A. F. Alonso Baquero, C. R. Niño Ojeda, C. F. Gonzales, J. D. Duran Gerena, S. Ramírez Muñoz y S. A. Ortiz

[[1]](#footnote-1)

Reconocimiento facial para el ingreso de los funcionarios en la alcaldía Municipal de Granada MVC (noviembre 2023)

Resumen - Este proyecto propone el desarrollo de un sistema de reconocimiento facial para la gestión de recursos humanos en una alcaldía municipal. La implementación se basa en la arquitectura Modelo Vista Controlador (MVC) y utiliza tecnologías como Python, OpenCV para el reconocimiento facial, PyQt para la interfaz gráfica y Docker para la creación de contenedores. El sistema busca optimizar la administración de recursos, fortalecer la seguridad interna y mejorar la eficiencia en los procesos de identificación y control de acceso. Se detallan las funcionalidades, requisitos y relaciones entre las clases del sistema.

**Índice de Términos - Reconocimiento Facial: Identificación de individuos mediante análisis facial.**

**Arquitectura Modelo Vista Controlador (MVC): Estructura y diseño esenciales del software.**

**Docker: Implementación de aplicaciones, gestión de dependencias.**

**Entrenamiento de Modelo: Proceso crucial, impacta precisión del sistema.**

# introduccion

**E**n el entorno actual, donde la seguridad y la eficiencia son elementos críticos para el desarrollo de comunidades locales, la implementación de tecnologías innovadoras se convierte en una necesidad imperante. Este trabajo se centra en la creación de un sistema de reconocimiento facial basado en la arquitectura Modelo Vista Controlador (MVC) para la gestión de recursos humanos en una alcaldía municipal específica: Granada. La adopción de esta tecnología no solo busca fortalecer la seguridad interna, sino también optimizar la administración de recursos y mejorar la eficiencia en los procesos de identificación y control de acceso.

Objetivo General:

El propósito fundamental es la implementación de un sistema de reconocimiento facial mediante la arquitectura Modelo Vista Controlador (MVC) para el registro de funcionarios en la alcaldía municipal de Granada. Este proyecto tiene como meta mejorar la gestión de recursos humanos, fortalecer la seguridad interna y perfeccionar los procesos de identificación y control de acceso.

Objetivos Específicos:

1. Gestionar de manera eficiente el ingreso a la alcaldía municipal de Granada.

2. Mapear de manera precisa el rostro de cada funcionario municipal.

3. Registrar y mantener un historial de la actividad de los funcionarios.

4. Establecer un control efectivo sobre los horarios de trabajo de los empleados municipales.

5. Supervisar y registrar los usuarios que acceden a la alcaldía municipal de Granada.

Planteamiento del Problema:

La falta de control en los ingresos a la alcaldía de Granada plantea una seria interrogante: ¿Cómo afecta esta carencia a la seguridad de los ciudadanos y al desarrollo de la comunidad local? Esta problemática resalta la necesidad urgente de implementar medidas que no solo garanticen la integridad de la alcaldía, sino también impulsen el progreso y bienestar de la comunidad.

Justificación:

En respuesta a la creciente inquietud por la seguridad y la búsqueda de procesos más eficientes, la introducción de un sistema de reconocimiento facial se presenta como una solución prometedora. Una interfaz de reconocimiento facial no solo añade una capa adicional de seguridad, sino que también agiliza el proceso de entrada para empleados y visitantes autorizados, eliminando la dependencia de tarjetas de identificación o documentos físicos. Este enfoque innovador tiene el potencial de transformar la gestión de recursos humanos y la seguridad en la alcaldía municipal de Granada, proporcionando beneficios tangibles para la comunidad local.

# Requerimientos funcionales y no funcionales

## Requerimientos Funcionales

1. Reconocimiento Facial:

- El sistema debe ser capaz de realizar el reconocimiento facial de los funcionarios registrados.

- Utilizar la biblioteca OpenCV para implementar el algoritmo de reconocimiento facial.

