



UNIVERSIDADE
FEDERAL DO PIAUÍ

Universidade Federal do Piauí

Disciplina: Engenharia de Software II

Professor: Armando Soares Sousa

Projeto: Microgram

Equipe:

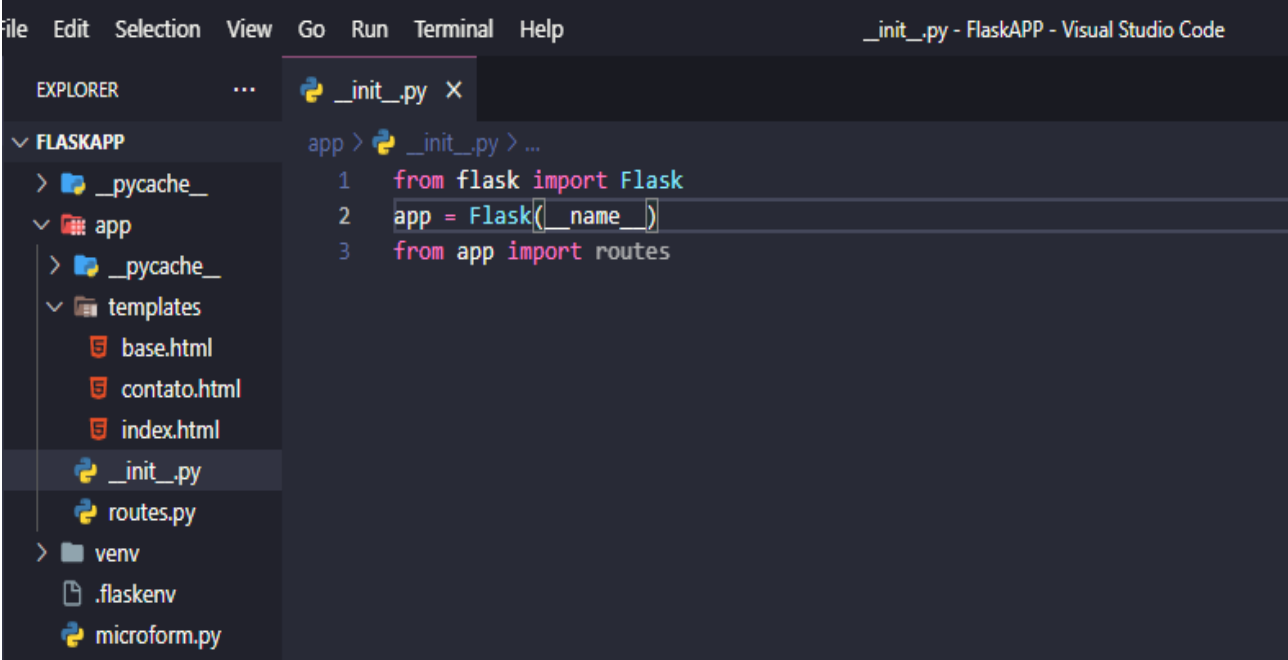
- Carlos Eduardo Mendes de Oliveira.
- José Pires Gayoso
- Fernando Vieira Rosa
- Markesley Ramos
- Bruno Estrella

Tutorial 2

Tutorial de como criar uma aplicação web básica
mostrando um formulário HTML

- 1) **Introdução:** É importante frisar que esse tutorial depende do tutorial 1 que mostra a instalação do framework Flask e configuração do ambiente de desenvolvimento. Além de conhecimentos em HTML.
- 2) **Utilizando o flask e inicializando a aplicação:** Com o flask já instalado crie uma pasta chamada app no

ambiente de desenvolvimento que terá as dependências do projeto, dentro dessa pasta crie um arquivo chamado `__init__.py` e outro arquivo chamado `routes.py`, o `__init__.py` que servirá para as solicitações HTTP, pois dentro desse arquivo haverá as rotas possíveis. Importe o objeto `Flask` do pacote `flask`. Em seguida, use-o para criar sua instância de aplicativo Flask com o nome `app`. Passe a variável especial `__name__` que guarda o nome do módulo Python atual. Ela é usada para dizer à instância onde ela está localizada. Ao final faça a importação das rotas de navegação que a página terá, importando o arquivo criado chamado `routes.py`, que será configurado a seguir.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure for 'FLASKAPP'. It includes a '__pycache__' folder, an 'app' folder containing another '__pycache__' folder and a 'templates' folder with 'base.html', 'contato.html', and 'index.html' files. Below these are the Python files '__init__.py' and 'routes.py', followed by a 'venv' folder, a '.flaskenv' file, and a 'microform.py' file. The main editor window shows the content of '__init__.py' with the following code:

