

DOCUMENTAÇÃO - TESTE EDESOFT

Tecnologias utilizadas

Python 3.11.0	
Flask	
Flask-SQLAlchemy	
Pandas	
Regex	
DateTime	
SQLite	

Conceito

A aplicação suporta requisições HTTP dos tipos GET e POST, onde ambas terão o mesmo resultado, ou seja, o acesso ao arquivo .csv no bucket AWS, o tratamento de seus dados e inserção no Banco de Dados.

Função lambda

<code>lambda_handler(bucket_name, object_key)</code>	
--	--

– Parâmetros:

bucket_name: o nome do bucket

ex: 'teste_vaga_edesoft'

object_key: o nome do arquivo

ex: 'arquivo_exemplo.csv'

Deste modo o arquivo no bucket, supondo que este seja público, será acessado através da url de acesso padrão formatada da seguinte maneira:

url: 'https://<bucket_name>.s3.amazonaws.com/<object_key>'

url exemplo: https://teste_vaga_edesoft.s3.amazonaws.com/arquivo_exemplo.csv

Retorno da função:

Caso todos os passos da função sejam executados de forma correta o retorno será o seguinte:

http = 200

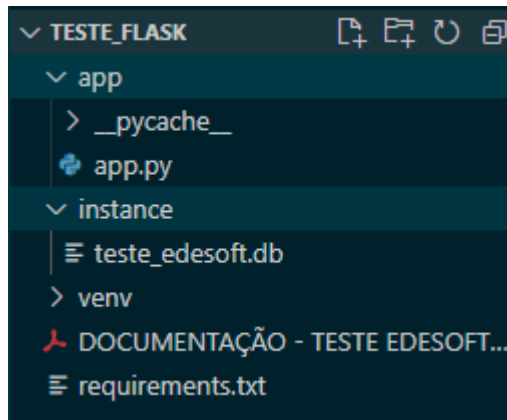
String contendo a seguinte informação: 'Sucesso - Banco de Dados atualizado'.

Execução

Para executar é muito simples, basta executar o arquivo app.py (está na pasta raiz do projeto), inclusive no próprio debug do editor de código:

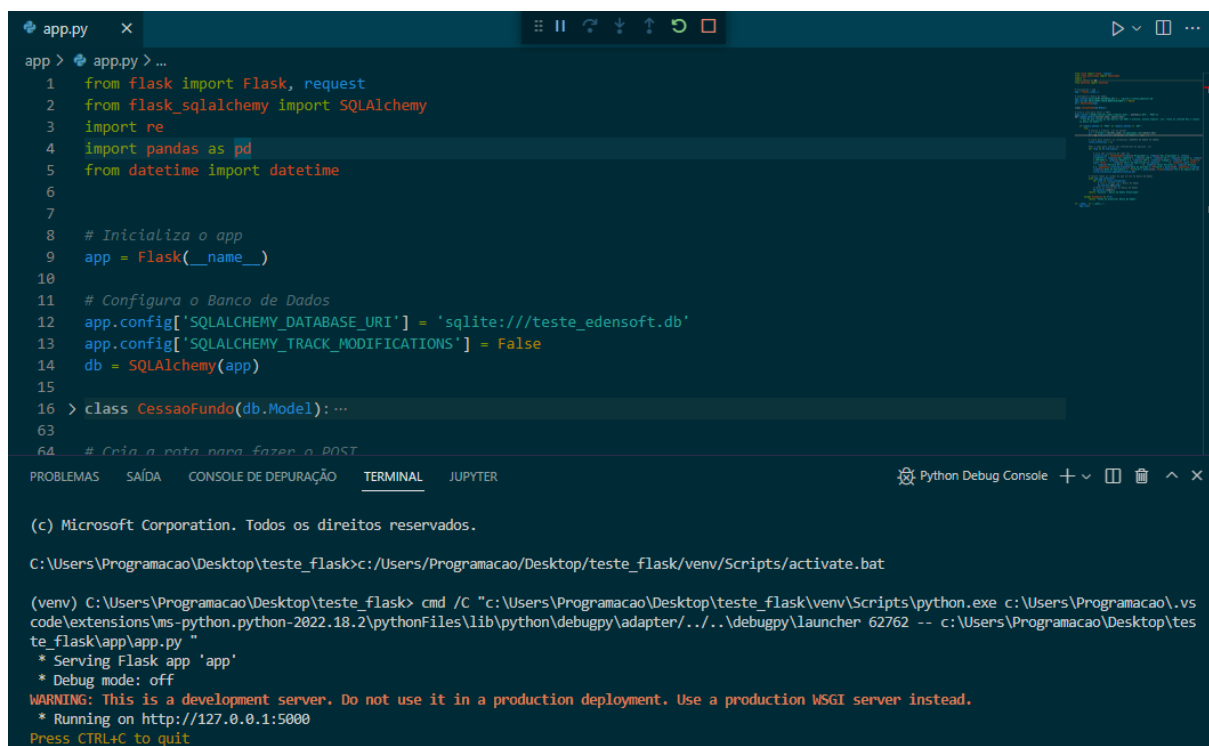
Antes de executar leia as informações sobre o ambiente virtual que está mais adiante nesta documentação.