2. Registro de funcionarios:

- Permitir el registro de nuevos funcionarios, incluyendo la captura y almacenamiento de imágenes faciales.

- Asociar información personal y laboral al rostro registrado.

3. Interfaz Gráfica con PyQt:

- Desarrollar una interfaz gráfica sencilla utilizando PyQt para facilitar la interacción del usuario.

- Incluir símbolos y escudos de la alcaldía para una identificación visual clara.

4. Control de Acceso:

- Integrar un sistema que permita el acceso únicamente a funcionarios reconocidos y registrados en la base de datos.

5. Docker como Entorno de Contenedores:

- Utilizar Docker para encapsular y desplegar la aplicación, facilitando la portabilidad y escalabilidad.

- Garantizar la compatibilidad con diferentes sistemas operativos mediante el uso de contenedores.

## Requerimientos No Funcionales

1. Eficiencia:

- El sistema debe operar de manera eficiente, con tiempos de respuesta rápidos en el reconocimiento facial y la gestión de datos.

2. Seguridad:

- Garantizar la seguridad de la información almacenada, implementando medidas de cifrado y control de accesos.

3. Escalabilidad:

- Diseñar la aplicación de manera que permita la incorporación sencilla de nuevas funcionalidades y la expansión de la base de datos.

4. Simplicidad de Interfaz:

- La interfaz gráfica debe ser intuitiva y fácil de usar, manteniendo la simplicidad para usuarios con diversos niveles de habilidad.

5. Portabilidad:

- Asegurar que la aplicación sea compatible con diferentes plataformas mediante el uso de contenedores Docker.

6.\*Documentación:

- Proporcionar documentación detallada sobre la instalación, configuración y uso del sistema.

7. Estabilidad:

- Minimizar la ocurrencia de errores y fallos, asegurando la estabilidad del sistema durante su operación normal.

8. Estilo de Código y Buenas Prácticas:

- Seguir estándares de codificación y buenas prácticas de desarrollo en Python para garantizar la legibilidad y mantenibilidad del código.

## Diagrama de procesos.

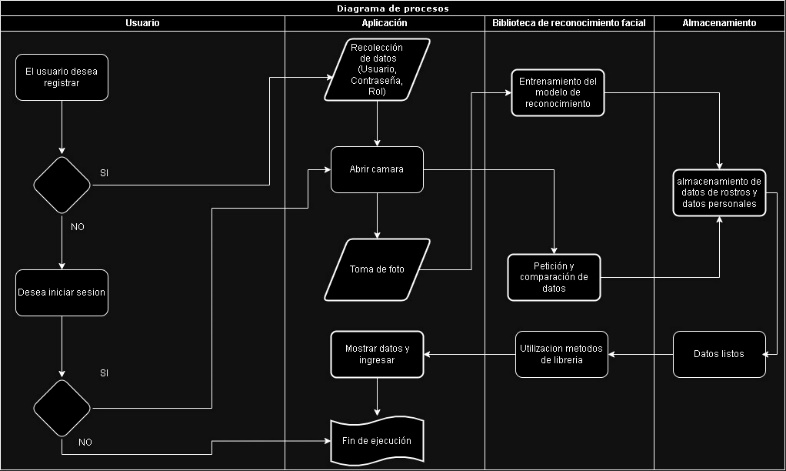


Fig. 1. La Figura 1 representa un Diagrama de Procesos, ofreciendo una visualización concisa y esquemática de las etapas o pasos secuenciales en un sistema o proceso específico. Este diagrama proporciona una representación gráfica para comprender la secuencia y la interconexión de las diferentes actividades o elementos dentro del proceso en cuestión.

