

**INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA BAIANO - CAMPUS
GUANAMBI**

GABRIEL SOARES ROCHA

**E-BOOK AUXILIADOR PSW
VOL 1**

**GUANAMBI-BA
2019**

GABRIEL SOARES ROCHA

E-BOOK PSW

VOL 1

Feito unicamente na intenção de ajudá-los(las) na disciplina de Programação de Sistemas Web do prof. Cleyton.

GUANAMBI-BA

2019

SUMÁRIO

| | |
|---|----|
| 1. Lista de passos para as primeiras criações e instalações:..... | 4 |
| 2. Criação de uma página simples com o flask:..... | 4 |
| 3. Programas a serem instalados:..... | 5 |
| 4. Lista de comandos para Linux:..... | 6 |
| 5. Browser – Banco de Dados:..... | 6 |
| 6. Criando um banco com o flask-sqlalchemy:..... | 7 |
| 7. Templates:..... | 8 |
| 8. Criando o Blog (simples):..... | 8 |
| 9. Anexos:..... | 10 |

1. Lista de passos para as primeiras criações e instalações:

1. Criar um diretório com seu nome para trabalhar o projeto;
2. Dentro dele abra o terminal e digite “sudo apt install virtualenv” para instalar a “virtualenv” caso não tenha;
3. Logo após digite “virtualenv -p python3 env” para criar uma MV(Máquina Virtual);
 1. nomes padrões “env” ou “venv”;
4. Criada a MV, ative-a com o comando “source env/bin/activate”, assim você terá ela pronta para uso;
 1. Ou o nome que você deu a MV, no caso acima “env”;
5. Depois você instalará o FLASK(mini framework), com este comando “pip install flask” dentro da env ativada;
 1. Para conferir a instalação digite “pip freeze”;
6. E para desativar a env use o comando “deactivate”.

2. Criação de uma página simples com o flask:

1. Abra um editor de texto, do próprio Linux ou um programa com os mesmos fins;
2. Agora você construirá a estrutura da página com códigos em python, salvando o arquivo como o nome que quiser, com a extensão “.py”;
 1. Padrão de nome -”site.py”;
 2. Visualizar (Anexo_1);

3. Feito isso, no terminal dentro da env, se você fez todo direito digite python3 e o nome do seu arquivo.py, ex: "python3 site.py";
4. Como esse comando será gerado um link, com o botão esquerdo click nele e escolha a opção "abrir link" e sua página logo será aberta.

3. Programas a serem instalados:

1. Abra o terminal, de preferência em sua pasta (a criada para o projeto com o seu nome);
2. Agora com os seguintes comandos instale os programas;
 1. "sudo apt install sqlite3" - SQLite é uma biblioteca em linguagem C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo SGBD separado, segundo o Wikipédia;
 2. "sudo apt install sqlitebrowser" - É o programa executável que se acessa ao procurá-lo em atividades e digitar "DB Browser for SQLite";
 3. "sudo snap install code" - com este você instala o Visual Studio Code, caso não funcione tente "sudo snap install code --classic";
 4. "pip install flask-sqlalchemy" - Este será instalado na env ativada. O Flask-SQLAlchemy é uma extensão do Flask que adiciona suporte ao SQLAlchemy ao seu aplicativo. O objetivo é simplificar o uso do SQLAlchemy com Flask, fornecendo padrões úteis e auxiliares extras que facilitam a realização de tarefas comuns, segundo o site documental oficial do flask-sqlalchemy;
 5. "pip install ipython" - Assim como o comando acima este também é dentro da env. Ipython é um interpretador interativo para várias linguagens de

programação, mas especialmente focado em Python. Ipython oferece "type introspection", "rich media", syntax shell, completção por tab e edição auxiliada por histórico de comando, segundo o Wikipédia sobre o ipython, e para usá-lo basta digitar "ipython" e para sair "exit", na env.

4. Lista de comandos para Linux:

1. Guia_de_500_comandos_Linux. Disponível em:
<https://www.linuxpro.com.br/dl/guia_500_comandos_Linux.pdf>. Acesso em: 13 de set. 2019.
2. Com este guia você poderá aprender alguns comandos que podem lhe ser útil na hora de programar.

