

Servidor

1-Parte:

Pré-requisitos:

- Está criação foi feita em uma Máquina Virtual
- Raspberry Pi com sistema devidamente instalado
- Instalação do Flask, Gunicorn, Nginx

2-Parte:

Nesta configuração irei usar Raspberry OS, modelo 32bits (raspios-bullseye-i386).

Atualize o sistema com os comandos no terminal:

```
sudo apt update
```

Depois de atualizar os repositórios, utilize o comando seguinte para fazer o download e instalação das atualizações:

```
sudo apt upgrade
```

Logo após instale os pacotes que permitirão construir o ambiente Python:

```
sudo apt install python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools
```

3-Parte:

Instalação do Nginx.

Criação do ambiente virtual e instalação do Flask e Gunicorn.

Comando para a instalação do Nginx:

```
sudo apt install nginx
```

Crie o diretório, pois será o diretório onde armazenara todos os arquivos do Flask e Gunicorn, e onde será criado o ambiente virtual e feitas algumas das configurações.

Mas primeiro instale o “python3-venv” pacote que instalará o módulo “venv”:

```
sudo apt install python3-venv
```

Em seguida crie o diretório e entre nele:

```
mkdir ~/origem
```

```
cd ~/origem
```

Crie seu ambiente virtual para armazenar todos os requisitos e funcionalidades do Flask:

```
python3 -m venv origemenv
```

Ative o ambiente virtual e faça a instalação do Flask e Gunicorn:

```
source origemenv/bin/activate
```

Mas antes disso instale o Wheel, que é nada mais que um arquivo compactado onde pode ser adquirido vários pacotes que podem ser necessários para a instalação. Comando para instalar o Wheel dentro do ambiente virtual:

```
pip install wheel
```

E logo depois o Flask e Gunicorn:

```
pip install flask gunicorn
```

4-Parte:

Continuando dentro do ambiente virtual iremos criar um aplicativo onde poderá colocar seu site como caminho:

```
nano ~/origem/origem.py
```

Nessa parte fica a gosto da pessoa de como será o app: mas irei colocar um exemplo bem simples:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "<h1 style='color:red'>Oi, Pessoas!</h1>"
if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

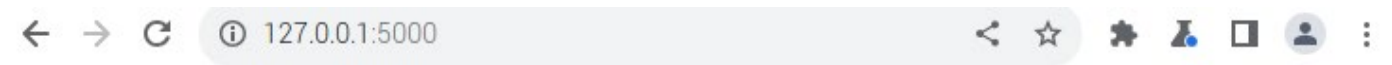
Ctrl + O para salvar o arquivo e Ctrl + X para sair, e isso em todos arquivos que forem criados com o nano

Dentro do ambiente virtual inicie o aplicativo:

```
python origem.py
```

```
(origemenv) origem@raspberrypi:~/origem $ python origem.py
* Serving Flask app 'origem'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
127.0.0.1 - - [05/Apr/2024 19:40:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Apr/2024 19:40:35] "GET /favicon.ico HTTP/1.1" 404 -
```

Digite o IP do seu servidor no seu navegador com a porta :5000:

A screenshot of a web browser's address bar. It shows the URL "127.0.0.1:5000" in the center. To the left of the URL are navigation icons: back, forward, and refresh. To the right are icons for bookmarks, extensions, and a user profile. The browser interface is light gray.

Oi, Pessoas!

5-Parte:

Criando o ponto de entrada WSGI e configurando o Gunicorn.

Crie o arquivo WSGI que servira como ponto de entrada e configure, isso mostrara como o Gunicorn deve agir com o app do Flask:

```
nano ~/origem/wsgi.py
```

```
from origem import app

if __name__ == "__main__":
    app.run()
```

Entre no diretório origem e verifique se o Gunicorn está respondendo corretamente ao app:

```
cd ~/origem
```

```
gunicorn --bind 0.0.0.0:5000 wsgi:app
```

E a resposta deverá ser essa:

```
(origemenv) origem@raspberrypi:~/origem $ gunicorn --bind 0.0.0.0:5000 wsgi:app
[2024-04-05 20:00:21 -0300] [12782] [INFO] Starting gunicorn 21.2.0
[2024-04-05 20:00:21 -0300] [12782] [INFO] Listening at: http://0.0.0.0:5000 (12782)
[2024-04-05 20:00:21 -0300] [12782] [INFO] Using worker: sync
[2024-04-05 20:00:21 -0300] [12783] [INFO] Booting worker with pid: 12783
```

Oi, Pessoas!