## Diagrama de clases

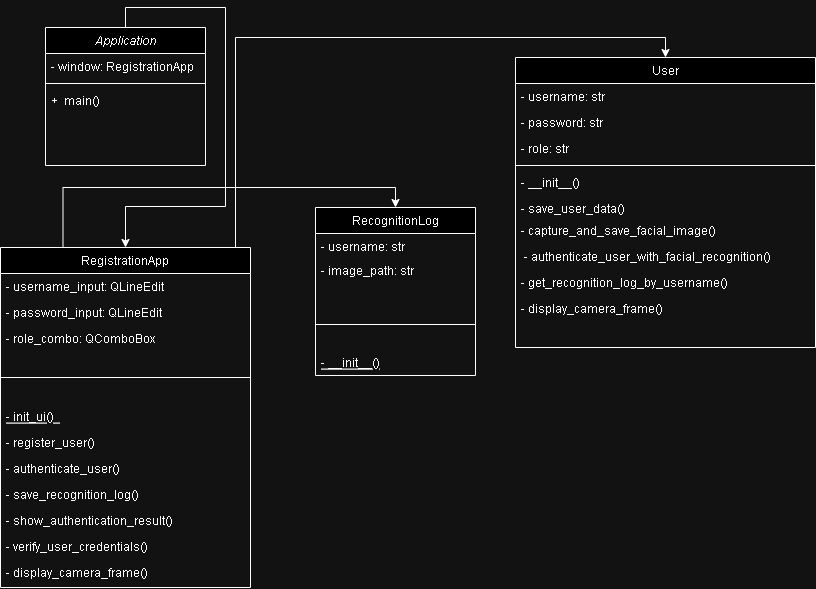


Fig. 2. La Figura 2 ilustra un Diagrama de Clases, proporcionando una representación gráfica de las clases en un sistema y sus relaciones. Este tipo de diagrama es esencial para comprender la estructura y la jerarquía de las clases, así como las asociaciones entre ellas en el contexto del desarrollo de software. La figura ofrece una visión visual clara de la arquitectura y la organización de las clases dentro del sistema analizado.

## Wireframes

se centra en la presentación y descripción visual de las estructuras esenciales de la interfaz de usuario. Aquí se proporcionan representaciones esquemáticas, de bajo detalle, pero fundamentales para la disposición y organización de elementos clave en el diseño de la interfaz. Los wireframes sirven como una herramienta inicial para la conceptualización y comunicación de la disposición general de la interfaz, antes de avanzar a etapas más detalladas del proceso de diseño.

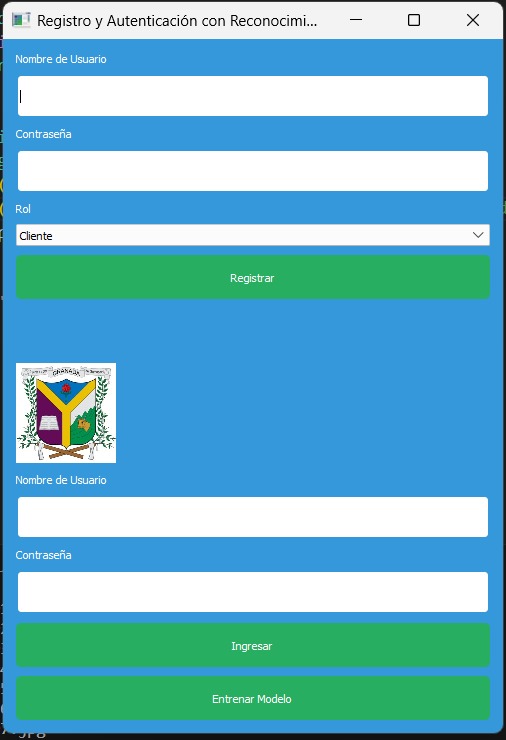


Fig. 3. Interfaz grafica del programa

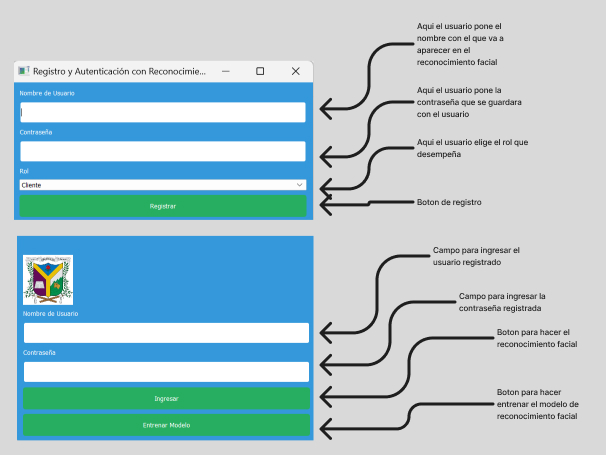


Fig .4. Funciones de la interfaz gráfica.

