

Postman

Postman

- É uma plataforma independente de software para testar APIs.
- **Site:** <https://www.postman.com/downloads/>
- **Obs:** Ao invés de baixar e instalar o Postman, você também pode instalar sua extensão no Visual Studio Code.



PIS / hello



Save



Share



GET



http://127.0.0.1:8000/

Send



Params

Authorization

Headers (6)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (4)

Test Results



200 OK

• 14 ms • 150 B •



Save Response



{ } JSON



▶ Preview

🖼 Visualize



```
1 {
2   "message": "Hello World"
3 }
```

Criando a rota Mensagens (Post)

```
from fastapi import FastAPI
from fastapi.params import Body
```

```
app = FastAPI()
```

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

```
async def root():
```

```
    return {"message": "Hello World"}
```

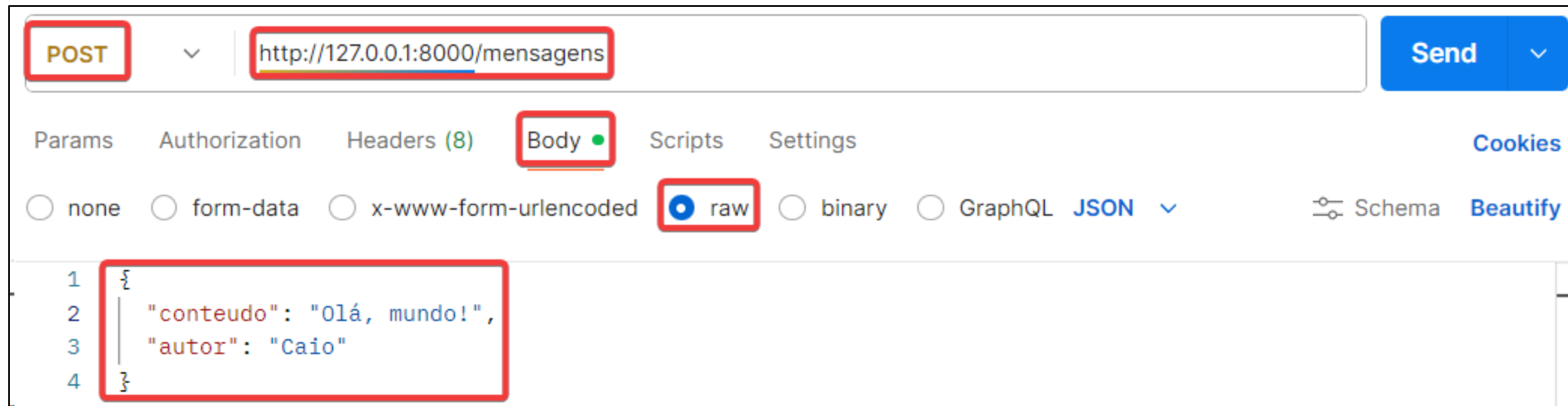
```
@app.post("/mensagens")
```

```
def criar_mensagem(res: dict = Body(...)):
```

```
    print(res)
```

```
    return {"Mensagem": "Criado com Sucesso!"}
```

Postman



Terminal

```
{'conteudo': 'Olá, mundo!', 'autor': 'Caio'}
```

Postman

The screenshot displays the Postman interface for a POST request. At the top, the method is set to **POST** and the URL is `http://127.0.0.1:8000/mensagens`. A **Send** button is located to the right. Below the URL bar, tabs for **Params**, **Authorization**, **Headers (8)**, **Body** (selected), **Scripts**, and **Settings** are visible. The **Body** tab shows the request body in **raw** **JSON** format:

```
{
  "conteudo": "Olá, mundo!",
  "autor": "Caio"
}
```

. The bottom section shows the response with a status of **200 OK**, a time of **9 ms**, and a size of **159 B**. The response body is also in **JSON** format:

```
{
  "Mensagem": "Criado com Sucesso!"
}
```

. The interface includes various utility buttons like **Schema**, **Beautify**, **Preview**, **Visualize**, and **Save Response**.