5. Browser – Banco de Dados:

1. Após a instalação do "sqlite3" e do "sqlitebrowser" com os comandos que você viu nos conteúdos **3.2.1** e **3.2.2**, iremos a sua utilização;
2. Primeiramente abra o programa "DB Browser for SQLite" ao digitá-lo em "Atividades". No programa aberto vá em "Novo banco de dados", depois escolha um diretório pra salvá-lo, com um nome e a extensão ".db";
 1. Exemplo, "meuBanco.db";
 2. Visualizar (Anexo_2);
3. Feito isso, abrirá uma tela na qual você irá defini um nome para sua tabela (Anexo_3);
 1. exemplo (Anexo_4);
4. Depois você irá em "Adicionar campo", que adicionará um atributo a tabela criada, na qual pode-se alterar o nome, tipo(integer, text, blob, real e numeric), além das opções Não nulo, PK(Primary Key), AI(automático) e U(Único);

1. Visualizar (Anexo_5);
2. Exemplo (Anexo_6);
5. Feito os campos, selecione a tabela criada (Anexo_7), e vá em “Navegar dados”, nessa área pode-se atribuir valores aos campos criados (Anexo_8);
 1. exemplo (Anexo_9);

6. Criando um banco com o flask-sqlalchemy:

1. Para esse processo você precisará já ter instalado todos programas do item 3, e ter a sua env;
2. No diretório que contém sua **env** crie um arquivo “.py” com um editor de texto contendo os seguintes códigos, visualizar(Anexo_10);
 1. exemplo “DataBase.py”;
3. Depois abrir no terminal a sua env , ative-a. Acesse o “ipython” e insira os dados comandos;
 1. from “nome_do_arq.py” import db;
 1. ex: from DataBase.py import db;
 2. Repita o primeiro - “from “nome_do_arq.py” import db”;
 3. db_create_all();
 4. visualizar (Anexo_11);
4. Após todo este processo você terá em seu diretório juntamente com a sua env um arquivo chamando “banco1.db” e uma pasta “__pycache__”, na qual você pode está acessando pelo DB Browser for SQLite no mesmo esquema do item **5.5**;
 1. OBS: O banco criado recebe o nome de “banco1”, pois está definido no código visualizado no (Anexo_10), no qual você pode alterá-lo com o nome que preferir, sem esquecer de colocar a extensão “.db” ao final.

7. Templates:

1. Dentro do diretório com sua **env** crie um nova pasta com o nome “templates”;
 1. É um nome padrão;
2. Nessa pasta serão armazenados arquivos “.HTML” e outros, que para serem acessados, é inserido ao “`arq.py`” o módulo “`render_template`”;
 1. Este “`arq.py`” pode-se tomar como exemplo o (Anexo_1);
 2. A inserção será feito no seguinte esquema: “`from flask import Flask, render_template`”;
 3. Visualizar (Anexo_12);
3. Feito isso, agora você criará uma rota que ligará o “`arq.html`” dentro da pasta “templates” a página web, no “`arq.py`” na qual está vinculada;
 1. Visualizar (Anexo_13);
 2. Isso após ter criado um “`arq.html`” de preferência com algum conteúdo no diretório “templates”, como no (Anexo_13), “`index.html`”.

8. Criando o Blog (simples):

1. O primeiro passo, é fazer todos os processos do item 1. Feito isso, crie um diretório templates e logo após instale o Virtual Studio Code, caso este não esteja instalado (para a instalação visualizar item 3.2.3);
2. Em seguida, no terminal, ative a sua env e abra o Virtual Studio Code com o comando “`code .`”. O programa será aberto e você visualizará todos os arquivos em seu diretório: Pasta env, pasta templates e `arq.py`;
 1. adicione na primeira linha de código do seu `arq.py` o módulo “`render_template`”, para as criações de rotas de arquivos “.html” da pasta templates, para que possam ser acessadas quando forem chamadas no navegador;
 2. Visualizar (Anexo_14);
3. Posterior, ainda no “code”, vá em seu diretório templates e crie um arquivo, “`base.html`”. No qual servirá de base para os futuros arquivos. Nela digite

