Framework

Luiza <CODE>

luizalabs MAGALU

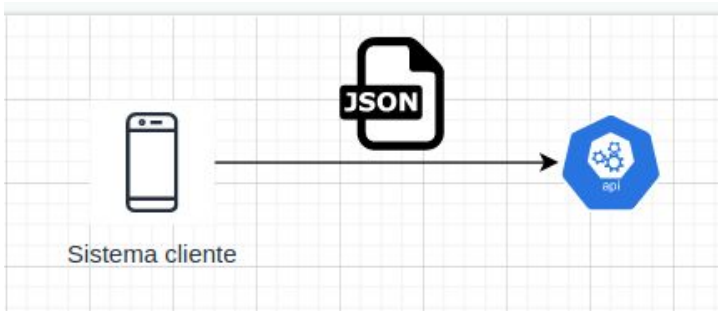
O que vamos ver?

- APIs
- *Frameworks* de APIs em Python
- Instalando o FastAPI
- Hello World
- Testando o Hello World
- FastApi
  - Métodos HTTP
  - Parâmetros
    - Query Parameters
    - Body request
  - Requisições e Respostas


APIs

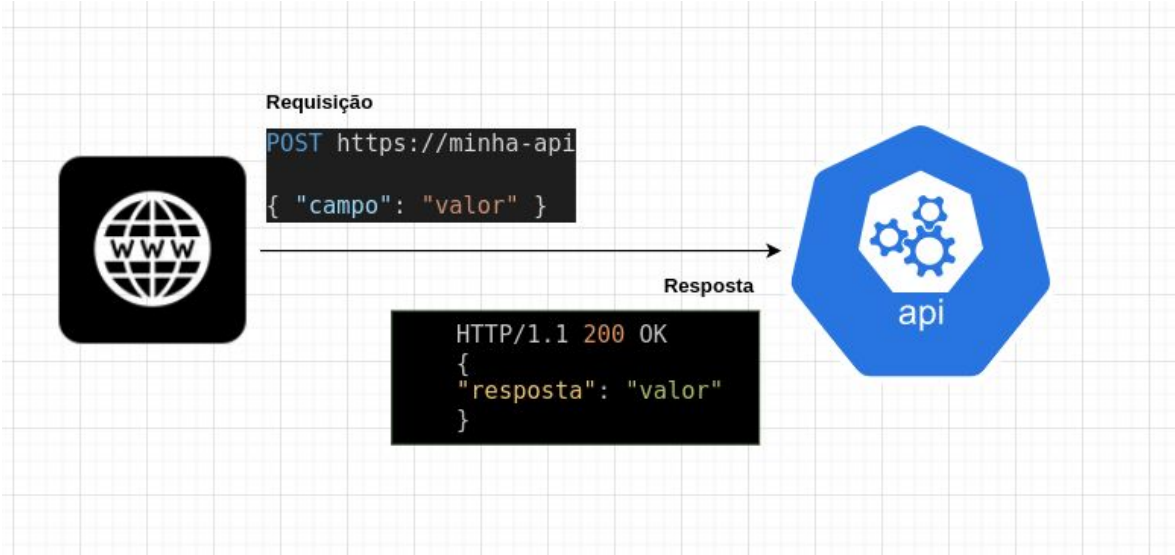
API Rest

- API (*Application Programming Interface*): Conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos (Techtudo)
- REST (*Representational State Transfer*): Conjunto de restrições para aplicações Web (*Red Hat*).
- API REST: Conjunto de restrições e normas para comunicação entre sistemas HTTP(s)/Web.



API Rest

- Para a comunicação:
  - Métodos (GET, POST)
  - Caminhos: “api/produtos”.
  - Parâmetros




Luiza
<CODE>
Frameworks do Python
labs
Luizalabs
Magalu

Frameworks

- Bibliotecas para *desenvolvimento rápido* para Web (Api Rest).