Iremos desativar o ambiente virtual com o comando:

```
sudo nano /etc/systemd/system/origem.service
```

O arquivo configura um serviço do Gunicorn para servir uma aplicação chamada "origem". Ele especifica o usuário e o grupo sob os quais o serviço será executado, o diretório de trabalho da aplicação e como iniciar o servidor Gunicorn com três trabalhadores que se refere aos processos e um arquivo de soquete Unix para conexões entre o cliente e o servidor.

[Unit]

Description=Gunicorn instance to serve origem

After=network.target

[Service]

User=origem

Group=www-data

WorkingDirectory=/home/origem/origem

Environment="PATH=/home/origem/origem/origemenv/bin"

ExecStart=/home/origem/origem/origemenv/bin /gunicorn --workers 3 --bind unix:origem.sock -m 007 wsgi:app

[Install] WantedBy=multi-user.target

Agora iremos iniciar o serviço Gunicorn e habilitá-lo para que sempre que ele inicie junto com o servidor, com os comandos:

```
sudo systemctl start origem
```

```
sudo systemctl enable origem
```

E verifique o status:

```
sudo systemctl status origem
```

Você receberá a resposta seguinte:

```
origem@raspberrypi:~/origem $ sudo systemctl status origem
● origem.service - Gunicorn instance to serve origem
   Loaded: loaded (/etc/systemd/system/origem.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2024-04-05 20:24:14 -03; 27s ago
     Main PID: 12958 (gunicorn)
        Tasks: 4 (limit: 4616)
      Memory: 34.4M
         CPU: 257ms
       CGroup: /system.slice/origem.service
               └─12958 /home/origem/origem/origemenv/bin/python3 /home/origem/origem/origemen>
                 └─12959 /home/origem/origem/origemenv/bin/python3 /home/origem/origem/origemen>
                   └─12960 /home/origem/origem/origemenv/bin/python3 /home/origem/origem/origemen>
                     └─12961 /home/origem/origem/origemenv/bin/python3 /home/origem/origem/origemen>

abr 05 20:24:14 raspberrypi systemd[1]: Started Gunicorn instance to serve origem.
abr 05 20:24:14 raspberrypi gunicorn[12958]: [2024-04-05 20:24:14 -0300] [12958] [INFO] Start>
abr 05 20:24:14 raspberrypi gunicorn[12958]: [2024-04-05 20:24:14 -0300] [12958] [INFO] Liste>
abr 05 20:24:14 raspberrypi gunicorn[12958]: [2024-04-05 20:24:14 -0300] [12958] [INFO] Using>
abr 05 20:24:14 raspberrypi gunicorn[12959]: [2024-04-05 20:24:14 -0300] [12959] [INFO] Booti>
abr 05 20:24:14 raspberrypi gunicorn[12960]: [2024-04-05 20:24:14 -0300] [12960] [INFO] Booti>
abr 05 20:24:14 raspberrypi gunicorn[12961]: [2024-04-05 20:24:14 -0300] [12961] [INFO] Booti>
lines 1-20/20 (END)
```

6-Parte:

Configuração do Nginx, para passar solicitações web para o arquivo do soquete.

Criando um arquivo de bloco de servidor no diretório de sites-available que está dentro do diretório do Nginx:

```
sudo nano /etc/nginx/sites-available/origem
```

Esse bloco de configuração Nginx, direciona todas as solicitações recebidas na porta 80 com o nome de host www.origem.local, para um servidor Gunicorn em execução no arquivo de soquete Unix `"/home/origem/origem/origem.sock"`. Ele atua como um intermediário, encaminhando solicitações HTTP, para o servidor Gunicorn.

```
server {
    listen 80; server_name www.origem.local;
    location / {
        include proxy_params;
        proxy_pass http://unix:/home/origem/origem/origem.sock;
    }
}
```

Para habilitar a configuração de bloco do servidor Nginx que acabou de criar, vincule o arquivo ao sites-enabled diretório com o comando:

```
sudo ln -s /etc/nginx/sites-available/origem /etc/nginx/sites-enabled
```

