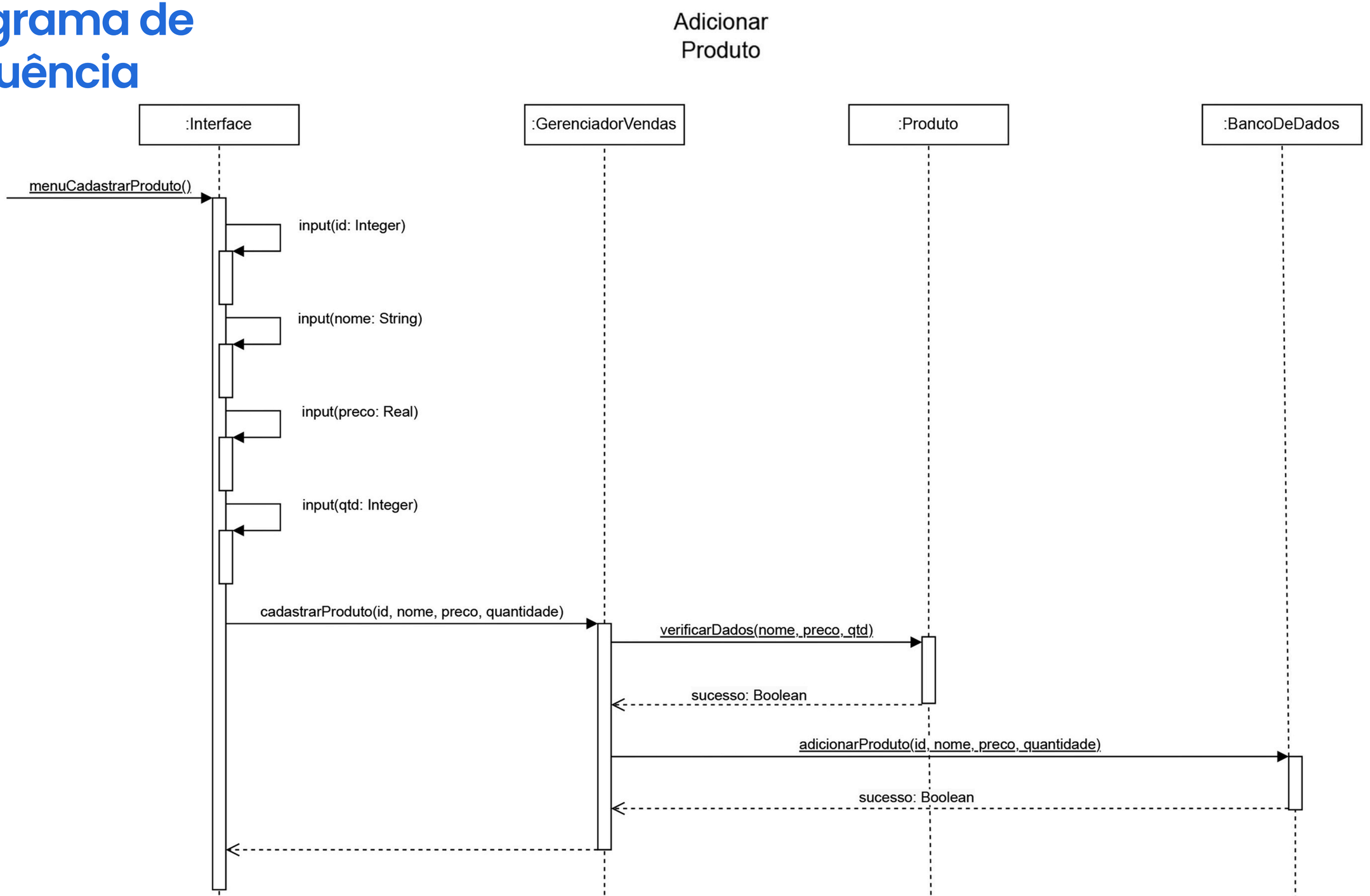
