**FACULTAD DE INGENIERIA**



**RECONONOCIMIENTO FACIAL COMO METODO DE REGISTRO PARA EMPRESAS**

**INTELIGENCIA ARTIFICIAL I**

**INTEGRANTES**

|  |  |
| --- | --- |
| **JHON BEIMAR CRUZ MONTAÑO** | **100** |
| **LIDER MORENO PADILLA** | **100** |
| **DANIEL ALBERTO PAZ CAMACHO** | **100** |
| **VAGNER ELDER VELAZCO ZEBALLOS** | **100** |

**Santa Cruz – Bolivia**

**Año 2023**

INDICE GENERAL

[CAPÍTULO 1 1](#_Toc131487735)

1. [MARCO TEÓRICO 1](#_Toc131487736)

[1.1. MARCO TEORICO 1](#_Toc131487737)

[CAPÍTULO 2 4](#_Toc131487739)

[2. PROYECTO 4](#_Toc131487740)

[2.1. Instalacion de librerias 4](#_Toc131487741)

[2.1.1. Numpy 4](#_Toc131487742)

[2.1.2. Tkinter 4](#_Toc131487743)

[2.1.3. Pillow 4](#_Toc131487744)

[2.1.4. Opencv 5](#_Toc131487745)

[2.1.5. Face\_recpgnition 5](#_Toc131487749)

[2.1.6. Dlib 5](#_Toc131487750)

[2.2. DISEÑO 6](#_Toc131487754)

[2.3. FUNCIONES PRINCIPALES 8](#_Toc131487758)

[2.3.1. Código fuente del proyecto 11](#_Toc131487764)

[CAPITULO 3 18](#_Toc131487765)

[3. CONCLUSIONES Y RECOMENDACIONES 18](#_Toc131487766)

[3.1. CONCLUSIONES 18](#_Toc131487767)

[3.2. RECOMENDACIONES 18](#_Toc131487768)

[REFERENCIAS 20](#_Toc131487769)

**RESUMEN EJECUTIVO**

|  |  |
| --- | --- |
| **TÍTULO:** | **RECONONOCIMIENTO FACIAL COMO METODO DE REGISTRO PARA EMPRESAS** |

Este proyecto tiene como objetivo ofrecer una solución de reconocimiento facial para empresas que buscan mejorar la eficiencia de sus procesos de registro y control de acceso. La solución se basará en tecnología de inteligencia artificial y aprendizaje automático para proporcionar un método de registro rápido, seguro y confiable.

El sistema permitirá a los empleados y visitantes registrarse a través de una simple toma de fotografía facial. Esto permitirá una identificación y autenticación de alta precisión, en la cual se almacenarán los datos personales del usuario, reduciendo la posibilidad de fraudes y suplantaciones de identidad.

Además, la solución permitirá a las empresas monitorear el acceso a sus instalaciones y áreas restringidas mediante el uso de cámaras de seguridad equipadas con tecnología de reconocimiento facial. Esto ayudará a garantizar la seguridad y el cumplimiento de las políticas de la empresa, al tiempo que reducirá el riesgo de robos o intrusiones.

Este proyecto también abordará las preocupaciones de privacidad y seguridad de los datos personales. Se implementarán medidas de seguridad para proteger la información personal, como el cifrado y el almacenamiento seguro de los datos. Además, se respetarán las regulaciones de privacidad y se obtendrán los consentimientos necesarios de los empleados y visitantes.

En resumen, este proyecto ofrecerá una solución de registro de alta precisión, seguridad y confiabilidad para empresas que buscan mejorar su gestión de acceso. La tecnología de reconocimiento facial permitirá una autenticación rápida y fácil, mejorando la eficiencia y reduciendo los riesgos de seguridad.

|  |  |  |
| --- | --- | --- |
| **CARRERA** | Ingeniería en Sistemas |  |
| **PROFESOR GUÍA** | Zambrana Chacon Jaime |
| **DESCRIPTORES O TEMAS** | Reconocimiento facial como método de registro para empresas. |  |
| **PERIODO DE INVESTIGACIÓN** | MARZO-ABRIL (2023) |  |

**ABSTRACT**

According to the provided text, the objective of this project is to offer a facial recognition solution to companies seeking to improve the efficiency of their registration and access control processes. The solution will be based on artificial intelligence and machine learning technologies to provide a fast, secure, and reliable registration method.

The system will allow employees and visitors to register through a simple facial photograph, which will be compared to a previously registered database, ensuring high precision identification and authentication, thereby reducing the possibility of fraud and identity theft.

Additionally, the solution will allow companies to monitor access to their facilities and restricted areas through security cameras equipped with facial recognition technology. This will help ensure the safety and compliance with company policies, while also reducing the risk of theft or intrusion.

Privacy and data security concerns will also be addressed in this project. Security measures will be implemented to protect personal information, such as encryption and secure data storage. Additionally, privacy regulations will be respected, and necessary consents from employees and visitors will be obtained.

