

P V T H O W

web war

Nas trincheiras do Desenvolvimento



Ednaldo Paes



# CAPITULO

## Um

Cavando Requests e Codigos de Retorno

python web war

DESENVOLVER APLICAÇÕES WEB COM PYTHON É UMA HABILIDADE VALIOSA E CADA VEZ MAIS REQUISITADA.

---

ESTE CAPÍTULO TEM COMO OBJETIVO APRESENTAR O AMBIENTE DE DESENVOLVIMENTO WEB COM PYTHON, EXPLICAR OS PRINCIPAIS TIPOS DE REQUESTS E OS CÓDIGOS DE RETORNO HTTP, SEMPRE DE MANEIRA SIMPLES E OBJETIVA, COM EXEMPLOS PRÁTICOS.

---

PARA DESENVOLVER APLICAÇÕES WEB COM PYTHON, UTILIZAMOS FRAMEWORKS QUE FACILITAM A CRIAÇÃO E GESTÃO DE PROJETOS. OS PRINCIPAIS FRAMEWORKS SÃO:

---

**FLASK:** UM MICROFRAMEWORK LEVE E SIMPLES, IDEAL PARA PROJETOS MENORES.

---

**DJANGO:** UM FRAMEWORK COMPLETO, RECOMENDADO PARA PROJETOS MAIORES E MAIS COMPLEXOS.

---

**FASTAPI:** UM FRAMEWORK MODERNO E RÁPIDO, IDEAL PARA CONSTRUIR APIS.

**VAMOS TRABALHAR COM O FLASK  
NO NOSSO EBOOK.**

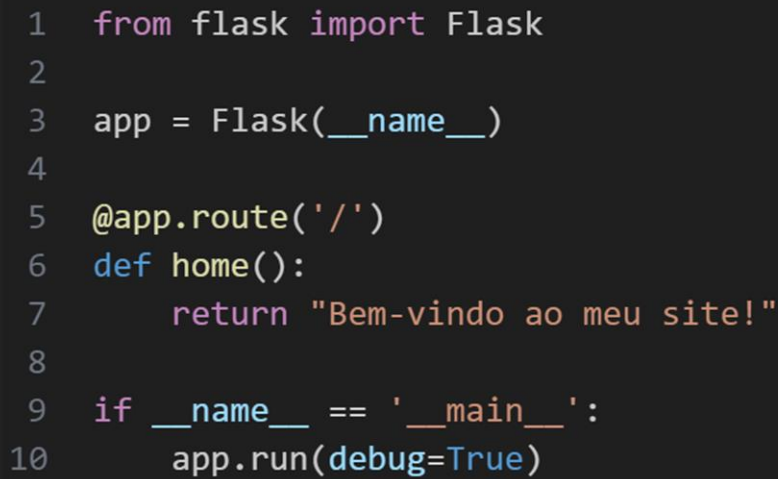
PARA ISSO,  
VAMOS RODAR O COMANDO

```
pip install flask
```

PARA INSTALAR O FLASK NO NOSSO  
AMBIENTE VIRTUAL

VAMOS CRIAR UMA APLICAÇÃO BÁSICA EM FLASK PARA INICIAR NOSSO PROJETO.

---



```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def home():
7      return "Bem-vindo ao meu site!"
8
9  if __name__ == '__main__':
10     app.run(debug=True)
```

COM APENAS ISTO, EXECUTANDO O FLASK RUN, JÁ TEMOS UMA APLICAÇÃO FUNCIONAL.

---

CLARO QUE O FLASK PODE SER, SIM, SIMPLES, MAS ELE TAMBÉM OFERECE UMA GRANDE GAMA DE BIBLIOTECAS E OPÇÕES PARA FAZER UM PROJETO COMPLETO, GARANTINDO UM AMBIENTE O MAIS LIMPO POSSÍVEL.

Veja mais: [Site Oficial do Flask](#)

---



## TIPOS DE REQUESTS E SUAS FUNÇÕES

---

REQUESTS SÃO MENSAGENS ENVIADAS PELO CLIENTE AO SERVIDOR, SOLICITANDO INFORMAÇÕES OU AÇÕES. OS PRINCIPAIS TIPOS DE REQUESTS SÃO: POST, GET, PUT AND DELETE. ESTES SÃO ALGUNS VERBOS HTTP (OU MÉTODOS HTTP) QUE TRADUZEM PARA A WEB O CONHECIDO CRUD: CREATE, READ, UPDATE AND DELETE.

---

AS REQUESTS NÃO SE LIMITAM APENAS NESSAS 4 REQUISIÇÕES. DEVEMOS ENFATIZAR QUE UM MÉTODO PODE SER SEGURO, IDEMPOTENTE E ARMAZENÁVEL EM CACHE, SE USADO CORRETAMENTE.



