

Curso de Programação em Python

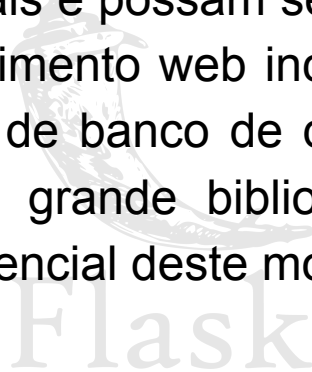
Módulo: Programação Web com
Flask

ARÖ-TEC



Desenvolvimento Web

O desenvolvimento web é o conjunto de processos na criação de websites e aplicações que sejam funcionais e possam ser executados em navegadores web. Além disso, o desenvolvimento web inclui integração com sistemas e ferramentas, armazenamento de banco de dados, cibersegurança e muito mais. O Python possui uma grande biblioteca para desenvolvimento o **Flask**, que é a ferramenta essencial deste módulo.



Ambiente Virtual em Python

É um espaço isolado onde você pode instalar bibliotecas (módulos), pacotes e versões específicas do Python sem interferir com o sistema Python global ou com outros projetos. É como criar uma “bolha” em torno do seu projeto, garantindo que suas dependências permaneçam separadas e controladas. Podemos criar ambiente virtual com o comando:

No Windows

```
python -m venv env
```

No Linux/Mac

```
python3 -m venv env
```

Obs: env é nome do ambiente poderia ser qualquer outro.

Activação do ambiente virtual

Após de criar o ambiente virtual há necessidade de activá-lo para que todas as instalações e execuções de comandos Python ocorram dentro desse ambiente isolado.

No Windows

```
env/Scripts/activate
```

No Mac

```
source env/bin/activate
```



PIP

O pip é o gerenciador de módulos (pacotes) para projetos Python. Com ele, você pode instalar, remover e atualizar pacotes módulos (pacotes) em seus projetos. Existe uma vasta quantidade de módulos em <https://pypi.org/>. Abaixo, o comando de instalação do PIP

Estrutura

```
pip install <nome do módulo>
```

No Mac

```
pip install
```

Cliente-Servidor

A web está essencialmente composta por clientes e servidores.

Clientes: São programas conectados a internet para acessar à web, como navegadores (por exemplo, Firefox ou Chrome). Quando você digita um endereço da web no navegador, ele envia uma solicitação para o servidor correspondente.

Servidores: são computadores que armazenam páginas, sites ou aplicativos. Quando um cliente deseja acessar uma página da web, o servidor envia uma cópia dessa página para a máquina do cliente. Essa cópia é então apresentada no navegador do usuário.

Fundamentos da Web

O DNS é fundamental para o funcionamento da Internet. Ele traduz nomes de domínio (como “google.com”) em endereços IP (como “172.217.6.238”). Isso permite que as pessoas encontrem páginas com URLs amigáveis, em vez de digitar números IP

Um endereço IP é um identificador único associado a um dispositivo na Internet ou em uma rede local. Ele permite que os sistemas se reconheçam e se comuniquem por meio do protocolo de Internet.

Fundamentos da Web

O HTTP é um protocolo usado para obter recursos na Web. Ele permite que os navegadores solicitem documentos HTML, imagens, vídeos e outros conteúdos de servidores web.

Na Internet, um recurso é qualquer coisa que pode ser acessada por um URL específico. Isso inclui páginas da web, imagens, arquivos PDF, vídeos, etc.

Os pacotes referem-se aos dados que são transmitidos pela rede. Eles são divididos em pequenas partes chamadas pacotes e reagrupados no destino. Isso permite a transferência eficiente de informações pela Internet.

Flask

Flask é uma biblioteca de estrutura leve e flexível para criação de aplicações web (Website e API's).

