PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

CARRERA DE: INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Trabajo de Titulación

Tema: Prototipo de Ecosistema de Identidad de Acceso con reconocimiento facial

AUTOR:

Freddy Alejandro Zapata Armas

DIRECTOR:

Edison Javier Guaña Moya

QUITO DM, ABRIL DE 2025

DEDICATORIA

El presente trabajo de titulación se lo dedico, con todo mi amor y gratitud, a mi madre, parte vital de mi vida. Cuando más lo necesitaba, su amor inquebrantable, su fortaleza en los momentos más difíciles, su tenacidad frente a las adversidades, su infinita paciencia y su apoyo incondicional fueron mi guía. De ella aprendí que el sacrificio, la valentía y la fe no solo acompañan a los sueños, sino que los hacen posibles. Gracias por ser un ejemplo para seguir, un refugio y una motivación durante todo este proceso.

Asimismo, quiero agradecer a mi padre por su presencia, sus consejos oportunos y su inquebrantable fe en mí. Su manera de afrontar las dificultades, sus sabios consejos y su apoyo incondicional me han proporcionado el equilibrio necesario para seguir adelante con determinación. Agradezco que me haya enseñado a valorar el esfuerzo diario y a ser resiliente.

Dedico también este trabajo a mis hermanos, Doménica y Leonardo, así como a mi mascota Goliat. Aunque hoy no están físicamente conmigo, siento su presencia en cada paso que doy. Sus recuerdos viven en mi corazón y me han brindado la fuerza emocional necesaria para no rendirme. Sus memorias han sido un impulso poderoso que me ha acompañado durante todo este proceso.

AGRADECIMIENTO

Agradezco este logro a toda mi familia, cuyo apoyo incondicional ha sido fundamental en cada paso de este camino. Agradezco especialmente a mi madre, por su amor infinito, su fortaleza y su confianza inquebrantable en mí. Su ejemplo y sacrificio han sido la luz que iluminó mi vida en los momentos más difíciles, y su cariño el refugio donde siempre encontré fuerzas para seguir adelante. Sin su apoyo, este logro no habría sido posible.

Agradezco a mi tutor**,** Javier Guaña, por su guía constante a lo largo de todo este proceso. Asimismo, extiendo mi gratitud a los docentes y a la Pontificia Universidad Católica del Ecuador quienes, con sus enseñanzas, consejos y un ambiente académico de excelencia enriquecieron mi formación y me inspiraron a dar lo mejor de mí en cada etapa.

También agradezco a mis amigos, quienes con su compañía, consejos y motivación hicieron que este camino fuera más acogedor, llevadero y único. Su apoyo en los momentos de incertidumbre fue clave para mantener el ánimo y seguir adelante.

A todos ustedes, gracias por acompañarme y creer en mí durante esta etapa tan importante de mi vida.

RESUMEN

El presente trabajo de titulación abordó el diseño, desarrollo e implementación de un prototipo de sistema de control de acceso basado en reconocimiento facial, utilizando Raspberry Pi, su módulo cámara y librerías de Python como Flask, OpenCV y face_recognition, apoyado en una base de datos PostgreSQL. Este prototipo integró técnicas de extracción y almacenamiento de codificaciones faciales mediante modelos HOG para realizar comparaciones en tiempo real, permitiendo identificar de manera automática y precisa a las personas autorizadas y registrar sus accesos. Además, facilita la visualización en tiempo real de quién entra, fortaleciendo la seguridad y el control en entornos físicos. Se establecieron metodologías para el entrenamiento con datasets variados para la captura y análisis continuo, buscando optimizar tanto la precisión como la eficiencia del reconocimiento. La evaluación funcional se fundamentó en pruebas estructuradas que miden métricas esenciales como la tasa de reconocimiento, tiempos de detección y comparación, y tasa de falsos negativos, bajo condiciones variables de ángulo, iluminación y uso de accesorios, validando así el desempeño y las limitaciones del sistema en escenarios reales de control biométrico. Este trabajo demostró la viabilidad técnica de implementar sistemas biométricos accesibles y propone un ecosistema integrable que asegura tanto seguridad como usabilidad en aplicaciones de control de acceso.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ABREVIATURAS