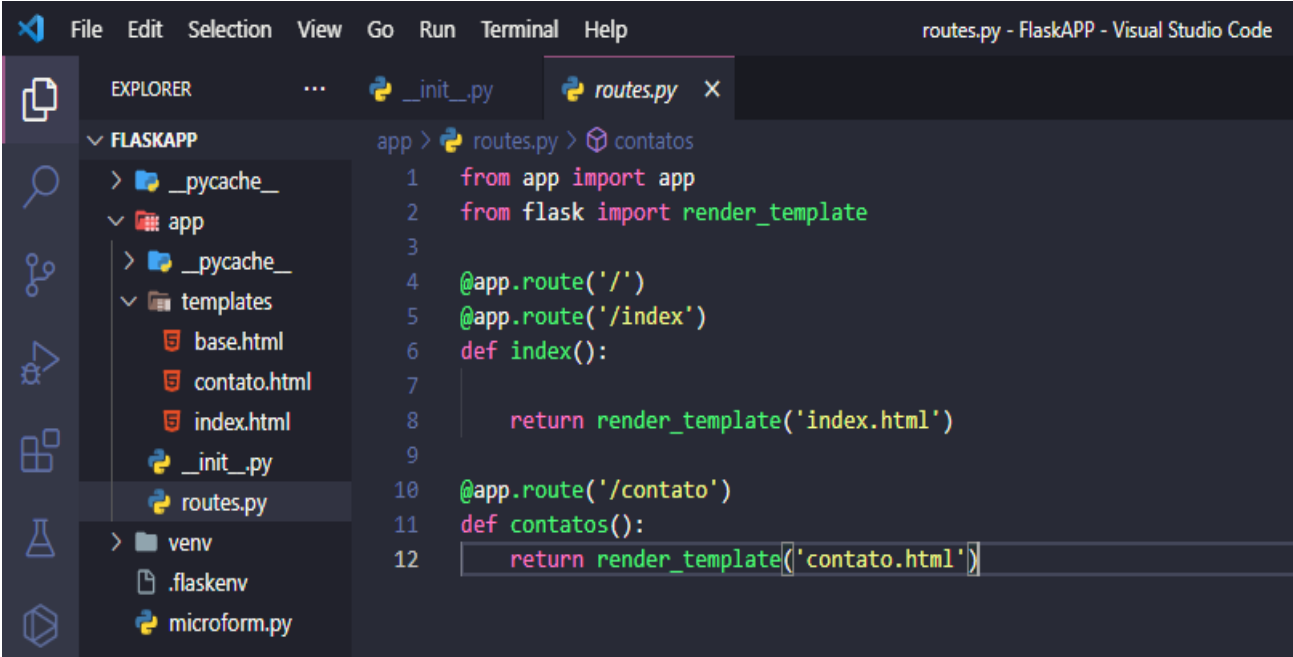
```
app > __init__.py > ...
1  from flask import Flask
2  app = Flask(__name__)
3  from app import routes
```

3) Configurando as rotas: Para criar as possíveis rotas da

sua aplicação crie uma instância de `app` nesse arquivo `routes.py`, utilizando “`from app import app`”, assim que criar a instância `app`, use-a para processar solicitações Web de entrada e enviar respostas ao usuário. O

`@app.route` é um decorador que transforma uma função Python comum em uma função *de visualização* Flask. Ela converte o valor de retorno da função em uma resposta

HTTP para ser mostrada por um cliente HTTP, como um navegador Web. Passe o valor `'/'` para `@app.route()` para indicar que esta função responderá às solicitações Web para a URL `/`, que é a URL principal. No caso do nosso formulário criaremos 2 rotas: index e contato, então criaremos duas rotas utilizando o decorador e passando os valores `/index` e `/contato` em `@app.route()` nas suas respectivas rotas. Em cada rota é preciso haver uma função logo abaixo que execute algo, no nosso caso precisamos integrar essas rotas com páginas HTML que serão mostradas a seguir.