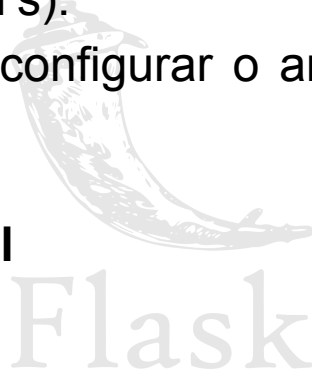
Para iniciar em Flask, debes configurar o ambiente virtual e instalá-lo, via pip.

Criação do ambiente virtual

```
python -m venv env
```


Instalação do Flask

```
pip install Flask
```



Estrutura básica de um servidor Flask

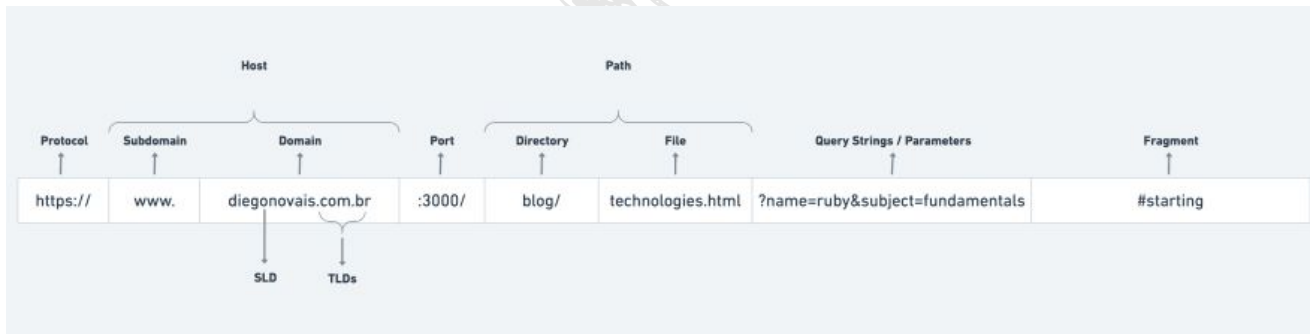
Crie um ficheiro chamado main.py. E implemente o seguinte código



```
1 from flask import Flask
2
3 app = Flask("Meu website")
4
5 app.run()
```

URLs

URL (Uniform Resource Locator) é o endereço que aponta para um recurso específico na web. Ela pode ser usada para acessar qualquer coisa, desde uma página da web até um arquivo específico.



Composição de uma URL

Tomando o exemplo do slide anterior, é possível notar a existência destes principais componentes na URL

- **Protocol (Protocolo)** é o protocolo de rede usado, como “http”, “https”, “mailto”, “ftp” etc.
- **Host:** é um nome (composto pelo subdomínio quando houver e pelo domínio) identificando um computador na internet (ou seu número de endereço IP).
- **Port (porta):** Portas em uma URL são números que especificam qual serviço está sendo associado
- **Path (caminho)** é localização para encontrar o recurso, arquivo ou objeto dentro do servidor

Rotas

As rotas são URLs que o cliente (navegador, aplicativo, etc.) pode solicitar. Você define rotas no seu código usando decoradores, cada rota é associada a uma função que responde a essa solicitação.



Criando Rotas

1. Definir a rota usando o decorator do Flask

```
@app.route("/")
```

2. Definir a função associada a essa rota, onde o retorno é resposta da solicitação.

```
def index():  
    return "Hello World!"
```

Recursos estáticos

No Flask é possível acessar recursos estáticos, a pasta “**static**” é usada por padrão para armazenar arquivos estáticos, como folhas de estilo CSS, arquivos JavaScript e imagens, que são servidos diretamente pelo servidor web sem processamento adicional. Estes recursos estáticos estão acessíveis a partir da rota **/static**. Ex: /static/style.css



Flask

Templates

Os templates desempenham um papel crucial na renderização de conteúdo dinâmico para sua aplicação web. São armazenados em um diretório chamado **“templates”**



Flask

Motor de Template Jinja

O Flask utiliza o motor de template Jinja para renderizar os templates. O Jinja permite que você insira dados dinâmicos e declarações de fluxo de controle dentro dos seus templates HTML.

