Curso: Análise e Desenvolvimento de Sistemas **Turma:** EADADS4A

Professor (a): Francisco Lopes Data: 07/08/2025

Disciplina: Lógica – Algoritmos e programação de computadores

Aluno (a): Lucas Gabriel Silva da Costa **Matrícula:** 2486101783

Atividade Ativa

Github: https://github.com/GandalfMagoDev/Coffe-Shop-Tea-Rosa

Descrição do Sistema: Fiz um sistema de gerenciamento simples e funcional para o Coffe Shop da Tia Rosa. O sistema foi pensado para ser operado por um garçom ou outro atendente do café da Tia Rosa. É executado em linha de comando e possui funcionalidades simples, porém eficazes para o gerenciamento de pedidos, produtos e clientes.

Explicação do Código

main.py > cadastrar_produto

```python
# Carregar a(as) bibliotecas que serão usadas
from tabulate import tabulate


# Criar dicionarios de produtos e clientes pra não inciar os sistema em branco
cardapio = [{ 'id_produto': 1, 'nome': "Cafe Coado", 'preco': 2.0}, { 'id_produto': 2, 'nome': "Expresso", 'p
]

clientes = [{'nome': "Ana", 'cpf': "78925431579"}, {'nome': "Daniel", 'cpf': "15967202361"}, {'nome': "Kleber
]

# Criar um dicionário para armazenar os pedidos do dia
pedidos = []

# Declarar a variável global de ID que será usada pra contar e identificar os produtos
id_produto = max([produto['id_produto'] for produto in cardapio], default = 0)


```

Nessa parte inicial do código, importei a biblioteca *tabulate,* para formatar os dados em tabelas, para melhor visualização.
Fiz também a criação dos dicionários principais, inserindo previamente alguns dados no cardápio e nos clientes para que o sistema não inicie completamente vazio, para fins de testes e apresentação.
Declarei a variável *id_produtos* como: global, para que possa ser utilizada e recuperada em outras funções do sistema.

```python
# Criar a função para cadastrar novos produtos
def cadastrar_produto():
    global id_produto
    # Obter o nome do produto a ser cadastrado
    nome = input("Que produto você quer cadastrar? ")
    # Obter o preço do produto a ser cadastrado
    # Usar try/except para evitar um preço inválido
    try:
        preco = float(input("Qual será o preço desse produto? R$ "))
    except ValueError:
        print("Erro: Preço inválido. Tente novamente.")
        return
    # Incrementar o ID do produto, para que cada produto tenha um ID único
    id_produto += 1
    # Inserir o novo produto no cardapio
    cardapio.append({'id_produto': id_produto, 'nome': nome, 'preco': preco})
    print(f"{nome} cadastrado com sucesso!\n")
```

Aqui criei a função de cadastrar um novo produto, utilizando o *id_produto*, sendo incrementado para atribuir um identificador único para cada produto no cardápio.

```python
# Criar a função usada pra visualizar os produtos do cardápio
def ver_produtos():
    # Criar o cabeçalho da tabela
    cabeçalho = ["ID", "       Nome       ", "Preço"]
    #Criar uma tabela que será usada para exibir os produtos
    tabela_produtos = [
        [produto['id_produto'], produto['nome'], f"R$ {produto['preco']:.2f}"]
        for produto in cardapio
    ]
    # Exibe um título da tabela que será mostrada
    print("\n------ Cardapio Atual ------")
    # Faz a listagem
    print(tabulate(tabela_produtos, headers = cabeçalho, tablefmt = "grid"))
```

Fiz a função de *ver_produtos* que usa a biblioteca *tabulate* para exibir os produtos em formato de tabela, com um cabeçalho para organizar a disposição dos dados e, facilitar a leitura e entendimento por parte do usuário.

```python
# Criar a função pra cadastrar novos clientes
def cadastrar_cliente():
    # Obter o nome do novo cliente
    nome = input("Qual o nome do cliente? ")
    # Obter o número de CPF e evitar que este seja digitado incorretamente
    cpf = input("Digite o cpf do cliente (11 digitos, apenas números):")
    # Checar se o CPF tem 11 dígitos
    if len(str(cpf)) != 11:
        print("Erro: O CPF deve ter exatamente 11 dígitos. Tente novamente.")
        return
    # Comparar o novo CPF para evitar clientes duplicados
    for cliente in clientes:
        if cliente['cpf'] == str(cpf):
            print("Erro: Esse CPF já está cadastrado. Tente novamente.")
            return
    # Se não for repetido, cadastrar o cliente
    novo_cliente = {'nome': nome, 'cpf': cpf}
    # Inserir o novo cliente no sistema
    clientes.append(novo_cliente)
    print(f"Cliente {nome} cadastrado com sucesso!\n")
```