“html”, desse modo pode lhe aparecer a opção “HTML:5”, selecione-a, mas se não apenas construa a estrutura básica de uma página HTML;

1. Visualizar (Anexo_15);

4. Dentro do seu arquivo base.html, entre as tags “<title>” e “<body>”, adicione os comandos do “Jinja”. No primeiro escreva “{% block title %}{% endblock title %}”, e no segundo “{% block conteudo %} {% endblock conteudo %}”

1. Visualizar (Anexo_16);

5. Ainda na pasta templates crie outro arquivo, com o nome “blog.html”. Nele você iniciará a primeira linha com o código “{% extends 'base.html' %}”, que fará uma ponte com o outro arquivo, no qual importará as funcionalidades dos códigos que forem sendo acrescentados nele, além dos códigos já presentes;
6. Agora com os códigos encontrados no item acima, **8.4**, adicione um título e um conteúdo ao seu arquivo.html (blog.html);

1. Visualizar (Anexo_17);

7. Após a criação e inserção dos devidos códigos no seu arquivo.html (blog.html), vá para o “arq.py” e declare uma rota para sua página;

1. Visualizar (Anexo_18);

8. Para finalizar, teste-o para confirmar sua funcionalidade, vá ao terminal com sua env ativada e digite “python3 nome_do_arquivo.extensão” (python3 arq.py), se aparecer as seguintes telas, seu blog simples estará funcionando;

1. OBS: não esqueça de escrever ao final do link a rota “/blog”, para o acesso da página;

2. Tela01 visualizar (Anexo_19), e tela02 visualizar (Anexo_20).

9. Anexos:

1. Anexo_1:

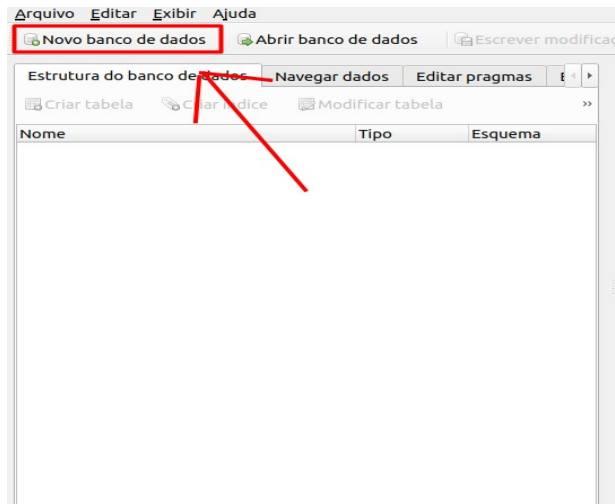
```
from flask import Flask

app = Flask(__name__)

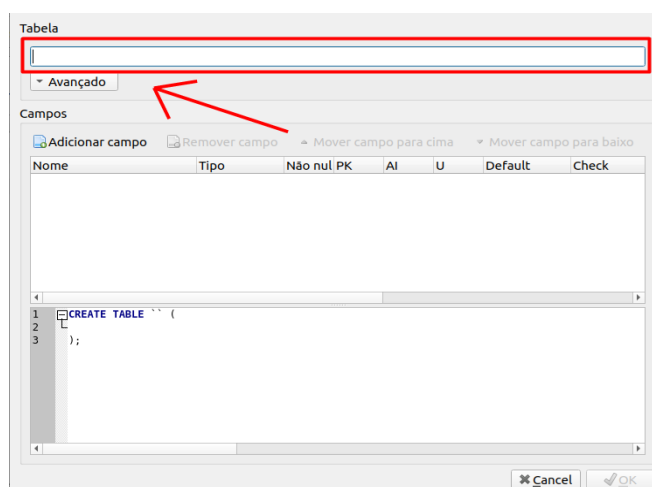
@app.route('/')
def index():
    return "Hello word!"

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

2. Anexo_2:



3. Anexo_3:



4. Anexo_4:

Tabela

Pessoa

Avançado

Campos

Adicionar campo Remove campo Mover campo para cima Mover campo para baixo

| Nome | Tipo | Não nul | PK | AI | U | Default | Check |
|------|------|---------|----|----|---|---------|-------|
|------|------|---------|----|----|---|---------|-------|

