



UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD CIENCIAS DE LA INGENIERÍA

INGENIERÍA EN SOFTWARE

INTEGRANTES:

- COELLO CASTILLO RAUL STEVEN
- HERRERA SILVA ALEXANDER JAVIER
- MOLINA MOSQUERA VICTOR JOSE
- VERA ESPINALES WILLIAM DENILSON
- VERA MACIAS JOHN KLEINER

HERRAMIENTAS DE PROGRAMACIÓN

DOCENTE: ING. PORTILLA OLVERA GILBERTO ELIAS

CREAR EJECUTABLE EN PYTHON

Integrantes del grupo:

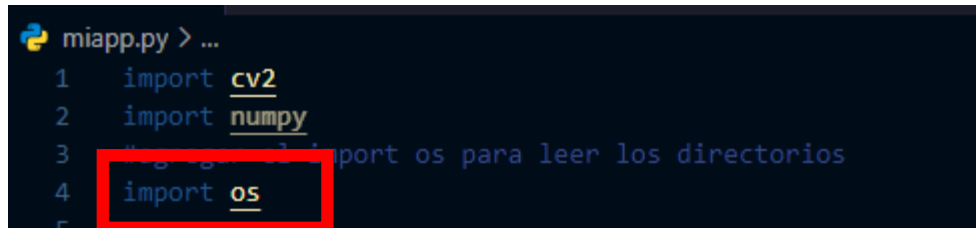
- Coello Castillo Raul Steven
- Herrera Silva Alexander Javier
- Molina Mosquera Victor Jose
- Vera Espinales William Denilson
- Vera Macias John Kleiner

Configuración de la computadora donde se desarrolló la aplicación:

- Laptop Marca HP, modelo 15-dy2xxx.
- Procesador: 11 th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz.
- 18gb Memoria RAM 2.400 MT/s ddr4 (2 módulos de 8 y 4 físicos) y (6 Virtuales).
- 250gb disco interno NVMe KBG40ZNV256G KIOXIA.
- Versión de Python 3.12.2
- Versión de OpenCV 4.10.0
- Versión de pyinstaller 6.9.0
- Version de numpy 2.0.0
- IDE utilizado: Visual Studio Code versión 1.91.1
- Microsoft Office Professional Plus 2016.

Pasos a seguir para crear el ejecutable de la aplicación para reconocer rostros

1. Agregar la librería "os" en la aplicación:



```
miapp.py > ...
1  import cv2
2  import numpy
3  # Agregar import os para leer los directorios
4  import os
```

2. Ahora hay que agregar una variable "haarcascade_path" que servirá para guardar la ruta donde se encuentra el archivo XML, esto va a ser importante para la construcción del ejecutable ya que esta ruta indica que este archivo siempre se encontrara en la misma ruta donde se encuentra la aplicación. Cabe decir que si se intenta ejecutar el proyecto con el comando Python va a dar error y no va a leer la ruta ya que este paso solo se agrega para la construcción del ejecutable.

```
#modificar la linea para anadir el directorio del xml
haarcascade_path = os.path.join(os.path.dirname(__file__),
'haarcascade_frontalface_default.xml')
```

La siguiente linea se modifica la variable "faceClassif" anteriormente estaba definida de la siguiente manera:

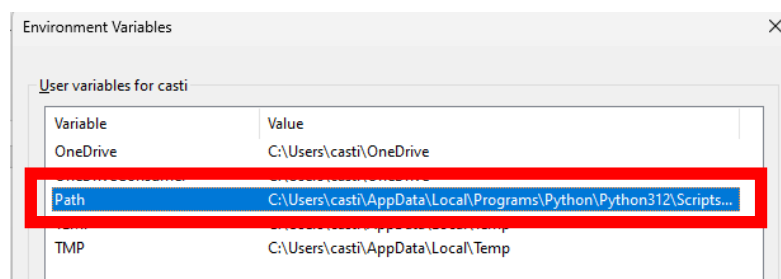
```
#faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
```

Ahora se tiene que definir simplemente pasándole la variable “haarcascade_path” a la función “cv2.CascadeClassifier” como parámetro el cual que contiene la ruta del XML, debe quedar de la siguiente manera:

```
#ahora el faceClassif
faceClassif = cv2.CascadeClassifier(haarcascade_path)
```

El resto del código sigue sin cambios.