# Pruebas

En el desarrollo de software, las pruebas son esenciales para garantizar la calidad. Las pruebas unitarias evalúan componentes individuales, mientras que las pruebas funcionales examinan el sistema completo. Ambos tipos son críticos para detectar errores y asegurar el rendimiento del programa.

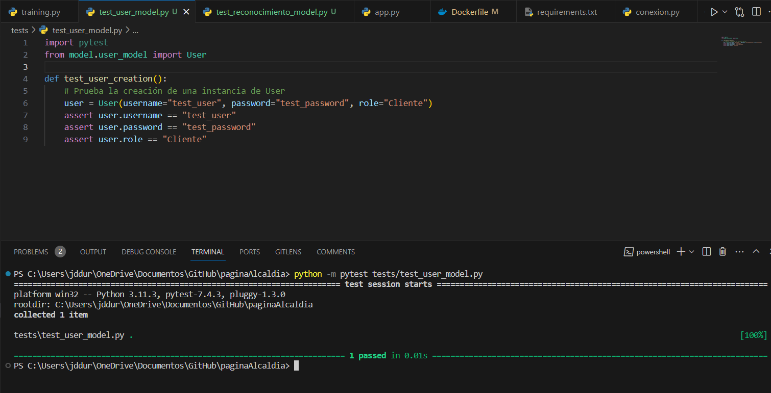


Fig.5 . Prueba al modelo User.

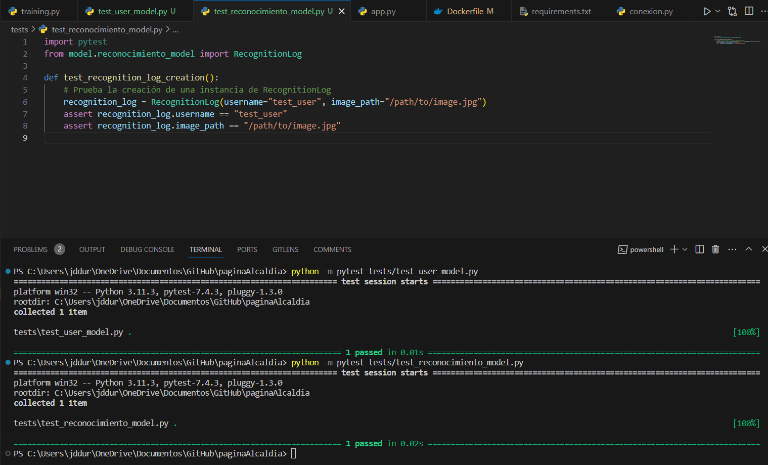


Fig.6 . Prueba al modelo de reconocimiento.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Referencia | Descripción | Si | No | Anotaciones |
| 1 | Intentar ingresar sin previo registro | x |  | Si no cuenta con un previo registro no será reconocido por el sistema. |
| 2 | Intentar realizar un entrenamiento sin registro | x |  | Si no ha culminado el registro no puede entrenar el modelo. |
| 3 | Registrar sin suministrar datos |  | x | Al no suministrar los datos en el registro no lo reconocerá posteriormente. |
| 4 | Consultar registro | x |  | No permite consultas. |
| 5 | Ingreso con tapabocas |  | x | Al no reconocer las facciones no permite el ingreso. |
| 6 | Ingreso con gorra | x |  | La gorra afecta mayormente en cuestiones de iluminación, pero si la gorra esta hacia atrás no hay problema. |
| 7 | Ingreso con gafas |  | x | las gafas no permiten el ingreso ya que oculta parte de las facciones. |
| 8 | Rendimiento por saturación |  | x | El sistema al recibir una alta saturación de registros sufre perdida de rendimiento. |
| 9 | Saturación por exceso de caracteres | x |  | Se pueden digitar bastantes caracteres el problema es a la hora de ingresar, no los mostrara en pantalla |

Tabla.1. Pruebas del funcionamiento

## Abreviaciones

RF (Reconocimiento Facial): Tecnología que identifica y verifica individuos mediante el análisis de características faciales.