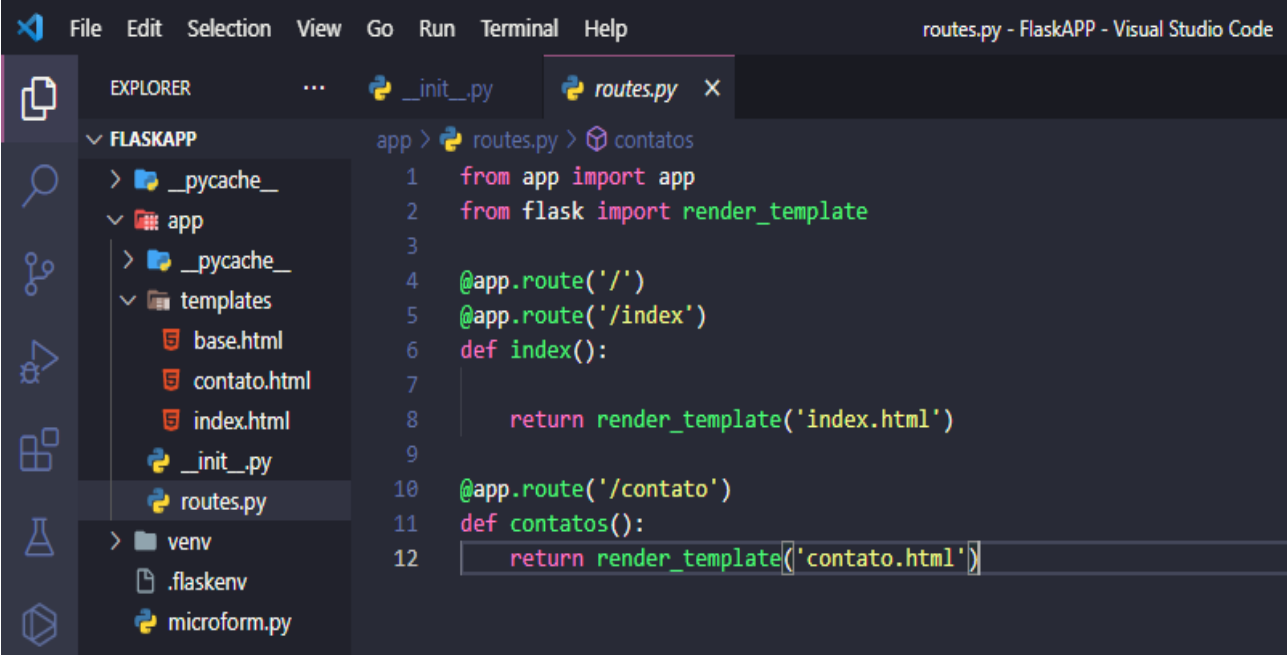


The screenshot shows the Visual Studio Code interface with the 'routes.py' file open. The Explorer sidebar on the left shows the project structure: 'FLASKAPP' containing a 'templates' folder with 'base.html', 'contato.html', and 'index.html', and a 'routes.py' file. The main editor displays the following Python code:

```
1 from app import app
2 from flask import render_template
3
4 @app.route('/')
5 @app.route('/index')
6 def index():
7     return render_template('index.html')
8
9
10 @app.route('/contato')
11 def contatos():
12     return render_template('contato.html')
```

- 4) **Integrar páginas HTML:** Agora, para integrar a aplicação a páginas HTML o Flask fornece uma função auxiliar `render_template()` que permite o uso do mecanismo modelo do Jinja. Isso facilitará bastante o gerenciamento do HTML ao escrever seu código HTML em arquivos `.html` e ao usar lógica em seu código HTML. Você usará esses arquivos HTML, para desenvolver todas as suas páginas do aplicativo. Para criar o formulário é interessante criar uma pasta chamada `templates` dentro da pasta `app`, em que você colocará os arquivos HTML,

para que a página exiba isso, utiliza-se no arquivo `routes.py` dentro de cada uma das rotas um `return render_template(aqui dentro coloca-se o nome do template desejado)`. No nosso caso colocaremos os templates de `index.html` que é a página principal de com as informações do formulário a serem preenchidas e na outra rota o template de `contato.html` que será uma página de informações contatos do fornecedor do serviço.