- Use `{{ expressão }}` para inserir expressões no template
- Use `{% controle %}` para estruturas de controle, como if e for.
- Use `{% endfor %}` ou `{% endif %}` para fechar estruturas de controle, como if e for

Tratamento de Formulários

Em muitas situações seu servidor necessita de dados provenientes do usuário, como uma página de cadastro, ou login. Para acessar os dados de entrada no Flask, você usará o objeto request.

```
from flask import Flask, request
```

Para recuperar os dados deve ser uma rota POST e podes pegar o valor do jeito, abaixo

```
@app.route('/add', methods=["POST"])
def add_post():
    nome = request.form.get('nome', "")
    print(nome)
```

Banco de dados

Um banco de dados é um sistema de armazenamento de informações que permite a coleta, o armazenamento, a recuperação e a manipulação de dados de maneira estruturada e eficiente. SQL é uma linguagem universal utilizada para gerenciar e manipular bancos de dados relacionais. SQLite é um banco de dados leve e fácil de usar, perfeito para projetos pequenos e médios.

ORM é uma técnica que preenche a lacuna entre a programação orientada a objetos (OOP) e os bancos de dados relacionais. Ele permite consultar e manipular dados de um banco de dados usando um paradigma orientado a objetos. Essencialmente, uma biblioteca ORM encapsula o código necessário para interagir com os dados, eliminando a necessidade de consultas SQL diretas.

Flask-SQLAlchemy

Flask-SQLAlchemy é uma extensão que permite trabalhar com o SQLAlchemy (um ORM popular para Python) no Flask. Abaixo configuramos.

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///mydatabase.db'  
db = SQLAlchemy(app)
```

Flask

Modelo em Flask-SQLAlchemy

Em Flask-SQLAlchemy, um "modelo" refere-se a uma classe Python que representa uma tabela específica em um banco de dados relacional.

```
class Usuario(db.Model):  
    id = db.Column(db.Integer, primary_key=True)  
    username = db.Column(db.String(80))  
    senha = db.Column(db.String(120))
```

Cria agora, crie a tabela no banco de dados

```
db.create_all()
```

Manipulação dos dados

Você pode adicionar, recuperar, atualizar e excluir dados usando o SQLAlchemy.

Por exemplo (adicionando):

```
new_usuario = Usuario(nome="Sebastião", senha="123")  
db.session.add(new_usuario)  
db.session.commit()
```

Flask

Recuperar dados

```
all_usuarios = Usuario.query.all()
```

Atualizar dados

```
usuario = Usuario.query.get(1)  
usuario.nome = "Lucas"  
db.session.commit()
```

Excluir dados

```
usuario = Usuario.query.get(1)  
db.session.delete(usuario)  
db.session.commit()
```



Sessões

As sessões no Flask permitem que você armazene dados específicos do usuário entre várias solicitações HTTP, com diversas finalidades como autenticação, carrinho de compras. No Flask, você pode usar cookies para implementar sessões. Os cookies são pequenos pedaços de dados armazenados no navegador do usuário

The Flask logo is a stylized illustration of a laboratory flask. It is filled with a liquid that has a wavy surface, suggesting movement or a chemical reaction. The flask is positioned centrally in the background, behind the word 'Flask'.

Flask

Armazenando dados na sessão

Precisas primeira definir uma secret key para o teu servidor flask

```
app.secret_key="123"
```

Importar o objecto session

```
from flask import Flask, request, session
```

Adicionar um novo valor na sessão

```
session['nome']='Sebastião'
```



flask

Acesso e limpeza da sessão

Para acessar a sessão, use o método get

```
print(session.get('nome'))
```

Para limpar a sessão, use o método pop

```
print(session.pop('nome', None))
```

Atividade final: CRUD

CRUD é um acrônimo que significa Create (Criar), Read (Ler), Update (Atualizar) e Delete (Excluir) — as operações básicas de um sistema de gerenciamento de banco de dados.

Nós iremos construir uma aplicação web simples para realizar as operações de CRUD de um banco de dados de pessoas.

Flask