A função *cadastrar_cliente* adiciona novos clientes no dicionário *clientes*, recebendo um nome de cliente, e um CPF que deve ser único, caso o CPF seja repetido o sistema avisa o usuário de que esse cliente já está registrado.

```python
# Criar a função para ver os clientes já cadastrados no sistema
def ver_clientes():
    # Ordenar os clientes dentro da tabela atribuindo um número de ordem (índice)
    cliente_numero = [[i + 1, c['nome'], c['cpf']] for i, c in enumerate(clientes)]
    # Criar o cabeçalho da tabela
    cabecalho = ["Cliente nº", "    Nome    ", "       CPF       "]
    # Título da tabela de clientes
    print("\n----------- Clientes Cadastrados -----------")
    # Mostrar a tabela de clientes cadastrados
    print(tabulate(cliente_numero, headers = cabecalho, tablefmt = "grid"))
```

A função *ver_clientes* gera uma tabela que exibe os clientes já cadastrados no sistema, ordenando-os através de um índice e mostrando os dados: Nome e CPF de todos os clientes cadastrados.

```python
#Criar a função de fazer pedidos
def novo_pedido():
    # Obter o CPF do cliente e comparar com os clientes cadastrados
    cpf = input("Digite o CPF do cliente que está pedindo: ")
    cliente = None
    for c in clientes:
        if c['cpf'] == cpf:
            cliente = c
            break
    # Exibir mensagem de erro caso o CPF não esteja atribuido a um cliente
    if not cliente:
        print("Erro: Cliente não cadastrado. Tente novamente.")
        return
    # Mostra o cardápio para que o atendente realize o pedido
    print(f"\nOlá {cliente['nome']} o que vai querer hoje?")
    ver_produtos()
    # Criar um dicionário para armazenar os produtos pedidos
    produtos_pedidos = []
    # Criar a variável que armazenará o total do pedido
    total_pedido = 0.0
```

Aqui é iniciada a função *novo_pedido*, que deve obter do usuário um cliente e os produtos que esse cliente vai consumir, gerar um resumo de pedido que será visualizado pelo atendente, bem como fazer um registro desse pedido para que ele possa ser recuperado na função *histórico_pedidos*.

Nessa parte o CPF do cliente que deseja fazer um pedido é requisitado e comparado aos CPF's já presentes no sistema, somente clientes cadastrados podem fazer um pedido. Logo em seguida é chamada a função *ver_produtos* que mostrará o cardápio para que o usuário possa selecionar e adicionar produtos ao pedido. É criado também o dicionário: *produtos_pedidos*, para armazenar o pedido que está sendo feito no momento e mostra-lo posteriormente no *resumo* de pedido; e a variável: *total_pedido* que armazena o preço de todos os produtos que formam esse pedido.

```python
    # Loop para adicionar produtos ao pedido
    while True:
        escolha = input("Digite o ID do produto que deseja adiciona-lo ao pedido: ").lower()
        # Verificar se a escolha é válida
        if not escolha.isdigit():
            print("Erro: esse ID não existe. Tente novamente.")
            continue
        # Obter a escolha e puxar o produto pelo ID
        produto_selecionado = next((p for p in cardapio if p['id_produto'] == int(escolha)), None)
        # Adicionar o produto à lista de produtos pedidos
        if produto_selecionado:
            produtos_pedidos.append(produto_selecionado)
            # Atualizar o total do pedido
            total_pedido += produto_selecionado['preco']
            # Confirmar que o produto foi adicionado ao pedido ou informar que não foi encontrado
            print(f"{produto_selecionado['nome']} adicionado ao pedido.")
        else:
            print("Erro: Produto não encontrado. Tente novamente.")
            continue
        # Perguntar se o cliente deseja finalizar o pedido
        finalizar = input("Quer finalizar o pedido? (sim/não): ").lower()
        if finalizar == 'sim':
```

```
            # Mostrar o resumo do pedido em forma de tabela
            print("\n----- Resumo do pedido -----")
            resumo = []
            for p in produtos_pedidos:
                resumo.append([p['nome'], f"R$ {p['preco']:.2f}"])
            print(tabulate(resumo, headers = ["Produto", "Preço/uni"], tablefmt = "grid"))
            print("-------------------------------------------------")
            print(f"{'Preço Total do Pedido':<45} R$ {total_pedido:.2f}")
            print("-------------------------------------------------")
            print(f"Pedido finalizado, Bom apetite {cliente['nome']}!!!")
            # Registrar o pedido no histórico e sai do registro de pedidos
            pedidos.append({
            'cliente': cliente['nome'],
            'cpf': cliente['cpf'],
            'produtos': [p['nome'] for p in produtos_pedidos],
            'total': total_pedido
            })
            print(f"Pedido registrado com sucesso!!!\n")
            break
        elif finalizar == 'não':
            print("Continue seu pedido:\n")
        else:
            print("Opção inválida. Tente novamente.")
```

