# 🧪 **Atividade Prática: Sistema de Carrinho de Compras com Flask

### 🎯 Objetivo Geral

Desenvolver uma aplicação web utilizando Flask, que permita o **cadastro e login de usuários**, o **gerenciamento de um carrinho de compras** (um carrinho por usuário), **sem utilizar banco de dados**, e aplicando as funcionalidades de **cookies** e **sessions** para persistência dos dados durante a navegação.

A aplicação deve ainda ser estilizada com **CSS** utilizando a pasta `static`, e ao final, deve ser adicionado um recurso de **criptografia de senhas**.

## 👥 Organização

- Trabalho em equipes de 2 a 4 alunos.
- Atividade dividida em **dois momentos**, cada um com duração de **1h30min**.
- A entrega deve incluir o arquivo `requirements.txt` com todas as dependências.

## 🧩 Momento 1 (1h30min) — Estruturação e Funcionalidades Básicas

### Objetivos:

- Criar o ambiente virtual e instalar o Flask.
- Estruturar o projeto com as pastas e arquivos essenciais:

```
/projeto/
  ├── static/
  │   └── style.css
  ├── templates/
  │   ├── base.html
  │   ├── index.html
  │   ├── login.html
  │   ├── cadastro.html
  │   ├── produtos.html
  │   └── carrinho.html
  ├── app.py
  └── requirements.txt
```

- Criar o sistema de **cadastro de usuários** e **login**, armazenando os dados dos usuários em uma **estrutura de dados na memória** (ex.: dicionário).
- Implementar o gerenciamento de **sessão** (`session`) para manter o estado de autenticação do usuário.
- Criar uma lista fixa de produtos exibida na rota `/produtos`, com a possibilidade de o usuário **adicionar produtos ao carrinho**.

- O carrinho deve ser **armazenado na `session`**, de modo que cada usuário tenha seu próprio carrinho durante a navegação.
- Implementar as seguintes rotas:
  - `/`: Página inicial.
  - `/cadastro`: Formulário para cadastrar usuário (usuário e senha).
  - `/login`: Formulário de login.
  - `/logout`: Efetuar logout, removendo o usuário da `session`.
  - `/produtos`: Lista de produtos com opção de adicionar ao carrinho.
  - `/carrinho`: Visualização do carrinho do usuário.

### Conteúdos Abordados:

- Criação do ambiente virtual e instalação do Flask.
- Estruturação do projeto e arquivos estáticos.
- `render_template`, `request`, `redirect`, `url_for`.
- `session` para controle de autenticação e armazenamento do carrinho.
- Envio e recebimento de dados com `request`.

## 🧩 Momento 2 (1h30min) — Cookies, Criptografia e Aprimoramentos

### Objetivos:

- Implementar a funcionalidade de **cookies** utilizando `make_response`:
  - Após o login, criar um **cookie** com o nome do usuário.
  - Na página inicial, exibir uma **mensagem personalizada** utilizando o valor armazenado no cookie.
- Permitir que o usuário possa **esvaziar o carrinho**.
- Adicionar **estilização** às páginas utilizando arquivos **CSS** na pasta `static`.
- Adicionar **criptografia das senhas** utilizando `werkzeug.security`:

```
from werkzeug.security import generate_password_hash, check_password_hash
```

  - As senhas devem ser **armazenadas já criptografadas**.
  - O login deve verificar a senha usando a função `check_password_hash`.
- Finalizar e revisar o `requirements.txt` para conter todas as bibliotecas utilizadas.

## 💡 Desafios de Pesquisa

1. Como servir arquivos **CSS** corretamente utilizando a pasta `static` no Flask?
2. Como armazenar e recuperar dados via **cookies** usando `make_response`?
3. Como usar a **session** para manter dados durante a navegação (como o carrinho)?

4. Como realizar a **criptografia de senhas** utilizando `werkzeug.security`?

## Orientações Finais:

- O sistema **não deve usar banco de dados** — todos os dados devem ser manipulados com `session` e `cookies`.
- É obrigatório o uso de **pasta `static`** para armazenar o CSS.
- O código deve estar **organizado e comentado**.
- Entregar a pasta do projeto completa e funcional.