In summary, this project will offer a high-precision, secure, and reliable registration solution for companies seeking to improve their access management. The facial recognition technology will allow for quick and easy authentication, improving efficiency and reducing security risks.

**UPDS**

**CAPÍTULO I**

**MARCO TEÓRICO**



# CAPÍTULO 1

# 1 MARCO TEÓRICO

## MARCO TEORICO

* La inteligencia artificial: (IA) es una rama de la informática que se centra en la creación de sistemas que pueden realizar tareas que normalmente requieren inteligencia humana.
* **Red neuronal:** es un sistema matemático modelado en el cerebro humano que aprende habilidades encontrando patrones estadísticos en los datos.
* **Aprendizaje automático:** es una rama de la IA que se centra en el desarrollo de algoritmos y técnicas que permiten a las máquinas aprender a partir de datos, sin ser programadas explícitamente para realizar una tarea en particular.
* **Aprendizaje profundo:** es un subconjunto del aprendizaje automático que utiliza redes neuronales con múltiples capas para mejorar el rendimiento de la máquina en tareas como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la conducción autónoma.
* **Procesamiento del lenguaje natural:** es una rama de la IA que se centra en el desarrollo de algoritmos y técnicas para permitir que las máquinas comprendan el lenguaje humano, tanto hablado como escrito.
* **Algoritmos genéticos:** son una técnica de optimización inspirada en la selección natural que utiliza principios biológicos para mejorar el rendimiento de los sistemas de IA.
* **Redes bayesianas:** son modelos gráficos probabilísticos que se utilizan en la IA para representar relaciones causales entre variables y para hacer inferencias y predicciones.
* **El reconocimiento facial con inteligencia:** artificial es una tecnología que utiliza algoritmos de aprendizaje automático para identificar y verificar la identidad de una persona a partir de una imagen facial.
* **Visión artificial:** es la tecnología que se encarga de la automatización de la extracción, análisis, clasificación y comprensión de imágenes mediante el uso de algoritmos de inteligencia artificial.
* **Detección facial:** es la tarea de identificar la presencia de un rostro humano en una imagen o video.
* **Reconocimiento facial**: es la tarea de identificar la identidad de una persona a partir de una imagen o video de su rostro. El reconocimiento facial utiliza algoritmos de aprendizaje automático para comparar las características faciales de la persona en la imagen con una base de datos de imágenes de rostros conocidos.
* **Algoritmos de detección de características faciales:** son algoritmos de aprendizaje automático que se utilizan para detectar características específicas en un rostro humano, como los ojos, la nariz, la boca y la forma de la cara. Estos algoritmos se utilizan en el proceso de reconocimiento facial para identificar patrones que ayuden a identificar a una persona específica.
* **Privacidad y seguridad:** el uso del reconocimiento facial con inteligencia artificial plantea preocupaciones importantes en relación a la privacidad y la seguridad de los datos personales de las personas. Por lo tanto, es importante que las empresas y los gobiernos implementen medidas adecuadas para proteger los datos personales y garantizar el uso ético de esta tecnología.
* **Visual Studio Code:** es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux.
* **Python:** es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes.
* **Tkinter:** es una biblioteca de Python para crear interfaces gráficas de usuario (GUI). Proporciona una forma fácil de crear ventanas, botones, etiquetas, campos de entrada y otros elementos de GUI.
* **PIL:** (Python Imaging Library) es una biblioteca que se utiliza para trabajar con imágenes en Python. Proporciona una forma fácil de abrir, manipular y guardar imágenes de diferentes formatos.
* **Os:** es una biblioteca que se utiliza para interactuar con el sistema operativo en Python. Proporciona funciones para crear, eliminar, renombrar y mover archivos y directorios.
* **cv2:** es una biblioteca de Python para procesamiento de imágenes y visión por computadora. Proporciona una gran cantidad de funciones para leer, escribir, manipular y mostrar imágenes y videos.
* **Matplotlib:** es una biblioteca de Python para visualización de datos. Proporciona una gran cantidad de funciones para crear gráficos, diagramas y otros tipos de visualizaciones.
* **Mtcnn:** es una biblioteca de Python para detección de rostros en imágenes. Proporciona una implementación de la red neuronal MTCNN (Multi-Task Cascaded Convolutional Networks) para detectar rostros en imágenes.
* **Numpy:** es una biblioteca de Python para cálculo numérico. Proporciona una gran cantidad de funciones para trabajar con matrices y operaciones matemáticas. Es ampliamente utilizada en procesamiento de imágenes y visión por computadora.

**UPDS**

**CAPÍTULO II**

**PROYECTO**



# CAPÍTULO 2

# PROYECTO

## Instalacion de librerias

### Numpy

NumPy es una librería numérica de Python de código abierto.

NumPy contiene una matriz multidimensional y estructuras de datos matriciales.