Aqui o loop para adicionar produtos ao pedido é iniciado, recebendo o *id_produto* para efetuar a escolha do produto e armazenando cada produto adicionado no dicionário *produtos_pedidos*. Após cada seleção o sistema pergunta ao usuário se ele deseja continuar adicionando produtos ao pedido ou finalizar o pedido. O loop continua até que o usuário decida *finalizar* o pedido, quando isso acontece, o *resumo* do pedido é exibido em formato de tabela mostrando os produtos que foram pedidos com os respectivos preços unitários, e logo abaixo dessa tabela e mostrado o valor total do pedido efetuado. A operação de registrar pedido é finalizada e o pedido é armazenado no *histórico_pedidos.*

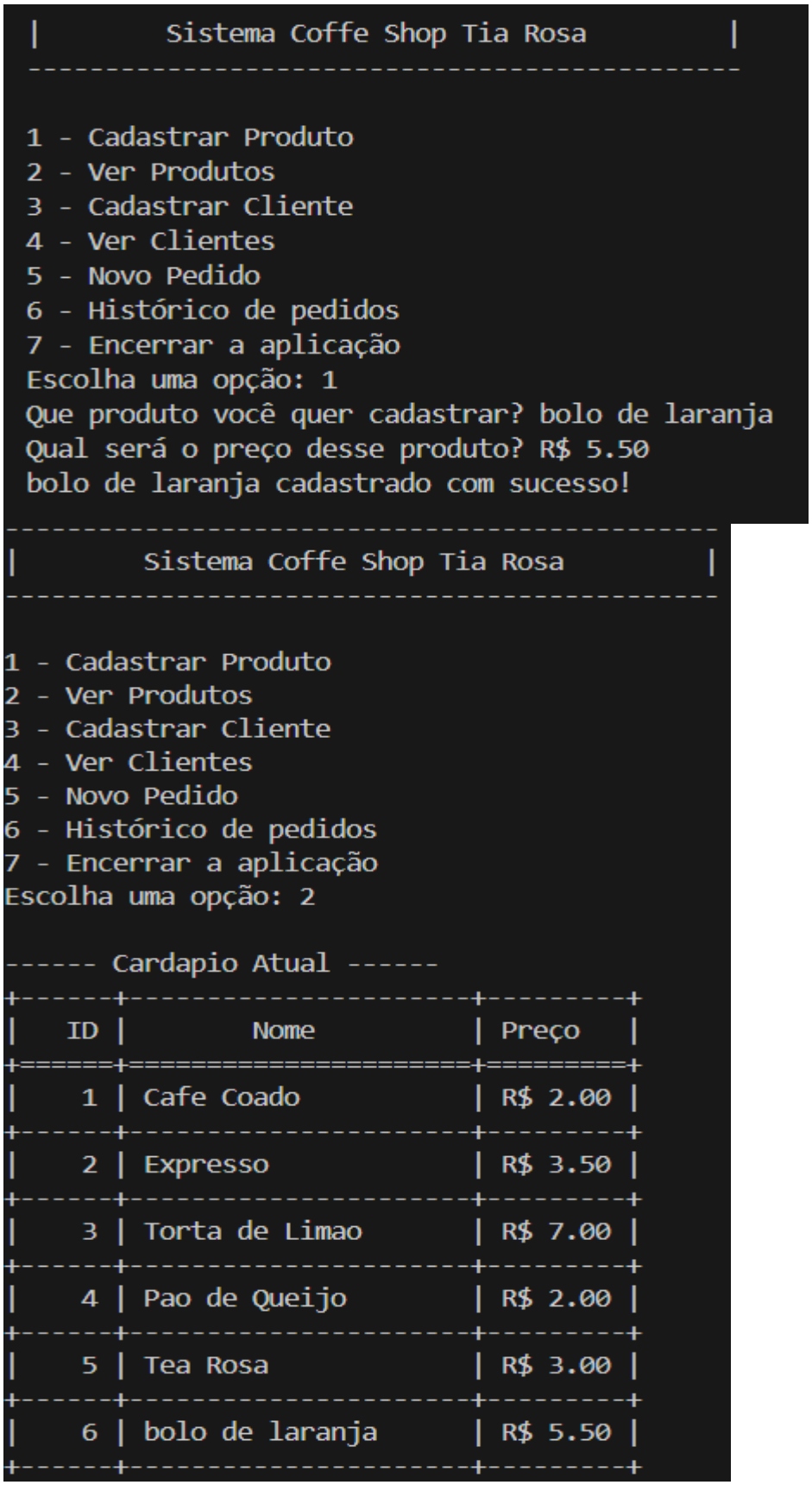
```
# Criar a função para ver o histórico de pedidos
def historico_pedidos():
    if not pedidos:
        print("Nenhum pedido foi feito ainda.")
        return
    # Criar o cabeçalho da tabela
    cabecalho = ["---- Cliente ----", "---- CPF ----", "----- Produtos -----", "-- Total --"]
    # Mostrar o histórico de pedidos
    print("\n----------- Histórico de Pedidos -----------")
    tabela_pedidos = [
        [pedido['cliente'], pedido['cpf'], ', '.join(pedido['produtos']), f"R$ {pedido['total']:.2f}"]
        for pedido in pedidos
    ]
    if pedidos:
        print(tabulate(tabela_pedidos, headers = cabecalho, tablefmt = "grid"))
    # Se não houver pedidos registrados ainda
    else:
        print("Não há pedidos no histórico.")
```