MVC (Arquitectura Modelo Vista Controlador): Patrón de diseño que separa la lógica de la aplicación en tres componentes: Modelo, Vista y Controlador.

PyQt: Conjunto de enlaces Python para la biblioteca Qt, utilizada para crear interfaces gráficas de usuario.

RF (Requerimientos Funcionales): Funcionalidades específicas que el sistema debe cumplir para satisfacer las necesidades del usuario.

RNF (Requerimientos No Funcionales): Características no específicas de la funcionalidad, como rendimiento, seguridad y escalabilidad.

DC (Diagrama de Clases): Representación gráfica de las clases en un sistema y sus relaciones.

RA (Relación de Asociación): Conexión entre clases que indica una relación semántica.

EC (Entorno de Contenedores): Ambiente aislado que encapsula una aplicación y sus dependencias para su implementación.

EM (Entrenamiento de Modelo): Proceso de enseñar a un modelo de reconocimiento facial mediante datos de entrada.

EP (Eficiencia en Procesos): Mejora en la ejecución de tareas para lograr un rendimiento óptimo.

CIF (Comparación de Imágenes Faciales): Evaluación de similitud entre imágenes para autenticación facial.

RR (Registro de Reconocimiento): Almacenamiento de información relacionada con el reconocimiento facial de usuarios.

IV. CONCLUSIÓN

El proyecto propuesto, enfocado en la implementación de un sistema de reconocimiento facial para la gestión de registros en la alcaldía municipal de Granada, destaca por la integración efectiva de tecnologías clave. La adopción de la arquitectura Modelo Vista Controlador (MVC) ha brindado una estructura organizada que facilita la separación de responsabilidades, mejora la disponibilidad y simplifica el mantenimiento del código, contribuyendo así al éxito y eficiencia del proyecto.