Se puede utilizar para realizar una serie de operaciones matemáticas en matrices como rutinas trigonométricas, estadísticas y algebraicas

Se instaló la versión 1.23.5

El comando para la instalación es el siguiente:

pip install numpy==1.23.5

### Tkinter

Se utilizó la versión 8.6

El paquete [tkinter](https://docs.python.org/es/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) («interfaz tk») es la interfaz por defecto de python para el kit de herramientas de gui tk. Tanto tk como [tkinter](https://docs.python.org/es/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) están disponibles en la mayoría de las plataformas unix, así como en sistemas windows (tk en sí no es parte de python, es mantenido por activestate).

### Pillow

La biblioteca de imágenes de python agrega capacidades de procesamiento de imágenes a su intérprete de python.

Esta biblioteca proporciona una amplia compatibilidad con formatos de archivo, una representación interna eficiente y capacidades de procesamiento de imágenes bastante potentes.

La biblioteca de imágenes principal está diseñada para un acceso rápido a los datos almacenados en unos pocos formatos básicos de píxeles. Debe proporcionar una base sólida para una herramienta general de procesamiento de imágenes.

Se utilizó la versión 9.4.0

El comando de instalación es el siguiente:

pip install Pillow==9.4.0

### Opencv

## Opencv es una [biblioteca](https://es.wikipedia.org/wiki/Biblioteca_(programaci%C3%B3n)) [libre](https://es.wikipedia.org/wiki/Software_libre) de [visión artificial](https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial) originalmente desarrollada por [intel](https://es.wikipedia.org/wiki/Intel_Corporation" \o "Intel Corporation). Opencv significa open computer vision (visión artificial abierta)

## Se utilizo la versión 4.7.0 con el comando de instalación:

## pip install opencv-python-headless==4.7.0

### Face\_recpgnition

Reconocer y manipular caras desde python o desde la línea de comandos conla biblioteca de reconocimiento facial más simple del mundo.

Comando de instalación:

pip install face-recognition

### Dlib

## Dlib es un kit de herramientas moderno que contiene algoritmos y herramientas de aprendizaje automático para crear software complejo

## El comando de instalación es el siguiente:

## pip install dlib

## diseño

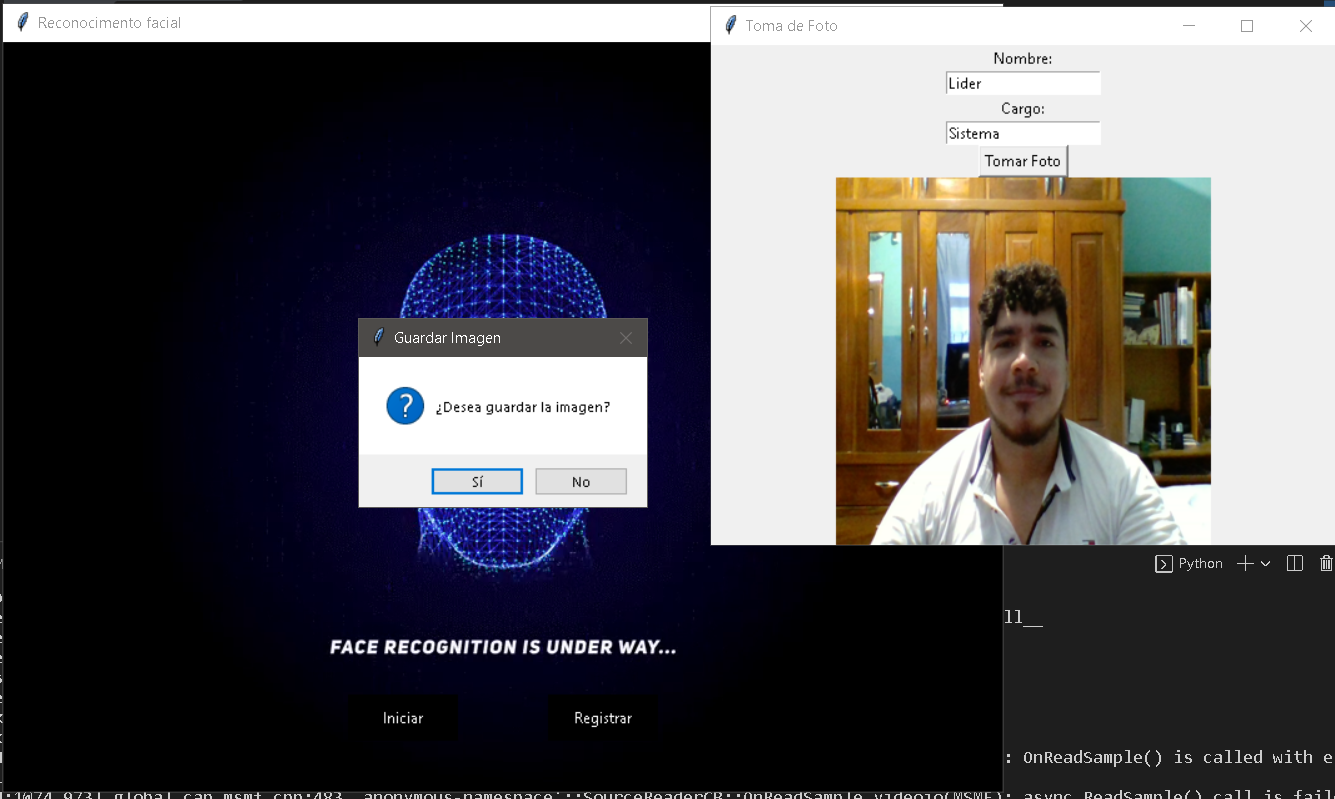
Nuestro diseño cuenta con tres pantallas, la primera siendo la pantalla de inicio donde elegimos si iniciamos sesión o si queremos registrar un nuevo usuario, la segunda es la pantalla de registro donde se coloca el nombre de usuario y cargo para después tomar la fotografía para el registro y por último esta la pantalla de inicio donde nos reconoce y valida el acceso.

