# **Manual Técnico: Sistema de Registro e Inicio de Sesión con Reconocimiento Facial**

## **1. Introducción**

El sistema implementado utiliza la biblioteca **face\_recognition** para registrar e iniciar sesión mediante reconocimiento facial. Este sistema proporciona un método seguro y eficiente para autenticar usuarios, capturando y verificando sus rostros en tiempo real.

Este manual describe los aspectos técnicos necesarios para implementar, mantener y operar el sistema.

## **2. Requisitos Técnicos**

### **2.1 Hardware:**

* **Cámara:**
  + Resolución mínima recomendada: 2 MP.
  + Resolución óptima: 5 MP o superior para mayor precisión.
  + Debe ser compatible con la biblioteca OpenCV para captura de video.
* **Computadora:**
  + Sistema operativo: Windows, macOS o Linux.
  + Procesador: Mínimo a 1.6ghz
  + RAM: Mínimo 4 GB, recomendado 8 GB.
  + Almacenamiento: 500 MB libres para guardar datos de usuarios.

### **2.2 Software:**

* **Python 3.7 o superior** (lenguaje de programación principal).
* Librerías necesarias (instalación mediante pip):

pip install face\_recognition opencv-python dlib numpy

* Sistema de almacenamiento: Base de datos MYSQL

### **2.3 Dependencias:**

1. **face\_recognition:** Para el procesamiento y comparación de rasgos faciales.
2. **opencv-python:** Para la captura y manipulación de video en tiempo real.
3. **dlib:** Librería subyacente utilizada por face\_recognition.
4. **numpy:** Para operaciones matemáticas y manipulación de datos.
5. **pickle:** Para almacenar y recuperar codificaciones faciales.

## **3. Arquitectura del Sistema**

El sistema está compuesto por los siguientes módulos:

### **3.1 Captura de Imagen:**

* Utiliza la cámara para capturar una imagen en tiempo real.
* Convierte la imagen a formato RGB para procesamiento.

### **3.2 Registro de Usuario:**

* Procesa la imagen capturada para generar una codificación facial única.
* Almacena el nombre del usuario y su codificación en un archivo o base de datos.

### **3.3 Inicio de Sesión:**

* Captura una imagen en tiempo real y genera su codificación facial.
* Compara esta codificación con las codificaciones almacenadas para autenticar al usuario.

### **3.4 Gestor de Datos:**

* Guarda y recupera las codificaciones faciales mediante un archivo binario (pickle).
* Gestiona la actualización o eliminación de registros de usuarios.

## **4. Estructura del Código**

### **4.1 Archivos Principales:**

* **main.py:** Contiene las funciones principales para registro e inicio de sesión.
* **user\_data/face\_encodings.pkl:** Archivo donde se almacenan las codificaciones faciales de los usuarios.

### **4.2 Funciones Principales:**

#### **Captura de Imagen:**

Captura una imagen desde la cámara en tiempo real.

#### **Registro de Usuario:**

Permite registrar un nuevo usuario almacenando su codificación facial.

#### **Inicio de Sesión:**

Autentica al usuario comparando su rostro con los datos almacenados.

#### **Gestor de Datos:**

Guarda y recupera las codificaciones faciales de los usuarios.

## **5. Descripción de los Módulos**

### **5.1 Base de datos**

import mysql.connector

class BaseDeDatos:

def \_\_init\_\_(self, host, user, password, database):

self.connection = mysql.connector.connect(

host=host,

user=user,

password=password,

database=database

)

self.cursor = self.connection.cursor()

def ejecutar\_consulta(self, consulta, datos=None):

self.cursor.execute(consulta, datos or ())

self.connection.commit()

return self.cursor

def obtener\_resultados(self, consulta, datos=None):

self.cursor.execute(consulta, datos or ())

return self.cursor.fetchall()

def cerrar\_conexion(self):

self.cursor.close()

self.connection.close()



### **5.2 Captura de Imagen**

Captura una imagen utilizando OpenCV.

import cv2

import face\_recognition

import numpy as np

class ReconocimientoFacial:

@staticmethod

def capturar\_datos\_facial():

video\_capture = cv2.VideoCapture(0)

ret, frame = video\_capture.read()#exito, matriz

video\_capture.release()#libe

if not ret:

return None

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)#Asi

face\_encodings = face\_recognition.face\_encodings(rgb\_frame) #detec y ge vec

if face\_encodings:

return face\_encodings[0] # Retorna el encoding como un ndarray

return None

#COMPARACION

@staticmethod

def comparar\_rostros(datos1, datos2):

if datos1 is None or datos2 is None:

return False

return face\_recognition.compare\_faces([datos1], datos2)[0]



### **5.3 Registro de Usuario**

Convierte la imagen del usuario en una codificación facial y la guarda.

def register\_user():

"""

Registra a un nuevo usuario capturando su rostro y almacenando su codificación.

"""

name = input("Introduce tu nombre: ")

print("Capturando tu imagen para el registro...")

image = capture\_image()

if image is not None:

rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

face\_encodings = face\_recognition.face\_encodings(rgb\_image)

if len(face\_encodings) > 0:

face\_encoding = face\_encodings[0]

known\_faces = load\_encodings()

known\_faces[name] = face\_encoding

save\_encodings(known\_faces)

print(f"Usuario {name} registrado con éxito.")

else:

print("No se detectó un rostro en la imagen.")



### **5.4 Inicio de Sesión**

Compara la codificación facial del usuario con las almacenadas.

def iniciar\_sesion\_contrasena(self):

correo = self.correo\_var.get()

contrasena = self.contrasena\_var.get()

usuario\_id = Usuario.autenticar(self.db, correo, contrasena)

if usuario\_id:

self.registrar\_ingreso(usuario\_id)

messagebox.showinfo("Éxito", "Inicio de sesión exitoso.")

else:

messagebox.showwarning("Error", "Credenciales incorrectas.")



### **6. Mantenimiento del Sistema**

1. **Actualización de la Base de Datos:**
   * Regularmente agregar o eliminar usuarios del archivo de codificaciones (face\_encodings.pkl).
2. **Depuración:**
   * Utilizar print o herramientas como logging para identificar errores en la captura de imágenes o el reconocimiento facial.
3. **Mejoras Futuras:**
   * Implementar almacenamiento en una base de datos en lugar de archivos binarios.
   * Diseñar una interfaz gráfica amigable utilizando bibliotecas como Tkinter o PyQt.

Este manual técnico detalla los aspectos necesarios