The screenshot shows the Visual Studio Code interface with the FlaskAPP project open. The Explorer sidebar on the left shows the project structure: `FLASKAPP` (containing `__pycache__`, `app`, and `venv`), `app` (containing `__pycache__`, `templates`, `__init__.py`, and `routes.py`), and `venv` (containing `.flaskenv` and `microform.py`). The `templates` folder is expanded, showing `base.html`, `contato.html`, and `index.html`. The `routes.py` file is open in the editor, showing the following code:

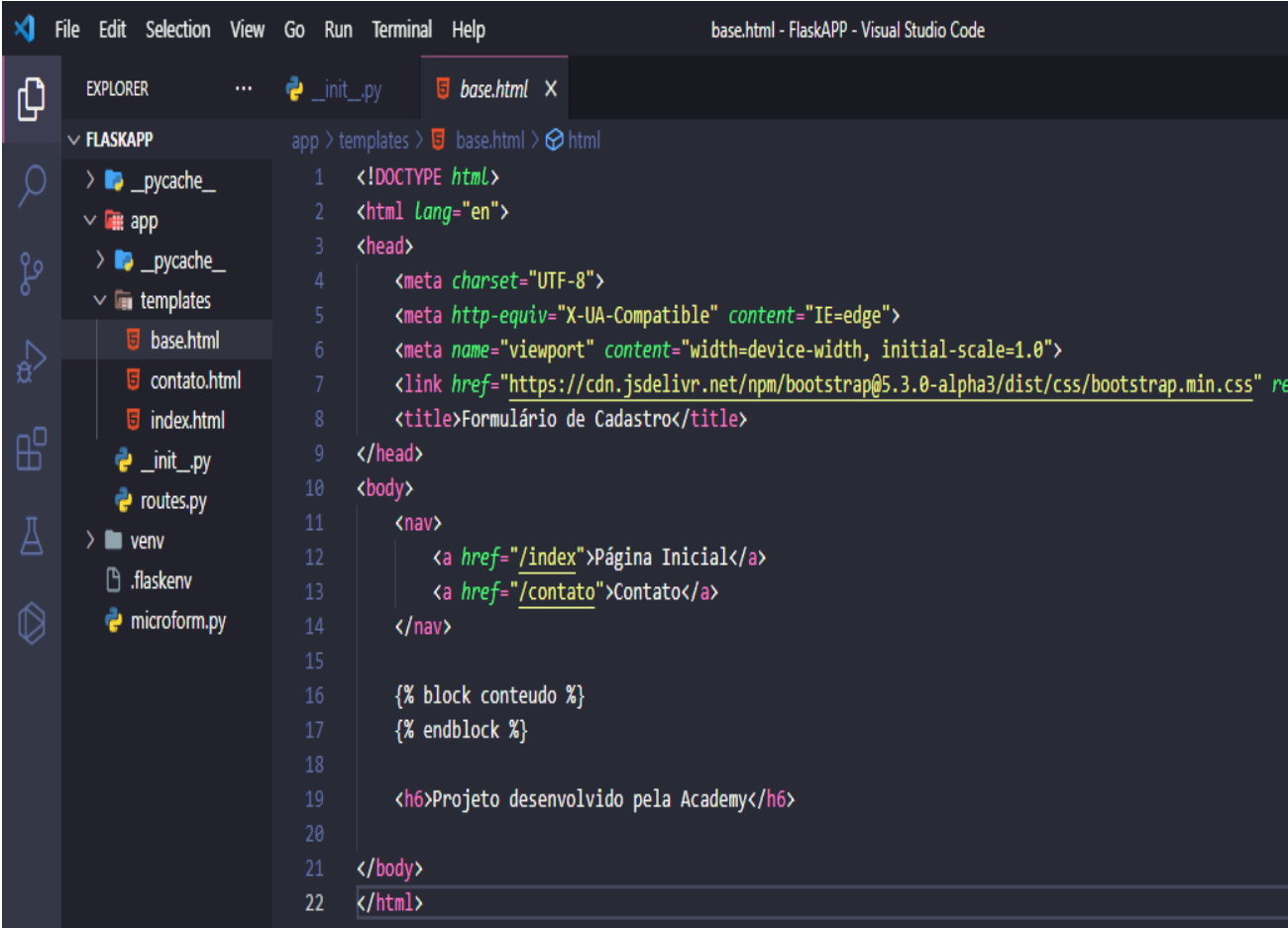
```
app > routes.py > contatos
1 from app import app
2 from flask import render_template
3
4 @app.route('/')
5 @app.route('/index')
6 def index():
7
8     return render_template('index.html')
9
10 @app.route('/contato')
11 def contatos():
12     return render_template('contato.html')
```