aiohttp

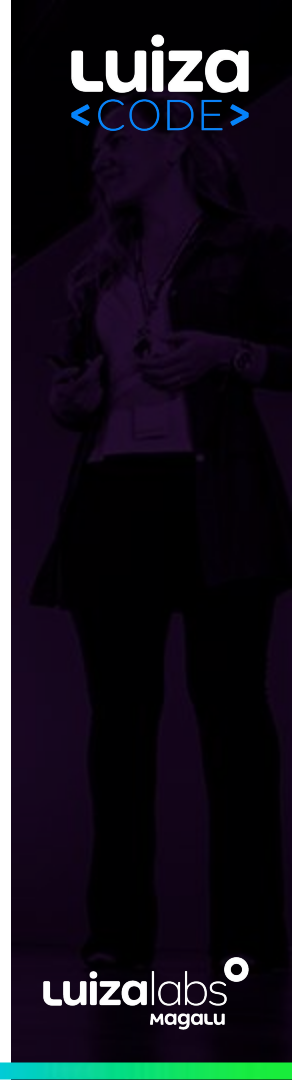
Instalando o FastAPI

FastAPI

- FastAPI: *Framework* Web para desenvolvimento de APIs Rest.
- https://fastapi.tiangolo.com/

```bash
$ pip install fastapi
```

```bash
$ pip install "uvicorn[standard]"
```



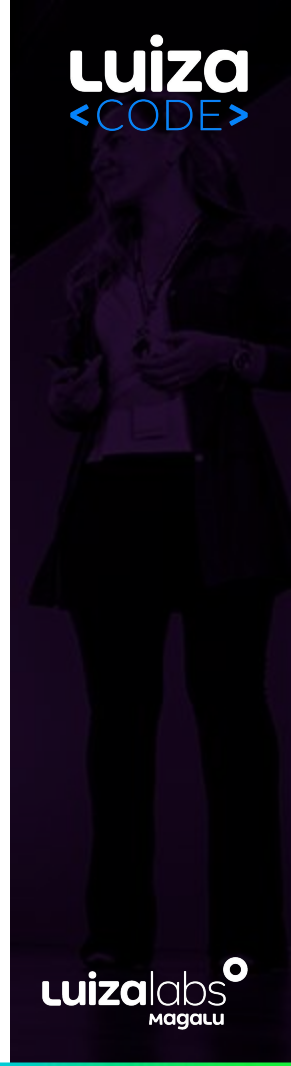

Portas TCP

Portas TCP

- Um servidor pode ter vários serviços rodando nele.
- O endereço do servidor é composto por IP + Porta.
- Porta está associada a um serviço rodando no servidor.
- localhost = 127.0.0.1

Exemplo

127.0.0.1:8000




FastAPI
Hello World

FastAPI - Hello World

```python
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
def read_root():
    return {"Hello": "World"}

```

Link

Dentro da mesma pasta, executar no terminal:

uvicorn <nome do arquivo sem o .py>:app --reload

Bibliotecas usadas pelo FastAPI

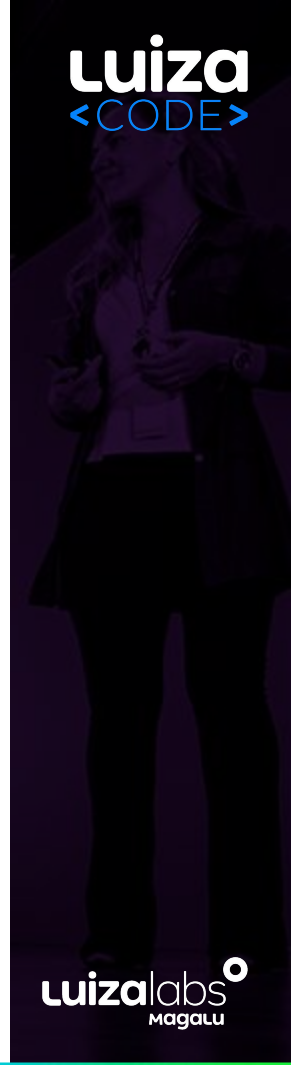
- uvicorn
  - ASGI (Asynchronous Server Gateway Interface)