### 

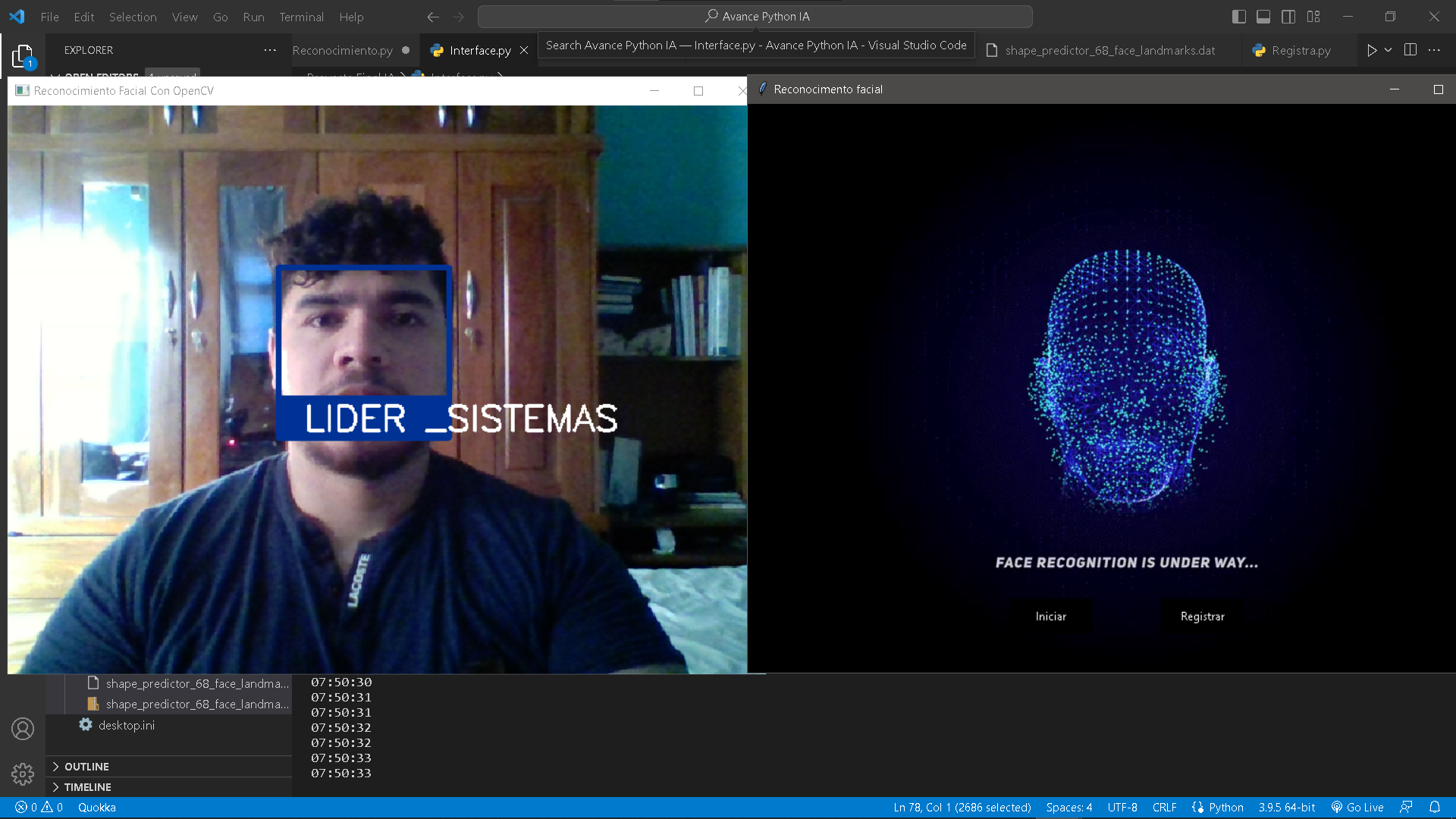
Este es el diseño de la página de inicio para el programa donde podemos verificarnos o registrarnos, se cuenta con una animación de escaneo y dos botones.

### 

### Esta es la pantalla de registro para los nuevos usuarios, donde colocamos el nombre de usuario y cargo, apretamos el botón para tomar la foto. Se guarda la fotografía para la posterior comparación y acceso con el reconocimiento facial.