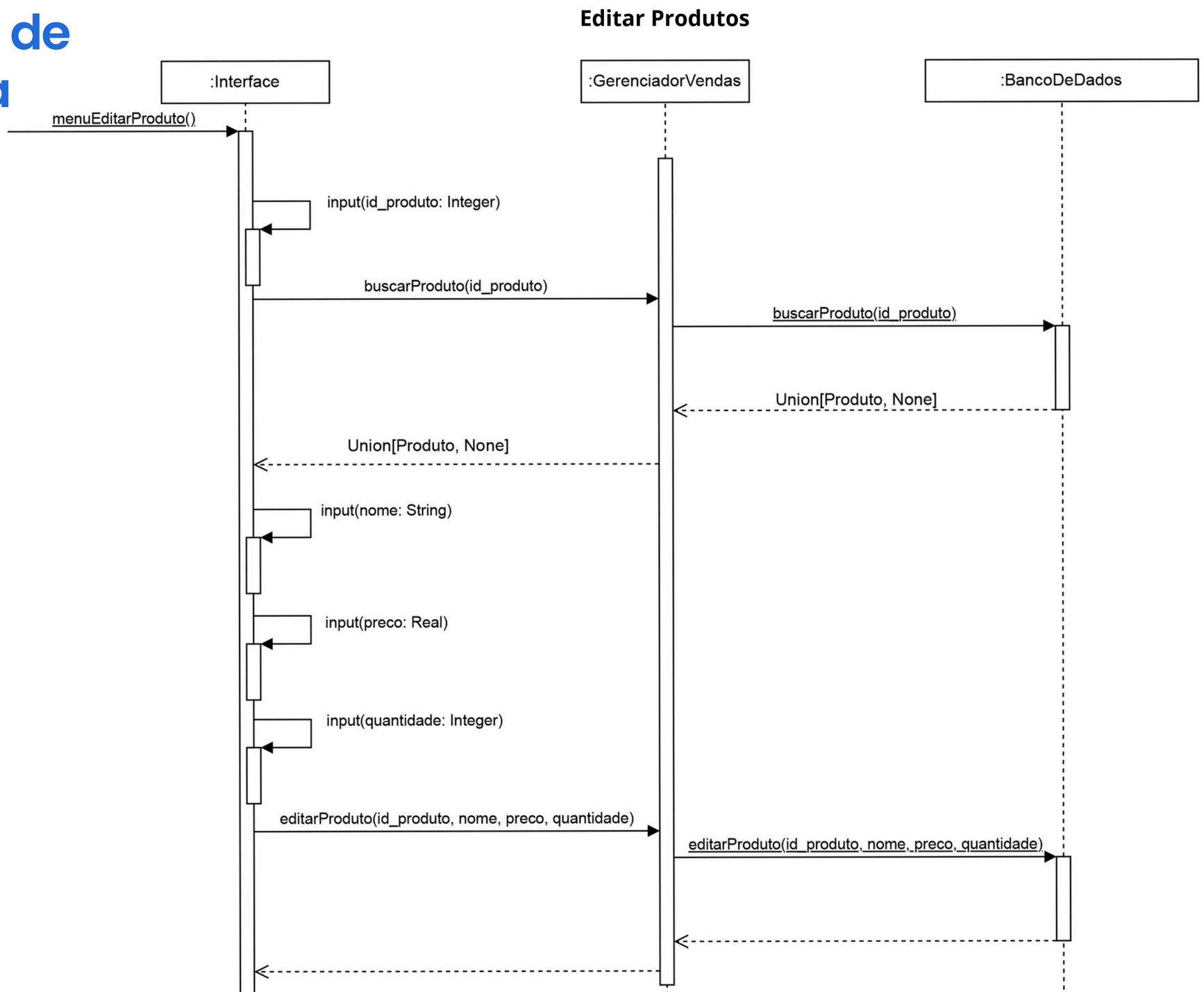


