



# **Curso completo de Python!**





**Instrutor: Vitor Mazuco**

**<http://facebook.com/vitormazuco>**

**Email: [vitor.mazuco@gmail.com](mailto:vitor.mazuco@gmail.com)**

**WebSite:**

**<http://vmzsolutions.com.br>**



# Flask

web development,  
one drop at a time



## Criando Rotas e Views

Até agora, temos apenas uma rota e um view em nosso arquivo. Mas o que são as rotas e views??





## Criando Rotas e Views

As rotas são as nossas URL's que você entra pelo navegador. Ex:

- / - seria a index ou raiz de um site
- /usuarios/ - seria a rota
- /grupos/ - seria a rota de grupos



## Criando Rotas e Views

O flask já usa o conceito de navegação de URL amigável.

Antigamente as URL's eram assim:

`http://localhost/?page=exemplo`

Agora com o Flask será assim:

`http://localhost/exemplo`



## Criando Rotas e Views

As views são as funções que nos retornam os elementos que o usuário veja as aplicações. Ex, a rota / possui uma função chamada de hello:

```
def hello():  
    return "Ola Mundo!!!!!"
```



## Criando Rotas e Views

A função não precisa ter o mesmo nome da sua rota, porém é de praxe colocar de mesmo nome. O retorno dessa função virá o que tem nessa página para o usuário.





## Criando Rotas e Views

As views podem nos retornar diversos tipos de formatos, como json, xml, plain text, etc. O que faz de fato a sua saída, é o seu content type que ao fazer as requests do tipo json de acordo com as nossas aulas anteriores.



## Criando Rotas e Views

Para a nossa aplicação nos retornar em XML no Flask, é preciso importar a classe response do Flask:

```
from flask import Response
```



## Criando Rotas e Views

E no retorno da View coloque assim:

```
return Response(xml, mimetype="text/xml")
```

O formato do tipo XML é muito usado em API's, logo é importante retornar esse tipo de formato.



## Criando Rotas e Views

Logo, caso queria criar uma nova view, podemos simplesmente copiar a função e colocar logo abaixo:

```
@app.route("/usuarios/")  
def index_usuarios():  
    return "Aqui gerenciamos os usuarios"
```

Depois disso é só salvar o seu arquivo e entrar na aba correta /usuarios/.



## Criando Rotas e Views

As funções nunca podem ter os mesmos nomes! Tome cuidado com o coipar e colar, pois assim você pode esquecer de trocar o nome das funções e o Flask irá retornar um erro!

Não existe um limite de funções, ou seja, você pode programar todo o seu site em um único arquivo!



## Criando Rotas e Views

Quando trabalhamos com API's, é preciso criar as requisições de forma diferente através dos métodos usados, como os CRUD (PUT, DELETE, POST e GET). Logo numa mesma rota, podemos usar 2 ou mais métodos diferentes como criar e remover usuários de um cadastro.



## Criando Rotas e Views

A nossa função `index_usuarios` que é a rota `/usuarios/`, caso você não especifique o seu método a ser usado, o Flask irá usar o GET por padrão. Para mudar isso, bastar colocar essas linhas abaixo:

```
@app.route("/usuarios/", methods=["POST"])
```



## Criando Rotas e Views

Ele recebe um parâmetro de uma lista, então podemos colocar mais que um:

```
@app.route("/usuarios/", methods=["POST","PUT","DELETE"])
```





## Criando Rotas e Views

Agora, podemos testar nossa aplicação com o RestClient para vermos o resultado. Deixe a nossa view assim:

```
@app.route("/usuarios/", methods=["POST"])  
def index_usuarios():  
    return "Aqui cadastramos os usuarios"
```



## Criando Rotas e Views

Podemos também passar as variáveis pelas rotas, ex:

```
@app.route("/usuarios/<ID>/", methods=["GET"])
```

Nessa rota, é dado o ID que deve ser retornado pelo BD. É possível também, criar pelo ID como parâmetro da função que na sequência.



## Criando Rotas e Views

Podemos especificar o tipo de valor que aquela rota irá receber, ex:

```
@app.route("/usuarios/<int:id>/", methods=["GET"])
```

Caso não seja informado o tipo de valor, ele dará um erro ao chamar essa view.



## Criando Rotas e Views

Nesse momento, já sabemos que as nossas views são nada mais nada menos que rotas e funções! Agora, vamos criar views com as operações de CRUD em um BD! (**SERÁ VISTO NAS PRÓXIMAS AULAS!**)

```
@app.route("/usuarios/", methods=["POST"]) # create
@app.route("/usuarios/<int:id>/", methods=["GET"]) # retrieve
@app.route("/usuarios/<int:id>/", methods=["PUT"]) # update
@app.route("/usuarios/<int:id>/", methods=["DELETE"]) # delete
```



## Criando Rotas e Views

As funções logo a seguir podem vim nessa sequência:

- `def insert_usuario():`
- `def select_usuario():`
- `def update_usuario():`
- `def delete_usuario():`

Já a função *index* não precisa ser alterada, pois todos os usuários serão listados por padrão.



## Criando Rotas e Views

Agora que temos as nossas views, precisamos fazer com que essas views tenham os dados de formato json e que enviem dados também em json. Também vimos que o Python é extremamente prático que os dicionários são praticamente idênticos.



## Criando Rotas e Views

O flask tem um módulo que faz esse tipo de conversão de um dicionário para um json e manda a resposta para o usuário.



## Criando Rotas e Views

Podemos importar o módulo jsonify(use o pip install para instalar), que faz essa conversação, e para testar basta você modificar a view index de nosso flask!

```
from flask import Flask,jsonify
```





## Criando Rotas e Views

A nossa view ficará assim:

```
@app.route("/usuarios/")  
def index_usuarios():  
    return jsonify({"message": "Aqui serao listados os usuarios"})
```

Depois, faça novamente o teste com o RestClient e veja que a resposta virá num outro formato!

\* Veja mais no arquivo flask\_application2.py