Melhorando a Interface do Gerenciador Financeiro

Desenvolvimento de Sistemas - Aula 24 - 2º Mtec DS

VISÃO GERAL E OBJETIVO

Construir interface mais agradável e melhor utilizável para o App Gerenciador Financeiro construído em Python. Introdução ao framework Flask.

Começando pela Interface Web

Como é o nosso primeiro App Web com Flask, vamos passar o código pronto e ir comentando o papel de cada trecho, a fim de compreender na prática as interações do HTML com Python utilizando o Flask. Você pode alterar a aparência da página como desejar, criando o seu próprio estilo CSS.

Template para a index.html

```
<!DOCTYPE html>

<html lang="pt-BR">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>App Gerenciador Financeiro</title>

</head>

<body>

    <div class="container">
```

```
    <h1>Meu App de Finanças</h1>

    <!-- Seção de Saldo -->
    <h2 class="{{ 'saldo-positivo' if saldo >= 0 else 'saldo-negativo' }}">
        Saldo Atual: R$ {{ "%.2f"|format(saldo) }}
    </h2>

    <!-- Formulários para Adicionar Transações -->
    <div class="form-container">
        <div class="form-section">
            <h3>Adicionar Receita</h3>
            <form action="/adicionar" method="post">
                <input type="hidden" name="categoria" value="Receita">
                <input type="text" name="nome" placeholder="Ex: Salário" required>
                <input type="number" name="valor" step="0.01" min="0.01"
placeholder="Valor" required>
                <input type="submit" value="Adicionar Receita" class="btn-receita">
            </form>
        </div>
        <div class="form-section">
            <h3>Adicionar Despesa</h3>
            <form action="/adicionar" method="post">
                <input type="hidden" name="categoria" value="Despesa">
                <input type="text" name="nome" placeholder="Ex: Aluguel" required>
```

```
            <input type="number" name="valor" step="0.01" min="0.01"
placeholder="Valor" required>
            <input type="submit" value="Adicionar Despesa" class="btn-despesa">
        </form>
    </div>
</div>

<!-- Tabela de Transações -->
<h2>Extrato</h2>
{% if transacoes %}
    <table>
        <thead>
            <tr>
                <th>nome</th>
                <th>Valor</th>
                <th>categoria</th>
            </tr>
        </thead>
        <tbody>
            {% for t in transacoes|reverse %}
            <tr>
                <td>{{ t.nome }}</td>
                <td class="{{ t.categoria|lower }}">{{ "%.2f"|format(t.valor) }}</td>
                <td class="{{ t.categoria|lower }}">{{ t.categoria }}</td>
```

```
                </tr>
                {% endfor %}
            </tbody>
        </table>
    {% else %}
        <p class="no-data">Nenhuma transação registrada ainda.</p>
    {% endif %}
  </div>
</body>
</html>
```

Conteúdo do Arquivo para o CSS (criar um arquivo com o nome style.css)

```
body {
        font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif;
        background-color: #f4f7f9; color: #333; margin: 0; padding: 20px;
}
.container {
        max-width: 800px;
        margin: 0 auto;
        background-color: #fff;
        padding: 30px;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}
```

```
h1, h2 {
        color: #2c3e50;
        text-align: center;
}
h2.saldo-positivo {
        color: #27ae60;
}
h2.saldo-negativo {
        color: #c0392b;
}
table {
        width: 100%;
         border-collapse: collapse;
         margin-top: 20px;
}
th, td {
        padding: 12px;
        border-bottom: 1px solid #ddd;
        text-align: left;
}
th {
        background-color: #f2f2f2;
}
.receita {
        color: #27ae60; }
```

```
.despesa {
        color: #c0392b;
}
.form-container {
        display: flex;
        justify-content: space-between;
        gap: 20px;
        margin-top: 30px;
        padding: 20px;
        background-color: #f9f9f9;
        border-radius: 8px;
}
.form-section {
        flex: 1;
}
form {
        display: flex;
        flex-direction: column;
}
input[type="text"], input[type="number"] {
        padding: 10px;
        margin-bottom: 10px;
        border: 1px solid #ccc;
        border-radius: 4px;
}
```

```
input[type="submit"] {
        padding: 10px 15px;
        border: none;
        border-radius: 4px;
        color: #fff;
        cursor: pointer;
}
.btn-receita {
        background-color: #2ecc71;
}
.btn-despesa {
        background-color: #e74c3c;
}
.no-data {
        text-align: center;
        color: #777;
        padding: 20px;
}
```

Explicando os elementos HTML e os Estilos

O arquivo `index.html` é um **template**, o que significa que ele é um HTML "turbinado". Ele mistura a estrutura HTML padrão com comandos especiais do **Jinja2** (o sistema de templates do Flask), que são processados no servidor antes da página ser enviada para o navegador.

