Roteiro para Construção de App Estante de Livros

Na aula de hoje, vamos criar um aplicativo web chamado "Minha Estante". A aplicação permitirá que você cadastre livros (título e autor) e exibirá a lista dos livros cadastrados.
O objetivo é que você conecte os conceitos de Programação Orientada a Objetos (POO), já vistos em console, a uma aplicação web real e interativa, utilizando o framework Flask.

Passo 1: Organizando o Projeto

Uma boa organização é o primeiro passo para um projeto web limpo. Inicie criando a seguinte estrutura de pastas e arquivos para o seu projeto:

```
minha_estante/
|
|-- app.py            # O controlador Flask (a "cola" de tudo)
|
|-- modelos.py        # Onde as classes serão definidas (o "cérebro")
|
|-- estante.json      # Arquivo de dados (será criado automaticamente)
|
|-- templates/
|   |-- estante.html  # A interface visual (o "rosto")
```

Passo 2: O Cérebro da Aplicação (As Classes em modelos.py)

Este é o coração da lógica da sua aplicação. Neste arquivo, você definirá o que é um Livro e como a Estante funciona para gerenciá-los. O ponto mais importante é que este arquivo não sabe nada sobre web, Flask ou HTML. Ele lida apenas com a lógica de livros.
Crie o arquivo modelos.py com o seguinte conteúdo:

```
import json
import os

class Livro:
    """
    Representa um único livro com título e autor.
    Esta classe é o "molde" para cada livro a ser cadastrado.
    """
    def __init__(self, titulo: str, autor: str):
        # Atributos de cada objeto Livro
        self.titulo = titulo
        self.autor = autor

class Estante:
    """
    Gerencia uma coleção de objetos Livro.
    É responsável por adicionar, carregar e salvar os livros.
    """
    def __init__(self, arquivo_json='estante.json'):
        self._arquivo = arquivo_json
        self._livros = []
        self.carregar_livros()

    @property
    def livros(self):
        """Propriedade para acessar a lista de livros de forma segura."""
        return self._livros

    def adicionar_livro(self, livro: Livro):
        """Adiciona um novo livro à estante e salva a lista no arquivo."""
        if isinstance(livro, Livro):
            self._livros.append(livro)
            self.salvar_livros()

    def salvar_livros(self):
        """Salva a lista atual de livros no arquivo JSON."""
        lista_para_salvar = [vars(livro) for livro in self._livros]
        try:
            with open(self._arquivo, 'w', encoding='utf-8') as f:
                json.dump(lista_para_salvar, f, ensure_ascii=False, indent=4)
        except IOError as e:
            print(f"Erro ao salvar o arquivo: {e}")
```

```
    def carregar_livros(self):
        """Carrega os livros do arquivo JSON ao iniciar."""
        if not os.path.exists(self._arquivo):
            return

        try:
            with open(self._arquivo, 'r', encoding='utf-8') as f:
                dados_json = json.load(f)
                self._livros = [Livro(dado['titulo'], dado['autor']) for dado in dados_json]
        except (json.JSONDecodeError, IOError) as e:
            print(f"Erro ao carregar ou decodificar o arquivo: {e}")
            self._livros = []
```

Explicação do código:

- **Separação Total:** Este arquivo contém apenas código Python. Ele poderia ser utilizado em um app de console ou, como neste caso, em um app web. Isso demonstra o princípio do **reaproveitamento de código**.
- **Classe Livro:** É uma classe simples, cuja função é agrupar os dados (atributos) de cada livro.
- **Classe Estante:** Funciona como o gerenciador. Ele contém a lista de livros e os métodos para interagir com essa lista (adicionar, salvar, carregar).
- **Persistência com JSON:** Em vez de CSV, utiliza-se o formato JSON, muito comum em aplicações web. O método salvar_livros converte os objetos em um formato compatível com JSON, e o carregar_livros faz o processo inverso.

Passo 3: O Rosto da Aplicação (O HTML em templates/estante.html)

Esta é a parte visual, a interface que o usuário final verá no navegador. Este arquivo não contém lógica complexa, apenas "marcações" onde o Flask irá inserir os dados dinamicamente.
Crie o arquivo estante.html com o seguinte conteúdo:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Minha Estante de Livros</title>
    <style>
        body { font-family: sans-serif; background-color: #f0f2f5; color: #333; }
        .container { max-width: 700px; margin: 40px auto; padding: 20px; background-color:
#fff; border-radius: 10px; box-shadow: 0 2px 10px rgba(0,0,0,0.1); }
```

