

Documentação da API Rest

# Desafio 1

---

Gleidson Felipe Pereira da Silva

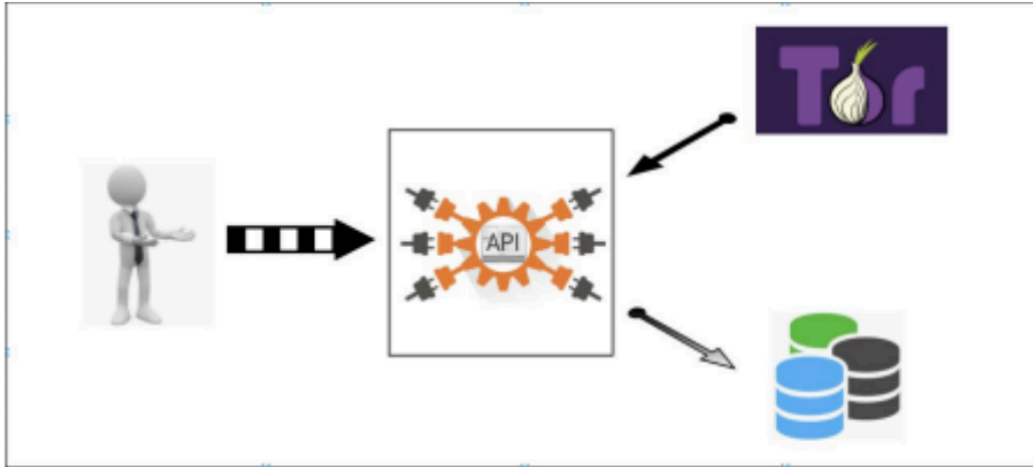
Novembro de 2024



## Resumo Executivo

A motivação do desafio vislumbra que todos os membros do time de segurança cibernética tenham um conhecimento básico de Networking, infraestrutura, desenvolvimento e base de dados. Nesse sentido, o desafio proposto tem como objetivo desenvolver um sistema para obter endereços IPs maliciosos, a fim de bloquear o tráfego proveniente desses IPs.

## 1. Visão Geral



Foi desenvolvida uma API REST em Python que contempla as seguintes funcionalidades:

1. Um endpoint GET que devolve IPs a partir da fonte externa com a seguinte URL:  
<https://check.torproject.org/torbulkexitlist>
2. Um endpoint POST que recebe um IP e adiciona este IP na base de dados (postgresql). Nesta base, estão os IPs que não queremos que apareçam no output do endpoint 3
3. Um endpoint GET que retorna os IPs da base de dados.
4. Um endpoint GET que retorna os IPs filtrados, ou seja, retornará a lista de IPs de saída do TOR (fonte externa) e excluindo os IPs adicionados no endpoint 2.
5. Um endpoint DELETE (extra) que faz a remoção de endereço IP.



## 2. Desenvolvimento

Para que aplicação seja executada com sucesso, faz-se necessário a instalação das seguintes dependências:

- Python 3.10
- Framework FastAPI

Justificativa: Este framework foi escolhido por permitir o desenvolvimento de APIs assíncronas e, consequentemente, ganho de performance nas requisições, bem como disponibilizar e facilitar uma documentação/swagger elegante e intuitiva.

- PostgreSQL
- SQLAlchemy
- asyncpg
- aiohttp
- requests

Como foi usado a ferramenta pipenv para gerenciamento de ambientes virtuais e dependências, foi gerado o arquivo pipfile onde pode ser encontrada todas as dependências.

## 2.1 Execução Local

- Instale o python 3.9+
- Instale e configure um ambiente virtual com **pipenv**:
- `pip install pipenv`
- `pipenv install fastapi`
- `pipenv install uvicorn`
- `pipenv install sqlalchemy`
- `pipenv install asyncpg`
- `pipenv install aiohttp`
- `pipenv install requests`

## Clonar Repositório

Faça o clone do repositório:

- `git clone`  
[https://C4B4N0:ghp\\_H5XxX8Z4lzlr9lkvfNc3PisO0p6sE61XIMhb@github.com/C4B4N0/API-REST-IPS-TOR.git](https://C4B4N0:ghp_H5XxX8Z4lzlr9lkvfNc3PisO0p6sE61XIMhb@github.com/C4B4N0/API-REST-IPS-TOR.git)