```
1 CREATE TABLE Pessoa (  
2  
3 );
```

Cancel OK

5. Anexo_5:

Tabela

pessoa

Avançado

Campos

Adicionar campo Remove campo Mover campo para cima Mover campo para baixo

| Nome | Tipo | Não nul | PK | AI | U | Default | Check |
|--------|---------|--------------------------|--------------------------|--------------------------|--------------------------|---------|-------|
| Field1 | INTEGER | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | | |

↖ ↑ ↑ ↑ ↑ ↑

```
1 CREATE TABLE 'pessoa' (  
2  
3 );
```

Field1 INTEGER

Cancel OK

6. Anexo_6:

Tabela

Pessoa

Avançado

Campos

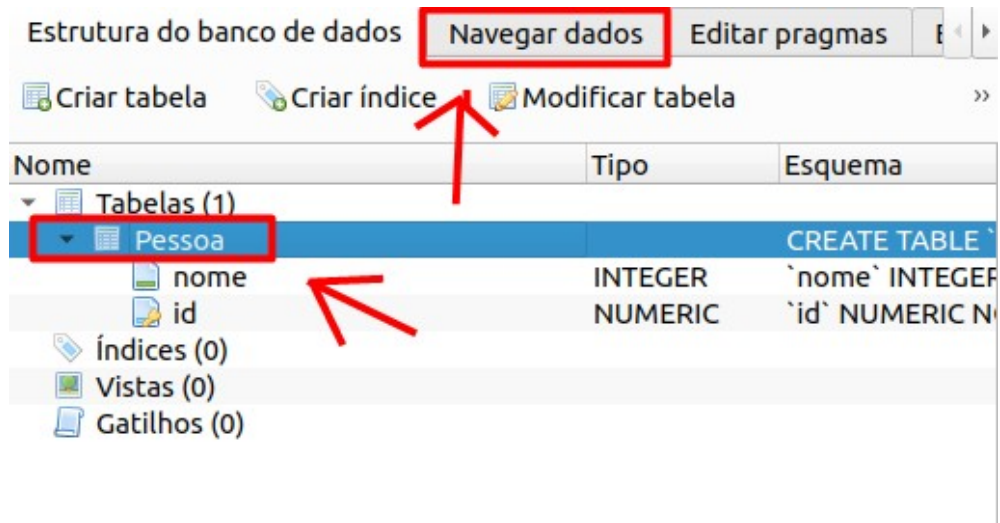
Adicionar campo Remove campo Mover campo para cima Mover campo para baixo

| Nome | Tipo | Não nul | PK | AI | U | Default | Check |
|--------|---------|-------------------------------------|--------------------------|--------------------------|-------------------------------------|---------|-------|