```
        h1, h2 { color: #1a237e; text-align: center; }
        .form-cadastro { background-color: #e8eaf6; padding: 20px; border-radius: 8px;
margin-bottom: 30px; }
        input[type="text"] { width: calc(100% - 22px); padding: 10px; margin-bottom: 10px;
border: 1px solid #c5cae9; border-radius: 4px; }
        input[type="submit"] { display: block; width: 100%; padding: 10px; background-color:
#3949ab; color: white; border: none; border-radius: 4px; cursor: pointer; font-size: 16px; }
        input[type="submit"]:hover { background-color: #303f9f; }
        .lista-livros ul { list-style: none; padding: 0; }
        .lista-livros li { background-color: #f9f9f9; border-left: 5px solid #3949ab; margin-bottom:
10px; padding: 15px; }
        .lista-livros .titulo { font-weight: bold; font-size: 1.1em; }
        .lista-livros .autor { color: #555; }
    </style>
</head>
<body>
    <div class="container">
        <h1>📚 Minha Estante de Livros</h1>

        <div class="form-cadastro">
            <h2>Adicionar Novo Livro</h2>
            <form action="/adicionar" method="post">
                <input type="text" name="titulo" placeholder="Título do Livro" required>
                <input type="text" name="autor" placeholder="Autor(a)" required>
                <input type="submit" value="Adicionar à Estante">
            </form>
        </div>

        <div class="lista-livros">
            <h2>Livros Cadastrados ({{ livros|length }})</h2>
            <ul>
                {% for livro in livros|reverse %}
                <li>
                    <div class="titulo">{{ livro.titulo }}</div>
                    <div class="autor">por {{ livro.autor }}</div>
                </li>
                {% else %}
                <li>Nenhum livro na estante ainda.</li>
                {% endfor %}
            </ul>
        </div>
    </div>
</body>
```

</html>

Passo 4: A Cola de Tudo (O Controlador em app.py)

Este arquivo é o maestro da sua aplicação. Ele importa as classes de modelos.py e o template de estante.html, fazendo-os trabalhar juntos. Sua função é receber as requisições do navegador e usar os objetos da aplicação para dar uma resposta.
Crie o arquivo app.py com o seguinte conteúdo:

```
from flask import Flask, render_template, request, redirect, url_for
from modelos import Livro, Estante

# Cria a aplicação Flask
app = Flask(__name__)

# --- PONTO CRÍTICO DA ORIENTAÇÃO A OBJETOS ---
# É criada UMA ÚNICA instância da Estante.
# Este objeto 'minha_estante' vai persistir enquanto o app estiver rodando.
# É ele que vai carregar os livros do arquivo e manter a lista em memória.
minha_estante = Estante()
# ---------------------------------------------

@app.route('/')
def pagina_inicial():
    """
    Rota principal. Exibe a estante de livros.
    """
    # Acessa a propriedade do objeto para pegar a lista de livros
    livros_na_estante = minha_estante.livros

    # Renderiza o HTML, passando a lista de livros para o template
    return render_template('estante.html', livros=livros_na_estante)

@app.route('/adicionar', methods=['POST'])
def adicionar_livro():
    """
    Rota que recebe os dados do formulário para adicionar um novo livro.
    """
    # 1. Pega os dados enviados pelo formulário
    titulo = request.form['titulo']
    autor = request.form['autor']

    # 2. Cria um NOVO OBJETO Livro com esses dados
    novo_livro = Livro(titulo=titulo, autor=autor)
```

```
    # 3. Usa o MÉTODO do objeto Estante para adicionar o novo livro
    minha_estante.adicionar_livro(novo_livro)

    # 4. Redireciona o usuário de volta para a página inicial
    return redirect(url_for('pagina_inicial'))

# Roda o servidor de desenvolvimento
if __name__ == '__main__':
    app.run(debug=True)
```

Explicação do código:

- **minha_estante = Estante()**: Esta é a linha central. Aqui, é criado **um objeto** Estante que servirá como o único gerenciador de livros para toda a aplicação. Ao ser instanciado, ele automaticamente chama seu método carregar_livros().
- **Rota / (GET):** Quando o usuário acessa a página principal, esta função pede a lista de livros ao objeto minha_estante e a envia para o template estante.html desenhar na tela.
- **Rota /adicionar (POST):** Quando o formulário é enviado, esta função é executada. Ela pega os dados, cria um novo **objeto Livro** e usa o **método adicionar_livro** do objeto minha_estante para realizar a operação.

Passo 5: Executando a Aplicação

Para ver a sua aplicação funcionando, siga estes passos:

1. Abra um terminal na pasta do projeto (minha_estante).
2. Execute o comando: flask run
3. Abra o navegador e acesse o endereço: http://127.0.0.1:5000

Ao final, você terá construído um aplicativo web completo e funcional, onde cada parte tem uma responsabilidade bem definida, conectada pelos princípios da Programação Orientada a Objetos.