

TRABALHANDO COM FLASK

O que é Flask ?

Flask é um micro framework que utiliza a linguagem Python para criar aplicativos Web.



TRABALHANDO COM FLASK

Para Rodar o Flask devemos criar nossa pasta de projeto e uma virtualenv.

Para criar a virtualenv basta digitar : `virtualenv nome_virtualenv` e digitar o comando `activate` dentro das pasta script da virtualenv

Instalar o pacote flask através do gerenciador de pacotes pip.

`pip install flask`

Vamos conectar o projeto ao VSCODE. Basta abrir o diretório no vscode

Dentro do vscode vamos criar uma pasta chamada app e vamos criar um arquivo chamado app.py

TRABALHANDO COM FLASK

Dentro do arquivo app.py vamos inserir o seguinte código para iniciarmos o aplicativo em flask.

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')  
def hello():
```

```
    return 'Olá Framework Flask em Execução'
```

Na linha de comando vamos setar a variável de ambiente de execução do nosso aplicativo flask

set FLASK_APP=app para Linux o **export FLASK_APP=app**

TRABALHANDO COM FLASK

Para executar o framework em linha de comando executamos:

`flask run`

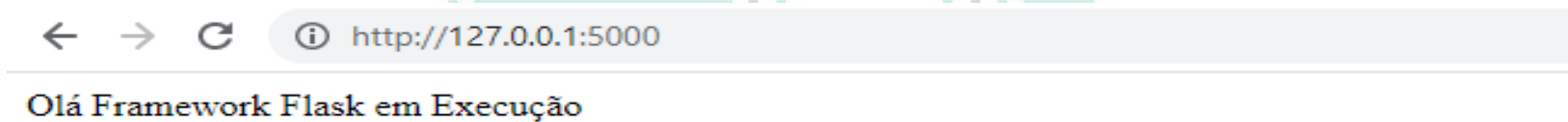
Irá aparecer a seguinte mensagem na tela

```
(flask_app) D:\temp\flask_app\flask_app\app>flask run
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Com o Navegador WEB vamos acessar o endereço descrito na saída do console

`http://127.0.0.1:5000`

TRABALHANDO COM FLASK



Nesse momento estamos com o Framework Rodando com uma configuração de pagina web Mínima. Podemos criar uma estrutura de pagina Web utilizamos HTML,CSS e JAVASCRIPT para Aprimorar nossa aplicação WEB. Bem como utilizar conexão de banco de dados para armazenar as informações do nosso sistema em Banco de dados SQL.





TRABALHANDO COM FLASK

Para criarmos a estrutura MVC de projetos vamos utilizar a seguinte estrutura:

Pasta Templates: Iremos colocar os arquivos HTML Template do Site

Pasta Static: Armazena arquivos de estilo CSS e imagens para o site

Vamos criar a estrutura de pastas templates,static e copiar os arquivos disponibilizados Na aula para estrutura do site.

Nome	Data de modificação
 <code>__pycache__</code>	18/07/2021 20:36
 <code>static</code>	18/07/2021 20:55
 <code>templates</code>	18/07/2021 20:49
 <code>app</code>	18/07/2021 20:36

TRABALHANDO COM FLASK

Iremos vincular o arquivo index.html da nossa aplicação web. Para isso iremos herdar Alguns estilos e estruturas HTML do formato bootstrap através de um arquivo disponibilizado na aula base.html

Iremos configurar nosso app.py para utilizar o index.html através do modulo render_template

```
from flask import Flask, render_template, request, url_for, redirect
```

```
app = Flask(__name__)
```

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

TRABALHANDO COM FLASK

Iremos criar a base de dados para gravar as informações de clientes no tabela do banco SQLITE3 (Banco Nativo do Python).

Iremos criar um arquivos cria_base.py e iremos deixar 2 cadastros inseridos

```
import sqlite3
```

```
conn = sqlite3.connect('app/basedados.db')
```

```
cursor = conn.cursor()
```

```
cursor.execute("""  
CREATE TABLE cadastro_clientes (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    Nome TEXT NOT NULL,  
    Sobrenome TEXT NOT NULL,  
    CPF TEXT NOT NULL  
    """)
```


TRABALHANDO COM FLASK

```
cursor.execute("INSERT INTO cadastro_clientes (Nome, Sobrenome, CPF) VALUES (?, ?,?)",  
              ("João", 'Cavichioli', '12345')  
              )
```

```
cursor.execute("INSERT INTO cadastro_clientes (Nome, Sobrenome, CPF) VALUES (?, ?,?)",  
              ("Pedro", 'Silva', '122444')  
              )
```

```
conn.commit()  
conn.close()
```

