

Seminário:

Introdução ao desenvolvimento back-end com Python e Flask

Walter Magri Temporal

Conteúdo:

- Como salvar, compartilhar e controlar versões: GIT – (GitHub, GitLab, local...);
- ambiente virtual: virtualenv;
- Pip e bibliotecas
- Flask: estrutura recomendada de diretórios;
- Jinja2: (HTML, Bootstrap, css);
- Simular requisições via Postman;
- MySQL .

DICAS DE FLASK, LINUX E PYTHON:

- Formatos de arquivos para envio e retorno;
- Threads;
- Linhas de comandos úteis de Linux;
- Api-rest: micro-serviços;
- IOT: Raspberry, Arduino, Esp8266;
- Bibliotecas legais/úteis Python.

PRÁTICA:

- Upload e download de arquivos para o servidor;
- Carregamento de páginas HTML e uso de formulários;
- Route para uma aplicação usando a biblioteca beautifulsoup para API de web scraping;
- Acessar o banco de dados MySQL via código;
- Projeto final livre (não obrigatório) - individual ou em dupla, sob orientação do professor.

- Opções do tipo de aplicação para projeto:
 - 1: Aplicação com CRUD;
 - 2: IA (OCR, detecção de faces - HAAR);
 - 3: Processamento de imagem;
 - 4: IoT
 - 5: ???

- Mais avançado... :
- Acessar o banco de dados sqlite3 via módulo (SQLAlchemy);
- Swagger;
- Segurança;
- Melhorias para desempenho;
- Documentação (pdoc, pydoc, etc);
- Controle de sessão;
- Blueprints.

VM – Ubuntu-Flask: Configurações úteis

Com a VM desligada:

- Alterar quantidade de RAM e núcleos ativos de acordo com a máquina hospedeira: Configurações => sistema (verificar nas abas “placa mãe”, e “processadores”);
- Rede no modo “Bridge”: Configurações => Rede (mudar o item “conectado a: placa em modo bridge”);
- Habilitar transferência bi-direcional e habilitar “arrastar e soltar” bidirecional: Configurações => avançado;
- Habilitar imagem do disco “Adicionais para convidados” e executar; (se necessário: Ctrl direito + C, clicar em “Dispositivos”, selecionar a última opção, dois clicks na imagem do cd que aparecer no desktop e clicar no botão no canto superior direito para “executar software”);
- Senha: 123456

AULA 01: FLASK

- Micro-framework – aplicações web, micro-serviços, APIs;
- Facilidade de uso e extremamente enxuto;
- Extensibilidade: permite usar as ferramentas que quiser;
- Werkzeug e Jinja2 – comunicação do “front” com o “back”;
 - (toolkit para WSGI, a interface padrão entre aplicações web Python e servidores HTTP para desenvolvimento e implantação. Jinja2 renderiza templates).

Antes de continuarmos, lembrando comandos úteis do Linux também...

- `sudo apt update`
- `sudo apt upgrade`
- `mkdir nome_da_pasta`
- `chmod 775 nome_da_pasta`
- `cd nome_da_pasta`
- `cd ..`
- `rm -rf nome_da_pasta`
- `gedit nome_do_arquivo`
- `nano nome_do_arquivo`
- `tree`

(OBS: pode ser necessário usar comando “sudo” na frente de alguns dos comandos acima)

Antes de continuarmos, lembrando comandos úteis do Linux também...

- ifconfig
- nproc
- pwd
- ls
- Ctrl + Alt + T
- Ctrl + Shift + T
- Ctrl + L
- Ctrl + Shift + C
- Ctrl + Shift + V

(OBS: pode ser necessário usar comando “sudo” na frente de alguns dos comandos acima)

... e breve revisão de Python!

Google, YouTube, BitTorrent, NASA - Jet Propulsion Lab, Eve Online, National Weather Service.