Veja mais: [HTTP request methods](https://requests.readthedocs.io/en/latest/user/quickstart/#http-verbs)

---

## TIPOS DE REQUESTS E SUAS FUNÇÕES

---

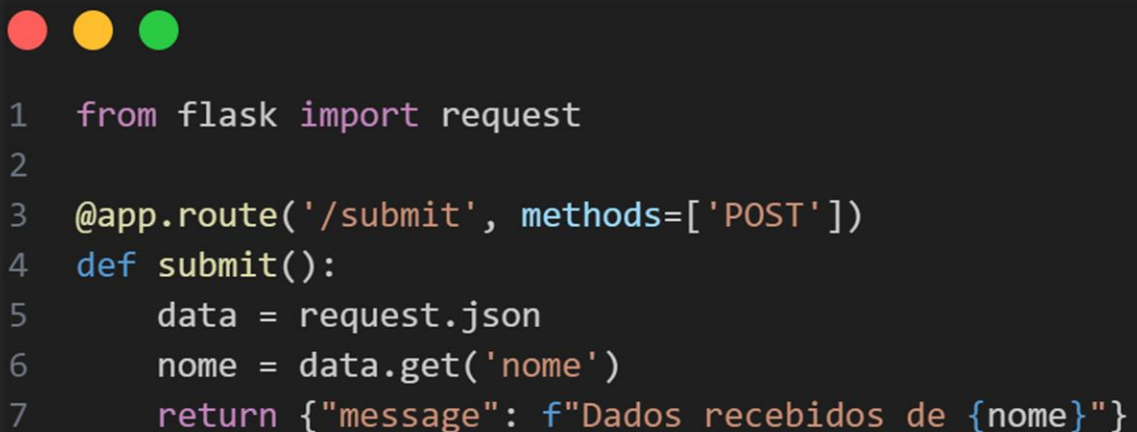
GET: UTILIZADO PARA SOLICITAR DADOS DE UM SERVIDOR. É O TIPO MAIS COMUM DE REQUEST.



```
1 @app.route('/dados')
2 def dados():
3     return {"nome": "João", "idade": 30}
```

---

POST: UTILIZADO PARA ENVIAR DADOS AO SERVIDOR, GERALMENTE PARA CRIAR OU ATUALIZAR RECURSOS.




```
1 from flask import request
2
3 @app.route('/submit', methods=['POST'])
4 def submit():
5     data = request.json
6     nome = data.get('nome')
7     return {"message": f"Dados recebidos de {nome}"}
```

## TIPOS DE REQUESTS E SUAS FUNÇÕES

---

PUT: UTILIZADO PARA ATUALIZAR RECURSOS  
EXISTENTES NO SERVIDOR.



```
1 @app.route('/update/<int:id>', methods=['PUT'])
2 def update(id):
3     data = request.json
4     novo_nome = data.get('nome')
5     return {"message": f"Recurso {id} atualizado para {novo_nome}"}
```

---

DELETE: UTILIZADO PARA DELETAR RECURSOS  
DO SERVIDOR.



```
1 @app.route('/delete/<int:id>', methods=['DELETE'])
2 def delete(id):
3     return {"message": f"Recurso {id} deletado"}
```

---




## CÓDIGOS DE RETORNO HTTP

---

DURANTE AS REQUESTS, O SERVIDOR E O CLIENTE TROCAM INFORMAÇÕES. OS CÓDIGOS, POR SUA VEZ, É A FORMA QUE O SERVIDOR AVISA QUE UMA REQUISIÇÃO FOI OU NÃO BEM-SUCEDIDA. ALGUNS DOS PRINCIPAIS CÓDIGOS DE RETORNO SÃO:

---

200 OK: INDICA QUE A SOLICITAÇÃO FOI BEM-SUCEDIDA.



```
1 @app.route('/success')
2 def success():
3     return {"message": "Requisição bem-sucedida"}, 200
```

201 CREATED: INDICA QUE UM RECURSO FOI CRIADO COM SUCESSO.



```
1 @app.route('/create', methods=['POST'])
2 def create():
3     data = request.json
4     return {"message": "Recurso criado"}, 201
```

---

## CÓDIGOS DE RETORNO HTTP

---

400 BAD REQUEST: INDICA QUE A SOLICITAÇÃO FOI INVÁLIDA.



```
1 @app.route('/badrequest')
2 def bad_request():
3     return {"message": "Requisição inválida"}, 400
```

404 NOT FOUND: INDICA QUE O RECURSO SOLICITADO NÃO FOI ENCONTRADO.



```
1 @app.route('/notfound')
2 def not_found():
3     return {"message": "Recurso não encontrado"}, 404
```

---

500 INTERNAL SERVER ERROR: INDICA UM ERRO NO SERVIDOR.



```
1 @app.route('/error')
2 def error():
3     return {"message": "Erro interno do servidor"}, 500
```

---

## RESUMÃO DO CAPÍTULO

---

COMPREENDER O AMBIENTE DE DESENVOLVIMENTO WEB COM PYTHON, OS PRINCIPAIS TIPOS DE REQUESTS E OS CÓDIGOS DE RETORNO HTTP É FUNDAMENTAL PARA CRIAR APLICAÇÕES WEB EFICIENTES E ROBUSTAS. UTILIZANDO FRAMEWORKS COMO FLASK, DJANGO OU FASTAPI, VOCÊ PODE DESENVOLVER DESDE PROJETOS SIMPLES ATÉ SISTEMAS COMPLEXOS, MANIPULANDO DADOS E GERENCIANDO RESPOSTAS DE FORMA ADEQUADA.

---

NO PRÓXIMO CAPÍTULO VEREMOS UMA ESTRUTURA INCRÍVEL, SIMPLES E EFICIENTE PARA MUITAS FINALIDADES, INCLUSIVE APIs.

---