Vestimenta

E-COMMERCE TRADICIONAL

Fundadores: Júlia Andrade, Geovanna Telles, Mathias Basílio e Nicolas Esteves.

```python
"""sistema-cadastro.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1XWtGXJcyTF0mxCokLW8HX8oiJOGc3fbL
"""

class SistemaClientes:
    def __init__(self):
        self._pilha_sessoes = []
        self._clientes = {}

    def registrar(self, email: str, nome: str, senha: str) -> bool:
        if email in self._clientes:
            return False
        self._clientes[email] = {"nome": nome, "senha": senha}
        return True

    def login(self, email: str, senha: str) -> bool:
        dados = self._clientes.get(email)
        if not dados or dados["senha"] != senha:
            return False
        self._pilha_sessoes.append(email)
        return True

    def logout(self) -> str | None:
        if not self._pilha_sessoes:
            return None
        return self._pilha_sessoes.pop()

    def cliente_atual(self) -> str | None:
        return self._pilha_sessoes[-1] if self._pilha_sessoes else None

clientes = {}
```

```python
while True:
    print("1 - Cadastrar cliente")
    print("2 - Mostrar dados do cliente")
    print("3 - Sair")
    opcao = input("Escolha uma opção: ")

    if opcao == "1":
        email = input("Email: ")
        nome = input("Nome: ")
        endereco = input("Endereco: ")
        clientes[email] = {"nome": nome, "endereco": endereco}
        print("Cliente cadastrado com sucesso\n")

    elif opcao == "2":
        email = input("Digite o email do cliente: ")
        if email in clientes:
            dados = clientes[email]
            print(f"Nome: {dados['nome']}")
            print(f"Endereco: {dados['endereco']}\n")
        else:
            print("Cliente nao encontrado\n")

    elif opcao == "3":
        break

    else:
        print("Opcao invalida\n")
```

LOGIN&LOGOUT

Para o login e logout foi utilizado pilha, para facilitar a organização e realização dos logins.

Linha & Luxo

```python
# -*- coding: utf-8 -*-
"""sistema-clientes.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1XWtGXJcyTF0mxCokLW8HX8oiJOGc3fbL
"""

clientes = {}

while True:
    print("1 - Cadastrar cliente")
    print("2   Mostrar dados do cliente")
    print("3 - Sair")
    opcao = input("Escolha uma opção: ")

    if opcao == "1":
        email = input("Email: ")
        nome = input("Nome: ")
        endereco = input("Endereco: ")
        clientes[email]   {"nome": nome, "endereco": endereco}
        print("Cliente cadastrado com sucesso\n")

    elif opcao  = "2":
        email = input("Digite o email do cliente: ")
        if email in clientes:
            dados   clientes[email]
            print(f"Nome: {dados['nome']}")
            print(f"Endereco: {dados['endereco']}\n")
        else.
            print("Cliente nao encontrado\n")

    elif opcao == "3":
        break

    else:
        print("Opcao invalida\n")
```

SISTEMA DO CLIENTE

No sistema do cliente utilizamos a matriz, que é uma lista que pode conter outras listas dentro da mesma. Esse processo facilita a aquisição dos dados e o armazenamento dos dados dos clientes

Linha & Luxo

```
# -*- coding: utf-8 -*-
"""AV60pontos.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1ogAcKxLEHM--pqYr7msX8Yd9dYqlqxUr
"""

print('================+=============================================')
print('                       LISTA DE ROUPAS                       |   ')
print('=============================================================')

roupas_dict = {'blusa de frio': ('19,99', 'cashmere'),
               'blusa de calor': ('17,00', 'algodão'),
               'calça jeans':('30,00', 'jeans'),
               'short': ('15,00', 'jeans'),
               'calca legging':('20,00', 'Poliéster'),
               'luva': ('15,00', 'algodão'),
               'gorro': ('15,00', 'algodão'),
               'chapeu': ('10,00', 'algodão')}

for produto in roupas_dict.keys():
    print(produto)
produto = str(input('Digite o produto que você deseja visualizar: ')).lower()
if produto in roupas_dict:
  print(roupas_dict[produto])

else:
    print("Perdão, ainda não temos esse produto!")

```

LISTA DE ROUPAS

DICT foi utilizado para adicionar elementos dentro das ‘Chaves’ que seriam as roupas. Como é possível ver no código quando você pergunta sobre uma determinada peça o sistema te devolve os seus elementos, como preço do produto e material utilizado na confecção do produto.

Linha & Luxo

```python
# -*- coding: utf-8 -*-
"""LL_solucoes_carrinho_de_compras.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1KZzUh9jGY7-I754jhyPegdRbFoo8DWGe

### **Carrinho de Compras**
"""

class Cart(list):
  def  _init__(self, *args):
    super().__init__(args)
  def add_item(self, item):
    self.append(item)
  def clear_cart(self):
    for i in self:
      self.remove(i)
  def rem_item(self, item):
    try:
      self.remove(item)
    except ValueError:
      print("Item não encontrado")
  def view_cart(self):
    if not self:
      print("Seu carrinho está vazio")
    else:
      for i in self:
       print(i)

meu_carrinho = Cart()
```

CARRINHO DE COMPRAS

Programação Orientada a Objetos utilizando a classe ‘Cart’ para adicionar ou remover itens de uma instância de carrinho por meio das funções definidas, os itens são guardados no carrinho utilizando o modelo de dados Vetor.

Linha & Luxo

FORMAS DE PAGAMENTO

A estrutura de dados utilizada foi Fila para ordenar as formas de pagamento. Desse modo, a primeira pessoa a solicitar o pagamento e pagar também será a pessoa que aguardará menos para que o pagamento seja verificado e aprovado.

```python
#4 concluído
class PaymentMethods:
    """
    Classe para gerenciar formas de pagamento usando fila (deque).
    Formas disponíveis: Débito, Crédito, Pix.
    """
    def __init__(self):

        self._fila: deque[str] = deque(["Cartão Débito", "Cartão Crédito", "Pix"])  # Fila (deque) com as formas de pagamento na ordem desejada

    def mostrar_formas(self) -> None:
        ""
        Exibe as formas de pagamento disponíveis, em ordem da fila.
        """
        print("\n--- Formas de Pagamento Disponíveis ---")
        temp_queue = deque()
        idx = 1
        while self._fila: # Percorremos a fila sem esvaziar, usando deque temporária
            metodo = self._fila.popleft()
            print(f"{idx} - {metodo}")
            temp_queue.append(metodo)
            idx += 1
        self._fila = temp_queue # Restauramos a fila original
        print("----------------------------------------\n")

    def calcular_pagamento(self, valor: float, opcao: int) -> float | None:
        """
        Retorna o valor final a pagar. Aqui, não há juros/descontos.  etorna valor original
        se a opção for válida (1, 2 ou 3). Caso contrário, retorna None.
        """
        if opcao in [1, 2, 3]:
            return valor
        print("Opção de pagamento inválida.")
        return None

    def metodo_por_indice(self, opcao: int) -> str | None:
        """
        Retorna a forma de pagamento correspondente ao índice (1-based), ou None se inválido.
        """
        if opcao < 1 or opcao > len(self._fila):
            return None
        return list(self._fila)[opcao - 1] # Como precisamos iterar sem alterar a fila, convertemos para lista temporária
```

Linha & Luxo