Tipos importantes e formas de acesso ao dado:

- Strings!!!
- Int, float, decimal, dict, json, set, list...
- OBS: testar no terminal: “\$ python --version; python”
- OBS2: se aparecer python2.7, ou que o python não está instalado, mudar o alias:
\$ sudo alias python=python3

Lists:

```
>>> L = []
```

```
>>> L.append(1) # Coloca na pilha
```

```
>>> L.append(2)
```

```
>>> L
```

```
[1, 2]
```

```
>>> L.pop() # Tira da pilha
```

```
2
```

```
>>> L
```

```
[1]
```

Json (tendência cada vez maior de uso):

```
>>> import json
```

```
>>> rec = {"job": ["dev", "mgr"], "name": {"last": "Smith", "first":  
      "Bob"}, "age": 40.5}
```

```
>>> rec['job']  
['dev', 'mgr']
```

```
>>> rec['name']  
{ 'last': 'Smith', 'first': 'Bob' }
```

```
>>> if rec['name']['last'] == 'Smith':  
...   print(rec['age'])
```

```
...  
40.5
```

Acessando os valores:

- Listas:

```
>>>minha_lista = [1,2,3,4,5]
```

```
>>>minha_lista[0]
```

```
1
```

- Json:

```
>>> {'cliente': {'nome': 'walter', 'sexo': 'M'}}
```

```
>>> dados['cliente']['nome']
```

```
'walter'
```

```
>>> dados['cliente']['sexo']
```

```
'M'
```

- Iterações:

```
>>> L = [1, 2, 3]
>>> for X in L:
>>>     print(X ** 2, end = ' ')
```

- Condições:

```
if:
    pass
elif:
    pass
else:
    pass
```

- Expressões regulares:

```
>>>import re
```

```
>>>text = re.sub("\n", " ", text)
```

```
>>> text = "eu quero comer salada"
```

```
>>> text = re.sub("salada", "churrasco", text)
```

```
>>> text
```

```
'eu quero comer churrasco'
```

```
x = re.search(r"[A-Z]{3}[0-9]{4}", placa)
```

```
inicio, fim = x.span()
```

```
placa = placa[inicio:fim]
```


AMBIENTE VIRTUAL

- Dependências de bibliotecas dentro do projeto;
- Incompatibilidade de versões de packages entre projetos
- Virtualenv - Virtualwrapper

Instalação do virtualenv (já instalado):

- **sudo apt install python3-pip**
- **sudo pip install virtualenv virtualenvwrapper**
- **sudo rm -rf ~/get-pip.py ~/.cache/pip**
- **echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.bashrc**
- **echo "export WORKON_HOME=\$HOME/.virtualenvs" >> ~/.bashrc**
- **echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.bashrc**
- **echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc**
- **source ~/.bashrc**

- *#cria ambiente com nome de nomedoambiente*
mkvirtualenv nomedoambiente -p python3
- *#lista todos os ambientes criados:*
lsvirtualenv -b
- *#remove um ambiente*
rmvirtualenv nomedoambiente
- *#verifica e habilita o ambiente*
workon nomedoambiente (No nosso caso: **workon ambiente_flask**)
- *#instala numpy no ambiente*
pip install numpy
- *#sai do ambiente*
deactivate

PIP – Python Package Index

- Ferramenta para instalação automática de bibliotecas e dependências para Python.

Comando úteis PIP:

- `pip3 install nome_do_package`
- `pip3 uninstall nome_do_package`
- `python -m pip3 freeze > requirements.txt`
- `pip3 install -r requirements.txt`
- Na dúvida consultar:
<https://pypi.org/>

Bibliotecas customizadas:

- Devem ser incluídas no \$PYTHONPATH do sistema;

import nome_modulo_customizado

- Podem ser usadas localmente, como explicado mais a frente.

GIT

- Controle de versões;
- Trabalho em equipe;
- Ideal: master e dev (ou mais...);
- Criar conta em:
<https://github.com/>

Join GitHub

Create your account

Verify your account

Please solve this puzzle so we
know you are a real person

Verify



Email preferences

☐ Send me occasional product updates, announcements, and offers.