TRABALHANDO COM FLASK

Vamos adicionar as conexões de banco de dados ao arquivo app.py

```
import sqlite3
```

```
def get_db_connection():  
    conn = sqlite3.connect('basedados.db')  
    conn.row_factory = sqlite3.Row  
    return conn
```

Para criarmos as subpáginas ou routes dentro do Flask apontando para uma pagina HTML. Estaremos criando as routes para nossa pagina listaclientes.html e addclintes.html

TRABALHANDO COM FLASK

Vamos adicionar a rota lista clientes atrelado a nossa aplicação.

```
@app.route('/listaclientes',methods=('GET', 'POST'))  
def listaclientes():  
    conn = get_db_connection()  
    clientes = conn.execute('SELECT * FROM cadastro_clientes').fetchall()  
    conn.close()  
    return render_template('listaclientes.html', clientes=clientes)
```

TRABALHANDO COM FLASK

Vamos adicionar a rota adicionar clientes atrelado a nossa aplicação.

```
@app.route('/addclientes', methods=('GET', 'POST'))
def addclientes():
    if request.method == 'POST':
        nome = request.form['nome']
        sobrenome = request.form['sobrenome']
        cpf = request.form['cpf']
        conn = get_db_connection()
        conn.execute('INSERT INTO cadastro_clientes (nome, sobrenome,cpf) VALUES (?, ?, ?)',
(nome, sobrenome,cpf))
        conn.commit()
        conn.close()
        return redirect(url_for('index'))

    return render_template('addclientes.html')
```

TRABALHANDO COM FLASK

Vamos subir nossa aplicação em um ambiente de nuvem para rodar nossa aplicação.
Vamos utilizar o python Anywhre que utiliza recurso de nuvem que rodam flask de forma gratuita.
Para isso e necessário criar a conta no Python AnyWhere do tipo beginner.

<https://www.pythonanywhere.com/registration/register/beginner/>

TRABALHANDO COM FLASK



Create your account

Username:

Email:

Password:

Password (again):

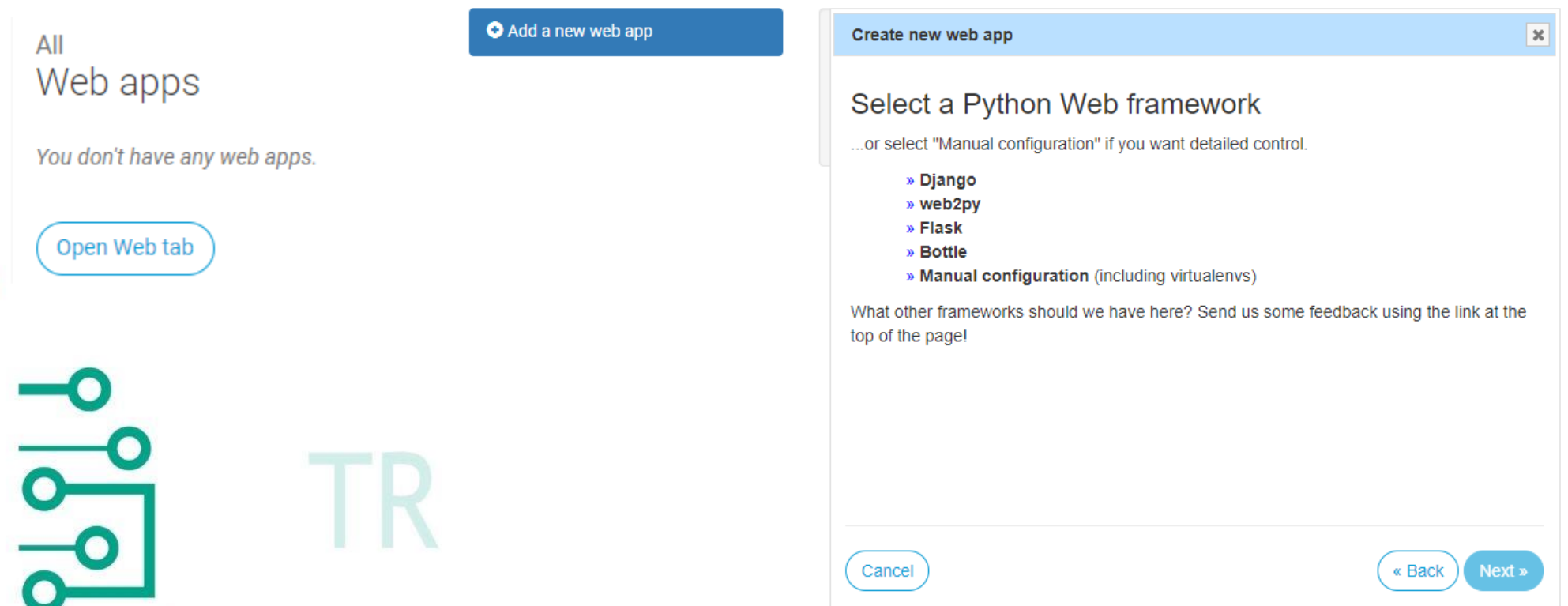
☐ I agree to the [Terms and Conditions](#) and the [Privacy and Cookies Policy](#), and confirm that I am at least 13 years old.