A função *histórico_pedidos* exibe uma tabela contendo os pedidos que foram registrados até então no sistema, organizando os dados para que sejam facilmente interpretados criando um sistema de gerenciamento simples, porém eficaz.

```python
def main():
    # Iniciar o loop do menu
    while True:
        # Mostrar as opçoes para que o usuário possa escolher a funcionalidade que deseja utilizar
        print("\n---------------------------------------------\n|        Sistema Coffe Shop Tia Rosa
        print("1 - Cadastrar Produto")
        print("2 - Ver Produtos")
        print("3 - Cadastrar Cliente")
        print("4 - Ver Clientes")
        print("5 - Novo Pedido")
        print("6 - Histórico de pedidos")
        # Mostrar uma opção para encerrar o sistema
        print("7 - Encerrar a aplicação")
        # Obter a escolha do usuário
        opcao = input("Escolha uma opção: ")
        # Iniciar a função que o usuário vai escolher
        if opcao == "1":
            cadastrar_produto()
        elif opcao == "2":
            ver_produtos()
        elif opcao == "3":
            cadastrar_cliente()
        elif opcao == "4":
            ver_clientes()
        elif opcao == "5":
            novo_pedido()
        elif opcao == "6":
            historico_pedidos()
        # Encerrar o sistema
        elif opcao == "7":
            print("Encerrando, até logo...")
            break
        # Retornar uma mensagem de erro caso o usuário digite um número que não corresponda a nenhuma opção
        else:
            print("!!Opcao invalida. Tente novamente!!\nDica: Digite um número entre 1 e 6\n")
main()
```