POST `http://127.0.0.1:8000/mensagens` **Send**

Params Authorization Headers (8) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ☐ Schema **Beautify**

```
1 {
2   "conteudo": "Olá, mundo!",
3   "autor": "Caio"
4 }
```

Body Cookies Headers (4) Test Results **200 OK** • 9 ms • 159 B • Save Response

{} **JSON** Preview Visualize

```
1 {
2   "Mensagem": "Criado com Sucesso!"
3 }
```

Criando a rota Mensagens (Post)

```
from fastapi import FastAPI
from fastapi.params import Body
```

Body é uma classe do FastAPI que indica que o valor de um parâmetro virá do corpo da requisição **HTTP** (body da requisição). É usado principalmente em rotas **POST** ou **PUT**, quando você quer enviar dados em **JSON**.

```
app = FastAPI()
```

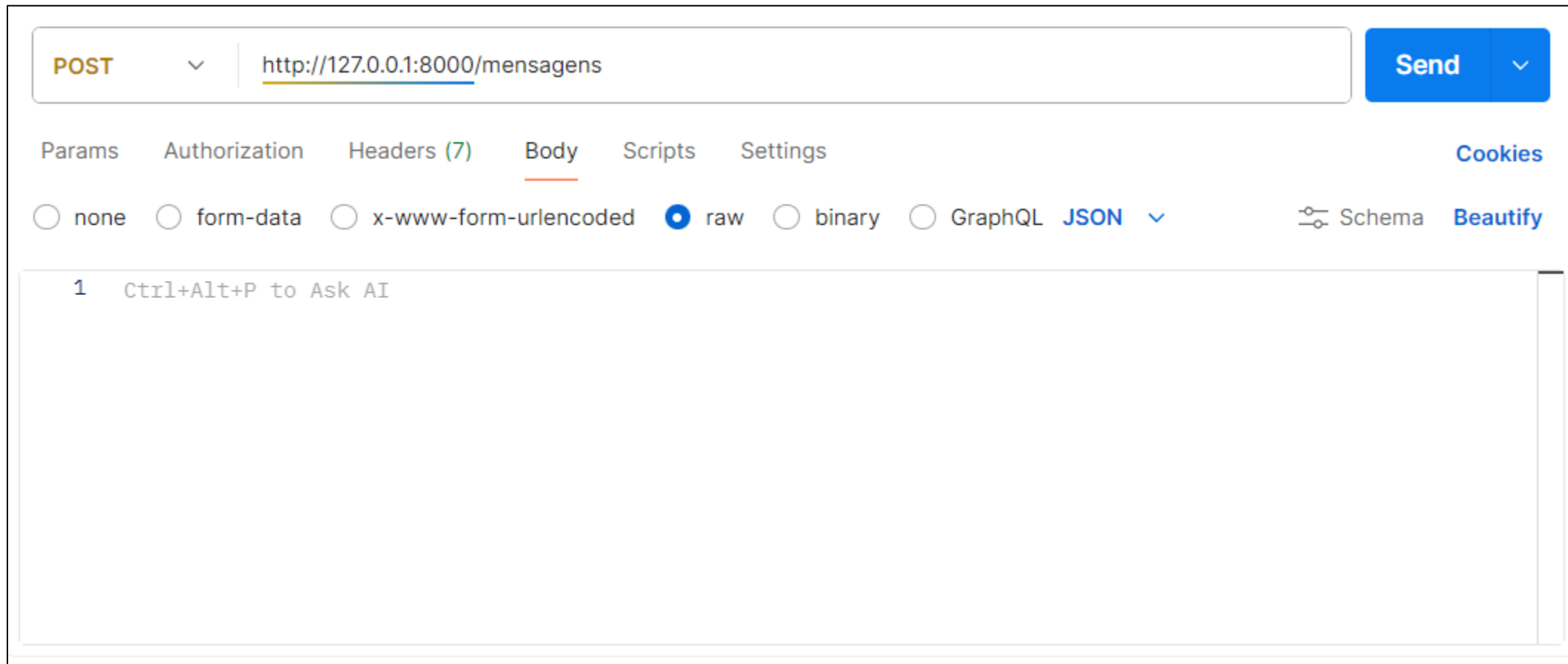
```
@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Define uma rota **POST** no caminho /mensagens. **POST** é usado para criar um novo recurso.