5) **Detalhamento das páginas HTML:** As páginas ficam a critério do desenvolvedor, como o nosso foco é o desenvolvimento de um formulário de cadastro, foi criado 3 páginas HTML, na pasta `templates`, no arquivo `base.html` é onde há a estrutura básica geral que toda página da nossa aplicação pode ter, assim em `index.html` e `contato.html`, são criadas também em `templates` e elas herdam essa estrutura básica utilizando a extensão de páginas que o flask permite chamado `extends`. A herança do modelo também dá a você a capacidade de reutilizar o código HTML que você tem em outros modelos (`base.html` neste caso), sem precisar repeti-lo sempre que ele for necessário. Em `index` toda a estrutura do

formulário é montada, com os campos de preenchimento usuário, nome, email e senha, e o botão de cadastro.

Além disso, utilizamos Bootstrap 5 na aplicação utilizando o link do Bootstrap que está em base.html. Já em no template contato, coloca-se as informações de contato do fornecedor.

base.html: Nela haverá a estrutura necessária para construir uma página HTML, na tag body haverá a tag nav que é uma barra de navegação no topo das páginas e um h6 que será o footer da página.



```
File Edit Selection View Go Run Terminal Help
base.html - FlaskAPP - Visual Studio Code

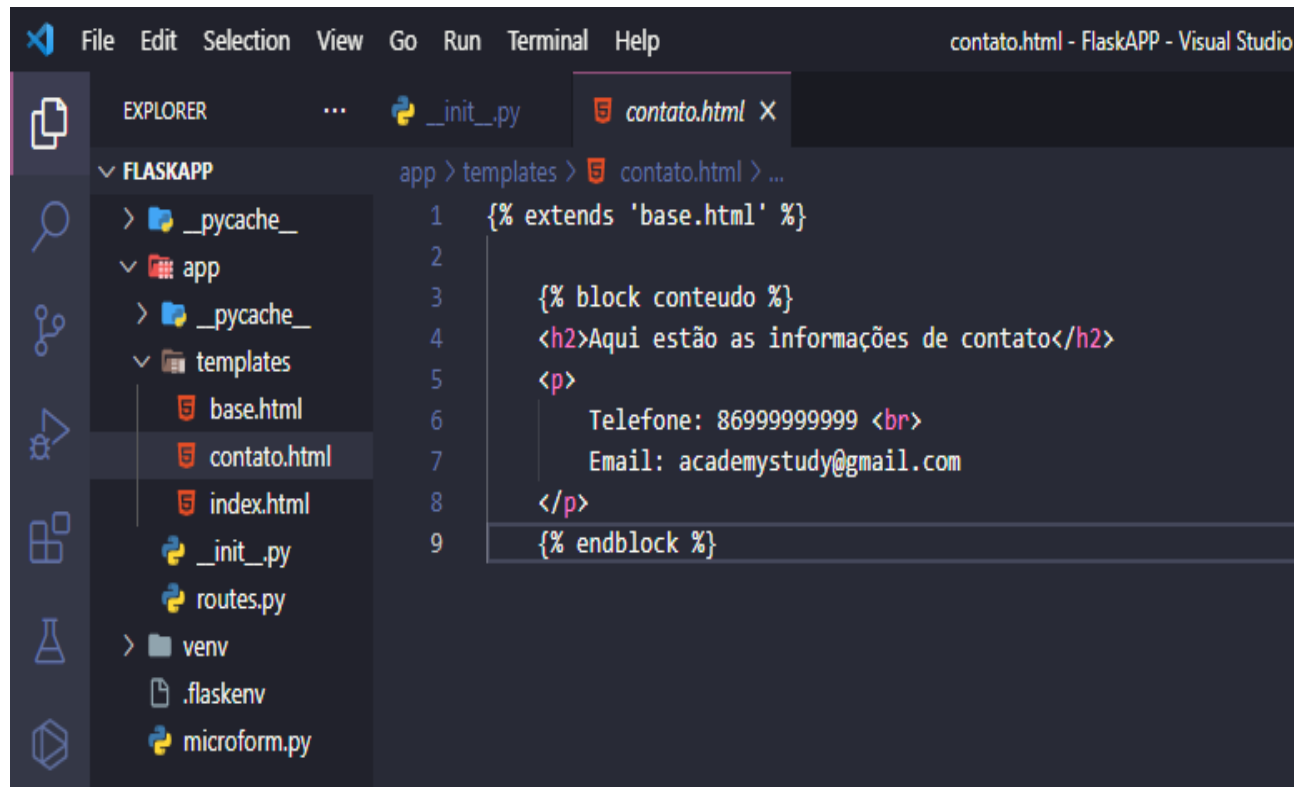
EXPLORER
FLASKAPP
  > _pycache_
  > app
  > _pycache_
  > templates
    base.html
    contato.html
    index.html
  _init_.py
  routes.py
  venv
  .flaskenv
  microform.py

app > templates > base.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
8   <title>Formulário de Cadastro</title>
9 </head>
10 <body>
11   <nav>
12     <a href="/index">Página Inicial</a>
13     <a href="/contato">Contato</a>
14   </nav>
15
16   {% block conteudo %}
17   {% endblock %}
18
19   <h6>Projeto desenvolvido pela Academy</h6>
20
21 </body>
22 </html>
```