Apéndice

Funcionalidad

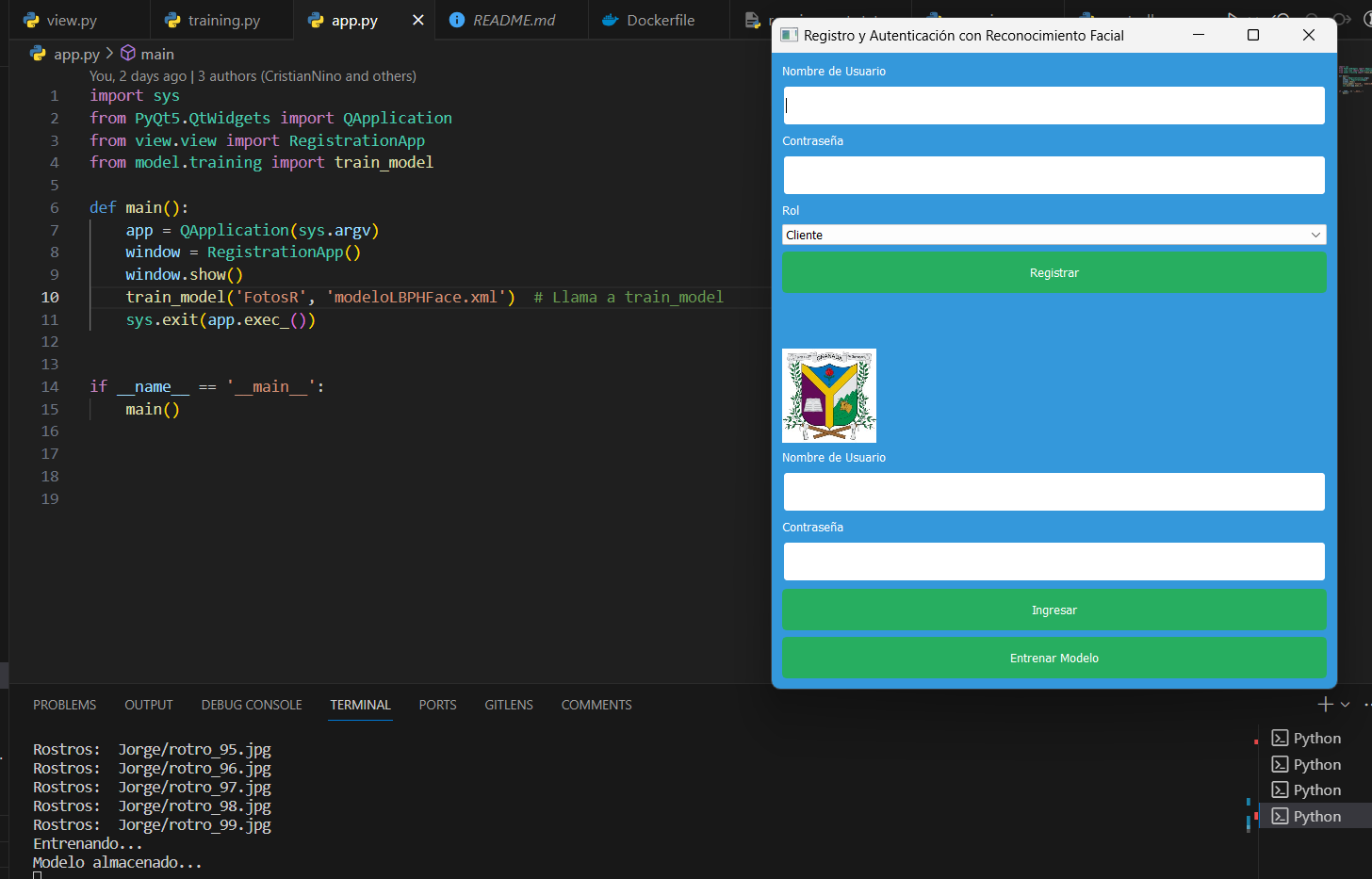


Fig.7 . Inicio del programa

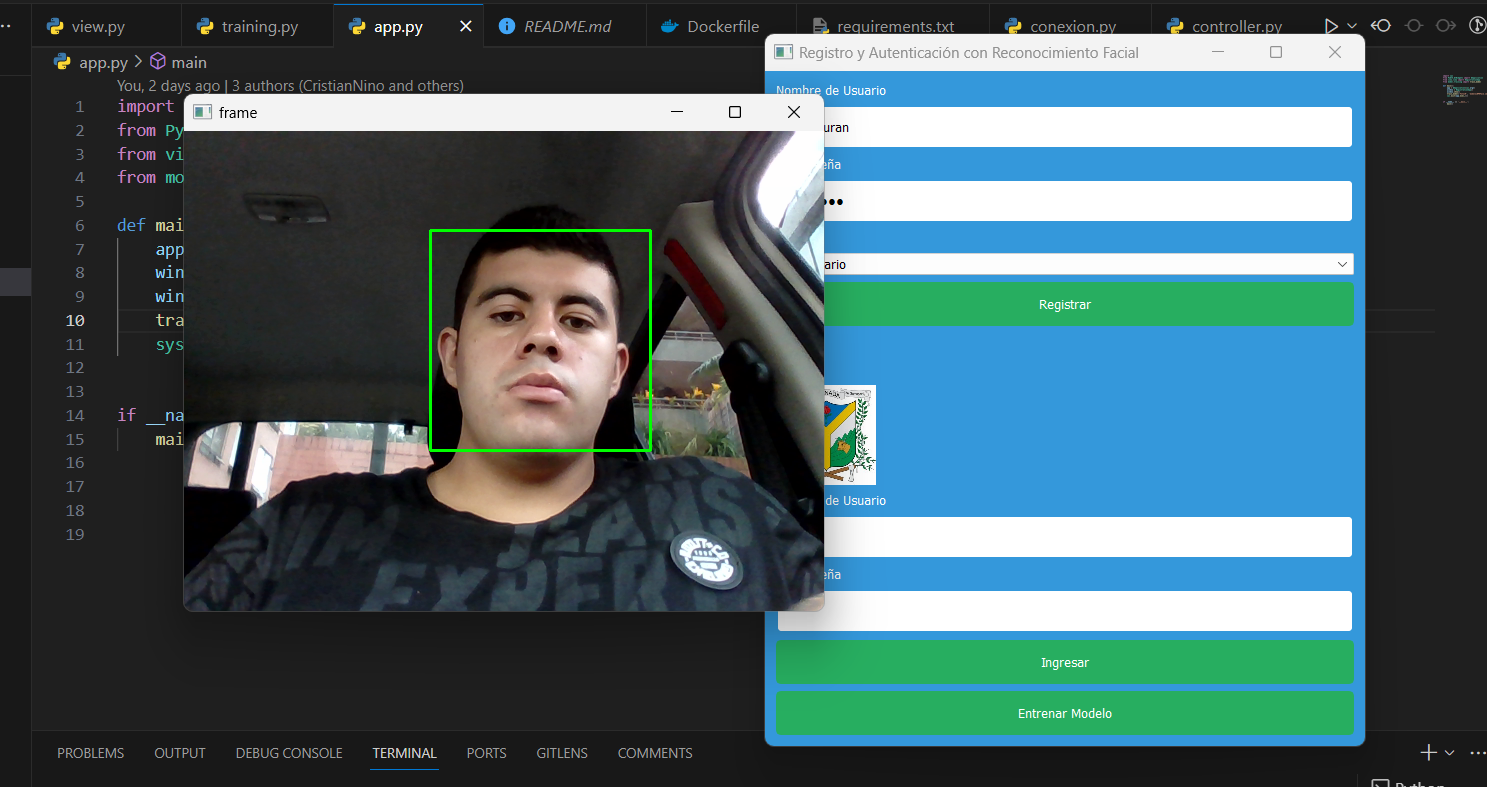
.

Fig.7 . Registro y toma de fotos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fig.8 . Entrenado del modelo para el reconocimiento facial.