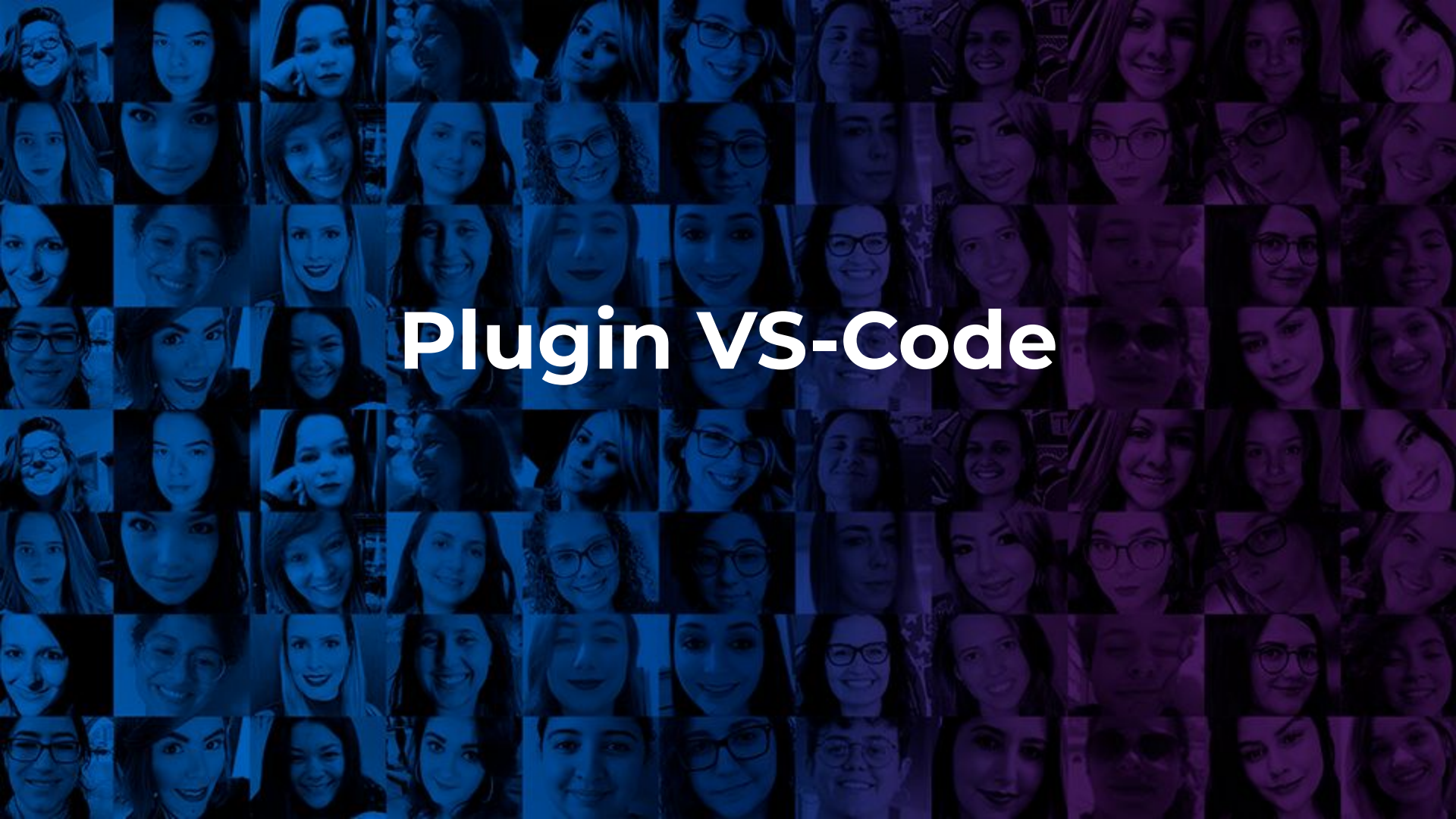
- Utiliza Pydantic.
  - Serialização.