index.html:

```
Go Run Terminal Help index.html - FlaskAPP - Visual Studio Code
__init__.py index.html X
app > templates > index.html > ...
1 {% extends 'base.html' %}
2
3 {% block conteudo %}
4 <h2>Bem-vindo ao cadastro de usuários</h2>
5 <form action="">
6     <div class="form-group">
7         <label for="usuario">Usuário:</label>
8         <input type="text" class="form-control" id="usuario" name="usuario" required>
9     </div>
10
11     <div class="form-group">
12         <label for="nome">Nome completo:</label>
13         <input type="text" class="form-control" id="nome" name="nome" required>
14     </div>
15
16     <div class="form-group">
17         <label for="email">Email:</label>
18         <input type="email" class="form-control" id="email" name="email" required>
19     </div>
20
21     <div class="form-group">
22         <label for="senha">Senha:</label>
23         <input type="password" class="form-control" id="senha" name="senha" required>
24     </div>
25
26     <button type="button" class="btn btn-success mt-3">Cadastrar</button>
27 </form>
28
29 {% endblock %}
```

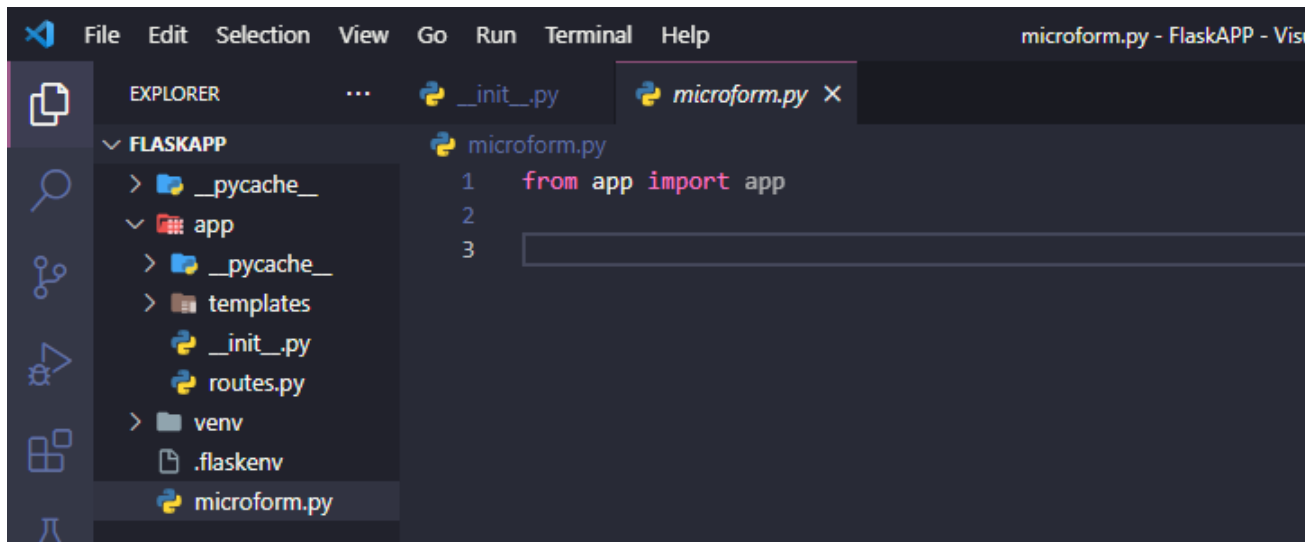
contato.html:



The screenshot shows the Visual Studio Code interface with the FlaskAPP project open. The Explorer sidebar on the left displays the project structure, including the templates folder which contains base.html, contato.html, and index.html. The contato.html file is selected and its content is visible in the editor. The file content is as follows:

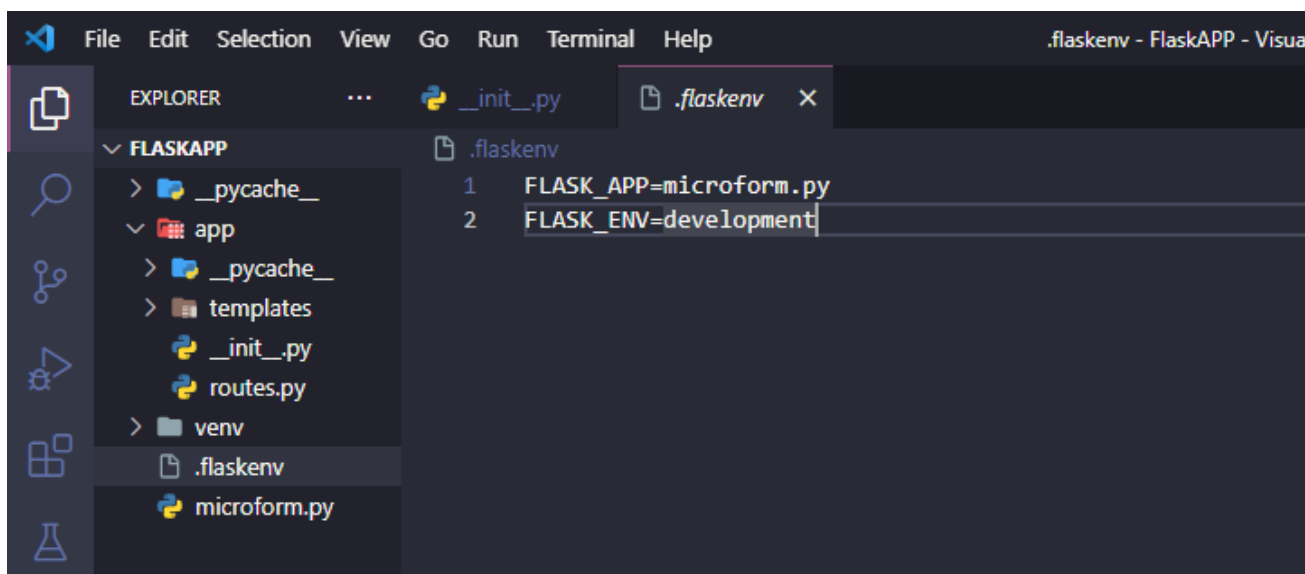
```
1 {% extends 'base.html' %}
2
3 {% block conteudo %}
4     <h2>Aqui estão as informações de contato</h2>
5     <p>
6         Telefone: 869999999999 <br>
7         Email: academystudy@gmail.com
8     </p>
9 {% endblock %}
```

- 6) **Execução:** Para facilitar a execução da aplicação pode-se criar um arquivo na pasta geral da aplicação, fora da pasta app, chamada microform, por exemplo, que também tem a instância de app e ainda criar outro arquivo .flaskenv que terá variáveis que facilitam a execução global da aplicação, mas antes disso no terminal execute o comando *pip install python-dotenv*. Assim, depois dos arquivos criados basta executar flask run no terminal para a aplicação funcionar.