Next: Select a plan

Choose a plan

Individual

Pick the plan that's right for you, personally.

Team

Choose a plan to help your team grow and collaborate.



Free

\$0 USD

The basics of GitHub for every



Pro

\$7 USD

Per month

Pro tools for developers with advanced

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

<p>Software Engineer</p> <p>I write code</p>	<p>Student</p> <p>I go to school</p>
<p>Product Manager</p> <p>I write specs</p>	<p>UX & Design</p> <p>I draw interfaces</p>
<p>Data & Analytics</p> <p>I write queries</p>	<p>Marketing & Sales</p> <p>I look at charts</p>
<p>Teacher</p> <p>I educate people</p>	<p>Other</p> <p>I do my own thing</p>



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.

An email containing verification instructions was sent to **seu email aqui**

[Resend verification email](#)[Change your email settings](#)

GitHub

Subscribe to our newsletter

Get product updates, company news, and more.

[Subscribe](#)

Product

[Features](#)[Security](#)[Enterprise](#)[Customer stories](#)[Pricing](#)[Resources](#)

Platform

[Developer API](#)[Partners](#)[Atom](#)[Electron](#)[GitHub Desktop](#)

Support

[Help](#)[Community Forum](#)[Professional Services](#)[Learning Lab](#)[Status](#)[Contact GitHub](#)

Company

[About](#)[Blog](#)[Careers](#)[Press](#)[Social Impact](#)[Shop](#)

Vão receber um email de verificação do Github... Clicar e preencher os campos



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner

Repository name *

 waltinho-temporal / primeiro_rep 





Great repository names are short and memorable. Need inspiration? How about [didactic-octo-broccoli](#)?

Description (optional)

meu primeiro repositorio



- ☒  **Public**
Anyone can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.



Skip this step if you're importing an existing repository.

- ☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.



Add .gitignore: **None** | Add a license: **None** 

Deixar público é opcional... Mas tem algumas vantagens....

waltinho-temporal / primeiro_rep

Unwatch

1

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security

Insights

Settings

meu primeiro repositório

Edit

Manage topics

1 commit

1 branch

0 packages

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

waltinho-temporal Initial commit

Latest commit 305b4ea now

README.md

Initial commit

now

README.md

primeiro_rep

meu primeiro repositório

Comandos úteis do Git:

- `$ git clone https://endereço_url_do_projeto_no_git`
- `$ git add .`
- `$ git commit -m "atualização do dia 11/04/2020 – criação da aplicação do Flask"`

Boa prática: explicar sobre o projeto no arquivo README

- `$ git config --global user.email "you@example.com"`
- `$ git config --global user.name "Your Name"`

Comandos básicos e úteis GIT:

- `git checkout -b dev`
- `git branch`
 - alterar o arquivo README;
- `git push --set-upstream origin dev`
- `git checkout master`
 - verificar o arquivo README;
- `git checkout dev`

Outro comando interessante:

- `git pull`

Quando ocorre alterações nas branches, aparece dessa forma no Github:

Helpful resources
[GitHub Community Guidelines](#)

2 commits 1 file changed 0 commit comments 1 contributor

Commits on Apr 04, 2020

- waltinho-temporal att2 ef268b1
- waltinho-temporal att2 0d214c1

Showing 1 changed file with 4 additions and 1 deletion.

Unified Split

5 README.md

```
... @@ -1,2 +1,5 @@
1 +
2 +
3 +
1 4 # primeiro_rep
2 - meu primeiro repositório de teste
5 + meu primeiro repositório de teste - teste do ramo dev
```

No commit comments for this range

Próximos passos:

- Configurar o git pra que não peça login e senha toda hora;
- Criar chave de autenticação;
- Comando para modificar pastas e arquivos únicos dentro do projeto;
- Pull e Merge entre branches como administradores.