## Como executar

- Adicione o caminho do projeto na variável PYTHONPATH no arquivo .env
- Inicie o banco de dados postgres e o pgadmin: `docker-compose up -d`
- Inicie o ambiente: `pipenv shell`
- Instale as dependências usando o comando `pipenv install "dependencia"` (verifique o arquivo pipfile)
- Inicie a aplicação: `uvicorn main:app --port 8080`

## 2.2 Execução no Docker

Pré-requisitos: Instale o docker e docker compose

### Clonar Repositório

Faça o clone do repositório:

- git clone  
[https://C4B4N0:ghp\\_H5XxX8Z4lzl9lkvfNc3PisO0p6sE61XIMhb@github.com/C4B4N0/API-REST-IPS-TOR.git](https://C4B4N0:ghp_H5XxX8Z4lzl9lkvfNc3PisO0p6sE61XIMhb@github.com/C4B4N0/API-REST-IPS-TOR.git)

### Como executar

- Adicione o caminho do projeto na variável PYTHONPATH no arquivo .env
- Construa e inicie o container: docker-compose up --build

## 2.3 Capturas de telas

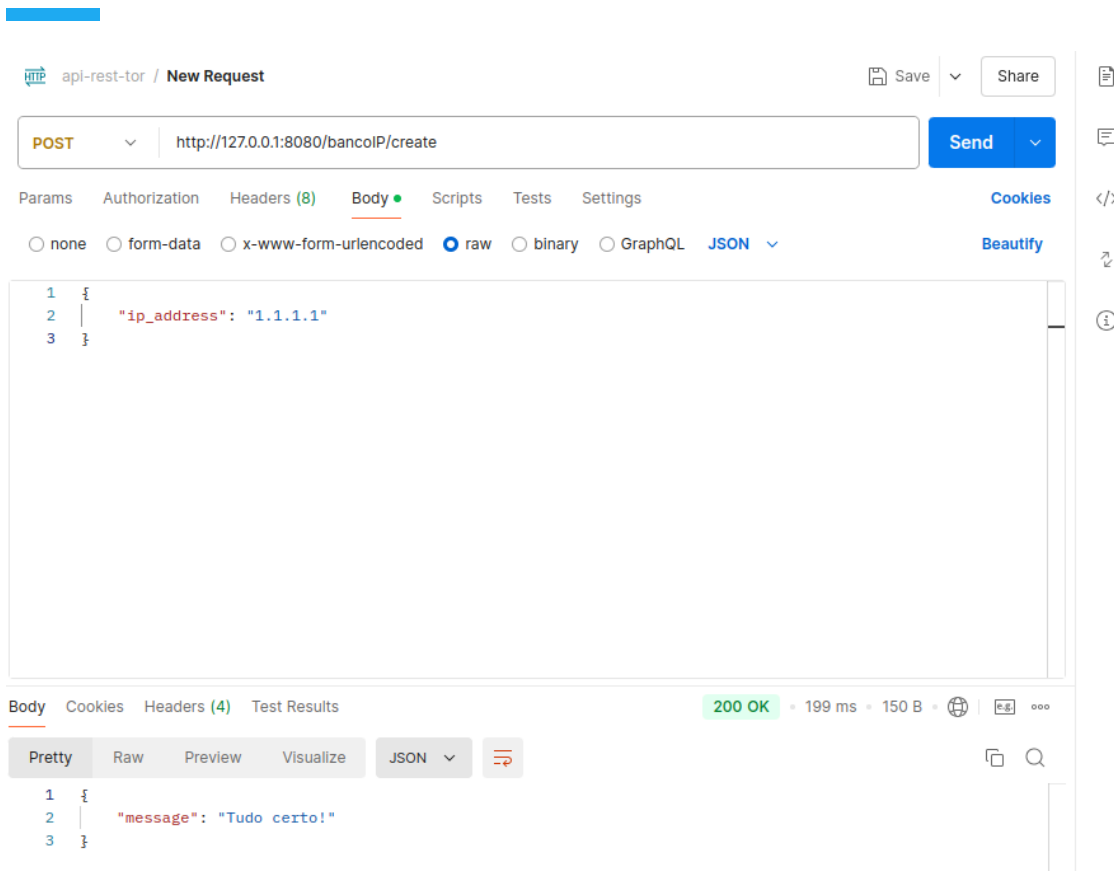
The screenshot displays the 'api-rest-tor' web application. At the top, the URL bar shows 'http://127.0.0.1:8080/tor-ips' with a 'GET' method selected. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Scripts', 'Tests', and 'Settings'. The 'Params' tab is active, showing a table with columns 'Key', 'Value', and 'Description'. Below this, there is a 'Query Params' section with a table that has columns 'Key' and 'Value'. The 'Body' tab is also visible, showing a '200 OK' status, a response time of 1727 ms, and a size of 18.65 KB. The response body is displayed in a 'Pretty' JSON format, showing a list of IP addresses under the key 'tor\_ips'.

```
1 {
2   "tor_ips": [
3     "171.25.193.25",
4     "80.67.167.81",
5     "192.42.116.187",
6     "198.98.51.189",
7     "89.58.26.216",
8     "109.70.100.4",
9     "149.56.22.133",
10    "5.45.102.93",
11    "192.42.116.196",
12    "185.220.101.4",
13    "45.141.215.62",
14    "94.182.51.15",
15    "192.42.116.213",
16    "45.141.215.80",
17    "193.26.115.61",
18    "192.42.116.175",
19    "149.56.44.47",
20    "87.118.116.103",
21    "178.17.171.102",
22    "192.42.116.208",
23    "89.58.41.156",
24    "2.58.56.43",
25    "104.192.1.138",
26    "107.189.8.56"
27  ]
28 }
```

