**BANCO DE DADOS**

Para fazer a conexão com um banco de dados usando Python em um ambiente de desenvolvimento backend e frontend, você pode usar o framework Flask para criar uma aplicação web simples. Vou mostrar um exemplo usando Flask e SQLite, que é um banco de dados leve e embutido. O exemplo inclui um backend básico que conecta ao banco de dados e uma interface frontend para interagir com os dados.

**Backend com Flask e SQLite**

1. **Instalação das Dependências** Primeiro, instale as bibliotecas necessárias:

bash

Copiar código

pip install flask flask\_sqlalchemy

1. **Código do Backend**

python

Copiar código

from flask import Flask, render\_template, request, redirect, url\_for

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///database.db'

db = SQLAlchemy(app)

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(50), nullable=False)

email = db.Column(db.String(50), nullable=False)

def \_\_repr\_\_(self):

return f'<User {self.name}>'

@app.route('/')

def index():

users = User.query.all()

return render\_template('index.html', users=users)

@app.route('/add', methods=['POST'])

def add\_user():

name = request.form.get('name')

email = request.form.get('email')

new\_user = User(name=name, email=email)

db.session.add(new\_user)

db.session.commit()

return redirect(url\_for('index'))

if \_\_name\_\_ == '\_\_main\_\_':

db.create\_all()

app.run(debug=True)

1. **Frontend (Templates)** Crie uma pasta chamada templates no mesmo diretório do script acima e adicione um arquivo chamado index.html com o seguinte conteúdo:

html

Copiar código

<!DOCTYPE html>

<html>

<head>

<title>Users</title>

</head>

<body>

<h1>Users</h1>

<form action="/add" method="post">

<input type="text" name="name" placeholder="Name" required>

<input type="email" name="email" placeholder="Email" required>

<button type="submit">Add User</button>

</form>

<ul>

{% for user in users %}

<li>{{ user.name }} - {{ user.email }}</li>

{% endfor %}

</ul>

</body>

</html>

**Explicação do Código**

1. **Backend**
   * A aplicação Flask é configurada com uma URI de banco de dados SQLite (sqlite:///database.db).
   * A classe User define um modelo de tabela com colunas para id, name e email.
   * A rota / renderiza uma lista de usuários armazenados no banco de dados.
   * A rota /add permite adicionar novos usuários ao banco de dados via formulário.
2. **Frontend**
   * O template index.html inclui um formulário para adicionar novos usuários e uma lista para exibir os usuários existentes.

**Executando a Aplicação**

1. Salve o código do backend em um arquivo chamado app.py.
2. Crie a pasta templates e adicione o arquivo index.html conforme mostrado.
3. Execute o backend com o comando:

bash

Copiar código

python app.py

1. Abra o navegador e acesse http://127.0.0.1:5000/ para ver a interface web.

Esse exemplo básico demonstra como conectar a um banco de dados, criar um modelo, e interagir com o banco de dados através de uma interface web utilizando Flask para backend e HTML para frontend.

my\_project/

├── my\_project/

│ ├── \_\_init\_\_.py

│ ├── main.py

│ ├── module1.py

│ ├── module2.py

│ └── ...

├── tests/

│ ├── \_\_init\_\_.py

│ ├── test\_module1.py

│ ├── test\_module2.py

│ └── ...

├── docs/

│ ├── conf.py

│ ├── index.rst

│ └── ...

├── scripts/

│ ├── script1.py

│ ├── script2.py

│ └── ...

├── data/

│ └── ...