Como podemos observar en la anterior captura tenemos el registro por reconocimiento facial, en donde se almacenan los datos de nombre y cargo del usuario correspondiente.



El registro del usuario fue exitoso, como podemos apreciar tenemos el reconocimiento facial de nuestro usuario, asignando automáticamente su nombre y cargo en el recuadro de identificación facial.

## Funciones principales

while True:

    #leemos los fotogramas

    ret, frame=cap.read()

    #reducimos las imagenes para mejorar el procesado

    frame2=cv2.resize(frame,(0,0),None,0.25,0.25)

    #realizamos una convercion de color

    rgb=cv2.cvtColor(frame2,cv2.COLOR\_BGR2RGB)

    #buscamos los rostros

    faces=fr.face\_locations(rgb)

    facescod=fr.face\_encodings(rgb,faces)

    facescod = np.array(facescod)

    #integramos

    for facecod,faceloc in zip(facescod,faces):

        #comparamos rostros

        comparacion=fr.compare\_faces(rostrocod,facescod)

        #calculamos la similitud

        simi=fr.face\_distance(np.array(rostrocod), np.array(facescod))

        #imprimimos simi

        #buscamos el alor mas bajo

        min=np.argmin(simi)

        if comparacion[min]:

            nombre=clases[min].upper()

            print(nombre)

            #extraemos cordenadas

            yi,xf,yf,xi=faceloc

            #escalamos

            yi,xf,yf,xi=yi\*4,xf\*4,yf\*4,xi\*4

            indice=comparacion.index(True)

            #comparamos

            if comp1 !=indice:

                #para dibujar lo que hacemos es cambiar los colores

                r=random.randrange(0,225,50)

                g=random.randrange(0,225,50)

                b=random.randrange(0,225,50)

                comp1=indice

            if comp1==indice:

                #dibujamos

                cv2.rectangle(frame,(xi,yi),(xf,yf),(r,g,b),3)

                cv2.rectangle(frame,(xi,yf-35),(xf,yf),(r,g,b),cv2.FILLED)

                cv2.putText(frame, nombre, (xi+6,yf-6),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,255),2)

                horario(nombre)

## Este código es un bucle que se ejecuta continuamente mientras se cumpla la condición "true". Dentro del bucle, se leen los fotogramas de un objeto "cap" y se realiza un procesamiento de imagen para buscar rostros y compararlos con un conjunto de rostros de referencia.

## Primero, se reduce la resolución de las imágenes para mejorar el procesamiento y se convierten del formato bgr al formato rgb. Luego, se utilizan funciones de la biblioteca "face\_recognition" para buscar rostros y codificarlos.

## Después de esto, se itera sobre cada rostro encontrado y se compara con los rostros de referencia utilizando la función "compare\_faces", que devuelve una lista de valores booleanos que indican si hay una coincidencia o no.

## Se calcula la similitud entre los rostros utilizando la función "face\_distance", que devuelve una lista de valores que indican cuánto se parecen los rostros. Luego, se busca el índice del valor más bajo en esta lista y se compara con la lista de comparación para determinar si hay una coincidencia.

## Si se encuentra una coincidencia, se dibuja un rectángulo alrededor del rostro en la imagen y se escribe el nombre de la persona correspondiente en el rectángulo. Además, se actualiza la variable "comp1" con el índice correspondiente para que el color del rectángulo cambie cuando se detecte un nuevo rostro. Finalmente, se llama a una función llamada "horario" con el nombre de la persona correspondiente como argumento.

def tomar\_foto():

    nombre = entry\_nombre.get()

    cargo = entry\_cargo.get()

    camara = cv2.VideoCapture(0)

    \_, imagen = camara.read()

    ruta = "Proyecto B\Personal\ "

    if not os.path.exists(ruta):

        os.makedirs(ruta)

    cv2.imwrite(ruta + nombre + '\_' + cargo + ".jpg", imagen)

    camara.release()

Esta función llamada "tomar foto()" toma una foto de la cámara del ordenador y la guarda en un directorio específico con el nombre y cargo proporcionados por el usuario en dos campos de entrada "entry\_nombre" y "entry\_cargo".

Primero, la función utiliza las bibliotecas "cv2" y "os" para acceder a la cámara del ordenador y leer una imagen. Luego, se crea una ruta para guardar la foto utilizando los datos de nombre y cargo proporcionados por el usuario. Si la carpeta especificada en la ruta no existe, la función crea la carpeta.

Finalmente, la función utiliza la función "cv2.imwrite()" para guardar la imagen en la ruta especificada con el nombre de archivo formateado como "nombre\_cargo.jpg". La función "release ()" se utiliza para liberar la cámara del ordenador después de que se haya tomado la foto.