Diagrama de Sequência



```python
class GerenciadorVendas:
    def __init__(self, codigo_admin: str, senha_admin: str, vendedores: list) -> None:
        """
        Inicializa o Gerenciador de Vendas com um administrador e uma lista de vendedores.
        """
        self.__codigo_admin = codigo_admin
        self.__senha_admin = senha_admin
        self.__vendedores = vendedores  # Lista de vendedores passada no momento da inicialização
        self.__login = False  # Indica se o vendedor está logado ou não

    def abrirCaixa(self, codigo: int, senha: str) -> bool:
        """
        Realiza o login e abre o caixa se as credenciais forem válidas.
        Verifica se o vendedor existe na lista e as credenciais são corretas.
        """
        # Verifica se é o administrador
        if codigo == self.__codigo_admin and senha == self.__senha_admin:
            self.__login = True
            return True

        # Verifica se o vendedor existe na lista e se a senha está correta
        for vendedor in self.__vendedores:
            if vendedor.obter_codigo() == codigo and vendedor.obter_senha() == senha:
                self.__login = True
                return True

        # Se não encontrou vendedor válido ou dados errados
        return False
```

```python
@staticmethod
def menuAbrirCaixa(gerenciador: GerenciadorVendas, codigo_admin: str, se
    """Menu para abrir o caixa (realizar login do vendedor)."""
    print("\n--- Abrir Caixa ---")
    codigo = (input("Digite o código do vendedor: "))  # Agora esperamos um
    senha = Interface._input_senha()

    # Verifica se é administrador
    if codigo == codigo_admin and senha == senha_admin:
        print("Acesso de administrador concedido.")
        return True  # Retorna True para login bem-sucedido

    # Verifica login do vendedor usando o Gerenciador
    if gerenciador.abrirCaixa(codigo, senha):
        print("Login bem-sucedido como vendedor.")
        return True  # Retorna True para login bem-sucedido
    else:
        print("Erro ao abrir o caixa. Verifique o código e a senha e tente novame
        return False  # Retorna False para login falho
```

```python
def main() -> None:
    print("Bem-vindo ao Sistema de Gerenciamento de Vendas!")

    # Credenciais de administrador
    codigo_admin = "admin"
    senha_admin = "admin123"
    # Criando vendedores diretamente
    vendedor1 = Vendedor("101", "senha101")

    # Instância do GerenciadorVendas
    gerenciador = GerenciadorVendas(codigo_admin, senha_admin,[vendedor1])

    # Instância da interface
    interface = Interface()

    while True:
        if interface.menuAbrirCaixa(gerenciador, codigo_admin, senha_admin):
            break
        else:
            print("Login inválido. Tente novamente.")
```

```python
while True:
    print("\n--- Menu Principal ---")
    print("1. Cadastrar Cliente")
    print("2. Cadastrar Produto")
    print("3. Editar Produto")
    print("4. Consultar Estoque")
    print("5. Fazer Compra")
    print("6. Fechar Caixa e Sair")
    opcao = input("Escolha uma opção: ")

    if opcao == "1":
        interface.menuCadastrarCliente(gerenciador)
    elif opcao == "2":
        interface.menuCadastrarProduto(gerenciador)
    elif opcao == "3":
        interface.menuEditarProduto(gerenciador)
    elif opcao == "4":
        interface.menuConsultarEstoque(gerenciador)
    elif opcao == "5":
        interface.menuFazerCompra(gerenciador)
    elif opcao == "6":
        interface.menuFecharCaixa()
        print("Encerrando o sistema. Até mais!")
        break
    else:
        print("Opção inválida. Escolha novamente.")
```

```python
class BancoDeDados:
    __produtos: Dict[int, Produto] = {}  # Armazena produtos por ID
    __clientes: Dict[str, Dict[str, Union[str, List[Dict]]]] = {}  # Clientes por CPF
    __pedidos: List[Dict[str, Union[str, List[Produto], float]]] = []

    @staticmethod
    def adicionarProduto(id: int, nome: str, preco: float, quantidade: int) -> bool:...

    @staticmethod
    def removerProduto(id: int) -> bool:...

    @staticmethod
    def buscarProduto(id: int) -> Union[Produto, None]:...

    @staticmethod
    def editarProduto(id: int, nome: str = "", preco: float = None, quantidade: int = None) -> bool:...

    @staticmethod
    def consultarProdutos() -> Dict[int, Produto]:...

    @staticmethod
    def adicionarCliente(nome: str, cpf: str, telefone: str) -> bool:...

    @staticmethod
    def buscarCliente(cpf: str) -> Union[Dict[str, Union[str, List]], None]:...
```

```python