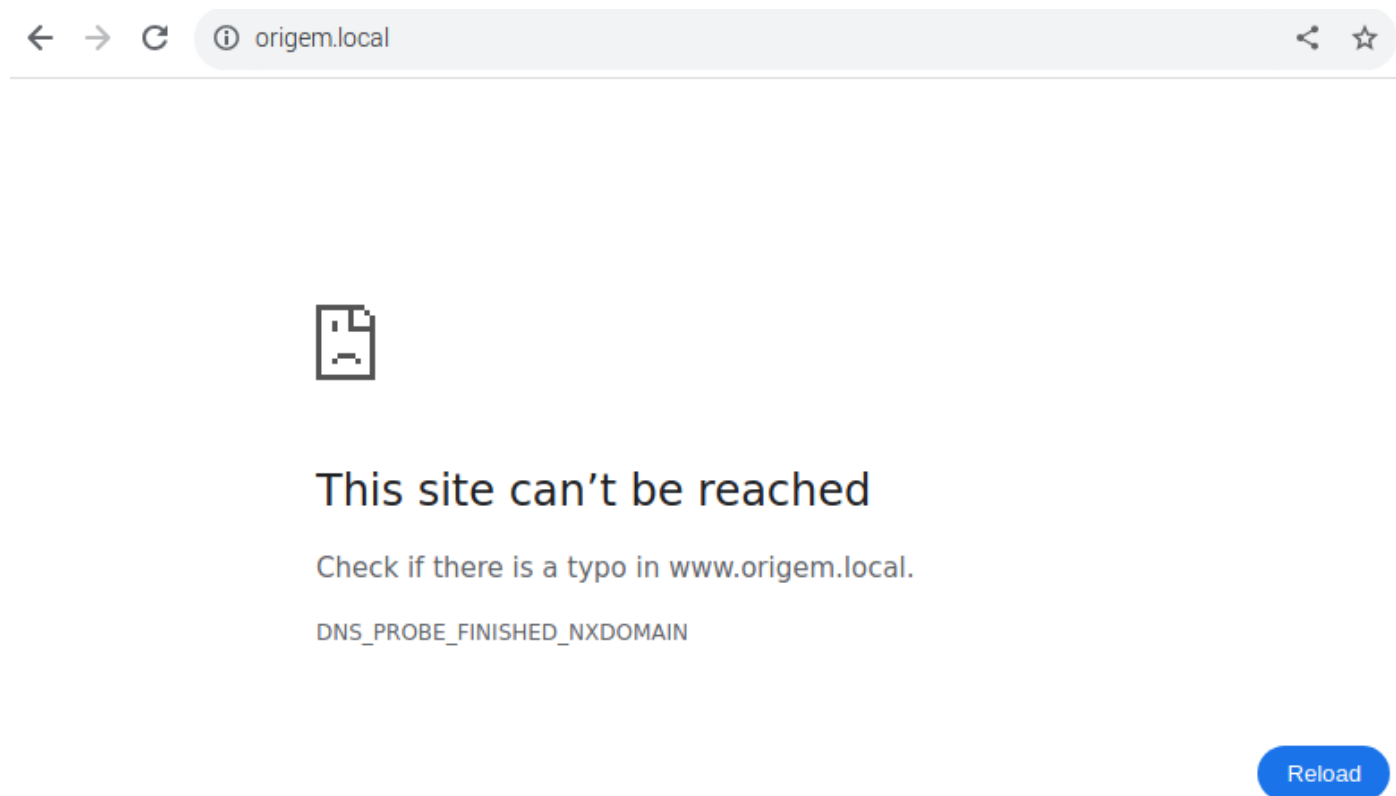
Antes de reiniciar o processo do Nginx e carregar a nova configuração de o seguinte comando para verificar se não a nenhum erro na sintaxe:

```
sudo nginx -t
```

E logo depois:

```
sudo systemctl restart nginx
```

Mas para aqueles que não tem um domínio próprio vai aparecer o seguinte



Mas podemos integrar o site no domínio local para a visualização:

```
sudo nano etc/hosts
```

```
GNU nano 5.4 /etc/hosts *
127.0.0.1 www.origem.local localhost
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.1.1 raspberrypi
```

E desta forma o site ira ser demonstrado na rede do próprio servidor. E claro é uma solução de aprendizagem onde depois se pode alocar em um domínio e disponibilizar seu site do app.

Oi, Pessoas!

Mensagem para criações futuras:

Observamos a criação de um servidor com Flask, Gunicorn e Nginx, Onde todas as configurações e criações poderão demonstrar o funcionamento entre um servidor e o usuário final, além da configuração do proxy reverso onde tem várias ferramentas de funcionamento do servidor. Mas ainda mais configurações de segurança podem ser feitas e isso é só a ponta do iceberg. O servidor é uma aplicação para computadores ou minicomputadores com um porte de processamento menor.