



Nome: João Victor Medeiros Rocha

RELATÓRIO SEMANA 09

Professor: Éder Alves de Moura

Uberlândia

2022



Roteiro de Atividades

Crie uma pasta em seu repositório GitHub, denominada ‘Semana09’. Nela você desenvolverá uma aplicação web utilizando o conjunto Python+Flask no Backend (server side).

1. Crie uma subpasta ‘web’ na pasta ‘Semana09’ e desenvolva as atividades que estão apresentadas no vídeo:

https://www.youtube.com/watch?v=Z1RJmh_OqeA

Este vídeo apresenta o desenvolvimento de um servidor web com o framework Flask, que utiliza a linguagem Python para a criação de páginas dinâmicas.

PREPARAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Para a realização da atividade, vamos utilizar a linguagem de programação Python na sua versão 3.8 em conjunto com o editor de texto Visual Studio Code (VSCode).

Para que possamos iniciar o desenvolvimento é necessário realizar a criação **de uma pasta chamada semana 09** pasta que irá armazenar todos os códigos do nosso projeto. Posteriormente será **criada uma pasta web** onde iremos armazenar todos os arquivos como por exemplo index.html , css, e o main.py que guardará o nosso servidor flask. Por fim como iremos criar um back-end utilizando o Flask para criar nossos end points , será necessário instalar a biblioteca do Flask e para isso vamos **utilizar o gerenciador de pacotes pip através do seguinte comando: pip install Flask para que possamos ter a biblioteca Flask** . Com a biblioteca em mãos é possível começar a criar nossas APIs e seus end points e dar início ao projeto proposto. Veja abaixo um exemplo:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + - [ ] [X] [ ] [ ] [ ] [X] [X]
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09> cd web
PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09\web> pip install Flask
```

Figura 1: Configuração do ambiente de desenvolvimento.



CODIFICAÇÃO E DESENVOLVIMENTO DO PROJETO

Realizado os passos do item anterior partimos para o desenvolvimento do nosso Back-End com Python para que tenhamos os end points prontos para poder lançar a nossa página web para este end point em Flask através de requisições HTTP do tipo: Get , Post , Put , Delete utilizando o modelo CRUD.

Sendo assim foi criado o arquivo app.py veja abaixo:

```
Terminal  Help  app.py - Untitled (Workspace) - Visual Studio Code

app.py web X

web > app.py

1 from flask import Flask, render_template, url_for, request, redirect
2 from flask_sqlalchemy import SQLAlchemy
3 from datetime import datetime
4
5 app = Flask(__name__)
6 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
7 db = SQLAlchemy(app)
8
9 class Todo(db.Model):
10     id = db.Column(db.Integer, primary_key=True)
11     content = db.Column(db.String(200), nullable=False)
12     date_created = db.Column(db.DateTime, default=datetime.utcnow)
13
14     def __repr__(self):
15         return '<Task %r>' % self.id
16
17
18 @app.route('/', methods=['POST', 'GET'])
19 def index():
20     if request.method == 'POST':
21         task_content = request.form['content']
22         new_task = Todo(content=task_content)
23
24     try:
```

Figura 2: Desenvolvimento do back-end com Flask



```
Terminal  Help  app.py - Untitled (Workspace) - Visual Studio Code

app.py web X
web > app.py

26         db.session.commit()
27         return redirect('/')
28     except:
29         return 'There was an issue adding your task'
30
31     else:
32         tasks = Todo.query.order_by(Todo.date_created).all()
33         return render_template('index.html', tasks=tasks)
34
35
36 @app.route('/delete/<int:id>')
37 def delete(id):
38     task_to_delete = Todo.query.get_or_404(id)
39
40     try:
41         db.session.delete(task_to_delete)
42         db.session.commit()
43         return redirect('/')
44     except:
45         return 'There was a problem deleting that task'
46
47 @app.route('/update/<int:id>', methods=['GET', 'POST'])
48 def update(id):
49     task = Todo.query.get_or_404(id)
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python Go Live Prettier