[Register](#)

We promise not to spam or pass your details on to anyone else.

TRABALHANDO COM FLASK

Iremos criar um WEBAPP no Python Anywhere. Iremos configurar o tipo de framework e a versão do Python.



The image shows a screenshot of the Python Anywhere interface. On the left, there's a sidebar with 'All Web apps' and a message 'You don't have any web apps.' with an 'Open Web tab' button. In the center, there's a blue button labeled 'Add a new web app'. On the right, a modal dialog box titled 'Create new web app' is open. Inside the dialog, it says 'Select a Python Web framework' followed by '...or select "Manual configuration" if you want detailed control.' Below this is a list of options: Django, web2py, Flask, Bottle, and Manual configuration (including virtualenvs). At the bottom of the dialog, there's a feedback prompt: 'What other frameworks should we have here? Send us some feedback using the link at the top of the page!'. Navigation buttons at the bottom include 'Cancel', '« Back', and 'Next »'.

All
Web apps

You don't have any web apps.

Open Web tab

Add a new web app

Create new web app

Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » Django
- » web2py
- » Flask
- » Bottle
- » Manual configuration (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

Cancel « Back Next »

TRABALHANDO COM FLASK

Iremos passar o nome do arquivo de configuração do site. Nosso arquivo é o **app.py**

Create new web app

Select a Python version

- » Python 3.6 (Flask 2.0.0)
- » Python 3.7 (Flask 2.0.0)
- » Python 3.8 (Flask 2.0.0)
- » Python 3.9 (Flask 2.0.0)

Note: If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are [instructions here](#).

Cancel

« Back

Next »

Create new web app

Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. If this file already exists, its contents will be overwritten with the new app.

Path

/home/joacavichioli/mysite/flask_app.py

Cancel

« Back


Next »

TRABALHANDO COM FLASK

Vamos carregar os arquivos do nosso site e aplicação dentro do Python AnyWhere

Code:

What your site is running.

Source code:	/home/joaocavichiolli/mysite	➔Go to directory
Working directory:	/home/joaocavichiolli/	➔Go to directory
WSGI configuration file:	/var/www/joaocavichiolli_pythonanywhere_com_wsgi.py	
Python version:	3.8	

TRABALHANDO COM FLASK

Utilizando Flask como API. Podemos utilizar o flask como uma API REST

O que é API REST.

API REST, também chamada de API RESTful, é uma interface de programação de aplicações (API ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful.

REST é a sigla em inglês para transferência representacional de estado.

Quando um cliente faz uma solicitação usando uma API RESTful, essa API transfere uma representação do estado do recurso ao solicitante ou endpoint. Essa informação (ou representação) é entregue via HTTP utilizando um dos vários formatos possíveis: Javascript Object Notation (JSON), HTML, XLT, Python, PHP ou texto sem formatação. O formato JSON é a linguagem de programação mais usada porque, apesar de seu nome, é independente de qualquer outra linguagem e pode ser lido por máquinas e humanos.

TRABALHANDO COM FLASK

Criando uma route e trabalhando com a biblioteca JSONIFY para transformar os dados em formato JSON.

Vamos criar um url rest onde iremos passar os valores de Nome e Endereço o mesmo irá Retornar o valor em JSON

Vamos importar o módulo jsonify dentro do flask.

```
from flask import jsonify
```

```
@app.route('/enderecos/<string:nome>,<string:end>')  
def hello(nome,end):  
    return jsonify({'Nome':nome,'Endereco':end})
```

Vamos configurar para o JSON não ordenar os valores em ordem alfabética

```
app.config['JSON_SORT_KEYS'] = False
```

TRABALHANDO COM FLASK

Para testar basta acessar o endereço do servidor web passando os parâmetros da API e validar o retorno.



← → ↻ ⓘ <http://127.0.0.1:5000/enderecos/Joao,RuaTiradentes>

```
{"Nome": "Joao", "Endereco": "RuaTiradentes"}
```