├── .gitignore

├── README.md

├── requirements.txt

├── setup.py

├── setup.cfg

├── pyproject.toml

└── MANIFEST.in

**Explicação de Cada Diretório/Arquivo:**

* **my\_project/**: O diretório principal do pacote contendo o código-fonte do seu projeto.
  + **\_\_init\_\_.py**: Indica que este diretório é um pacote Python.
  + **main.py**: O arquivo de entrada principal da aplicação (se houver).
  + **module1.py, module2.py, ...**: Módulos que contêm o código do seu projeto.
* **tests/**: Diretório para os testes unitários e de integração.
  + **\_\_init\_\_.py**: Indica que este diretório é um pacote Python.
  + **test\_module1.py, test\_module2.py, ...**: Arquivos de teste correspondentes aos módulos no diretório principal.
* **docs/**: Diretório para a documentação do projeto.
  + **conf.py**: Arquivo de configuração para ferramentas de documentação como Sphinx.
  + **index.rst**: Arquivo principal da documentação.
* **scripts/**: Diretório para scripts utilitários ou ferramentas que suportam o desenvolvimento.
  + **script1.py, script2.py, ...**: Scripts de utilidade.
* **data/**: Diretório para armazenar dados estáticos ou arquivos de configuração usados pelo projeto.
* **README.md**: Arquivo de leitura inicial com informações sobre o projeto.
* **requirements.txt**: Arquivo de lista de dependências do projeto.
* **setup.py**: Script de instalação para o projeto.
* **setup.cfg**: Arquivo de configuração para ferramentas de construção e distribuição.
* **pyproject.toml**: Arquivo de configuração para build system e metadata do projeto.
* **MANIFEST.in**: Arquivo que inclui arquivos adicionais no pacote distribuído.
* **.gitignore**: Arquivo para especificar quais arquivos/diretórios devem ser ignorados pelo Git.

**Recomendações Adicionais:**

1. **Virtual Environment**: Use ambientes virtuais (venv ou virtualenv) para isolar as dependências do projeto.
2. **Linters e Formatters**: Utilize ferramentas como flake8 para linting e black para formatação de código.
3. **Automação de Testes**: Configure um CI (Continuous Integration) para rodar seus testes automaticamente (por exemplo, usando GitHub Actions, Travis CI, ou Jenkins).
4. **Documentação**: Manter uma boa documentação é fundamental. Utilize ferramentas como Sphinx para gerar documentação a partir do seu código.

**RODANDO UM PROJETO NO GITHUB**

Para rodar projetos Python (.py) do GitHub, siga os passos abaixo:

1. **Clone o repositório**:
   * Primeiro, você precisa clonar o repositório do GitHub para o seu ambiente local. Abra o terminal e use o comando:

git clone <URL-do-repositório>

Substitua <URL-do-repositório> pelo link do repositório que você deseja clonar.

1. **Navegue para o diretório do projeto**:
   * Após clonar o repositório, entre no diretório do projeto com:

cd <nome-do-repositório>

1. **Crie um ambiente virtual (opcional, mas recomendado)**: DENTRO DO ENDEREÇO CLONADO – será criado uma pasta venv
   * É uma boa prática criar um ambiente virtual para isolar as dependências do projeto. No terminal, execute:

python -m venv venv

* + Ative o ambiente virtual:
    - No Windows:

bash

.\venv\Scripts\activate

.\activate.bat

* + - No macOS/Linux:

bash

Copiar código

source venv/bin/activate

1. **Instale as dependências do projeto**:
   * Normalmente, os projetos Python têm um arquivo requirements.txt ou Pipfile que lista todas as dependências necessárias. Para instalar as dependências a partir de um requirements.txt, use:

pip install -r requirements.txt

* + Se o arquivo não exiter crie o (pip freeze > requirements.txt)
  + Se o projeto usar Pipenv:

pipenv install

1. **Execute o script Python**:
   * Agora você pode executar o script Python. Por exemplo, se o arquivo principal do projeto for main.py, você pode rodar:

python main.py

1. **Configuração adicional**:
   * Alguns projetos podem ter configurações adicionais ou arquivos específicos que precisam ser configurados. Certifique-se de ler o README.md ou a documentação do projeto para quaisquer instruções especiais.

**Exemplo prático**

Suponha que você queira rodar um projeto disponível no repositório https://github.com/exemplo/projeto-python.