Figura 3: Desenvolvimento do back-end com Flask, segunda parte do código.



```
54     try:
55         db.session.commit()
56         return redirect('/')
57     except:
58         return 'There was an issue updating your task'
59
60     else:
61         return render_template('update.html', task=task)
62
63
64 if __name__ == "__main__":
65     app.run(debug=True)
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python Go Live Prettier

Figura 4: Desenvolvimento do back-end com Flask, terceira parte do código.

Feito o Back-end da nossa aplicação, agora o próximo passo é realizar a criação do nosso cliente que irá consumir de forma REST a APIs dos end points criados com Flask, e para tal iremos utilizar a antiga e conhecida linguagem de marcação HTML , além de utilizarmos a linguagem de estilização ou de folhas de estilo CSS3 que irá trazer para nosso site um “melhor visual”. Logo teremos os seguintes códigos:

- Códigos HTML:



```
1 {% extends 'base.html' %}
2
3 {% block head %}
4 <title>Task Master</title>
5 {% endblock %}
6
7 {% block body %}
8 <div class="content">
9     <h1 style="text-align: center">Task Master</h1>
10     {% if tasks|length <= 1 %}
11     <h4 style="text-align: center">There are no tasks. Create one below</h4>
12     {% else %}
13     <table>
14         <tr>
15             <th>Task</th>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + - [] [X] ^ X

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell <https://aka.ms/pscore6>

PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09> |

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML Go Live Prettier [] [X] [] [X]

Figura 5: Desenvolvimento do front-end com HTML, primeira parte do código index.html.



```
14      </tr>
15      <th>Task</th>
16      <th>Added</th>
17      <th>Actions</th>
18    </tr>
19    {% for task in tasks %}
20      <tr>
21        <td>{{ task.content }}</td>
22        <td>{{ task.date_created.date() }}</td>
23        <td>
24          <a href="/delete/{{task.id}}">Delete</a>
25          <br>
26          <a href="/update/{{task.id}}">Update</a>
27        </td>
28      </tr>
29    {% endfor %}
30  </table>
31  {% endif %}
32
33  <div class="form">
34    <form action="/" method="POST">
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + - [] [X]

PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09> []

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML Go Live Prettier [] []

Figura 6: Desenvolvimento do front-end com HTML, segunda parte do código index.html.

```
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     <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css')}}">
8     {% block head %}{% endblock %}
9 </head>
10 <body>
11     {% block body %}{% endblock %}
12 </body>
13 </html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + - [] [X] [Y] [Z]

PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09>

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML Go Live Prettier [] [X] [Y] [Z]

Figura 7: Desenvolvimento do front-end com HTML, código base.html.



```
1  {% extends 'base.html' %}
2
3  {% block head %}
4  <title>Task Master</title>
5  {% endblock %}
6
7  {% block body %}
8  <div class="content">
9      <h1 style="text-align: center">Update Task</h1>
10
11      <div class="form">
12          <form action="/update/{{task.id}}" method="POST">
13              <input type="text" name="content" id="content" value="{{task.content}}">
14              <input type="submit" value="Update">
15          </form>
16      </div>
17  </div>
18  {% endblock %}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + - [] [X] [Y] [Z]

PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09> []

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML Go Live Prettier [] [] []

Figura 8: Desenvolvimento do front-end com HTML, código update.html.

- Código CSS:



```
16 table {
17     border-collapse: collapse;
18     width: 100%;
19 }
20
21 th {
22     height: 30px;
23 }
24
25 td {
26     text-align: center;
27     padding: 5px;
28 }
29
30 .form {
31     margin-top: 20px;
32 }
33
34 #content {
35     width: 70%;
36 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + - [] [X] [Y]

PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09> []

Ln 1, Col 1 Spaces: 4 UTF-8 LF CSS Go Live Prettier [] []

Figura 10: Desenvolvimento do front-end com CSS, código style.css parte 2.

Sendo assim, com tudo criado rodamos a aplicação e obtivemos como resultado a seguinte página HTML onde é possível cadastrar tarefas ou atividades e as remover veja abaixo uma imagem que mostra como ficou o projeto:



Task Master

Task	Added	Actions
do the dishes	2019-05-20	Delete Update

Figura 11: Tela final da aplicação.