Primeira aplicação do git:

- Ir para o diretório “Documentos”
- Digitar o comando:
\$ git clone https://github.com/walter-temporal/curso_flask_puc.git
- Digitar o comando:
\$ cd AULA01
- Executar aplicação com:
\$ python app.py
- Visualizar em: <http://localhost:5000>
- Dica: Se você não quiser abandonar o terminal ao abrir o navegador para testar a URL:
\$ curl http://127.0.0.1:5000/

Primeira aplicação solo:

- Navegar ate a pasta que baixou do git;
- Ou criar os seguintes diretórios com o comando mkdir:

/home/flaskman/Documentos/curso_flask_puc/ATIVIDADE01

- Dentro do diretorio criar o arquivo: **app.py**

```
from flask import Flask
```

```
app = Flask(__name__)
```

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

```
def hello():
```

```
    return "Hello World!"
```

```
if __name__ == "__main__":
```

```
    app.run()
```

Erros e respostas mais comuns da request:

- 200: sucesso
- 500: server error
- 404: not found

Mensagens de retorno para o cliente devem ser criadas.

- Numero de funções cresce com o aumento de número de rotas
- O código começa a ficar extenso e mais difícil de ser entendido (templates e scripts também vão se acumulando no subfolder...)
- **PEP** - “Python Enhancement Proposal.”
<https://legacy.python.org/dev/peps/pep-0008/>
- Docstrings – documentação
<https://legacy.python.org/dev/peps/pep-0257/>

- Espaçamento: 4 espaços para os blocos (uso de tab pode variar)
- Docstrings: importante para documentação de funções.

ESTRUTURA USADA/RECOMENDADA

application

├─application

| ├─__init__.py

| ├─models.py

| ├─static

| | ├─app.js

| | └─styles.css

| ├─templates



| | ├─index.html

| | └─layout.html

| └─views.py



└─run.py



config.py

Uma boa prática é adotar um arquivo de configuração para o sistema. Exemplo:

- `DEBUG = False`
- `BASE_DIR = '/home/walter/Documentos/flaskapp/'`

Se precisarmos alterar algo no projeto podemos modificar somente pela configuração e não em vários pontos do código.

Próximos passos:

- Segurança de arquivos;
- Armazenamento de senhas e criptografia;
- Autenticação de usuários;
- Documentação (Swagger);
- Sistema mandar email para o responsável/ desenvolvedor caso ocorra um erro;
- Log de erros e eventos.

AULA 02:

- Jinja 2
- try/except

```
>>> try:  
    raise IndexError # ativando manualmente  
except IndexError:  
    print('got exception')
```

HTML

```
<strong>{% if msg %}  
Warning: {{ msg }}  
{% endif %}</strong>
```

```
<div>  
<h2>Meus arquivos:</h2>  
{% if onlyfiles %}  
    {% for item in onlyfiles %}  
        <p> id: {{ loop.index }} - file: {{ item }} <span class="glyphicon  
        glyphicon-ok-sign" style="color:rgb(15, 175, 15)"></span></p>  
    {% endfor %}  
{% endif %}  
</div>
```

CSS E BOOTSTRAP

- Diretório: static

<head>

```
<link href="../static/bootstrap.min.css"
      rel="stylesheet">-->
```

```
<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width,
      initial-scale=1">
```

...