class Produto:
    def __init__(self, id: int, nome: str, preco: float, quantidade: int):
        """Inicializa os atributos da classe Produto."""
        self.__id = id
        self.__nome = nome
        self.__preco = preco
        self.__quantidade = quantidade

    # Métodos para obter os dados do produto
    def obter_id(self) -> int:
        return self.__id

    def obter_nome(self) -> str:
        return self.__nome

    def obter_preco(self) -> float:
        return self.__preco

    def obter_quantidade(self) -> int:
        return self.__quantidade

    # Métodos para definir os dados do produto
    def definir_nome(self, nome: str) -> None:
        self.__nome = nome

    def definir_preco(self, preco: float) -> None:
        self.__preco = preco

    def definir_quantidade(self, quantidade: int) -> None:
        self.__quantidade = quantidade
```

```python
    @staticmethod
    def adicionarProduto(id: int, nome: str, preco: float, quantidade: int) -> bool:
        """
        Adiciona um novo produto ao banco de dados.
        Retorna True se a operação for bem-sucedida, caso contrário, False.
        """
        if id in BancoDeDados.__produtos:
            print(f"Erro: Já existe um produto com o ID {id}.")
            return False

        if not Produto.verificarDados(nome, preco, quantidade):
            return False

        BancoDeDados.__produtos[id] = Produto(id, nome, preco, quantidade)
        print(f"Produto {nome} adicionado com sucesso!")
        return True
```

```python
    @staticmethod
    def menuCadastrarProduto(gerenciador: GerenciadorVendas) -> None:
        """Exibe o menu para cadastrar um produto."""
        try:
            id = int(input("Digite o ID do produto: "))
            nome = input("Digite o nome do produto: ")
            preco = float(input("Digite o preço do produto: "))
            quantidade = int(input("Digite a quantidade do produto: "))

            # Chama o método do GerenciadorVendas para cadastrar o produto
            gerenciador.cadastrarProduto(id, nome, preco, quantidade)
        except ValueError:
            print("Erro: Entrada inválida. Por favor, insira os dados corretamente.")
```

```python
class Cliente:
    def __init__(self, nome: str, cpf: str, telefone: str) -> None:
        """Inicializa os atributos da classe Cliente."""
        self.__nome = nome
        self.__cpf = cpf
        self.__telefone = telefone

    # Métodos de acesso (Getters)
    def obter_nome(self) -> str:
        return self.__nome

    def obter_cpf(self) -> str:
        return self.__cpf

    def obter_telefone(self) -> str:
        return self.__telefone

    # Métodos de modificação (Setters)
    def alterar_nome(self, nome: str) -> None:
        self.__nome = nome

    def alterar_cpf(self, cpf: str) -> None:
        self.__cpf = cpf

    def alterar_telefone(self, telefone: str) -> None:
        self.__telefone = telefone
```

```python
    @staticmethod
    def adicionarCliente(nome: str, cpf: str, telefone: str) -> bool:
        """
        Adiciona um cliente ao banco de dados.
        Retorna True se bem-sucedido, False caso contrário.
        """
        if cpf in BancoDeDados.__clientes:
            print(f"Erro: Cliente com CPF {cpf} já existe.")
            return False

        BancoDeDados.__clientes[cpf] = {"nome": nome, "telefone": telefone, "pedidos": []}
        print(f"Cliente {nome} adicionado com sucesso.")
        return True
```

```python
@staticmethod
def menuCadastrarCliente(gerenciador: GerenciadorVendas) -> None:
    """Menu para cadastrar um novo cliente."""
    print("\n--- Cadastrar Cliente ---")
    nome = input("Digite o nome do cliente: ")
    cpf = input("Digite o CPF do cliente (somente números): ")
    telefone = input("Digite o telefone do cliente (formato: (XX) XXXXX-XXXX): ")

    gerenciador.cadastrarCliente(nome, cpf, telefone)
```

VAMOS PARA O CÓDIGO

Comentário Final da Cliente

“ Gostaria de agradecer imensamente pelo excelente trabalho e atenção no desenvolvimento do sistema de controle de estoque para a minha loja. O resultado superou todas as minhas expectativas, atendendo perfeitamente a cada requisito que propus. A dedicação, atenção aos detalhes e compromisso de vocês foram fundamentais para o sucesso do projeto. Estou muito satisfeita de conseguir ajudar vocês com o projeto, ser ajudada e confiante de que este sistema será essencial para o crescimento do meu negócio. Obrigada! “

- Néia (Cliente)

Obrigado!

- Carlos H.
- Guilherme F.
- Gustavo A.
- Kendy Outi