### Endpoint 1 - GET

Endpoint GET que devolve IPs a partir da fonte externa com a seguinte URL:

<https://check.torproject.org/torbulkexitlist>

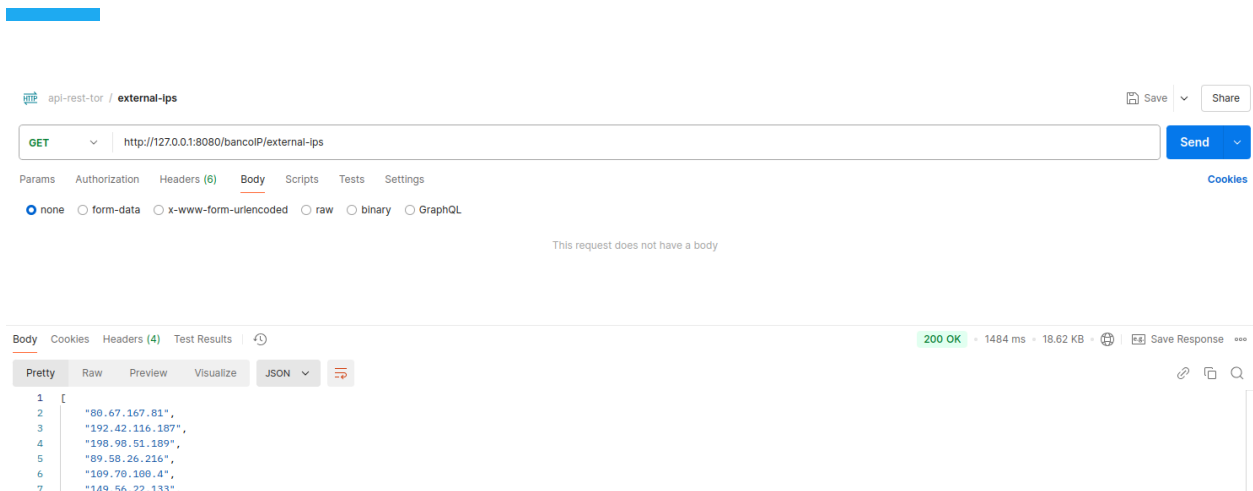


## Endpoint 2 - POST

Endpoint POST que recebe um IP e adiciona este IP na base de dados (postgresql). Nesta base, estão os IPs que não queremos que apareçam no output do endpoint 3.

Observação: Não foi criado um script para popular a base de dados. O IP foi adicionado inserindo um JSON via ferramenta POSTMAN, conforme pode ser observado na imagem.