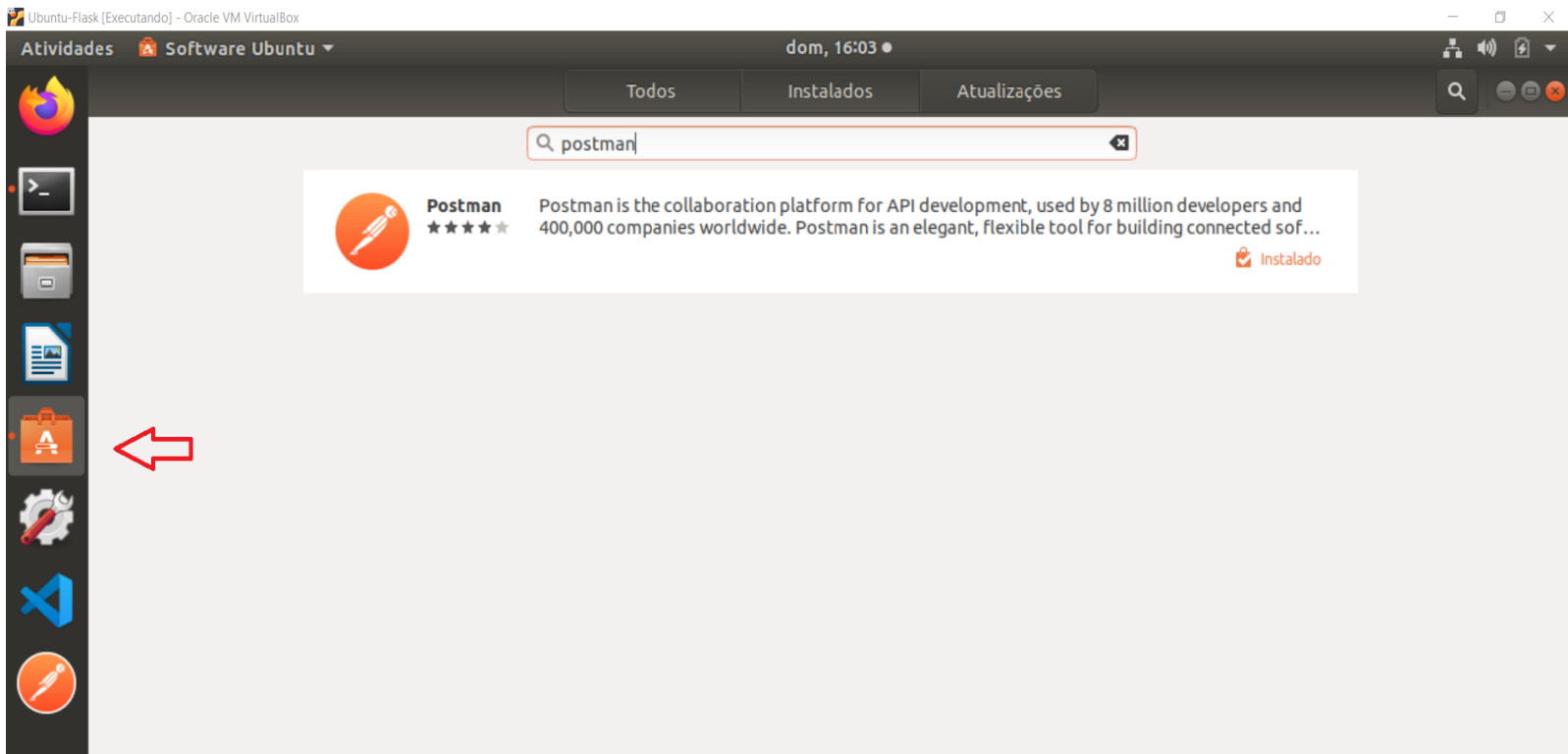
...