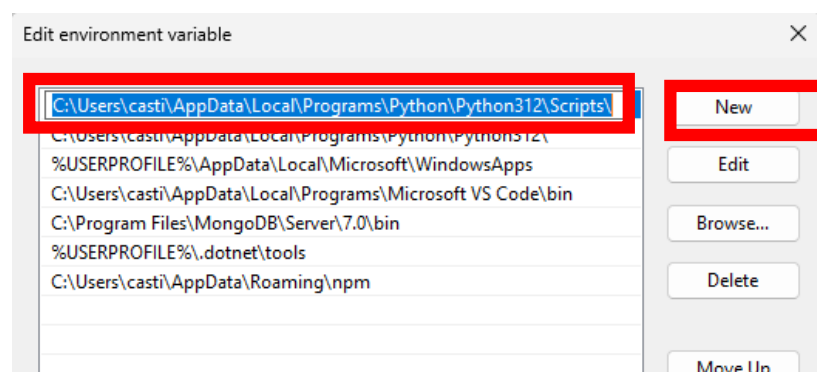
3. En las variables de entorno de Windows hay que verificar si existe una requerida para Python. Para verificarla hay que entrar en “Path” dentro de las variables de entorno.



Verificar si existe esta ruta dentro del “Path”:

C:\Users\casti\AppData\Local\Programs\Python\Python312\Scripts

Si no existe entonces agregarla dando click en el botón New. Cabe decir que la ruta debe tener su usuario, en este caso “casti” en el usuario actual donde se realiza la configuración del proyecto.



4. Instalar Pyinstaller en caso de no tenerlo instalado con el siguiente comando en el cmd:

```
pip install pyinstaller
```

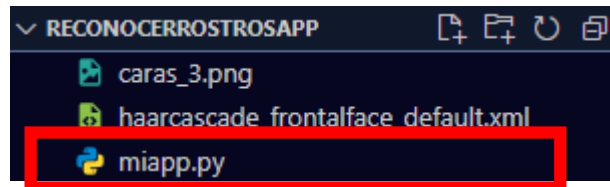
En caso de tenerlo instalado se puede omitir este paso. Para verificar si está instalado basta con ejecutar el siguiente comando en el cmd para saber la versión:

```
pyinstaller --version
```

Debe de dar como resultado lo siguiente:

```
C:\Users\casti>pyinstaller --version
6.9.0
```

5. Verificar la estructura del proyecto para identificar los archivos que no forman parte de Python como imágenes o videos ya que estos no se empaquetaran en el ejecutable. En este caso el xml será empaquetado mediante el comando "add-data".



6. Para empaquetar la aplicación hay que ejecutar un comando, recordar que el cmd tiene que tener la ubicación del proyecto. Ese comando creara un ejecutable de un solo archivo, es decir, la aplicación solo será el .exe que contendrá todos los archivos .py incluido el xml.

```
pyinstaller --onefile --add-data "haarcascade_frontalface_default.xml;."
miapp.py
```

De la siguiente manera tiene que estar el cmd:

Ruta del proyecto	Comando
PROBLEM PS C:\Users\casti\Documents\Universidad\8vo semestre\HERRAMIENTAS DE PROGRAMACIÓN\ReconocerRostrosApp\nv]	pyinstaller --onefile --add-data "haarcascade_frontalface_default.xml;." miapp.