### Código fuente del proyecto

**interface.py**

import tkinter as tk

from tkinter import ttk

import cv2

import numpy as np

from PIL import Image, ImageTk

import subprocess

def abrir\_registro():

    subprocess.Popen(['python', 'Proyecto B\Proyecto B\Registra.py'])

def abrir\_reconocimiento():

    subprocess.Popen(['python', 'Proyecto B\Proyecto B\Reconocimiento.py'])

# Función que carga y repite el video de fondo

def load\_video(filename):

    fondo = cv2.VideoCapture(filename)

    while True:

        ret, frame = fondo.read()

        if not ret:

            fondo = cv2.VideoCapture(filename)

            ret, frame = fondo.read()

        frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

        yield frame

# Crear la ventana

root = tk.Tk()

# Obtener las dimensiones de la pantalla

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

# Calcular la posición de la ventana en el centro de la pantalla

x = int((screen\_width/2) - (800/2))

y = int((screen\_height/2) - (600/2))

# Configurar la geometría de la ventana

root.geometry(f"800x600+{x}+{y}")

root.title("Reconocimento facial ")

root.configure(bg="black")

# Agregar el video de fondo

video\_canvas = tk.Canvas(root, width=800, height=600, highlightthickness=0)

video\_canvas.pack()

video\_frames = load\_video("Proyecto B\Proyecto B\Fondo\Video2.mp4")

current\_frame = next(video\_frames)

video\_image = ImageTk.PhotoImage(Image.fromarray(current\_frame))

video\_canvas.create\_image(0, 0, anchor=tk.NW, image=video\_image)

# Función que actualiza el video en el fondo

def update\_video():

    global current\_frame, video\_image

    current\_frame = next(video\_frames)

    video\_image = ImageTk.PhotoImage(Image.fromarray(current\_frame))

    video\_canvas.create\_image(0, 0, anchor=tk.NW, image=video\_image)

    root.after(10, update\_video)

# Configurar estilo para el botón black

style = ttk.Style()

style.theme\_use('clam')

style.configure("TButton", background="black", foreground="white", borderwidth=0)

# Configurar estilo para el botón cuando se pone el puntero sobre él

style.map("TButton", foreground=[("active", "white")], background=[("active", "blue")])

# Agregar los botones con el nuevo estilo y tamaño ajustado

button1 = ttk.Button(root, text="Iniciar", style="TButton", command=abrir\_reconocimiento, padding=(10, 10, 10, 10))

button1.place(relx=0.4, rely=0.9, anchor=tk.CENTER)

button2 = ttk.Button(root, text="Registrar", style="TButton", command=abrir\_registro, padding=(10, 10, 10, 10))

button2.place(relx=0.6, rely=0.9, anchor=tk.CENTER)

# Iniciar la actualización del video en el fondo

root.after(0, update\_video)

# Ejecutar la aplicación

root.mainloop()

**Reconocimiento.py**

#importamos las librerias

import cv2

import numpy as np

import face\_recognition as fr

import os

import random

from datetime import datetime

#accedemos a la carpeta

path ='Personal'

imagenes = []

clases = []

listas = os.listdir(path)

#print (listas)

#variables

comp1=100

#leemos los rostros

for lis in listas:

    # leemos las imagenes

    imgdb=cv2.imread(f'{path}/{lis}')

    #almacenamos la imagen

    imagenes.append(imgdb)

    #almacenamos el nombre

    clases.append(os.path.splitext(lis)[0])

print(clases)

#funcion para codificar los rostros

def codrostros(imagenes):

    listacod=[]

    for img in imagenes:

        #corregmos color

        img= cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

        #codificamos la imageb

        cod=fr.face\_encodings(img)[0]

        #almacenamos

        listacod.append(cod)

    return listacod

#hora de ingreso

def horario(nombre):

    #abrimos eñ arcivo en modo lectura

    with open ('Proyecto B\Proyecto B\horario.csv','r+') as h:

        #leemos la info

        data=h.readline()

        #creamos lista d nombres

        listanombres=[]

        #integramos cada linea de codigo

        for line in data:

            #buscamos las entradas

            entrada=line.split(',')

            #almacenamos los nombres

            listanombres.append(entrada[0])

            #verificamos si lo almacenamos

            if nombre not in listanombres:

                #extraemos info actual

                info=datetime.now()

                #extraemos fecha

                fecha=info.strftime('%Y:%M:%D:')

                #extraemos hora

                hora=info.strftime('%H:%M:%S')

                #guardamos info

                h.writelines(f'\n{nombre},{fecha},{hora}')

                print(info)

#leemos la funcion

rostrocod=codrostros(imagenes)

#realizamos la captura del video

cap=cv2.VideoCapture(0)

#inicio

while True:

    #leemos los fotogramas

    ret, frame=cap.read()

    #reducimos las imagenes para mejorar el procesado

    frame2=cv2.resize(frame,(0,0),None,0.25,0.25)

    #realizamos una convercion de color

    rgb=cv2.cvtColor(frame2,cv2.COLOR\_BGR2RGB)

    #buscamos los rostros

    faces=fr.face\_locations(rgb)

    facescod=fr.face\_encodings(rgb,faces)

    facescod = np.array(facescod)

    #integramos

    for facecod,faceloc in zip(facescod,faces):

        #comparamos rostros

        comparacion=fr.compare\_faces(rostrocod,facescod)

