

Prof. Éder Alves de Moura Semana 09 – Servidor Web com Flask

Nome: João Victor Medeiros Rocha

## RELATÓRIO SEMANA 09

Professor: Éder Alves de Moura

Uberlândia

2022

Prof. Éder Alves de Moura Semana og – Servidor Web com Flask



#### 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 'Semanao9' 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 bilbioteca em mãos é possivel começar a criar nossas APIs e seus end points e dar ínicio ao projeto proposto. Veja abaixo um exemplo:

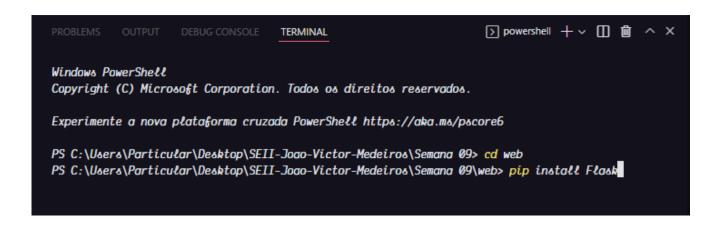


Figura 1: Configuração do ambiente de desenvolvimento.

Prof. Éder Alves de Moura Semana o9 – Servidor Web com Flask



### 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
                                                                                        П ...

 web > app.py

          from flask import Flask, render_template, url_for, request, redirect
          from flask_sqlalchemy import SQLAlchemy
          from datetime import datetime
          app = Flask(__name__)
          app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
          db = SQLA(chemy(app)
         class Todo(db.Model):
              id = db.Column(db.Integer, primary_key=True)
              content = db.Column(db.String(200), nullable=False)
              date_created = db.Column(db.DateTime, default=datetime.utcnow)
              def __repr__(self):
                  return '<Task %r>' % self.id
          @app.route('/', methods=['POST', 'GET'])
          def index():
              if request.method = 'POST':
                  task_content = request.form['content']
                  new_task = Todo(content=task_content)
                  try:
                                 Ln 1, Col 1 Spaces: 4 UTF-8 LF Python @ Go Live ⊘ Prettier 🔊 🚨
```

Figura 2: Desenvolvimento do back-end com Flask

#### Universidade Federal de Uberlândia

#### Engenharia de Controle e Automação / Engenharia Mecatrônica Sistemas Embarcados II/Sistemas Digitais para Mecatrônica



```
Terminal Help
                   app.py - Untitled (Workspace) - Visual Studio Code
                                                           Ⅲ ...
 e app.py web X
  web >  app.py
                    db.session.commit()
                    return redirect('/')
                 except:
                    return 'There was an issue adding your task'
             else:
                 tasks = Todo.query.order_by(Todo.date_created).all()
                 return render_template('index.html', tasks=tasks)
         @app.route('/delete/<int:id>')
         def delete(id):
             task_to_delete = Todo.query.get_or_404(id)
             try:
                db.session.delete(task_to_delete)
                db.session.commit()
                 return redirect('/')
             except:
                 return 'There was a problem deleting that task'
         @app.route('/update/<int:id>', methods=['GET', 'POST'])
         def update(id):
             task = Todo.query.get_or_404(id)
```

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



Prof. Éder Alves de Moura Semana 09 – Servidor Web com Flask

```
try:

db.session.commit()

return redirect('/')

except:

return 'There was an issue updating your task'

else:

return render_template('update.html', task=task)

if __name__ = "__main__":

app.run(debug=True)