Vamos dividir a explicação em partes:

---

1. Estrutura Básica e Estilos (CSS)

- **NO arquivo CSS:**
- **`.container`**: Cria a caixa central branca com bordas arredondadas e uma sombra, limitando a largura para que o layout fique bom em telas grandes.
- **`.saldo-positivo`** e **`.saldo-negativo`**: Classes que definem a cor do texto do saldo (verde ou vermelho).
- **`.receita`** e **`.despesa`**: Classes que definem a cor do texto para os valores na tabela.
- **`.form-container`**: Usa `display:flex` que ativa o "FlexBox Layout" para colocar os formulários de receita e despesa lado a lado.

---

2. Corpo da Página (`<body>`)

Esta é a parte visível da página, organizada dentro de uma `<div class="container">`.

Seção do Saldo

HTML{% if transacoes %}

```
<table>
    ...
    <tbody>
        {% for t in transacoes|reverse %}
        <tr>
            <td>{{ t.nome }}</td>
            <td class="{{ t.categoria|lower }}">{{ "%.2f"|format(t.valor) }}</td>
            <td class="{{ t.categoria|lower }}">{{ t.categoria }}</td>
        </tr>
        {% endfor %}
```

```
        </tbody>
    </table>
{% else %}
    <p class="no-data">Nenhuma transação registrada ainda.</p>
{% endif %}
```

Esta é a parte mais dinâmica:

- **`{% if transacoes %}`**: O Flask envia uma lista chamada `transacoes`. Este comando verifica se a lista não está vazia. Se estiver vazia, ele pula para o `{% else %}` e mostra a mensagem "Nenhuma transação registrada...".
- **`{% for t in transacoes|reverse %}`**: Se a lista tiver itens, este comando funciona como um `for` em Python. Ele percorre cada objeto da classe `Transacao` na lista.
    - **`|reverse`**: É um "filtro" do Jinja2 que inverte a ordem da lista, fazendo com que as transações mais recentes apareçam no topo da tabela.
- **`{{ t.nome }}`**, **`{{ t.valor }}`**, **`{{ t.categoria }}`**: Para cada transação `t` no loop, acessamos seus atributos (exatamente como faríamos em Python: `t.nome`) e os inserimos nas células da tabela.
- **`class="{{ t.categoria|lower }}"`**: De novo, uma classe dinâmica. Se `t.categoria` for "Receita", a classe será "receita" (verde); se for "Despesa", será "despesa" (vermelho).

O código Python

```
# app.py
from flask import Flask, render_template, request, redirect, url_for
from classes import Transacao, ControleFinanceiro

# Cria a aplicação Flask
app = Flask(__name__)
```

```
# Cria uma instância de nosso gerenciador. Em uma aplicação real, isso seria gerenciado de forma mais complexa.

# Para este exemplo, uma instância global simplifica as coisas.

gerenciador = ControleFinanceiro()


@app.route('/')

def index():

"""

    Rota principal que exibe a lista de transações e o saldo. Uma rota é um caminho indicado pelo usuário no formato de URL da página, que ao ser chamada, exibe um conteúdo com alguma funcionalidade do App. No Flask, associamos um rota a um método como acima index()

"""

# Acessa os dados através dos métodos e propriedades do nosso objeto

transacoes = gerenciador.transacoes

saldo = gerenciador.calculaSaldo()

# Renderiza o template HTML, passando os dados para ele

return render_template('index.html', transacoes=transacoes, saldo=saldo)


@app.route('/adicionar', methods=['POST'])

def adicionar():

"""

 Rota para adicionar uma nova transação (receita ou despesa).

"""
```

```
# Pega os dados enviados pelo formulário HTML
categoria = request.form['categoria']
nome = request.form['nome']
valor = float(request.form['valor'])

# Cria um novo objeto Transacao com os dados do formulário
nova_transacao = Transacao(nome=nome, valor=valor, categoria=categoria)

# Usa nosso objeto gerenciador para adicionar a transação
gerenciador.adicionaTransacao(nova_transacao)

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

# Garante que o servidor de desenvolvimento só rode quando o script for executado diretamente
if __name__ == '__main__':
# O modo debug recarrega o servidor automaticamente a cada alteração no código
app.run(debug=True)
```