CAPTURAS DEL CODIGO DEL **SERVIDOR PYTHON.**

• Importamos los paquetes que utilizaremos

if not os.path.exists(UPLOAD FOLDER):

os.makedirs(UPLOAD FOLDER)

• Tambien utilizamos las variables de entorno

```
SemiT4 201314439 > Servidor-Python > → server.py > .
      Click here to ask Blackbox to help you code faster
      from flask import Flask, request, jsonify
      from dotenv import load dotenv
      import boto3
      import base64
      import os
      app = Flask(__name___
     # Cargar variables de entorno desde el archivo .env
      load dotenv(
11
                                                                                          SemiT4_201314439 > Servidor-Python > ♥ server.py > ...
      AWS ACCESS KEY ID = os.getenv('AWS ACCESS KEY ID')
      AWS SECRET ACCESS KEY = os.getenv('AWS SECRET ACCESS KEY')
      AWS_REGION = os.getenv('AWS_REGION'
17 # Configurar el cliente de Rekognition
     rekognition_client = boto3.client('rekognition',
                                        aws access key id=AWS ACCESS KEY ID,
                                        aws secret access key=AWS SECRET ACCESS KEY,
                                        region name=AWS REGION)
     # Ruta al directorio de subidas
      UPLOAD FOLDER = './uploads'
```

- Cargamos las imagenes al servidor y despues sacamos su base 64
- utilizamos boto3 para el cliente de aws

```
@app.route('/tarea4-201314439', methods=['POST'])
def procesar imagen():
     # Verificar si se proporcionaron ambos archivos
    if 'file1' not in request.files or 'file2' not in request.files:
        return jsonify({'error': 'Se deben proporcionar dos archivos'}), 400
    file1 = request.files['file1']
    file2 = request.files['file2']
    # Guardar las imágenes en el servidor
   filename1 = file1.filename
   filename2 = file2.filename
   file path1 = os.path.join(UPLOAD FOLDER, filename1
   file path2 = os.path.join(UPLOAD FOLDER, filename2
   file1.save(file path1)
   file2.save(file path2)
   # Leer las imágenes en formato base64
   with open(file_path1, "rb") as image_file1:
        encoded_string1 = base64.b64encode(image_file1.read()).decode()
   with open(file path2, "rb") as image file2:
        encoded string2 = base64.b64encode(image file2.read()).decode()
   # Eliminar las imágenes del servidor
   os.remove(file path1)
   os.remove(file_path2)
```

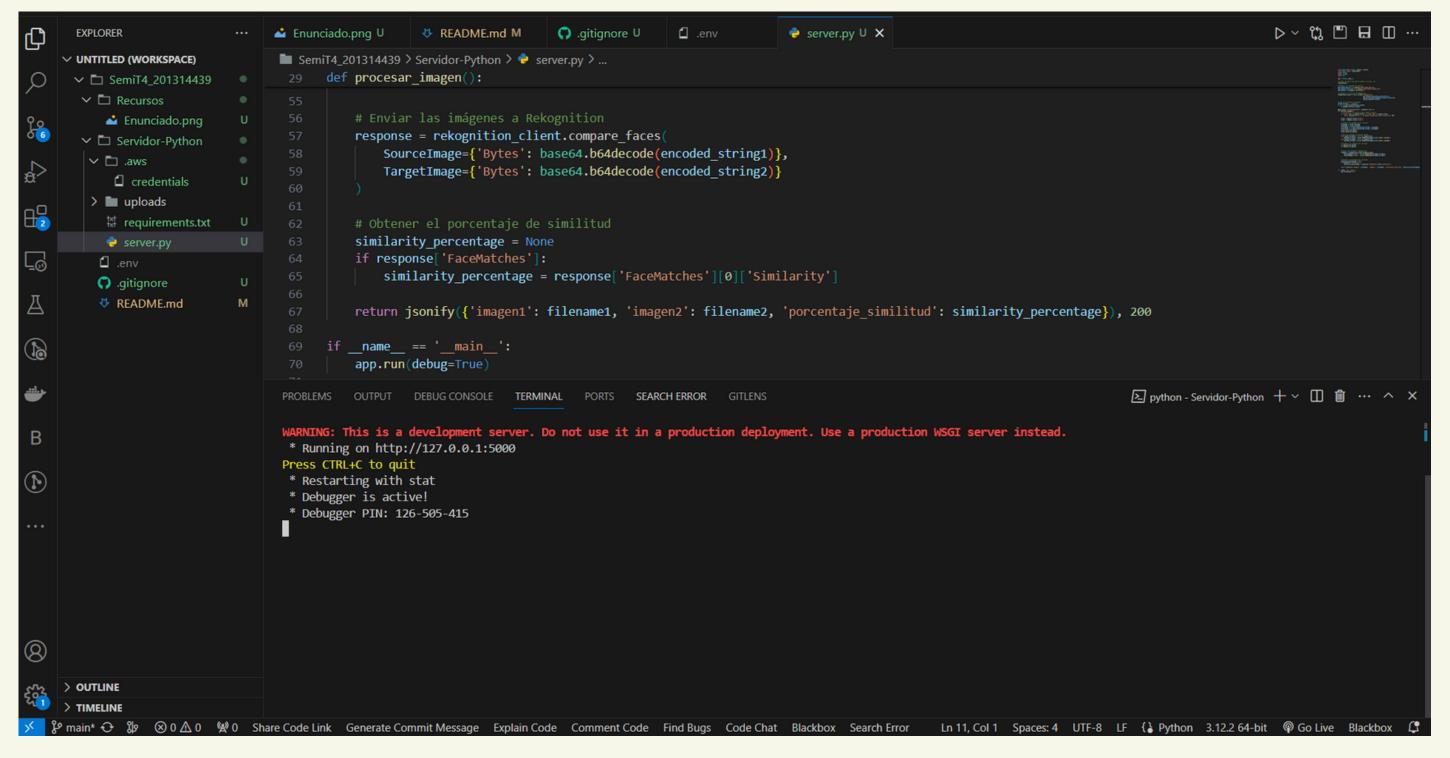
- Recibimos la respuesta y parsemos los campos que
- Devolvemos los campos en forma de json

```
■ SemiT4_201314439 > Servidor-Python > ♦ server.py >
 29 def procesar_imagen():
          response = rekognition client.compare faces(
              SourceImage={'Bytes': base64.b64decode(encoded_string1)},
              TargetImage={'Bytes': base64.b64decode(encoded_string2)}
          similarity_percentage = None
          if response['FaceMatches']:
             similarity_percentage = response['FaceMatches'][0]['Similarity']
          return jsonify({'imagen1': filename1, 'imagen2': filename2, 'porcentaje_similitud': similarity_percentage}), 200
      if __name__ == '__main__':
          app.run(debug=True)
```