1. Clone o repositório:

git clone https://github.com/exemplo/projeto-python

1. Navegue para o diretório do projeto:

cd projeto-python

1. Crie e ative um ambiente virtual:

python -m venv venv

source venv/bin/activate # Para macOS/Linux

.\venv\Scripts\activate # Para Windows

1. Instale as dependências:

pip install -r requirements.txt

1. Execute o script Python:

python main.py

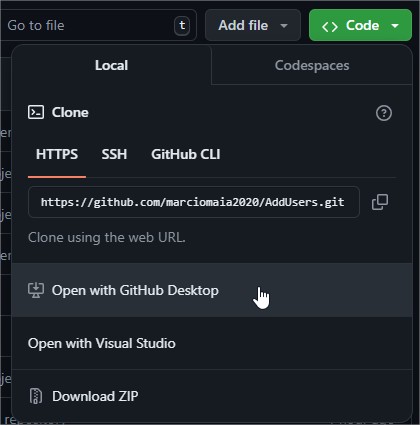
<https://www.pythonanywhere.com/user/marciomaia/>

Crie uma conta e confirme o e-mail no pythonanywhere

Crie uma conta e confirme o email no Github

Instale o Github Desktop

Se já tiver uma conta no Github então abra o repositório usando o githubDesktop



Abra no Visual Studeo Code ( CTRL+Shif+A)

Vá no site <https://flask.palletsprojects.com/en/3.0.x/> (para ver a documentação)

Crie um arquivo no visual chamado app.py

from flask import Flask

App = Flask(\_\_name\_\_)

@app.route(\*/\*)

Def hello.word():

return ‘<p>Hello, World!</p>’

Rode a aplicação Flask run (certifica que a app esta rodando)

Instale o

pip install pipenv

pipenv install

python -m venv venv

pipenv shell

python -m venv venv

source venv/bin/activate # Para macOS/Linux

.\venv\Scripts\activate # Para Windows

Successfully created virtual environment!

Virtualenv location: C:\Users\Marcio Fernando Maia\.virtualenvs\AddUsers-lIDYujiH