The screenshot shows the Visual Studio Code interface. The Explorer view on the left displays the project structure for 'FLASKAPP', including folders like '__pycache__', 'app', 'templates', and 'venv', and files like '__init__.py', 'routes.py', '.flaskenv', and 'microform.py'. The editor view on the right shows the 'microform.py' file with the following code:

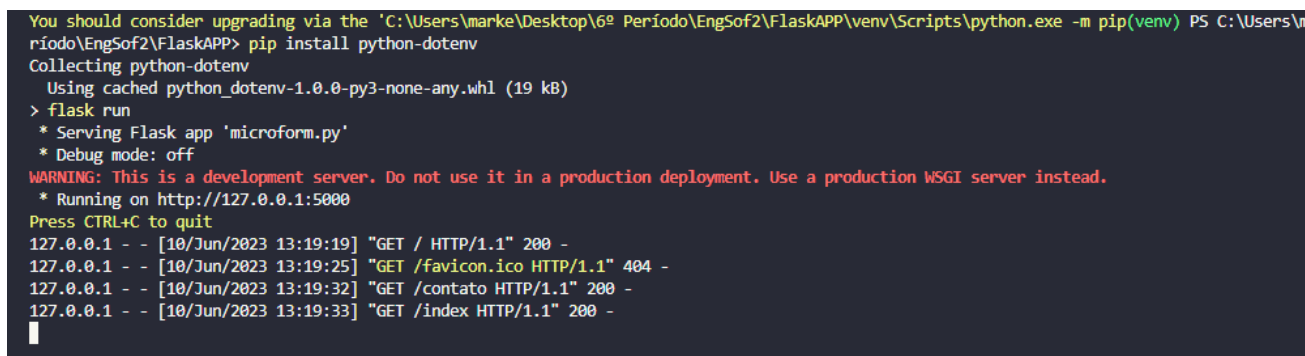
```
1 from app import app
2
3
```



The screenshot shows the Visual Studio Code interface with the '.flaskenv' file open in the editor. The Explorer view on the left shows the same project structure. The editor view on the right shows the following code in the '.flaskenv' file:

```
1 FLASK_APP=microform.py
2 FLASK_ENV=development
```

Por fim, depois do flask run no terminal a aplicação executará e o aplicativo está funcionando localmente na URL `http://127.0.0.1:5000/`, `127.0.0.1` é o IP que representa o `localhost` de sua máquina e `:5000` é o número da porta.



The screenshot shows a terminal window with the following output:

```
You should consider upgrading via the 'C:\Users\marke\Desktop\6º Período\EngSof2\FlaskAPP\venv\Scripts\python.exe -m pip(venv) PS C:\Users\marke\Desktop\6º Período\EngSof2\FlaskAPP> pip install python-dotenv
Collecting python-dotenv
  Using cached python_dotenv-1.0.0-py3-none-any.whl (19 kB)
> flask run
* Serving Flask app 'microform.py'
* 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 - - [10/Jun/2023 13:19:19] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2023 13:19:25] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [10/Jun/2023 13:19:32] "GET /contato HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2023 13:19:33] "GET /index HTTP/1.1" 200 -
```

Quando esse endereço for aberto o formulário será mostrado no navegador:

← ↻ 🏠 ⓘ 127.0.0.1:5000/index

[Página Inicial](#) [Contato](#)

Bem-vindo ao cadastro de usuários

Usuário:

Nome completo:

Email:

Senha:

[Cadastrar](#)

Projeto desenvolvido pela Academy

← ↻ 🏠 ⓘ 127.0.0.1:5000/contato

[Página Inicial](#) [Contato](#)

Aqui estão as informações de contato

Telefone: 869999999999

Email: academystudy@gmail.com

Projeto desenvolvido pela Academy

Para acesso ao código:

[markesley/Tutorial2FLASK: Tutorial de como integrar Flask com páginas HTML na criação de um formulário de cadastro. \(github.com\)](https://github.com/markesley/Tutorial2FLASK)