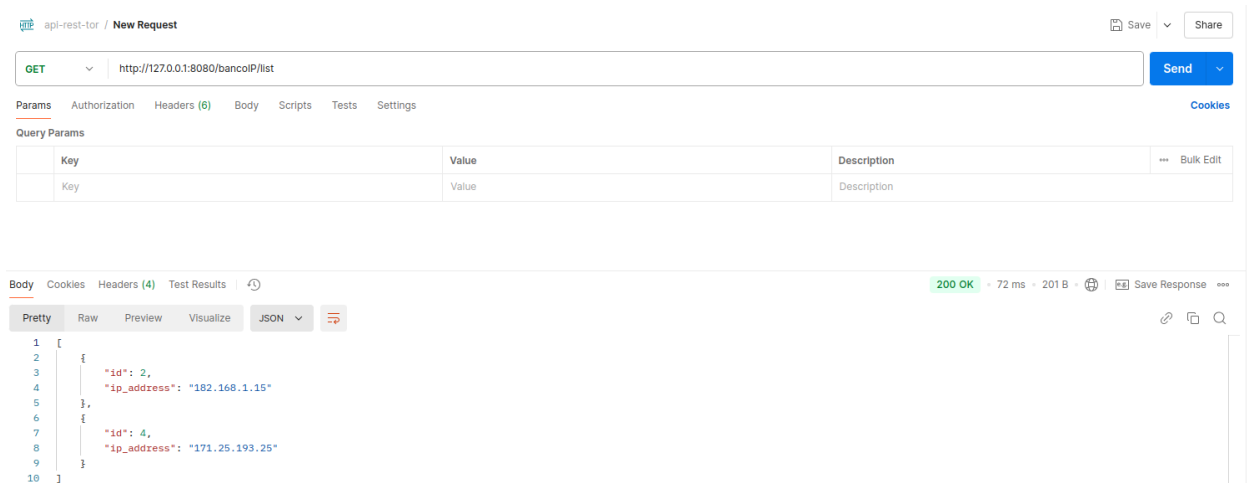




### Endpoint 3 - GET

Endpoint GET que retorna os IPs filtrados, ou seja, retornará a lista de IPs de saída do TOR (fonte externa) e excluindo os IPs adicionados no endpoint 2.

Para testar este Endpoint, foi adicionado pelo endpoint 2 o primeiro IP retornado pelo endpoint 1.



### Endpoint 4 - GET - Endpoint GET que retorna os IPs da base de dados



api-rest-tor / New Request

SaveShare

DELETE

http://127.0.0.1:8080/bancoIP/delete/3

Send

arams

Authorization

Headers (6)

Body

Scripts

Tests

Settings

Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

dy

Cookies

Headers (4)

Test Results

200 OK · 37 ms · 150 B · Save Response

Pretty

Raw

Preview

Visualize

JSON

1 {

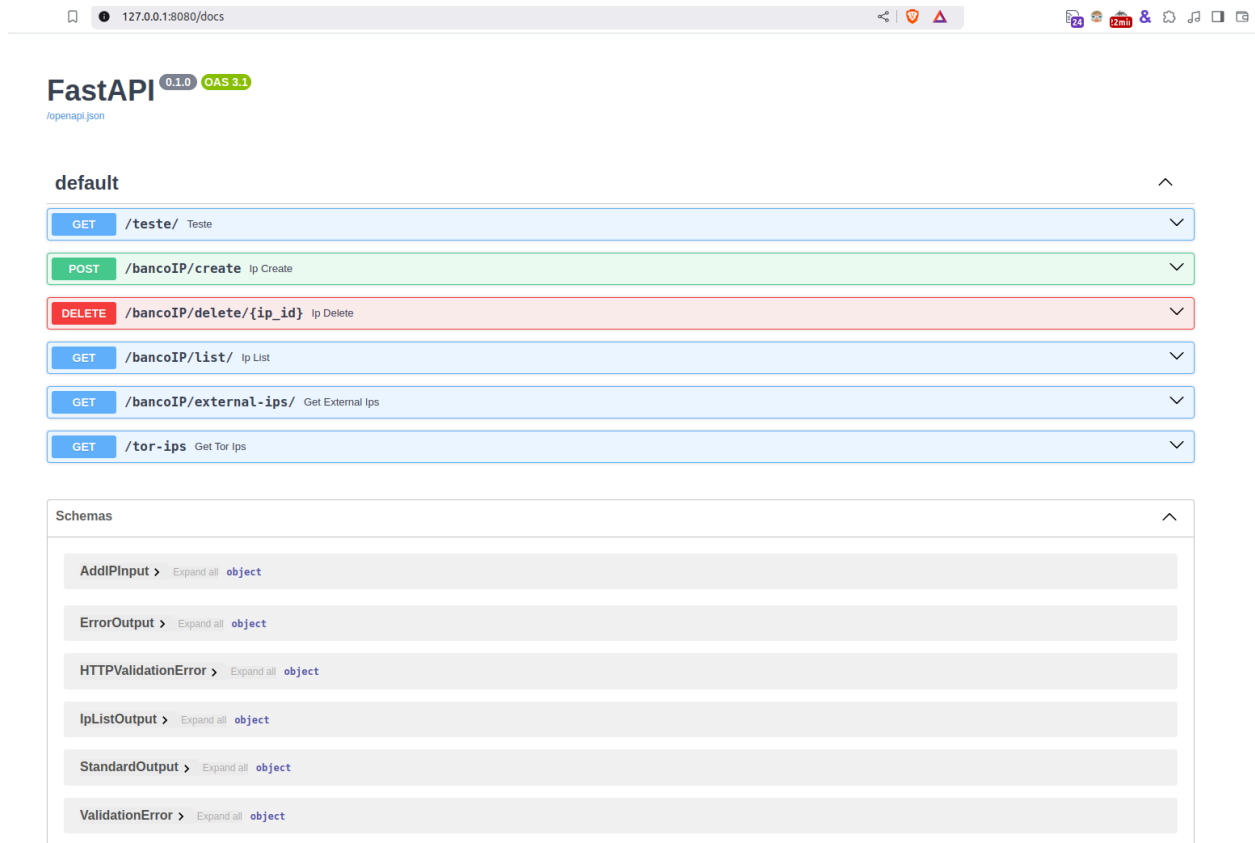
2 | "message": "Tudo certo!"