Para entender melhor o código, a formatação da pasta raiz abaixo é essencial:



Considere que o nome da pasta raiz do projeto é `'teste_flask'`, apenas uma referência ao microframework utilizado.

Dando início à execução da aplicação, através do debug:



```
app.py
app > app.py > ...
1  from flask import Flask, request
2  from flask_sqlalchemy import SQLAlchemy
3  import re
4  import pandas as pd
5  from datetime import datetime
6
7
8  # Inicializa o app
9  app = Flask(__name__)
10
11 # Configura o Banco de Dados
12 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///teste_edesoft.db'
13 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
14 db = SQLAlchemy(app)
15
16 > class CessaoFundo(db.Model): ...
63
64 # cria a rota para fazer o POST

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  JUPYTER
Python Debug Console

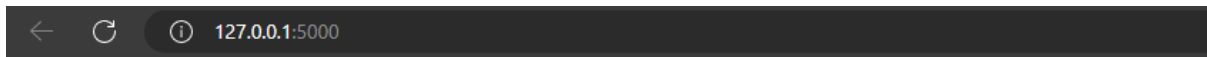
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Programacao\Desktop\teste_flask>c:\Users\Programacao\Desktop\teste_flask\venv\Scripts\activate.bat

(venv) C:\Users\Programacao\Desktop\teste_flask> cmd /C "c:\Users\Programacao\Desktop\teste_flask\venv\Scripts\python.exe c:\Users\Programacao\code\extensions\ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher 62762 -- c:\Users\Programacao\Desktop\teste_flask\app\app.py"
* Serving Flask app 'app'
* 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 http://127.0.0.1:5000
Press CTRL+C to quit
```

No caso do exemplo o endereço localhost foi: <http://127.0.0.1:5000>.

Se este endereço for clicado/acessar (CTRL+CLICK) resultará no seguinte erro, mas não se preocupe, ainda falta o complemento do endpoint:



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

O próprio terminal ressalta o erro na consulta:

```
* Serving Flask app 'app'
* 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 http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [07/Dec/2022 10:50:30] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [07/Dec/2022 10:50:30] "GET /favicon.ico HTTP/1.1" 404 -
```

Para executar corretamente a função basta completar o endpoint (url) no navegador, como por exemplo:

<seu endereço localhost/lambda/<nome_do_bucket>/<object_key>

Para testar a aplicação foi utilizado o seguinte endpoint:

127.0.0.1:5000/lambda/programacaosoftcurriculo1/arquivo_exemplo.csv

127.0.0.1:5000 - representa meu endereço localhost

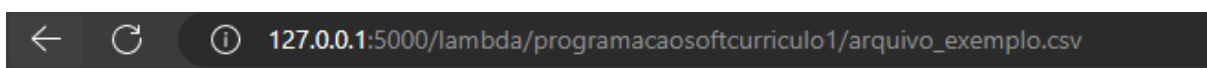
/lambda/ - representa uma parte do endereço

/programacaosoftcurriculo1/ - representa o nome do bucket (bucket_name)

/arquivo_exemplo.csv/ - representa o nome do arquivo no bucket (object_key)

Nenhum dos parâmetros acima é opcional, são todos obrigatórios.

Veja que se executar corretamente, em poucos segundos a própria tela retorna uma mensagem de sucesso.



Sucesso - Banco de Dados Atualizado

Importante ressaltar que é possível executar a aplicação sem utilizar a tela, bastando executar em linha de comando ou através do próprio postman ou outra ferramenta/plataforma similar.

Passos da função

1º —

Acessar o arquivo no bucket através da url:

Para esta etapa, a fim de não realizar o download do arquivo.csv, o acesso é feito através da função `pd.read_csv()` da biblioteca Pandas.

2º —

Fazer o tratamento dos dados:

Nesta etapa também é utilizada a biblioteca Pandas, para acessar os itens e tratá-los dentro de um DataFrame, o que traz melhor organização e controle dos dados.

3º —

Salvar as informações no Banco de Dados:

Esta etapa é a última, onde os dados são salvos no Banco de Dados, já criado na pasta da aplicação, cujo nome é `teste_edesoft.db`.

