**Carpeta con API para ser contenerizada y puesta en marcha en la nube**

En la imagen se visualizan las carpetas y los archivos necesarios para el funcionamiento de la API

Comparto ubicación del repositorio https://github.com/JuanEncizo/Landing\_Page

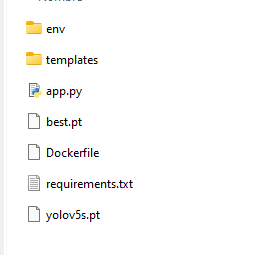


Ilustración Carpetas y Archivos

Se hicieron las pruebas con Python 3.10.3

En requirements.txt están todas las librerías que se deben instalar.

App.py es el programa que se corre y deja expuesta la dirección IP en la imagen

import argparse

import io

from PIL import Image

import base64

from io import BytesIO

import torch

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

DETECTION\_URL = "/detect"

@app.route(DETECTION\_URL, methods=["POST"])

def predict():

    if not request.method == "POST":

        return

    if request.files.get("image"):

        image\_file = request.files["image"]

        image\_bytes = image\_file.read()

        img = Image.open(io.BytesIO(image\_bytes))

        results = model(img, size=640)

        #results.show()

        data = results.pandas().xyxy[0]#.to\_json(orient="split")

        results.imgs # array of original images (as np array) passed to model for inference

        results.render()  # updates results.imgs with boxes and labels

        for img in results.imgs:

          buffered = BytesIO()

          im\_base64 = Image.fromarray(img)

          im\_base64.save(buffered, format="JPEG")

          #print(base64.b64encode(buffered.getvalue()).decode('utf-8'))  # base64 encoded image with results

          #

        return str(data[['name']])+'\n'+ base64.b64encode(buffered.getvalue()).decode('utf-8')

@app.route('/none')

def none():

    return render\_template('index.html')

if \_\_name\_\_ == "\_\_main\_\_":

    parser = argparse.ArgumentParser(description="Flask api exposing yolov5 model")

    parser.add\_argument("--port", default=5000, type=int, help="port number")

    args = parser.parse\_args()

    #model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

    model = torch.hub.load('ultralytics/yolov5', 'custom', path='best.pt', force\_reload=True)

    model.conf = 0.7

    #model = torch.hub.load(

       # "ultralytics/yolov5", "yolov5s", pretrained=True, force\_reload=True

   # )#.autoshape()  # force\_reload = recache latest code

    model.eval()

    app.run(host="0.0.0.0", port=4000, debug=True)  # debug=True causes Restarting with stat

Al correr arroja esta la IP marcada con lo que se han hecho pruebas

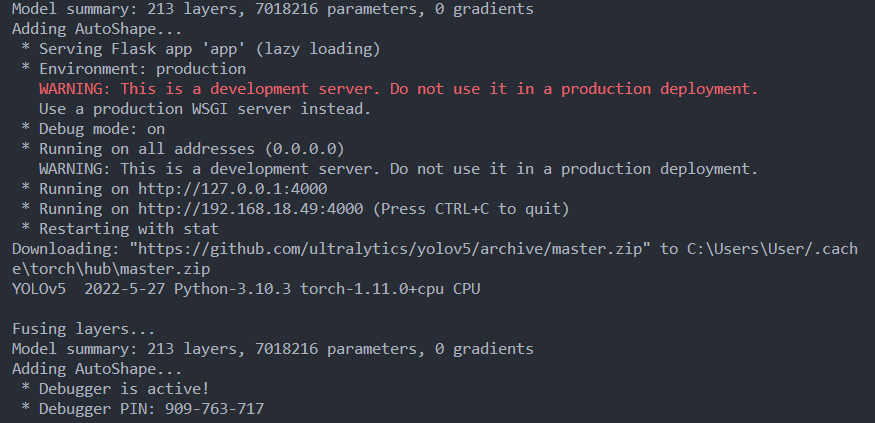
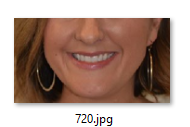
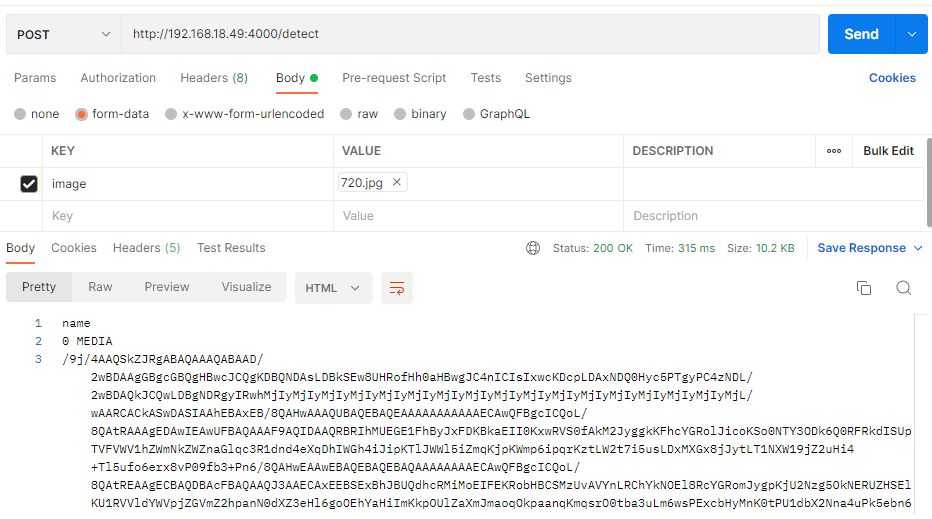


Ilustración IP local

**Petición Con Postman**



Los dos resultados de interés son: **el texto que aparece subrayado** y la cadena de texto de la detección de la imagen en base64.

La cadena de texto que se genera **“/9j/4AAQSkZ…”** corresponde a la detección. En la imagen se puede ver la imagen decodificada



**Contenerizacion de prueba**

Se utilizo Docker para hacer una prueba de contenerizado.

En la carpeta de archivos se encuentra el archivo Dockerfile con las siguientes instrucciones

FROM python:3.10.3

RUN python -m pip install --upgrade pip

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

RUN apt-get update && apt-get install libgl1 -y

CMD [ "python", "app.py" ]

Se creo el contenedor con la siguiente línea: **docker build -t sonrisas .**

Despues de la creación de la imagen , se corrio esta misma y con el siguiente comando:

**docker run -it --publish 7000:4000 sonrisas**

Finalmente se hicieron las mismas pruebas que en local, pero en este caso con el puerto 7000, funcionando correctamente.