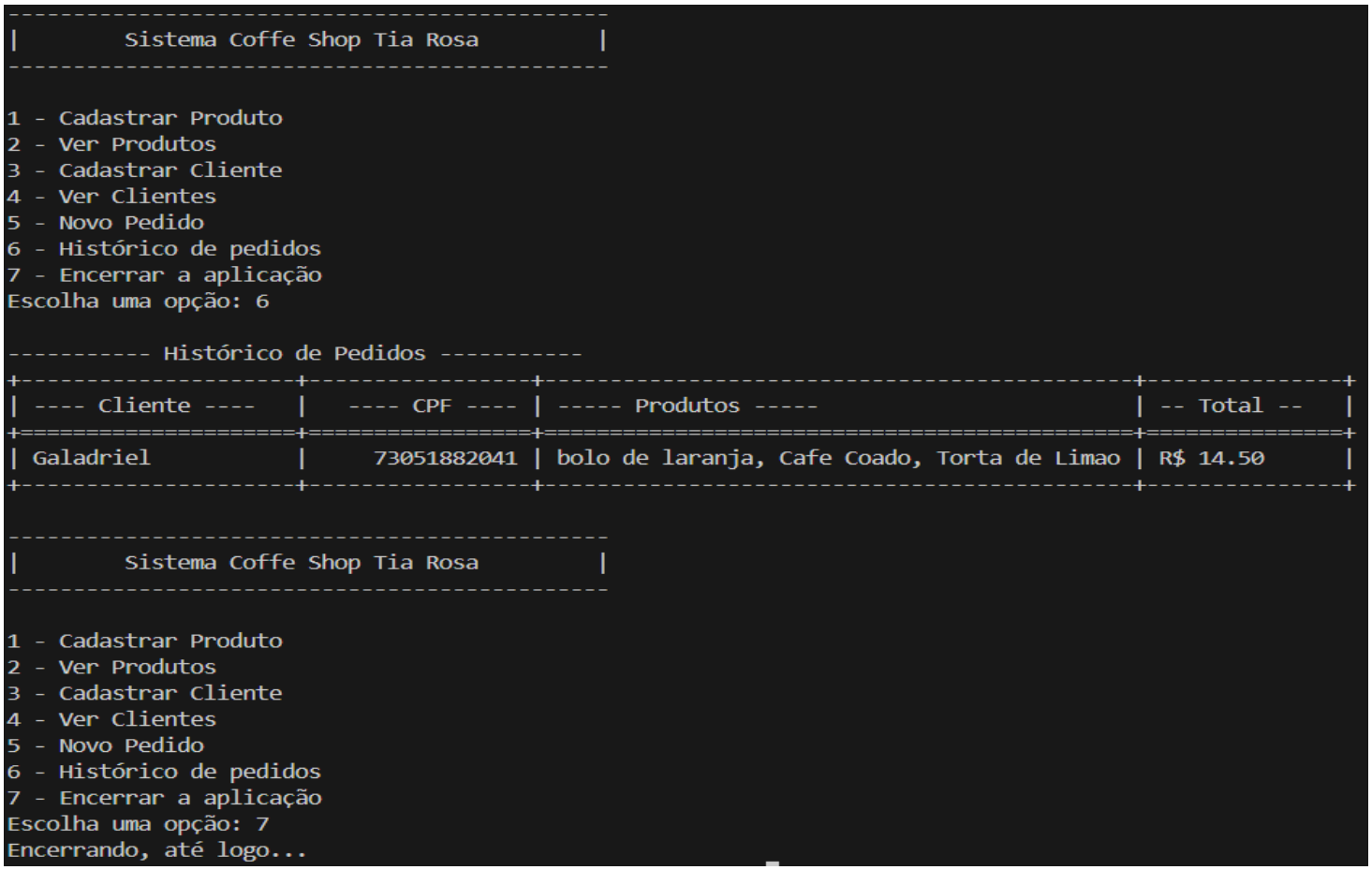
A função *main* é a ponte que conecta todas as funções do sistema ao usuário por meio de uma interface de linha de comando que recebe uma instrução do usuário e realiza a função correspondente.