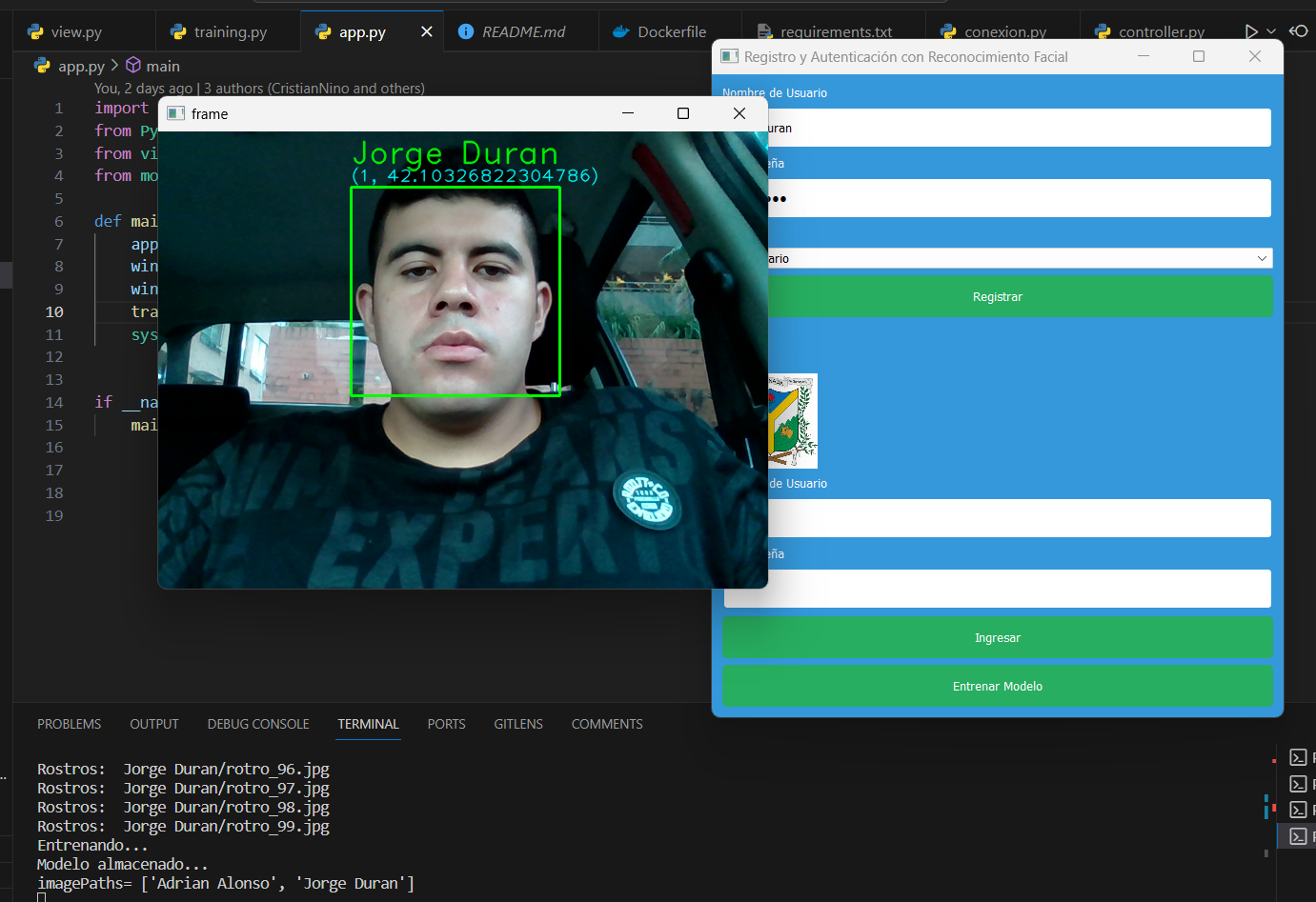


Fig.8 . Reconocimiento facial.

Reconocimiento

Quisiera expresar mi sincero agradecimiento al grupo en general, compuesto por Adrián Fernando Alonso Baquero, Cristian Ricardo Niño Ojeda, Cristian Felipe Gonzales, Santiago Ramírez Muñoz y Sergio Alejandro Ortiz, por su colaboración, apoyo y dedicación a lo largo de este proyecto. La contribución de cada miembro ha sido invaluable para el logro de nuestros objetivos comunes.

J. D. Duran Gerena

References

1. Python Software Foundation. "Python 3.11.0 Documentation.,[En línea]" Python 3.11.0 Documentation. Disponible en: <https://docs.python.org/3.11/library/index.html>.
2. AprendeeIngenia. "CURSO#1: SISTEMA DE RECONOCIMIENTO FACIAL & LIVENESS desde 0 | Python Tkinter OpenCV". YouTube, [En línea]. Disponible en: <https://www.youtube.com/watch?v=ut2jSVonUbM&ab_channel=AprendeeIngenia>.
3. DATACLOUDER. "Crear un ambiente de desarrollo con Docker y Python". YouTube, [En línea]. Disponible en: <https://www.youtube.com/watch?v=BvvH3ohis6E&t=530s&ab_channel=DATACLOUDER>.
4. FaztCode. "Docker, Instlación en Windows (más WSL, Window Subsystem for Linux)". YouTube, [En línea]. Disponible en: <https://www.youtube.com/watch?si=9EQ4SKZMkrlysWBb&v=ZO4KWQfUBBc&feature=youtu.be>.
5. Docker Documentation. [En línea]. Disponible en: <https://docs.docker.com/>.

1. [↑](#footnote-ref-1)