        #calculamos la similitud

        simi=fr.face\_distance(np.array(rostrocod), np.array(facescod))

        #imprimimos simi

        #buscamos el alor mas bajo

        min=np.argmin(simi)

        if comparacion[min]:

            nombre=clases[min].upper()

            print(nombre)

            #extraemos cordenadas

            yi,xf,yf,xi=faceloc

            #escalamos

            yi,xf,yf,xi=yi\*4,xf\*4,yf\*4,xi\*4

            indice=comparacion.index(True)

            #comparamos

            if comp1 !=indice:

                #para dibujar lo que hacemos es cambiar los colores

                r=random.randrange(0,225,50)

                g=random.randrange(0,225,50)

                b=random.randrange(0,225,50)

                comp1=indice

            if comp1==indice:

                #dibujamos

                cv2.rectangle(frame,(xi,yi),(xf,yf),(r,g,b),3)

                cv2.rectangle(frame,(xi,yf-35),(xf,yf),(r,g,b),cv2.FILLED)

                cv2.putText(frame,nombre,(xi+6,yf-6),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,255),2)

                horario(nombre)

    #mosramos frames

    cv2.imshow("Reconocimiento Facial Con OpenCV",frame)

    #leemos el teclado

    t=cv2.waitKey(5)

    if t==27:

        break

cv2.destroyAllWindows()

cap.release()

**Registra.py**

import cv2

import os

import tkinter as tk

from PIL import Image, ImageTk

from tkinter import messagebox

# Función que toma una foto y la guarda con el nombre y cargo ingresados

def tomar\_foto():

    nombre = entry\_nombre.get()

    cargo = entry\_cargo.get()

    camara = cv2.VideoCapture(0)

    \_, imagen\_bgr = camara.read()

    ruta = "Personal\ "

    if not os.path.exists(ruta):

        os.makedirs(ruta)

    imagen\_rgb = cv2.cvtColor(imagen\_bgr, cv2.COLOR\_BGR2RGB)  # Convertir de BGR a RGB

    # Previsualización de la imagen tomada

    img = Image.fromarray(imagen\_rgb)

    img = img.resize((300, 300))

    img\_tk = ImageTk.PhotoImage(image=img)

    label\_imagen.config(image=img\_tk)

    label\_imagen.image = img\_tk

    # Preguntar si desea guardar la imagen

    respuesta = messagebox.askyesno("Guardar Imagen", "¿Desea guardar la imagen?")

    if respuesta:

        cv2.imwrite(ruta + nombre + '\_' + cargo + ".jpg", imagen\_rgb)

        messagebox.showinfo("Imagen Guardada", "La imagen se ha guardado correctamente.")

        limpiar\_campos()

    else:

        if os.path.exists(ruta + nombre + '\_' + cargo + ".jpg"):

            os.remove(ruta + nombre + '\_' + cargo + ".jpg")

            messagebox.showinfo("Imagen No Guardada", "La imagen no se ha guardado.")

        else:

            messagebox.showinfo("Imagen No Guardada", "La imagen no se ha guardado.")

        limpiar\_foto()

    camara.release()

#limpiamos

def limpiar\_campos():

    entry\_nombre.delete(0, tk.END)

    entry\_cargo.delete(0, tk.END)

    label\_imagen.config(image=None)

    label\_imagen.image = None

#limpiamos solo img

def limpiar\_foto():

    label\_imagen.config(image=None)

    label\_imagen.image = None

# Creación de la ventana

ventana = tk.Tk()

ventana.title("Toma de Foto")

ventana.geometry("500x400")

ventana.eval('tk::PlaceWindow %s center' % ventana.winfo\_toplevel())

# Labels para los campos de nombre y cargo

label\_nombre = tk.Label(ventana, text="Nombre:")

label\_nombre.pack()

entry\_nombre = tk.Entry(ventana)

entry\_nombre.pack()

label\_cargo = tk.Label(ventana, text="Cargo:")

label\_cargo.pack()

entry\_cargo = tk.Entry(ventana)

entry\_cargo.pack()

# Botón para tomar la foto

boton\_foto = tk.Button(ventana, text="Tomar Foto", command=tomar\_foto)

boton\_foto.pack()

# Label para mostrar la imagen tomada

label\_imagen = tk.Label(ventana)

label\_imagen.pack()

ventana.mainloop()

**UPDS**

**CAPÍTULO III**

**CONCLUSIONES Y**

**RECOMENDACIONES**



# CAPITULO 3

# CONCLUSIONES Y RECOMENDACIONES

## CONCLUSIONES

En conclusión, la tecnología de reconocimiento facial con inteligencia artificial puede ser una herramienta valiosa para mejorar la seguridad de las empresas, siempre y cuando se adopten medidas adecuadas para proteger la privacidad del usuario y evitar el uso inapropiado de los datos personales. En este sentido, es crucial que las empresas evalúen cuidadosamente los beneficios y riesgos del uso de esta tecnología y tomen medidas de seguridad adecuadas para garantizar la privacidad de sus clientes y empleados.