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python @ Go Live ② Prettier R Q
```

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:



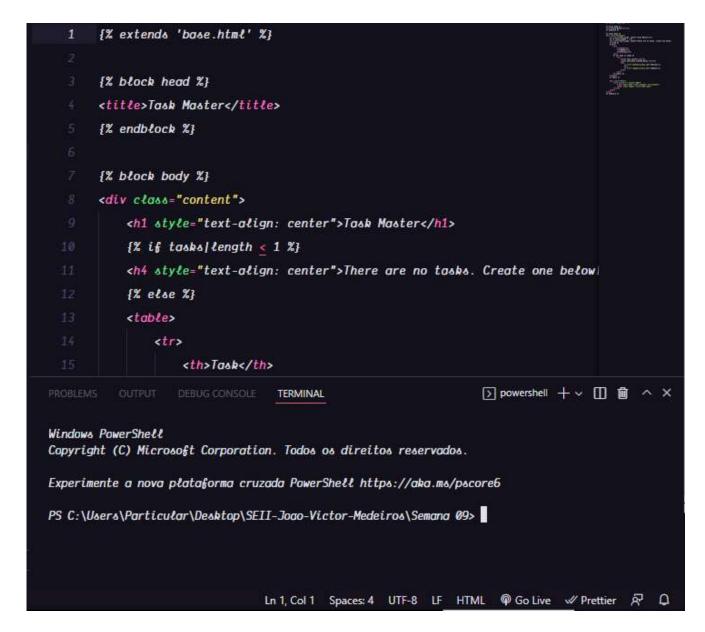


Figura 5: Desenvolvimento do front-end com HTML, primeira parte do código index.html.



```
Task
               Added
               Actions
            {% for task in tasks %}
               {{td>{{ task.content }}
                   {{ task.date_created.date() }}
                   <a href="/delete/{{task.id}}">Delete</a>
                      <br>
                      <a href="/update/{{task.id}}">Update</a>
                   {% endfor %}
         {% endif %}
         <div class="form">
            <form action="/" method="POST">
                                                   TERMINAL
PS C:\Users\Particular\Desktop\SEII-Joao-Victor-Medeiros\Semana 09> 🛚
                         Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML @ Go Live & Prettier R Q
```

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



```
<1DOCTYPE html>
       <html lang="en">
       <head>
           <meta charset="UTF-8">
           <meta name="viewport" content="width=device-width, initial-scale=1.(</pre>
           <meta http-equiv="X-UA-Compatible" content="ie=edge">
           <link rel="stylesheet" href="{{ url_for('static', filename='css/mair</pre>
           {% block head %}{% endblock %}
       </head>
      <body>
           {% block body %}{% endblock %}
       </body>
      </html>
                                                              TERMINAL
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 7: Desenvolvimento do front-end com HTML, código base.html.



Prof. Éder Alves de Moura Semana 09 – Servidor Web com Flask

```
{% extends 'base.html' %}
       {% block head %}
      <title>Task Master</title>
       {% endblock %}
       {% block body %}
       <div class="content">
          <h1 style="text-align: center">Update Task</h1>
          <div class="form">
               <form action="/update/{{task.id}}" method="POST">
                   <input type="text" name="content" id="content" value="{{task</pre>
                   <input type="submit" value="Update">
               </form>
          </div>
       </div>
       {% endblock %}
                                                             TERMINAL
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:



```
body, html {
          margin: 0;
          font-family: sans-serif;
          background-color: | lightblue;
      1
      .content [
          margin: 0 auto;
          width: 400px;
      3
       table, td, th [
          border: 1px solid = #aaa;
      1
       table [
          border-collapse: collapse;
          width: 100%;
                               TERMINAL
                                                           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 9: Desenvolvimento do front-end com CSS, código style.css parte 1.



Prof. Éder Alves de Moura Semana og – Servidor Web com Flask

```
table {
           border-collapse: collapse;
           width: 100%;
       th f
           height: 30px;
       td [
           text-align: center;
           padding: 5px;
       .form {
           margin-top: 20px;
       #content {
           width: 70%;
                                  TERMINAL

    powershell + ∨ Ⅲ 前 ^ ×

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:

Prof. Éder Alves de Moura Semana 09 – Servidor Web com Flask



# **Task Master**

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

Figura 11: Tela finalda aplicação.