pipenv shell

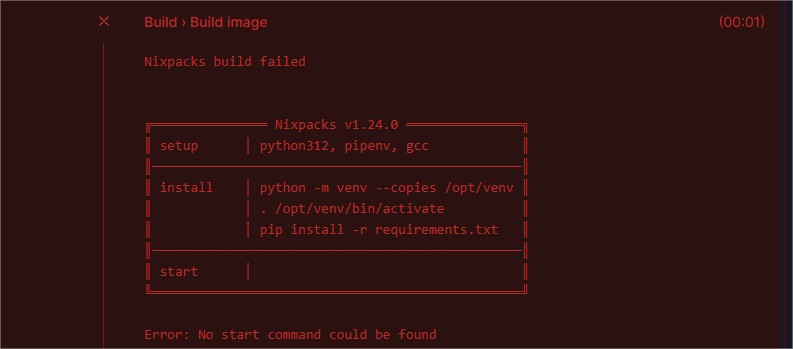
pipenv install flask

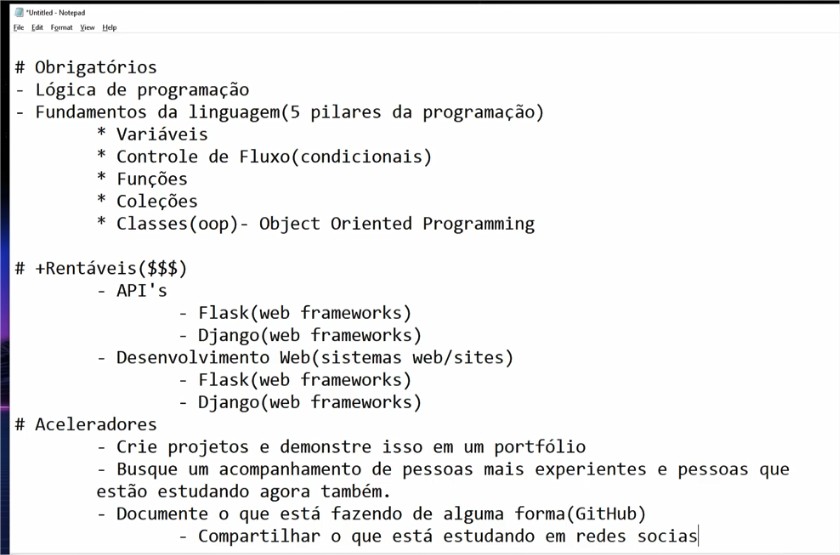
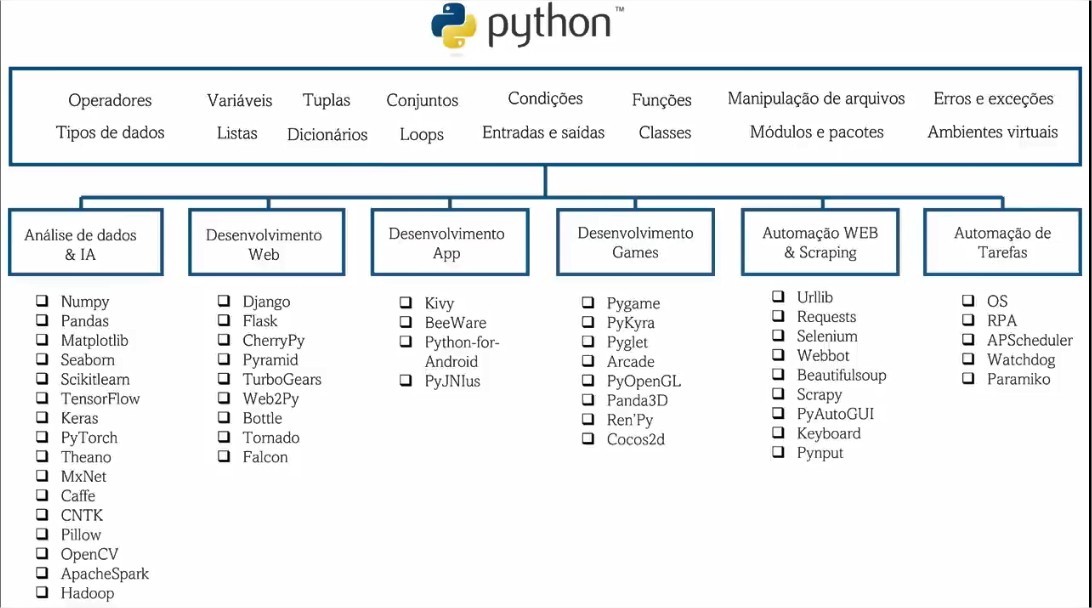
git init

git pul (no link do repositório)

<https://railway.app>

Está funcional porém preciso analisar a questão desta mensagem





1. Para colocar on-line <https://railway.app>
2. Para configurar variáveis conforme instalação do python <https://nixpacks.com/docs/providers/python> **NIXPACKS\_PYTHON\_VERSION**  3.12
3. Configure em <https://railway.app> Settings o domínio da aplicação.
4. Publicar no portifólio

**O QUE PODE CAIR NA PROVA**

Aqui estão alguns passos que você pode seguir para resolver esse problema:

**1. Verifique a Ativação do Ambiente Virtual**

Certifique-se de que o ambiente virtual está corretamente ativado. No Windows, você deve usar:

venv\Scripts\activate

E no Linux/Mac:

source venv/bin/activate

**2. Verifique se o Python e o Pip Estão Corretamente Instalados**

Você pode verificar se o Python está corretamente instalado executando:

python --version

pip --version

**3. Recrie o Ambiente Virtual**

Se a ativação do ambiente virtual não resolver o problema, tente recriar o ambiente virtual:

1. Exclua a pasta venv existente:

rm -rf venv # Linux/Mac

rmdir /s /q venv # Windows

1. Crie um novo ambiente virtual:

python -m venv venv

3 Instalar o flask neste ambienta virtual:

pipenv install flask

4 Ative o novo ambiente virtual:

venv\Scripts\activate # Windows

source venv/bin/activate # Linux/Mac

**4. Instale as Dependências**

Com o ambiente virtual ativado, tente novamente instalar as dependências:

pip install flask flask-login flask-mail