Como funciona a aplicação utilizando apenas Python, HTML e CSS, sem JavaScript, e com as limitações que isso implica.

Estrutura geral:

Nesta abordagem, usamos o **Flask**, um framework web Python, para servir tanto o **HTML** quanto processar os dados do formulário e gerar as respostas dinamicamente. O **CSS** é usado apenas para estilizar a página, e a interação é realizada de forma mais simples, sem a necessidade de JavaScript.

Funcionamento do Fluxo:

1. **Início da aplicação (Flask)**
    - O **Flask** é responsável por criar o servidor e rodar a aplicação web.
    - Quando o usuário acessa a página pela primeira vez, ele vê o formulário HTML.
2. **Formulário de entrada (HTML)**
    - O usuário escolhe um **gênero de filme** no formulário, e quando clica no botão "Obter Recomendação", os dados são enviados ao **Flask** através de uma requisição HTTP **POST**.
3. **Processamento no backend (Python)**
    - O **Flask** recebe os dados do formulário (o gênero escolhido) e usa o código Python para filtrar os filmes que correspondem ao gênero.
    - A lista de filmes que corresponde ao gênero escolhido é então preparada para ser enviada de volta para a página HTML.
4. **Renderização da resposta (HTML)**
    - O Flask então **renderiza a página HTML novamente**, mas desta vez com as sugestões de filmes.
    - Como o Flask gera o HTML no backend, a lista de filmes sugeridos é incluída diretamente no HTML que é enviado de volta para o navegador.
5. **Exibição de resultados (HTML + CSS)**
    - O navegador exibe o conteúdo atualizado com as sugestões de filmes.
    - O CSS é usado para estilizar a página, tornando a interface visualmente agradável.

Detalhamento das partes envolvidas:

1. O Formulário HTML

Aqui está o formulário que o usuário usa para enviar os dados para o backend. Ele contém uma lista de opções de gêneros de filmes e um botão de envio:

html

```
<form method="POST">
```

```
  <label for="genero">Gênero:</label>
  <select id="genero" name="genero">
    <option value="Ação">Ação</option>
    <option value="Comédia">Comédia</option>
    <option value="Terror">Terror</option>
    <option value="Animação">Animação</option>
  </select>
  <button type="submit">Obter Recomendação</button>
</form>
```

- **<form method="POST">**: Esse formulário envia os dados para o servidor usando o método **POST**.
- **<select id="genero" name="genero">**: O usuário escolhe um gênero de filme da lista.
- **<button type="submit">Obter Recomendação</button>**: O botão submete o formulário.

2. O Código Python (Flask)

O **Flask** recebe os dados do formulário (gênero escolhido) e processa a recomendação de filmes. No backend, o Python filtra os filmes com base no gênero selecionado:

python

```
@app.route('/', methods=['GET', 'POST'])
def index():
  sugestoes = []
  if request.method == 'POST':
    genero_preferido = request.form['genero'].capitalize()
    # Filtra os filmes com o gênero escolhido
    sugestoes = [f['nome'] for f in filmes if f['genero'] == genero_preferido]
  return render_template('index.html', sugestoes=sugestoes)
```

- **request.method == 'POST'**: Quando o formulário é enviado, ele faz uma requisição POST. O Flask então processa essa requisição.
- **request.form['genero']**: O Flask captura o gênero selecionado pelo usuário.
- **filmes = [...]**: Lista de filmes disponíveis, com nome e gênero.
- **sugestoes = [...]**: Filtra os filmes que correspondem ao gênero escolhido.

- **render_template('index.html', sugestoes=sugestoes)**: O Flask renderiza a página HTML novamente, agora incluindo a lista de sugestões de filmes.

3. O HTML com as Sugestões

Depois de o Python processar a entrada, ele gera o HTML atualizado com as sugestões de filmes:

html

```
{% if sugestoes %}
  <div id="sugestoes" class="sugestoes">
    <p>Recomendamos os seguintes filmes de {{ request.form['genero'] }}: </p>
    <ul>
      {% for filme in sugestoes %}
        <li>{{ filme }}</li>
      {% endfor %}
    </ul>
  </div>
{% endif %}
```

- **{% if sugestoes %}**: Verifica se há sugestões para exibir.
- **{{ filme }}**: Exibe cada filme da lista sugestoes que foi processada no backend.
- **{% for filme in sugestoes %}**: Itera sobre a lista de sugestões de filmes e os exibe na tela.

4. O CSS

A parte de **CSS** serve para dar uma aparência visual mais agradável à página. O código CSS pode ser algo simples como:

css

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f9;
  margin: 0;
  padding: 0;
}

.container {
```

```
    max-width: 600px;
    margin: 50px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

h1 {
    color: #333;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    font-size: 1.1em;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

.sugestoes {
    margin-top: 20px;
    font-size: 1.2em;
    font-weight: bold;
    color: #444;
}
```

- **font-family**: Define a fonte da página.
- **background-color**: Define a cor de fundo da página.
- **button:hover**: Estiliza o botão quando o mouse passa sobre ele.

Vantagens e Limitações:

Vantagens:

- **Simples e Direto**: O código é simples, sem a necessidade de JavaScript ou manipulação assíncrona.
- **Backend com Python**: Toda a lógica de processamento e filtragem dos filmes é feita no backend, o que pode ser interessante para quem deseja manter o controle total no servidor.

Limitações:

- **Sem Interatividade Dinâmica**: A página precisa ser recarregada para exibir novas sugestões. Não há atualizações em tempo real sem o uso de JavaScript.
- **A experiência do usuário é menos fluida**: Ao clicar em "Obter Recomendação", o navegador precisa fazer uma nova requisição e recarregar a página, o que pode parecer menos fluido para o usuário comparado a uma aplicação JavaScript (como com AJAX).

Conclusão:

Esse tipo de aplicação é simples e adequada para situações onde você deseja trabalhar com um backend robusto, sem a necessidade de tecnologias de frontend como JavaScript. O **Flask** lida com a lógica e a renderização dinâmica da página, enquanto o **HTML** e o **CSS** cuidam da apresentação visual. A interação é limitada, mas é perfeitamente funcional e fácil de entender para iniciantes.