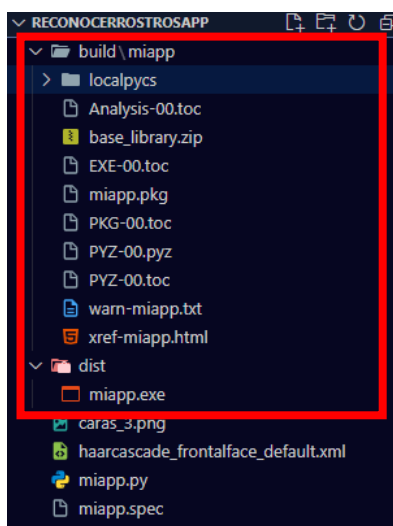
7. Una vez ejecutado el comando se comenzará a construir un ejecutable de la aplicación, solo hay que esperar que este proceso termine. Este proceso puede tardar dependiendo del tamaño de la aplicación y las librerías usadas ya que estas van a ser empaquetadas para que la aplicación pueda ser ejecutada en cualquier computadora sin necesidad de tener Python instalado.

```
321 INFO: PyInstaller: 6.9.0, contrib hooks: 2024.7
321 INFO: Python: 3.12.2
399 INFO: Platform: Windows-11-10.0.22631-SP0
399 INFO: Python environment: C:\Users\casti\AppData\Local\Programs\Python\Python312
400 INFO: wrote C:\Users\casti\Documents\Universidad\8vo semestre\HERRAMIENTAS DE PROGRAMACIÓN\ReconocerRostrosApp\miapp.spec
412 INFO: Module search paths (PYTHONPATH):
['C:\Users\casti\AppData\Local\Programs\Python\Python312\Scripts\pyinstaller.exe',
'C:\Users\casti\AppData\Local\Programs\Python\Python312\python312.zip',
'C:\Users\casti\AppData\Local\Programs\Python\Python312\DLLs',
'C:\Users\casti\AppData\Local\Programs\Python\Python312\Lib',
'C:\Users\casti\AppData\Local\Programs\Python\Python312',
'C:\Users\casti\AppData\Local\Programs\Python\Python312\Lib\site-packages',
'C:\Users\casti\AppData\Local\Programs\Python\Python312\Lib\site-packages\setuptools\_vendor',
'C:\Users\casti\Documents\Universidad\8vo semestre\HERRAMIENTAS DE '
'PROGRAMACIÓN\ReconocerRostrosApp']
733 INFO: Appending 'datas' from .spec
733 INFO: checking Analysis
734 INFO: Building Analysis because Analysis-00.toc is non existent
734 INFO: Running Analysis Analysis-00.toc
734 INFO: Target bytecode optimization level: 0
735 INFO: Initializing module dependency graph...
737 INFO: Caching module graph hooks...
```

La siguiente línea del cmd nos indica que se ha terminado de construir la aplicación:

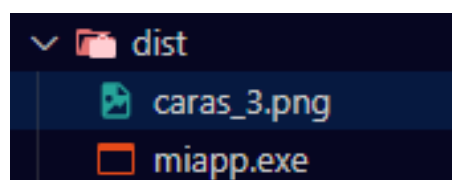
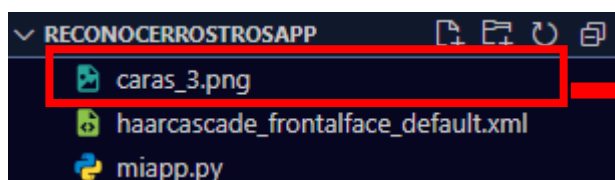
```
29261 INFO: Copying icon to EXE
29306 INFO: Copying 0 resources to EXE
29306 INFO: Embedding manifest in EXE
29356 INFO: Appending PKG archive to EXE
29400 INFO: Patching EXE headers
29839 INFO: Building EXE from EXE-00.toc completed successfully.
PS C:\Users\CASTI\Documents\UNIVERSIDAD\8VO SEMESTRE\HERRAMIENTAS DE PROGRAMACION\reconocerRostrosApp>
```

- Una vez finalizado exitosamente la construcción del proyecto la estructura del proyecto quedará de la siguiente manera:

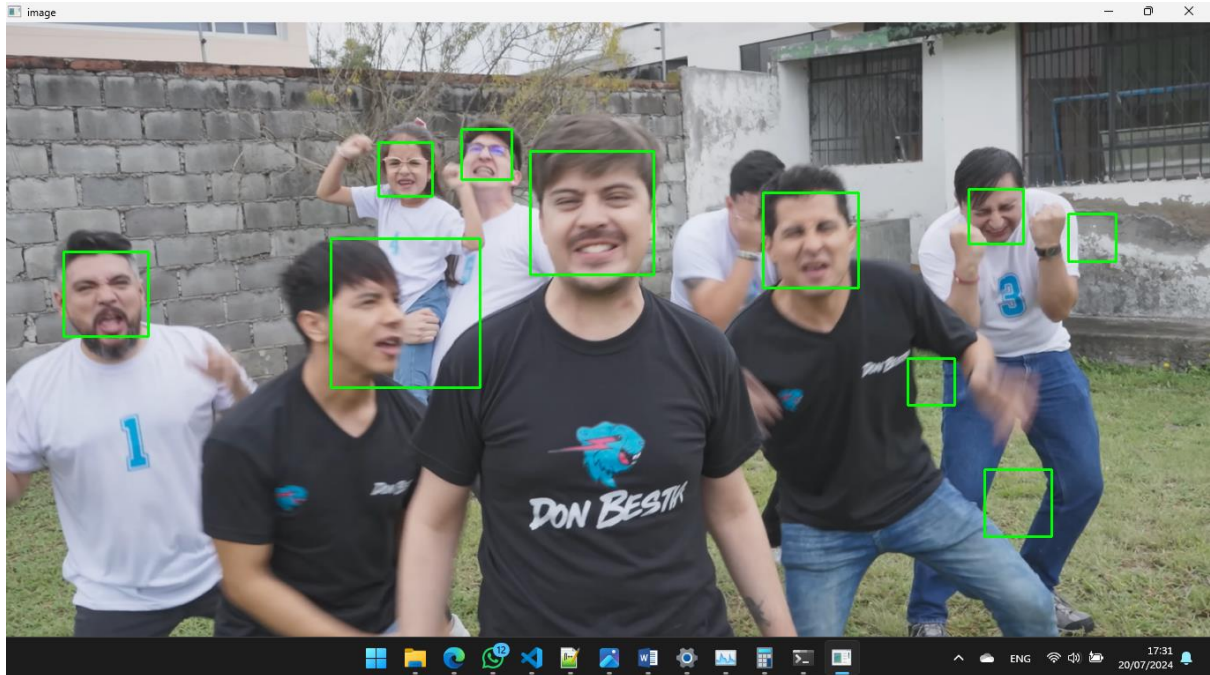


De las carpetas creadas solo nos enfocaremos en la siguiente “dist” la cual contiene la aplicación empaquetada. La carpeta “build” es utilizada durante el proceso de construcción del ejecutable, contiene archivos temporales y de trabajo que son generados por PyInstaller.

- Ahora los archivos que anteriormente se identificaron como externos, es decir, que no tienen terminación en .py como imágenes o videos se tienen que copiar y pegar dentro de la carpeta “dist” con los mismos nombres y en caso de estar ordenados en carpetas pues estas también tienen que ser copiadas y pegadas.



10. Realizados todos estos pasos la aplicación debe de ejecutar sin problemas y sin la necesidad de tener instalado Python o las librerías.



Realizar pruebas en otra computadora sin Python instalado

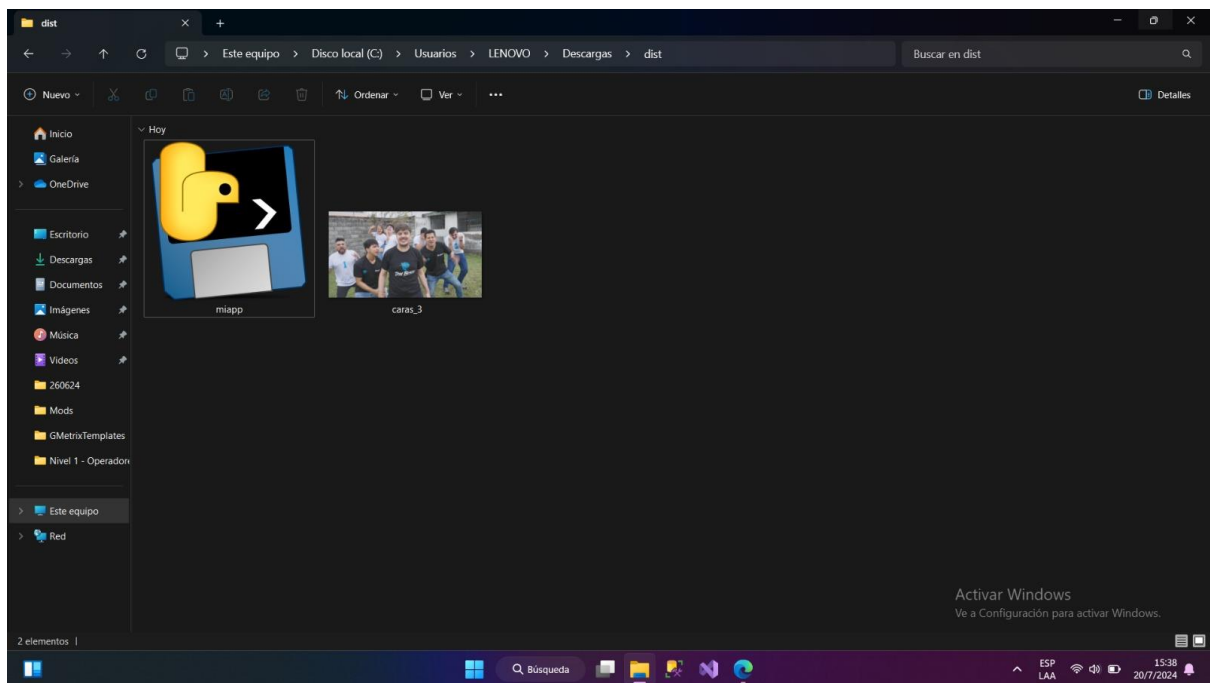
Computadora a realizar las pruebas, marca Lenovo:



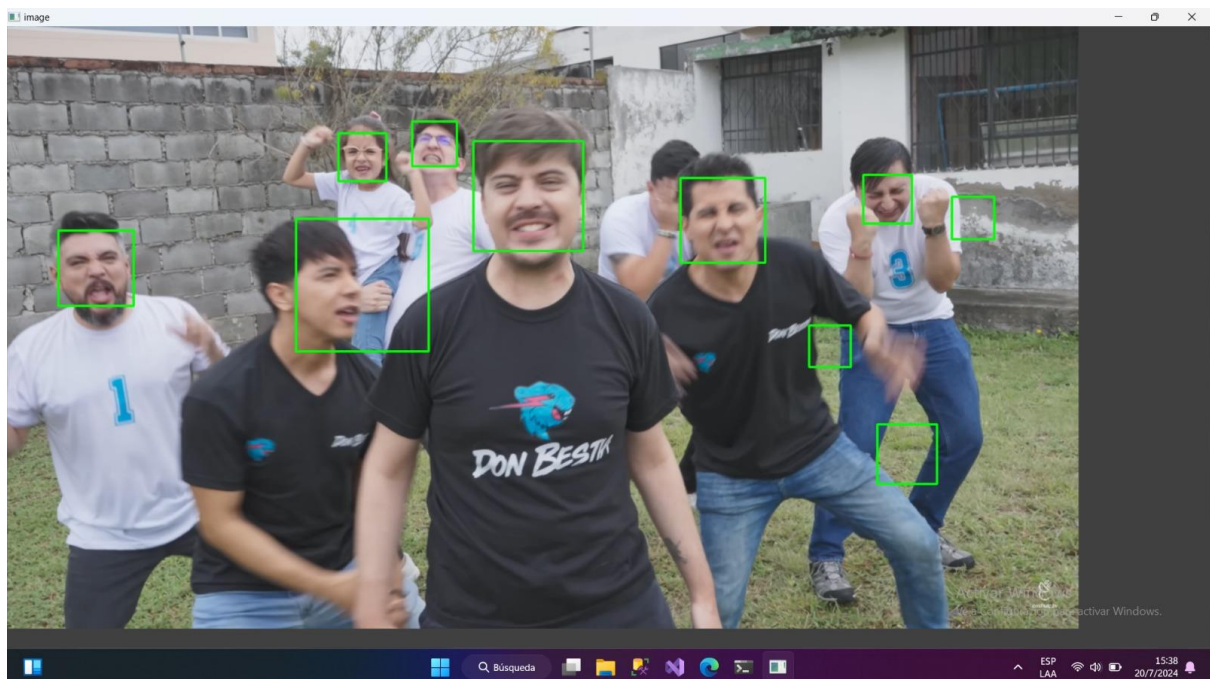
Características de la computadora donde se realizaron las pruebas:

- Procesador: 11 th Gen Intel(R) Core(TM) i3-1 1 15G4 @ 3.00GHz.
- 12gb Memoria RAM 2.400 MT/s ddr4 (2 módulos de 8 y 4 físicos)
- 250gb disco interno NVMe SKHynix_HFS256GEJ4X112N.
- Sin Python.
- Sin OpenCV.
- Sin pyinstaller.
- Sin numpy.

Carpeta con el contenido de la aplicación (ejecutable + imagen):



Ejecución del proyecto:



Repositorio de GitHub:

[ripderek/ReconocimientoRostrosPyhtonEjec \(github.com\)](https://github.com/ripderek/ReconocimientoRostrosPyhtonEjec)