



# Framework Web em Python

# Framework Web em Python

- Estrutura de projeto em Flask.
- Renderização de templates.
- Inserção de variáveis e loops nos templates.
- Boas práticas de organização para projetos Flask.

# Estrutura de Projeto em Flask

- Um projeto Flask organizado é essencial para aplicações maiores.
- Estrutura típica de um projeto Flask:

```
csharp  
Copiar código  
projeto/  
    └─ app.py          # Arquivo principal  
    └─ static/         # Arquivos estáticos (CSS, JS, imagens)  
    └─ templates/      # Arquivos HTML  
        └─ base.html   # Template base  
        └─ index.html   # Template inicial  
    └─ requirements.txt # Dependências
```

# Estrutura de Projeto Flask Detalhada

- **Arquivo Principal:** Este é o arquivo principal da sua aplicação Flask. Ele contém o código Python que define as rotas, a lógica da aplicação e a inicialização do servidor Flask.
- **Static/:** Arquivos Estáticos: Esta pasta armazena arquivos estáticos, como arquivos CSS, JavaScript e imagens.
- **Templates/:** Arquivos HTML: Esta pasta armazena os arquivos HTML que definem a estrutura e o conteúdo das páginas da sua aplicação.

# Estrutura de Projeto Flask Detalhada

- Requirements.txt: Dependências: Este arquivo lista as dependências da sua aplicação, ou seja, as bibliotecas Python que você precisa instalar para que a aplicação funcione corretamente.

Exemplo de requirements.txt :

```
Flask==2.0.1
Jinja2==3.0.1
```

# Renderização de Templates com Flask

O Flask permite renderizar páginas HTML dinâmicas usando a função `render_template`.

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')
```

- Template `index.html` :

HTML ▾

`html`

`Copiar código`

`<!DOCTYPE html>`

`<html>`

`<head>`

`<title>Página Inicial</title>`

`</head>`

`<body>`

`<h1>Bem-vindo à Página Inicial</h1>`

`</body>`

`</html>`

# Passagem de Variáveis para Templates

É possível passar dados do Python para o template usando render\_template.

```
@app.route('/usuario/<nome>')
def usuario(nome):
    return render_template('usuario.html', nome=nome)
```

- Template `usuario.html`:

HTML ↴

`html`

`Copiar código`

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Perfil</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Bem-vindo, {{ nome }}!</h1>
```

```
</body>
```

```
</html>
```

# Loops e Condições nos Templates

A biblioteca Jinja2 permite usar loops e condições no HTML.

## Loop em um template:

```
@app.route('/produtos')
def produtos():
    lista_produtos = ["Computador", "Celular", "Câmera"]
    return render_template('produtos.html', produtos=lista_produtos)
```

- Template `produtos.html` :

HTML ▾

`html`

`Copiar código`

`<!DOCTYPE html>`

`<html>`

`<head>`

`<title>Produtos</title>`

`</head>`

`<body>`

`<h1>Lista de Produtos</h1>`

`<ul>`

`{> for produto in produtos %}>`

`<li>{{ produto }}</li>`

`{> endfor %}>`

`</ul>`

`</body>`

`</html>`

# Exercícios Práticos

# Exercício 1: Criar Rotas Dinâmicas

## Exercício assitido

**Enunciado:** Configure uma rota `/saudacao/<nome>` que receba um nome e exiba "Olá, <nome>!" usando um template.

# Exercício 2: Trabalhar com Loops

Enunciado: Configure uma rota `/cursos` que exibe uma lista de cursos disponíveis usando um template.

## EXEMPLO TEMPLATE

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Saudação</title>
</head>
<body>
    <h1>Olá, {{ nome }}!</h1>
</body>
</html>
```

# Resposta

```
@app.route('/cursos')

def cursos():
    cursos = ["Python", "JavaScript", "HTML & CSS", "Django"]
    return render_template('cursos.html', cursos=cursos)
```

# Resposta

```
html
Copiar código
<!DOCTYPE html>
<html>
<head>
    <title>Cursos</title>
</head>
<body>
    <h1>Cursos Disponíveis</h1>
    <ul>
        {% for curso in cursos %}
            <li>{{ curso }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

# Exercício 3: Utilizar Condições no Template

Enunciado: Configure uma rota `/login` que exiba "Bem-vindo de volta!" se o usuário estiver logado e "Por favor, faça login." caso contrário.