Observações:

Ambiente virtual: A aplicação utiliza ambiente virtual, o qual por sua vez acompanha o projeto na sua pasta raiz, porém, é possível que o mesmo não funcione diretamente em outras máquinas, sendo necessário seguir os seguintes passos para sanar este problema.

- 1 - Exclua a pasta venv que está na pasta raiz do projeto;
- 2 - Crie um novo ambiente virtual:
comando CMD: `python -m venv venv`
- 3 - Ative o novo ambiente virtual:
comando CMD: `venv/scripts/activate.bat`
- 4 - Faça a instalação dos pacotes que estão no arquivo requirements.txt:
`pip install -r requirements.txt`

Após isso, o ambiente virtual não será um problema para testar ou executar a aplicação.

Publicação em repositório público

1º—

Crie um repositório público:

- Crie uma conta no github (ou em outra plataforma de hospedagem e versionamento de código de sua preferência, o exemplo será pelo github).

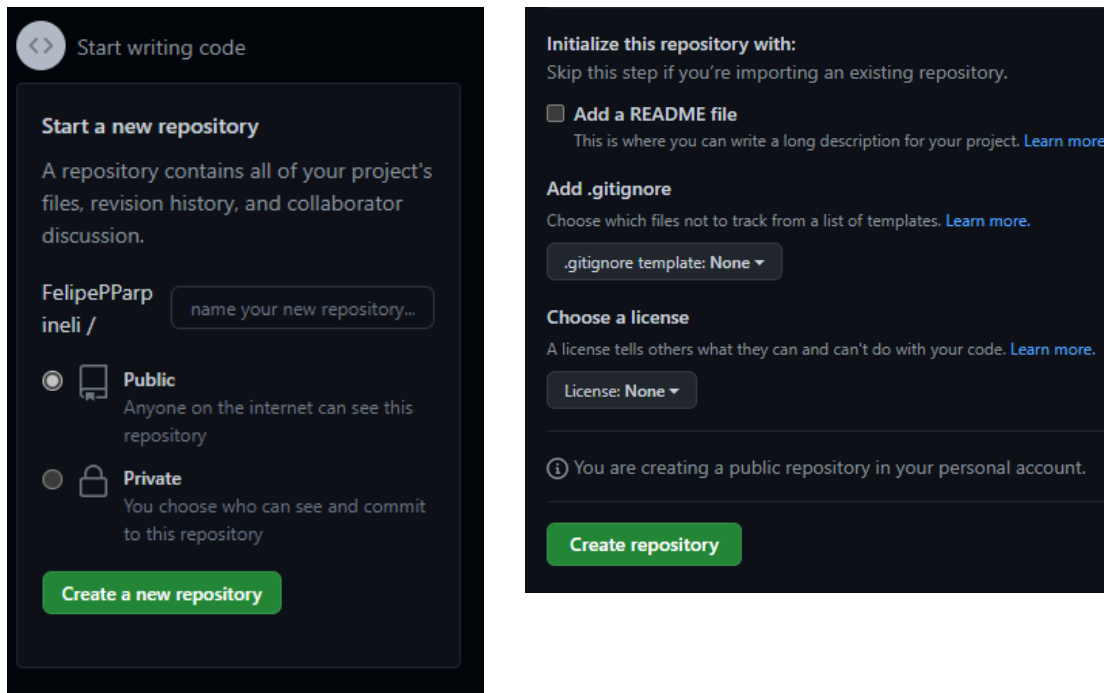
link: [Join GitHub · GitHub](#)

- Crie um repositório, seguindo este passo a passo oficial:

link: [Create a repo - GitHub Docs](#)

Marque a opção de repositório público

- **Algumas imagens que facilitarão o entendimento:**



2º—

Salve o projeto no repositório público:

- Para inserir o projeto no repositório público basta fazer o seguinte, supondo que você já tenha o **GIT** instalado na sua máquina:

- 1 - no próprio terminal abrir o git bash (terminal do GIT);
- 2 - execute o comando `git init` (certifique-se de que você está na pasta do seu projeto);
- 3 - adicionar o repositório remoto ao projeto (será o endereço do origin):
`git remote add origin <endereço do repositório>`
- 4 - Confirme os status:
`git status`
- 5 - faça o commit:
`git commit -m "coloque aqui a mensagem que quiser"`
- 6 - envie para o repositório remoto:
`git push origin main`

Pronto, o repositório público já contém o seu projeto, assim você pode fazer o controle de versionamento de forma segura, facilitando a contribuição do seu

time ou colegas para a manutenção do projeto através das branches.