Testando o
FastAPI

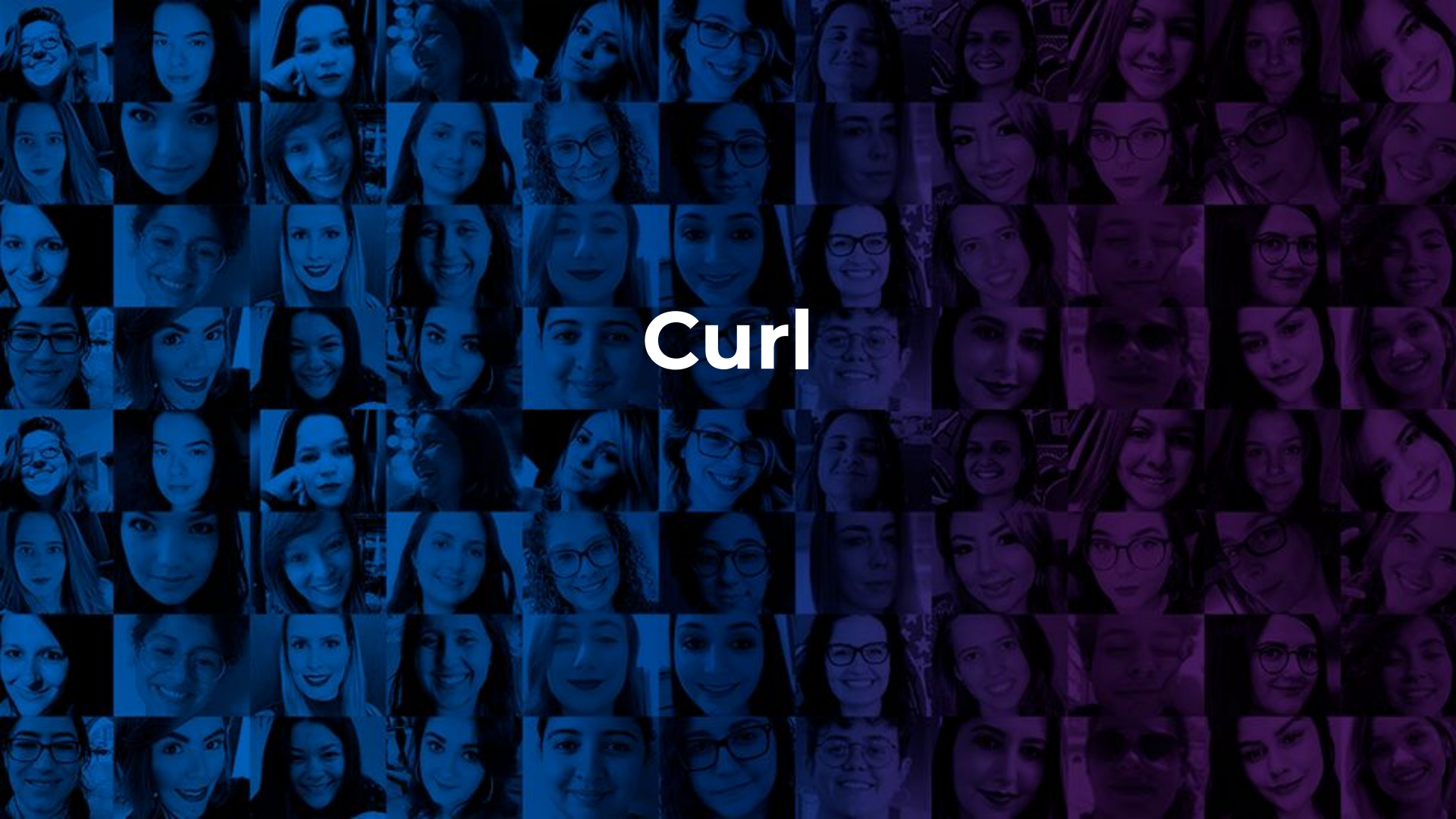
Testando o FastAPI

- Primeiro exemplo: Olá mundo!
- Testando do navegador:
  - http://localhost:8000
- Testando com o VS-Code
  - https://marketplace.visualstudio.com/items?itemName=humao.rest-client
- Testando com o Postman
- Testando com a linha de comando:
  - `curl http://localhost:8000`