# Resposta

```
@app.route('/login')
def login():
    logado = True # Alterar para False para testar
    return render_template('login.html', logado=logado)
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    {% if logado %}
        <h1>Bem-vindo de volta!</h1>
    {% else %}
        <h1>Por favor, faça login.</h1>
    {% endif %}
</body>
</html>
```

# Integração com Bibliotecas, Módulos e Manipulação de Formulários

- Utilização de bibliotecas e módulos externos.
- Integração com APIs.
- Manipulação de formulários HTML no Flask.

# Integração com Bibliotecas e Módulos Externos

- Módulos: Arquivos Python que contêm funções, classes ou variáveis reutilizáveis.
- Bibliotecas: Conjuntos de módulos que podem ser instalados com gerenciadores de pacotes, como pip.

bash

Copiar código

pip install requests

python

Copiar código

import requests

```
response = requests.get("https://api.github.com")
```

```
print(response.json())
```

# Integração com APIs

APIs permitem que sistemas se  
comuniquem.

**Consuma a API do GitHub para listar  
repositórios públicos:**

```
import requests

url = "https://api.github.com/repositories"
response = requests.get(url)
repos = response.json()
for repo in repos[:5]:
    print(repo['name'])
```

# Manipulação de Formulários HTML no Flask

Formulários permitem interagir com os usuários.

```
from flask import Flask, request, render_template

app = Flask(__name__)

@app.route('/formulario', methods=['GET', 'POST'])
def formulario():
    if request.method == 'POST':
        nome = request.form['nome']
        return f"Olá, {nome}!"
    return render_template('formulario.html')
```

## Template formulario.html :

HTML ▾

html  
Copiar código

<!DOCTYPE html>

<html>

<head>

    <title>Formulário</title>

</head>

<body>

    <form method="post">

        <label for="nome">Nome:</label>

        <input type="text" id="nome" name="nome">

        <button type="submit">Enviar</button>

    </form>

</body>

</html>

# Exercícios Durante a Aula

Assistido

## Exercício 1: Consumir uma API

Enunciado: Utilize a biblioteca `requests` para consumir a API pública do GitHub e exibir no terminal os 5 repositórios mais populares.

# Exercício 2: Formulário para Cálculo de Idade

Enunciado: Crie uma página com um formulário que receba o ano de nascimento do usuário e exiba sua idade.

Senac

```
from flask import Flask, request, render_template
from datetime import datetime

app = Flask(__name__)

@app.route('/calcule_idade', methods=['GET', 'POST'])
def calcule_idade():
    if request.method == 'POST':
        ano = int(request.form['ano'])
        idade = datetime.now().year - ano
        return f"Você tem {idade} anos."
    return render_template('calcule_idade.html')
```

- Template `calculo_idade.html`:

HTML ▾

`html`

Copiar código

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Cálculo de Idade</title>
```

```
</head>
```

```
<body>
```

```
    <form method="post">
```

```
        <label for="ano">Ano de nascimento:</label>
```

```
        <input type="number" id="ano" name="ano">
```

```
        <button type="submit">Calcular</button>
```

```
    </form>
```

```
</body>
```

```
</html>
```

# Exercício 3: Usar uma Biblioteca Externa

Enunciado: Instale a biblioteca `matplotlib` e crie um gráfico simples de uma função quadrática.

```
import matplotlib.pyplot as plt

x = range(-10, 11)
y = [i ** 2 for i in x]

plt.plot(x, y)
plt.title("Gráfico de y = x²")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.savefig("grafico.png")
```

## Exercício 1: Formulário para Exibir Nome

- Enunciado: Crie uma aplicação Flask que receba o nome e a cidade do usuário em um formulário e exiba uma mensagem como: "Olá, [nome]! Você mora em [cidade]."

## Exercício 2: Consumir API de Clima

- Enunciado: Utilize a API de clima da OpenWeatherMap para exibir a temperatura atual de uma cidade fornecida pelo usuário.

## Exercício 3: Gráfico Interativo

- Enunciado: Crie uma aplicação Flask que receba dois números no formulário e plote o gráfico da função linear  $y=ax+b$ .

$$y=ax+by = ax + b$$

## Exercício 4: Lista de Produtos Dinâmica

- Enunciado: Crie uma página que exibe uma lista de produtos fornecida pelo usuário através de um formulário.