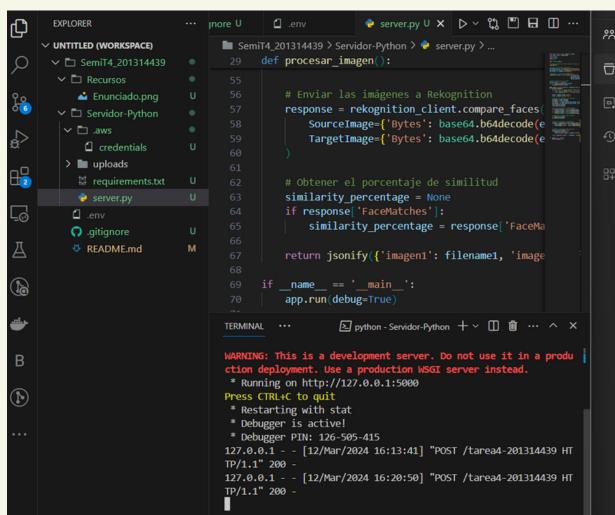
CAPTURA DEL SERVIDOR CORRIENDO.

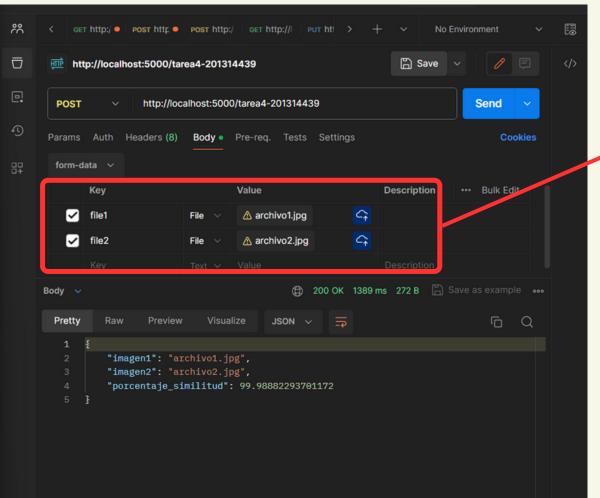
• vemos que el puerto esta en el la siguiente ruta

http://localhost:5000



CAPTURA DEL JSON INGRESADO Y RUTA CREADA (POSTMAN).