| nome | TEXT | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | | |
| id | NUMERIC | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | |
| altura | REAL | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | | |
| xxx | INTEGER | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | | |
| xxxx | BLOB | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | | |

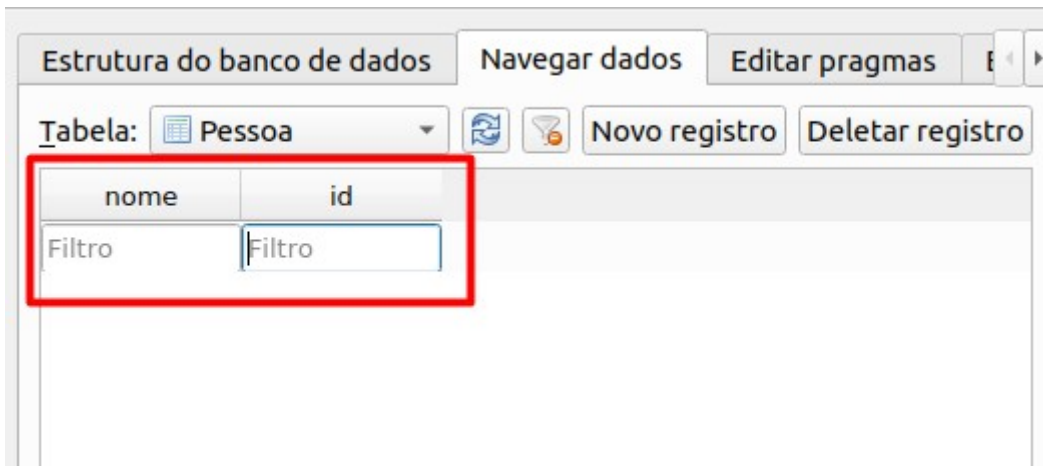
```
1 CREATE TABLE 'Pessoa' (  
2 'nome' TEXT NOT NULL,  
3 'id' NUMERIC NOT NULL UNIQUE,  
4 'altura' REAL,  
5 'xxx' INTEGER,  
6 'xxxx' BLOB  
7 );
```

Cancel OK

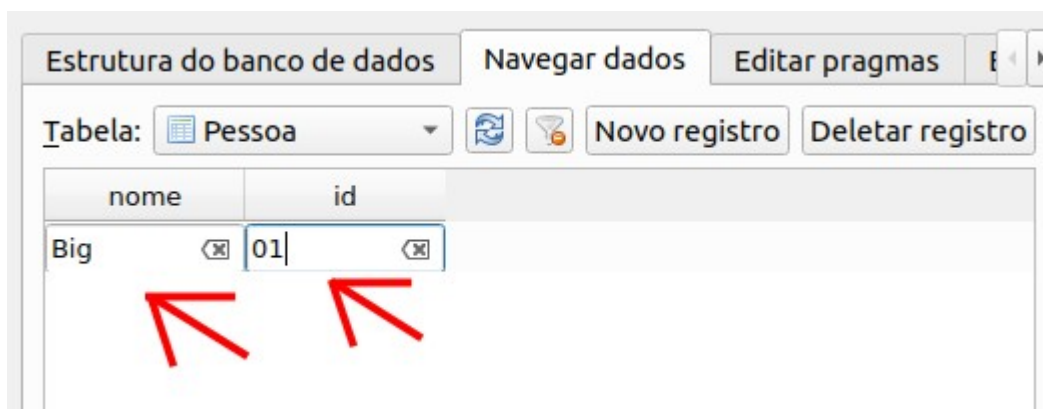
7. Anexo_7:



8. Anexo_8:



9. Anexo_9:



10. Anexo_10:

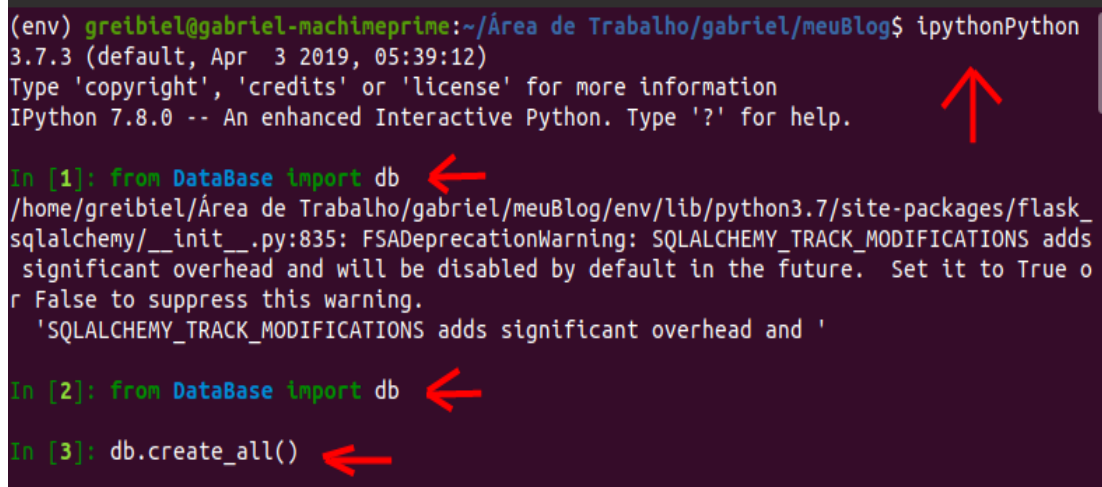
```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///banco1.db'
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)

    def __repr__(self):
        return '<User %r>' % self.username
```

11. Anexo_11:



The image shows a terminal window with the following content:

```
(env) greibiel@gabriel-machineprime:~/Área de Trabalho/gabriel/meuBlog$ ipythonPython
3.7.3 (default, Apr  3 2019, 05:39:12)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from DataBase import db
/home/greibiel/Área de Trabalho/gabriel/meuBlog/env/lib/python3.7/site-packages/flask_sqlalchemy/__init__.py:835: FSADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '

In [2]: from DataBase import db

In [3]: db.create_all()
```

Red arrows point to the import statements and the `db.create_all()` command. A red arrow also points to the IPython prompt.

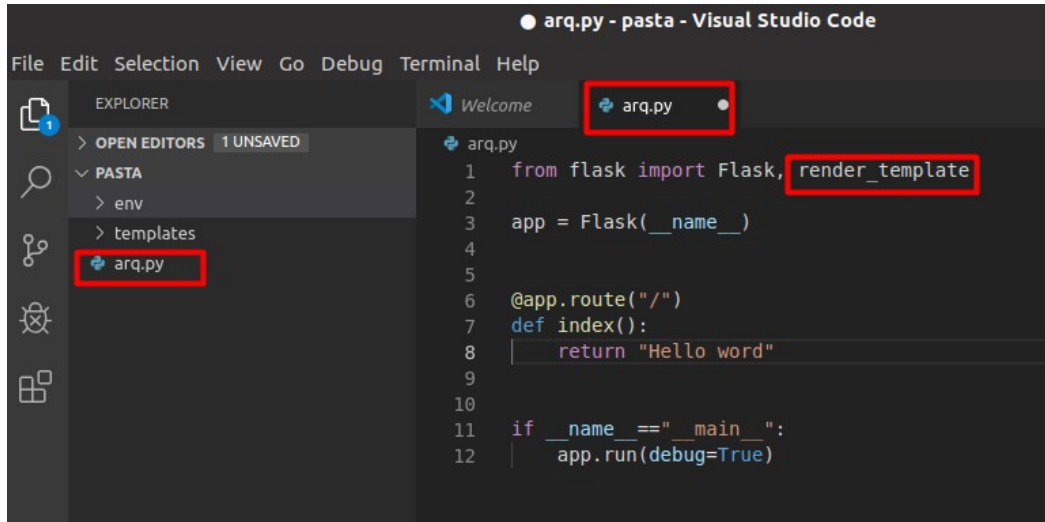
12. Anexo_12:

```
from flask import Flask, render_template
```

13. Anexo_13:

```
@app.route('/')
def index():
    return render_template("index.html")
```

14. Anexo_14:

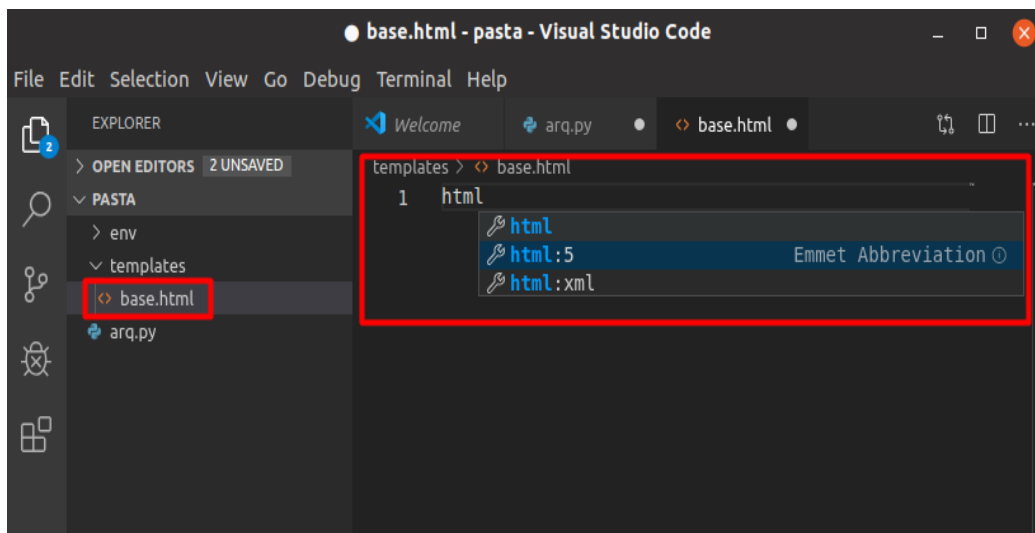


The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure: PASTA > env > templates > arq.py. The arq.py file is open in the editor, showing the following code:

```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5
6 @app.route("/")
7 def index():
8     return "Hello word"
9
10
11 if __name__ == "__main__":
12     app.run(debug=True)
```

The `render_template` function in the first line is highlighted with a red box.

15. Anexo_15:



The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure: PASTA > env > templates > base.html. The base.html file is open in the editor, showing the following code:

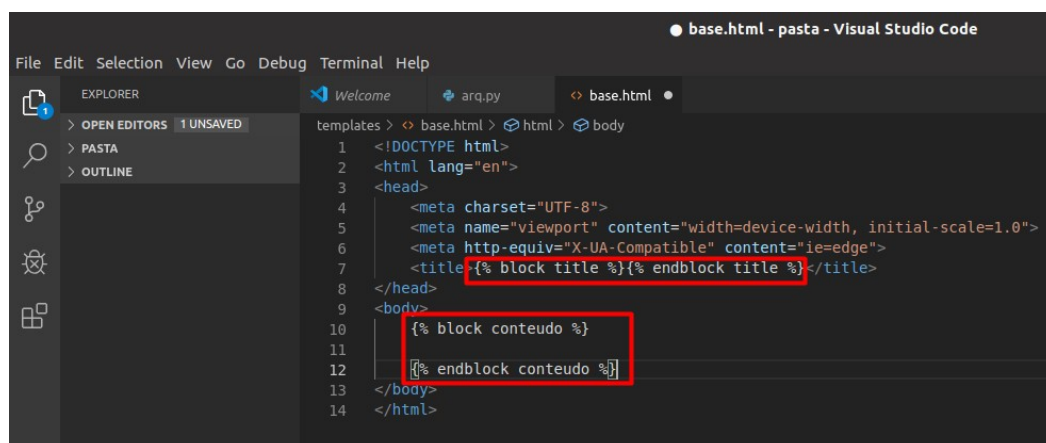
```
1 html
```

The Emmet abbreviation menu is open, showing the following options:

- html
- html:5
- html:xml

The menu is highlighted with a red box.

16. Anexo_16:




The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure: PASTA > env > templates > base.html. The base.html file is open in the editor, showing the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <meta http-equiv="X-UA-Compatible" content="ie=edge">
7 <title>{% block title %}{% endblock title %}</title>
8 </head>
9 <body>
10 {% block conteudo %}
11
12 {% endblock conteudo %}
13 </body>
14 </html>
```

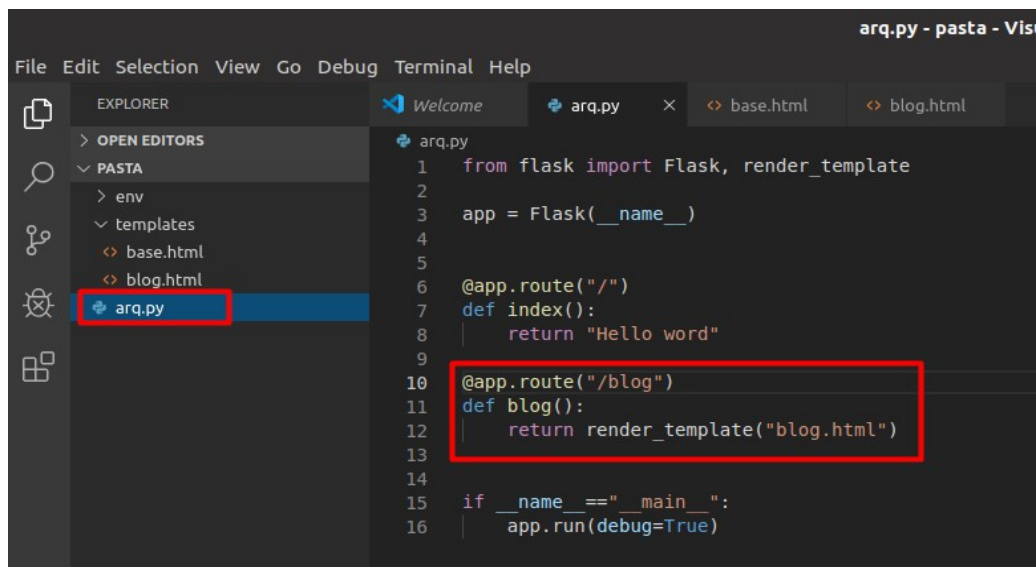
The `{% block title %}{% endblock title %}` and `{% block conteudo %}` blocks are highlighted with red boxes.