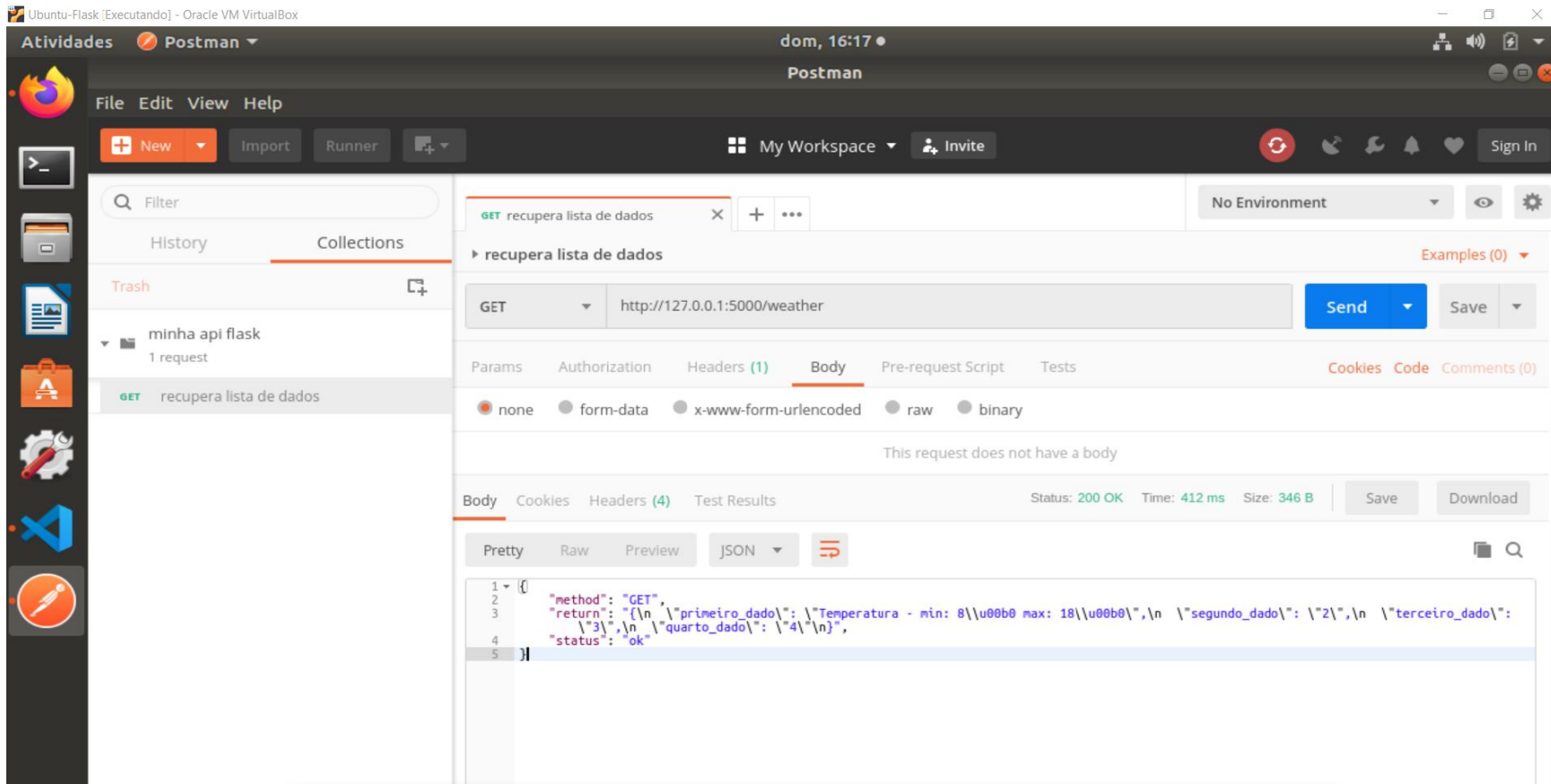
AULA 03:

- Projeto upload/download arquivos
- Projeto Clima Tempo e como lidar com chegada de json
- Postman

Métodos de requests:

- GET, POST, POST JSON...
- POSTMAN





AULA 04:

- Banco de dados: mysql - Criar a tabela com:

```
CREATE TABLE usuarios(  
  pk_user INT NOT NULL AUTO_INCREMENT,  
  login VARCHAR(50) NOT NULL,  
  senha VARCHAR(50) NOT NULL,  
  nome VARCHAR(30) NOT NULL,  
  sobrenome VARCHAR(30) NOT NULL,  
  email VARCHAR(50) NOT NULL,  
  tel01 VARCHAR(50) NOT NULL,  
  rg VARCHAR(20) NOT NULL,  
  data_cadastro VARCHAR(20) NOT NULL,  
  acesso_liberado int NOT NULL,  
  status_conta" int NOT NULL,  
  PRIMARY KEY (pk_user)  
);
```