3 }

Endpoint 5 - DELETE - Endpoint extra que faz a remoção de IP

## 2.4 Acesso à API

- Interface Swagger: <http://127.0.0.1:8080/docs>
- Interface Redoc: <http://127.0.0.1:8080/redoc>



Swagger da aplicação

## 2.5 Deploy em Cloud Público

2.5.1 - Escolha da Cloud Pública: AWS EC2.

2.5.2 - Pré-requisitos:

- Chave SSH
- Ubuntu instalado e atualizado
- Docker e docker compose instalado

2.5.3 - Configuração do Workflow do GitHub Actions

Criar o arquivo `.github/workflows/actions.yml` no repositório com o seguinte conteúdo:

name: Deploy FastAPI Application

on:

push:

branches:

- main

jobs:

deploy:


runs-on: ubuntu-latest

steps:

# Checkout do código

- name: Checkout code

uses: actions/checkout@v3



```
# Configuração do ambiente
```

```
- name: Set up Python
```

```
uses: actions/setup-python@v4
```

```
with:
```

```
python-version: '3.9'
```

```
# Instalação de dependências
```

```
- name: Install dependencies
```

```
run: |
```

```
python -m pip install --upgrade pip
```

```
pip install pipenv
```

```
pipenv install
```

```
# Build da aplicação usando Docker
```

```
- name: Build Docker image
```

```
run: |
```

```
docker build -t fastapi-app .
```

```
# Login no servidor remoto via SSH
```

```
- name: Deploy to EC2
```

```
uses: appleboy/ssh-action@v0.1.10
```

```
with:
```

```
host: ${ secrets.EC2_HOST }}
```

```
username: ubuntu
```

```
key: ${ secrets.EC2_SSH_KEY }}
```

```
port: 22
```

```
script: |
```

```
docker stop fastapi-app || true
```

```
docker rm fastapi-app || true
```

```
docker run -d -p 80:80 --name fastapi-app fastapi-app
```

#### 2.5.4 Configurar Secrets no Github

Adicionar as informações sensíveis no GitHub Action Secrets para segurança:

- EC2\_HOST: Endereço IP do servidor EC2
- EC2\_SSH\_Key: Chave SSH privada usada para acessar o servidor

#### 2.5.5 - Configuração do Servidor EC2

- Instalação no servidor remoto:  

```
sudo apt update
```

```
sudo apt install -y docker.io
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```
- Configurar portas: Verificar se a porta 80 está aberta no Security Group do EC2.

#### 2.5.6 Fluxo do Deploy

- Fazer um commit no branch main
- O GitHub Actions será acionado automaticamente
- A aplicação será empacotada em um container Docker e enviada para o servidor EC2
- O container será iniciado no servidor, expondo a API REST na porta 80

#### 2.5.7 Teste do Deploy

Acessar o endereço público do servidor usando o comando curl.



## 2.6 Dificuldade encontrada

Devido ao fator tempo não foi possível desenvolver a aplicação de forma full assíncrona, bem como os entregáveis extras.