```
@app.post("/mensagens")
def criar_mensagem(res: dict = Body(...)):
    print(res)
    return {"Mensagem": "Criado com Sucesso!"}
```

- Define que a função espera um dicionário **JSON** no corpo da requisição.
- ... significa obrigatório (não pode ser vazio).
- **res** será um dicionário Python contendo os dados enviados pelo cliente.

1) O que vai acontecer?



The image shows a REST client interface with the following elements:

- Method:** POST (indicated by a dropdown arrow)
- URL:** `http://127.0.0.1:8000/mensagens`
- Buttons:** Send (blue button with a dropdown arrow)
- Tabs:** Params, Authorization, Headers (7), Body (active), Scripts, Settings, Cookies
- Body Type Selection:** ☐ none, ☐ form-data, ☐ x-www-form-urlencoded, ☒ raw, ☐ binary, ☐ GraphQL
- Format:** JSON (with a dropdown arrow)
- Actions:** Schema (with a dropdown arrow), Beautify
- Body Content:** A text area containing the text: `1 Ctrl+Alt+P to Ask AI`

POST

<http://127.0.0.1:8000/mensagens>

Send



Params

Authorization

Headers (7)

Body

Scripts

Settings

Cookies



none



form-data



x-www-form-urlencoded



raw



binary



GraphQL

JSON



Schema

Beautify

1 Ctrl+Alt+P to Ask AI

Body

Cookies

Headers (4)

Test Results



422 Unprocessable Content

• 5 ms • 226 B •



e.g.

Save Response



JSON



▶ Preview



Debug with AI



```
1  {
2    "detail": [
3      {
4        "type": "missing",
5        "loc": [
6          "body"
7        ],
8        "msg": "Field required",
9        "input": null
10     }
11   ]
12 }
```

2) O que vai acontecer?

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** <http://127.0.0.1:8000/mensagens>
- Buttons:** Send, Cookies, Beautify
- Tabs:** Params, Authorization, Headers (8), Body (selected), Scripts, Settings
- Body Type:** raw (selected), none, form-data, x-www-form-urlencoded, binary, GraphQL
- Body Content:** A JSON object `{}` is entered on a single line. The line is numbered 1, 2, and 3 in the left margin, and the closing brace `}` is on the third line. A red box highlights the first three lines of the body content.

POST ▼ | <http://127.0.0.1:8000/mensagens> Send ▼

Params Authorization Headers (8) Body ● Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Schema Beautify

```
1 {  
2  
3 }
```

Body Cookies Headers (4) Test Results 🕒 200 OK • 6 ms • 159 B • 🌐 | e.g. Save Response ⋮

{} JSON ▼ ▶ Preview 🖼️ Visualize ▼ ↻ ≡ 🔍 📄 🔗

```
1 {  
2   "Mensagem": "Criado com Sucesso!"  
3 }
```

`{}` vazio ainda é válido!!!

Qual o problema deste código?

POST ▼ Send ▼

Params Authorization Headers (8) Body ● Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Schema Beautify

```
1 {
2   "conteudo": "Olá, mundo!",
3   "autor": "Caio",
4   "lala": "lala",
5   "lele": "lele",
6   "lili": "lili"
7 }
```

Body Cookies Headers (4) Test Results ↺ 200 OK • 9 ms • 159 B • 🌐 | 📄 Save Response ⋮

{} JSON ▼ ▶ Preview 🖼️ Visualize ▼ ↺ ≡ 🔍 📄 🔗

```
1 {
2   "Mensagem": "Criado com Sucesso!"
3 }
```

Eu posso enviar qualquer coisa!
Temos que validar os dados que os clientes (neste caso o Postman) enviam!