Asimismo, buscar el asesoramiento de expertos en el uso de la tecnología de reconocimiento facial puede ser útil para tomar decisiones informadas y responsables. En este sentido, nuestro proyecto puede ofrecer asesoramiento y apoyo para garantizar que se implementen medidas de seguridad adecuadas y se respeten los derechos de privacidad de los usuarios. En definitiva, se trata de utilizar la tecnología de manera responsable y ética para garantizar una mayor seguridad y protección de la privacidad de los usuarios.

## RECOMENDACIONES

El uso de tecnología de reconocimiento facial en empresas puede ser beneficioso para mejorar la seguridad y la eficiencia en los procesos de autenticación de los usuarios. Sin embargo, también es importante tener en cuenta ciertas recomendaciones para garantizar que la tecnología se utilice de manera responsable y proteger la privacidad de los empleados. A continuación, les presentaremos algunas recomendaciones para el buen uso de estas tecnologías:

* Establecer políticas claras: Es importante que las empresas establezcan políticas claras sobre el uso de la tecnología de reconocimiento facial, incluyendo cuándo y cómo se utilizará, quién tendrá acceso a los datos recopilados y cómo se protegerán los datos de los empleados.
* Obtener el consentimiento informado de los empleados: Las empresas deben obtener el consentimiento informado de los empleados antes de implementar la tecnología de reconocimiento facial. Es importante que los empleados comprendan cómo se utilizará la tecnología y que tengan la oportunidad de optar por no participar si así lo desean.
* Proteger los datos de los empleados: Es importante que las empresas implementen medidas de seguridad adecuadas para proteger los datos de los empleados recopilados por la tecnología de reconocimiento facial. Esto incluye medidas como el cifrado de datos, la limitación del acceso a los datos y la eliminación de los datos cuando ya no sean necesarios.
* Considerar la diversidad: Las empresas deben tener en cuenta la diversidad de los empleados al implementar tecnología de reconocimiento facial. La tecnología debe ser capaz de reconocer y autenticar a todos los empleados de manera efectiva, independientemente de su género, raza o etnia
* Evaluar regularmente la tecnología: Las empresas deben evaluar regularmente la tecnología de reconocimiento facial para garantizar que se utilice de manera responsable y efectiva. Esto incluye la evaluación de la precisión de la tecnología y la revisión de las políticas y procedimientos de la empresa en torno al uso de la tecnología
* En resumen, el uso de tecnología de reconocimiento facial en empresas puede ser beneficioso, siempre y cuando se implemente de manera responsable y se proteja la privacidad de los empleados. Las empresas deben establecer políticas claras, obtener el consentimiento informado de los empleados, proteger los datos de los empleados, considerar la diversidad y evaluar regularmente la tecnología para garantizar su efectividad y seguridad.

**UPDS**

**CAPÍTULO IV**

**REFERENCIAS**



# REFERENCIAS

Bhattacharya, S. y Majumder, K. (2018). Avances recientes en la tecnología de reconocimiento facial: una encuesta. Revisión de inteligencia artificial, 50(3), 313-347.

Jain, AK, Ross, A. y Nandakumar, K. (2016). Introducción a la biometría. Saltador.

Li, SZ y Jain, AK (2011). Manual de reconocimiento facial (2ª ed.). Saltador.

Parkhi, OM, Vedaldi, A. y Zisserman, A. (2015). Reconocimiento facial profundo. Conferencia británica sobre visión artificial, 41.1-41.12.

Schroff, F., Kalenichenko, D. y Philbin, J. (2015). FaceNet: una integración unificada para el reconocimiento facial y la agrupación. Actas de la Conferencia IEEE sobre visión artificial y reconocimiento de patrones, 815-823.

Sun, Y., Chen, J., Wang, X. y Tang, X. (2013). Representación facial de aprendizaje profundo a partir de la predicción de 10,000 clases. Actas de la Conferencia IEEE sobre visión artificial y reconocimiento de patrones, 1891-1898.

Taigman, Y., Yang, M., Ranzato, M. y Wolf, L. (2014). DeepFace: cerrar la brecha con el rendimiento a nivel humano en la verificación facial. Actas de la Conferencia IEEE sobre visión artificial y reconocimiento de patrones, 1701-1708.

Turco, MA y Pentland, AP (1991). Reconocimiento facial mediante autocaras. Actas de la Conferencia IEEE sobre visión artificial y reconocimiento de patrones, 586-591.

Viola, P. y Jones, M. (2001). Detección rápida de objetos utilizando una cascada potenciada de funciones simples. Actas de la Conferencia IEEE sobre visión artificial y reconocimiento de patrones, 511-518.

Zhang, K., Zhang, Z., Li, Z. y Qiao, Y. (2016). Detección y alineación de caras conjuntas utilizando redes convolucionales en cascada multitarea. Cartas de procesamiento de señales IEEE, 23(10), 1499-1503.

<https://www.youtube.com/watch?v=eBYlOGRUsCw>

https://www.youtube.com/watch?v=OArR9c\_OXaY