Plugin VS-Code

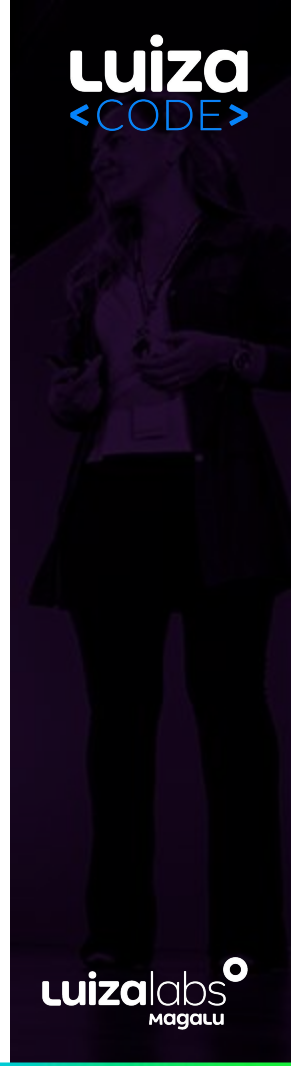

Postman


Curl

Curl

- Testando com a linha de comando:
  - curl http://localhost:8000
  - Link de Referência
  - curl

```
-X POST
-H "Content-Type: application/json"
-d '{"option1":"value1", "option2":"value2"}'
https://www.example.com
```




Métodos HTTP

Métodos HTTP

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods

GET

O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.

HEAD

O método HEAD solicita uma resposta de forma idêntica ao método GET, porém sem conter o corpo da resposta.

POST

O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.

PUT

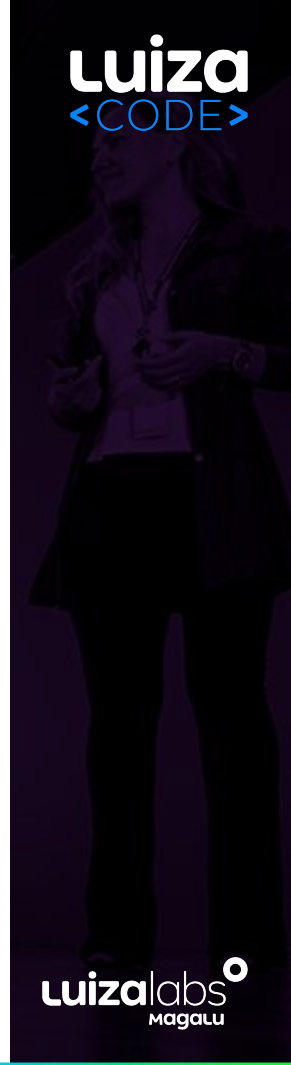
O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.

DELETE

O método DELETE remove um recurso específico.

OPTIONS

O método OPTIONS é usado para descrever as opções de comunicação com o recurso de destino.

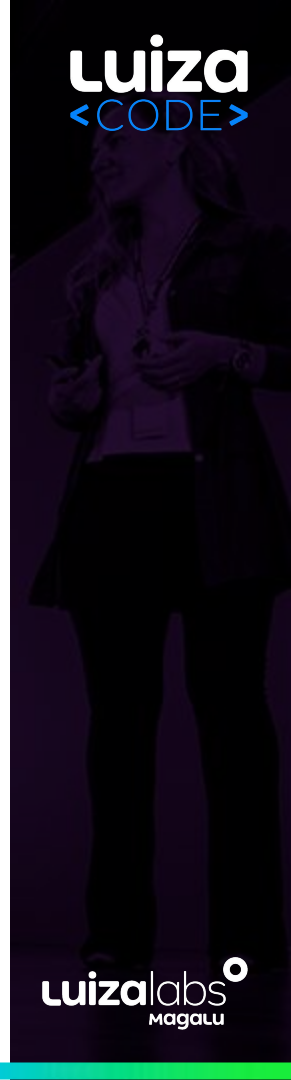


Parâmetros
Query Parameters

Parâmetros - Query Parameters

https://fastapi.tiangolo.com/tutorial/query-params/

```python
dados = [{"1": "A"}, {"2": "B"}, {"3": "C"}]
@app.get("/itens/")
async def read_item(numero: int = 0):
    return dados[numero]
```




Parâmetros
Body request

Parâmetros - Body request

https://fastapi.tiangolo.com/tutorial/body/

```python
from pydantic import BaseModel


class Item(BaseModel):
    nome: str
    id: int


@app.post("/elementos/")
async def create_item(item: Item):
    return item.id, item.nome
```

Perguntas?


magalu
#VemSerFeliz