Prints de funcionamento



```
-----------------------------------------------
|        Sistema Coffe Shop Tia Rosa          |
-----------------------------------------------

1 - Cadastrar Produto
2 - Ver Produtos
3 - Cadastrar Cliente
4 - Ver Clientes
5 - Novo Pedido
6 - Histórico de pedidos
7 - Encerrar a aplicação
Escolha uma opção: 3
Qual o nome do cliente? Patricia
Digite o cpf do cliente (11 digitos, apenas números):58861038257
Cliente Patricia cadastrado com sucesso!


-----------------------------------------------
|        Sistema Coffe Shop Tia Rosa          |
-----------------------------------------------

1 - Cadastrar Produto
2 - Ver Produtos
3 - Cadastrar Cliente
4 - Ver Clientes
5 - Novo Pedido
6 - Histórico de pedidos
7 - Encerrar a aplicação
Escolha uma opção: 3
Qual o nome do cliente? Lucas
Digite o cpf do cliente (11 digitos, apenas números):50297628106
Cliente Lucas cadastrado com sucesso!
-----------------------------------------------
|        Sistema Coffe Shop Tia Rosa          |
-----------------------------------------------

1 - Cadastrar Produto
2 - Ver Produtos
3 - Cadastrar Cliente
4 - Ver Clientes
5 - Novo Pedido
6 - Histórico de pedidos
7 - Encerrar a aplicação
Escolha uma opção: 4

---------- Clientes Cadastrados -----------
+--------------+-----------------+----------------------+
|   Cliente nº |      Nome       |         CPF          |
+==============+=================+======================+
|            1 | Ana             |         78925431579  |
+--------------+-----------------+----------------------+
|            2 | Daniel          |         15967202361  |
+--------------+-----------------+----------------------+
|            3 | Kleber          |         61820973106  |
+--------------+-----------------+----------------------+
|            4 | Galadriel       |         73051882041  |
+--------------+-----------------+----------------------+
|            5 | Patricia        |         58861038257  |
+--------------+-----------------+----------------------+
|            6 | Lucas           |         50297628106  |
+--------------+-----------------+----------------------+
```

```
------------------------------------------------
|        Sistema Coffe Shop Tia Rosa           |
------------------------------------------------

1 - Cadastrar Produto
2 - Ver Produtos
3 - Cadastrar Cliente
4 - Ver Clientes
5 - Novo Pedido
6 - Histórico de pedidos
7 - Encerrar a aplicação
Escolha uma opção: 5
Digite o CPF do cliente que está pedindo: 73051882041

Olá Galadriel o que vai querer hoje?

------ Cardapio Atual ------
+------+----------------------+---------+
|   ID |         Nome         | Preço   |
+======+======================+=========+
|    1 | Cafe Coado           | R$ 2.00 |
+------+----------------------+---------+
|    2 | Expresso             | R$ 3.50 |
+------+----------------------+---------+
|    3 | Torta de Limao       | R$ 7.00 |
+------+----------------------+---------+
|    4 | Pao de Queijo        | R$ 2.00 |
+------+----------------------+---------+
|    5 | Tea Rosa             | R$ 3.00 |
+------+----------------------+---------+
|    6 | bolo de laranja      | R$ 5.50 |
+------+----------------------+---------+
Digite o ID do produto que deseja adiciona-lo ao pedido: 6
bolo de laranja adicionado ao pedido.
Quer finalizar o pedido? (sim/não): não
Continue seu pedido:

Digite o ID do produto que deseja adiciona-lo ao pedido: 1
Cafe Coado adicionado ao pedido.
Quer finalizar o pedido? (sim/não): não
Continue seu pedido:

Digite o ID do produto que deseja adiciona-lo ao pedido: 3
Torta de Limao adicionado ao pedido.
Quer finalizar o pedido? (sim/não): sim

----- Resumo do pedido -----
+-----------------+-------------+
| Produto         | Preço/uni   |
+=================+=============+
| bolo de laranja | R$ 5.50     |
+-----------------+-------------+
| Cafe Coado      | R$ 2.00     |
+-----------------+-------------+
| Torta de Limao  | R$ 7.00     |
+-----------------+-------------+
------------------------------------------------
Preço Total do Pedido                   R$ 14.50
------------------------------------------------
Pedido finalizado, Bom apetite Galadriel!!!
Pedido registrado com sucesso!!!
```



Conclusão e aprendizados

Com a utilização da lógica e da programação é possível transformar e automatizar um algoritmo manual, possivelmente feito com blocos de anotações em papel, e uso de cadernos ou livros que registram as atividades rotineiras do Coffe Shop da Tia Rosa, em um sistema de gerenciamento que continua utilizando os algoritmos no meio digital. Um simples sistema com interface em linha de comando, pode fazer grande diferença na rotina dos funcionários e clientes do Coffe Shop Tia Rosa, os processos antes manuais se tornam mais ágeis, o risco de perda de informação é mitigado, a possibilidade de erro no registro de um cliente ou pedido, por caligrafia, ruim feita em um caderno é completamente eliminada, e a integração do cardápio digital com identificador único para cada produto facilita a adição de novidades para os clientes.

Com relação ao código, as possibilidades são quase infinitas e sempre podem ser melhoradas ou adicionadas novas funcionalidades, por isso, é muito importante manter a organização e usar os comentários para explicar o que cada trecho de código faz, o código limpo e explicado com clareza pode ser facilmente modificado, para realizar correções de bugs e, implementar ou atualizar as funcionalidades do sistema de acordo com a necessidade e conveniência do usuário.