- Criar o primeiro registro no phpmyadmin:

```
INSERT into usuarios( login, senha, nome, sobrenome, email, tel01,  
    rg, data_cadastro, acesso_liberado, status_conta )  
VALUES (  
    'flaskman',  
    '1234',  
    "  
    ",  
    "  
    ",  
    "  
    ",  
    "  
    ",  
    "  
    ",  
    "  
    ",  
    '1',  
    '1');
```

AULA 05:

- Configuração ambiente de produção:
nginx, gunicorn e supervisor

Nginx: proxy reverso e oculta portas usadas;

Gunicorn: ativa threads e processos;

Supervisor: mantém o sistema online automaticamente caso ocorra falha.

Alternativas: Apache e systemd

#Acessar o ambiente virtual e digitar os comandos:

```
$ sudo pip install gunicorn
```

```
$ which gunicorn
```

```
$ sudo apt install nginx
```

```
$ sudo systemctl enable nginx
```

```
$ /etc/init.d/nginx start
```

```
$ sudo rm /etc/nginx/sites-enabled/default
```

```
$ sudo touch /etc/nginx/sites-available/flask_config
```

```
$ sudo ln -s /etc/nginx/sites-available/flask_config /etc/nginx/sites-enabled/flask_config
```

```
$ sudo nano /etc/nginx/sites-enabled/flask_config
```

o arquivo tem que conter os seguintes parametros:

```
server {  
    listen 80;  
    location / {  
        proxy_pass http://127.0.0.1:5000;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
    }  
}
```

OBS: Pode ser necessário ter que reiniciar o terminal. Feche o terminal e abra novamente.

#aumentar limite de upload do usuário para 5 MB
em nginx.conf, na parte de http, colocar essa linha:
client_max_body_size 5M;

/etc/init.d/nginx restart

se necessario

\$ ufw disable

\$ sudo systemctl status nginx

\$ sudo systemctl stop nginx

\$ sudo systemctl restart nginx

verificar se o arquivo de configuracao esta ok

\$ sudo nginx -t -c /etc/nginx/nginx.conf

#para rodar a aplicação: primeiro item é o modulo e o segundo item é o nome da variável

gunicorn --bind 0.0.0.0:5000 app:app --preload --timeout 0

configurando o supervisor

\$ sudo apt install supervisor

\$ sudo nano /etc/supervisor/conf.d/flaskapp.conf

OBS: ver conteúdo do arquivo abaixo

\$ sudo mkdir -p /var/log/flaskapp/

\$ sudo touch /var/log/flaskapp/flaskapp.err.log

\$ sudo touch /var/log/flaskapp/flaskapp.out.log

\$ sudo supervisorctl reload

OBS: o arquivo tem que conter os seguintes parametros:

[program:flaskapp]

directory=/home/flaskman/Documentos/curso_flask_puc/AULA03/

**command=/home/flaskman/.virtualenvs/ambiente_flask/bin/gunicorn --bind 0.0.0.0:5000 app:app --preload
--timeout 0**

user=root

autostart=true

autorestart=true

stopasgroup=true

killasgroup=true

stderr_logfile=/var/log/flaskapp/flaskapp.err.log

stdout_logfile=/var/log/flaskapp/flaskapp.out.log

#executar:

\$ sudo systemctl start supervisor

\$ sudo supervisorctl reload

\$ sudo systemctl start nginx

REFERÊNCIAS:

- <https://flask.palletsprojects.com/en/1.1.x/>
- <https://jinja.palletsprojects.com/en/2.11.x/>
- <https://werkzeug.palletsprojects.com/en/1.0.x/>
- Páginas do github
- Stackoverflow!!
- Indianos abençoados no YouTube!