17. Anexo_17:



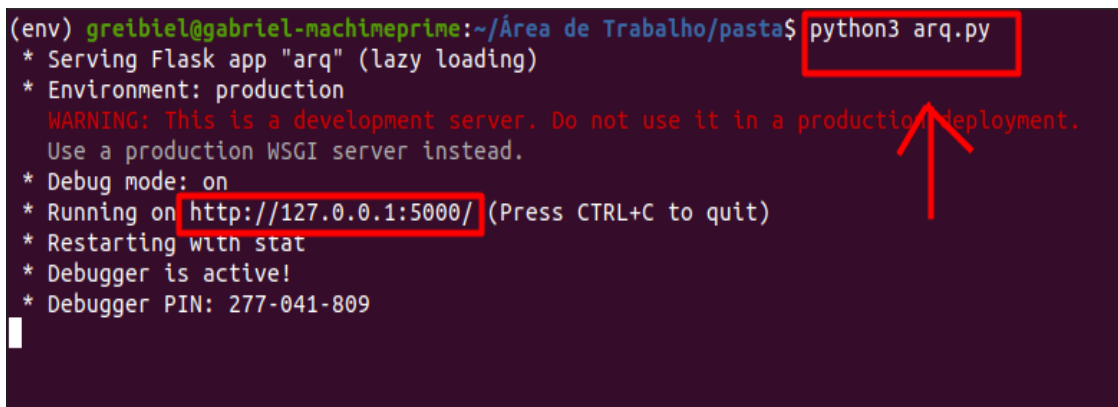
```
File Edit Selection View Go Debug Terminal Help
EXPLORER
  > OPEN EDITORS
  > PASTA
    > env
    > templates
      <> base.html
      <> blog.html
  arq.py
arq.py
templates > <> blog.html > ...
1  {% extends 'base.html' %}
2
3  {% block title %} BLOG {% endblock title %}
4
5  {% block conteudo %}
6
7      <h1>Blog</h1>
8
9  {% endblock conteudo %}
```

18. Anexo_18:



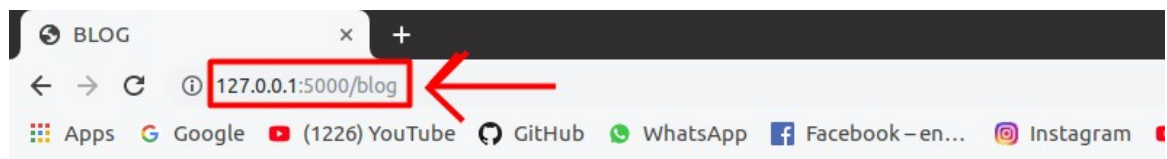
```
File Edit Selection View Go Debug Terminal Help
EXPLORER
  > OPEN EDITORS
  > PASTA
    > env
    > templates
      <> base.html
      <> blog.html
  arq.py
arq.py
arq.py
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5
6  @app.route("/")
7  def index():
8      return "Hello word"
9
10 @app.route("/blog")
11 def blog():
12     return render_template("blog.html")
13
14
15 if __name__ == "__main__":
16     app.run(debug=True)
```

19. Anexo_19:



```
(env) greibiel@gabriel-machineprime:~/Área de Trabalho/pasta$ python3 arq.py
* Serving Flask app "arq" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 277-041-809
```

20. Anexo_20:



Blog