• Se envia la imagen como tal no como base 64 si no se hace la conversion dentro del servidor como podemos ver en esta area

• Nos devuelve el porcentaje de simulitud en un 99.98 %

• hacemos uso del endpoint:

http://localhost:5000/tarea4-201314439

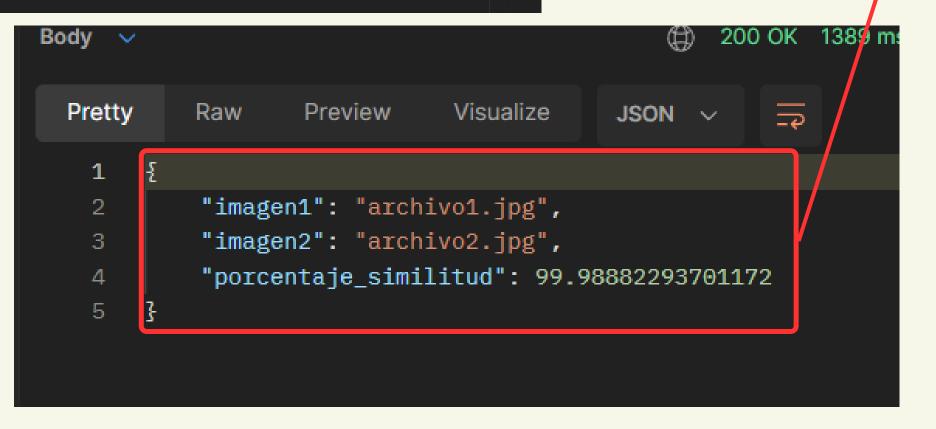
lmagen es que se cargan

archivo1.png

archivo2.png



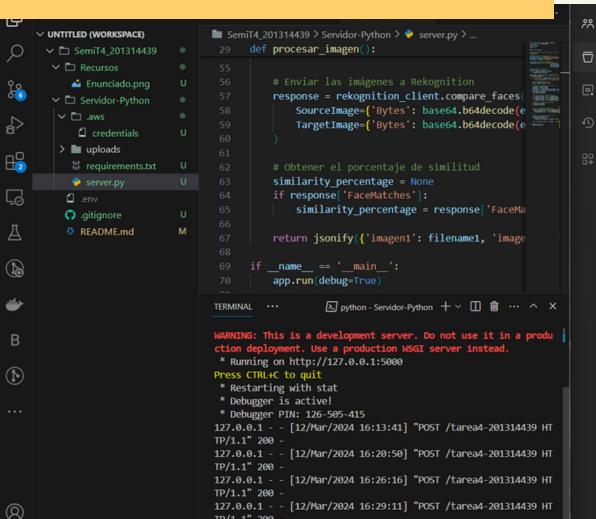




CAPTURA DEL JSON INGRESADO Y RUTA CREADA (POSTMAN).

⟨ GET http:/ ● POST http:/ GET http:// PUT htt >

"porcentaje_similitud": null



POST V http://localhost:5000/tarea4-201314439

Params Auth Headers (8) Body Pre-req. Tests Settings

Cookies

Form-data V

Key Value Description

Key Text V Value Description

File A archivo3.jpg

File A archivo4.jpg

Key Text V Value Description

Body V B 200 OK 1263 ms 258 B Save as example

**Save A preview Visualize JSON A pre-req. Tests Settings

Pretty Raw Preview Visualize JSON A pre-req. Tests Settings

**Imagen2*: *archivo4.jpg*,

**imagen2*: *archivo4.jpg*,

**imagen2*: *archivo4.jpg*,

**imagen2*: *archivo4.jpg*,

• Se envia la imagen como tal no como base 64 si no se hace la conversion dentro del servidor como podemos ver en esta area

 Nos devuelve el porcentaje de simulitud en un nulo ya que no se parecen

• hacemos uso del endpoint:

http://localhost:5000/tarea4-201314439

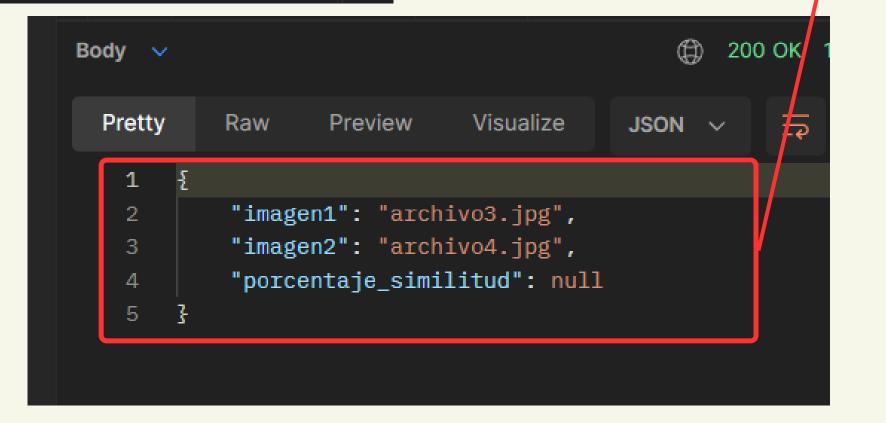
Imagen es que se cargan

archivo3.png



archivo4.png





LINK GITHUB DE CODIGO EJECUTADO: HTTPS://GITHUB.COM

<u>HTTPS://GITHUB.CO</u> /HENRRYBRAN-

HUB/SEMIT4_2013144

<u> 39</u>