ACID	*Atomicity, Consistency, Isolation, Durability* (**Atomicidad, Consistencia, Aislamiento, Durabilidad)**
API	*Application Programming Interface* **(Interfaz de Programación de Aplicaciones)**
BGR	*Blue Green Red* **(Azul Verde Rojo)**
BYTEA	*Byte Array* **(Arreglo de Bytes)**
CNN	*Convolutional Neuronal Network* **(Redes Neuronales Convolucionales)**
CRUD	*Create, Read, Update, Delete* **(Crear, Leer, Actualizar, Eliminar)**
CSS	*Cascading Style Sheets* **(Hoja de Estilos en Cascada)**
CSI	*Camera Serial Interface* **(Interfaz Serial para Cámara)**
DBMS	*Database Management System* **(Sistema de Gestión de Base de Datos)**
ER-D	*Entity-Relationship Diagram* **(Diagram Entidad-Relación)**
DOM	*Document Object Model* **(Modelo de Objetos del Documento)**
DNN	*Deep Neuronal Network* **(Red Neuronal Profunda)**
GPIO	*General Purpose Input/Output* **(Entrada/Salida de Propósito General)**
HOG	*Histogram of Oriented Gradients* **(Histograma de Gradientes Orientados)**
HTML	*HyperText Markup Language* **(Lenguaje de Etiquetas de Hipertexto)**
HTTP	*HyperText Trasnfer Protocol* **(Protocolo de Transferencia de Hipertexto)**
IDE	*Integrated Development Envirnoment* **(Ambiente Integrado de Desarrollo)**
IoT	*Internet of Things* **(Internet de las Cosas)**
JSON	*JavaScript Object Notation* **(Notación de Objetos de JavaScript)**
NFC	*Near Field Communication* **(Comunicación de Campo Cercano)**
PIN	*Personal Identificaion Number* **(Número de Identificación Personal)**
RESTful	*Representational State Transfer* **(Trasnferencia de Estado Representacional)**
RGB	*Red Green Blue* **(Rojo Verde Azul)**
RTC	*Real Time Clock* **(Reloj en Tiempo Real)**
SBC	*Single-Board Computer* **(Computadora de Placa Única)**

SVM	*Support Vector Machine* **(Máquina de Vectores de Soporte)**
SSH	*Secure Shell* **(Shell Seguro)**
URL	*Uniform Resource Locator* **(Localizador de Recursos Uniforme)**
Venv	*Virtual Environment* **(Ambiente Virtual)**
VLC	*VideoLAN Client* **(Cliente VideoLAN)**

CAPÍTULO I: INTRODUCCIÓN

1.1 Justificación

El mundo actual se ha visto profundamente afectado por el continuo desarrollo de las tecnologías, lo que convierte a la ciberseguridad en un campo crucial para garantizar la seguridad de la información y el acceso a diversos sistemas e instalaciones. Las amenazas cibernéticas han incrementado la necesidad de desarrollar soluciones innovadoras que aseguren el control de acceso a diferentes entornos como oficinas, residencias o industrias.

Dada esta necesidad, surge la oportunidad de proponer un prototipo de un ecosistema de identidad de acceso basado en reconocimiento facial, que permitirá una autenticación biométrica segura y eficiente, proporcionando una mejora a los sistemas tradicionales mediante el uso de tecnologías emergentes.

La importancia de este trabajo de titulación radica en su capacidad de integrar tecnologías emergentes para ofrecer un prototipo de un sistema de acceso seguro, fácil de implementar y accesible desde el punto de vista económico, adaptado a las necesidades de un entorno moderno que busca fortalecer la seguridad de las instalaciones y permita registrar detalladamente accesos, generando estadísticas relevantes.

1.2 Planteamiento del Problema

Debido al aumento de amenazas cibernéticas, la seguridad y el control de acceso, tanto en instalaciones físicas como digitales, son más importantes que nunca. Los sistemas de autenticación tradicionales, basados en contraseñas, tarjetas o códigos PIN, presentan vulnerabilidades que han sido explotadas a través de técnicas como el phishing, la clonación o el uso indebido de credenciales. Esto pone en riesgo no solo la seguridad física de las instalaciones, sino también la integridad de los datos.

Frente a esta situación, el presente trabajo de titulación buscó desarrollar un prototipo de un ecosistema de identidad de acceso basado en reconocimiento facial, que incremente la seguridad mediante autenticación biométrica.

El problema principal es, por tanto, la vulnerabilidad de los sistemas de autenticación tradicionales y la falta de integración de tecnologías modernas que puedan aumentar la seguridad en tiempo real. Entre los problemas secundarios se incluyen la complejidad en la implementación de tecnologías IoT en sistemas de acceso, así como la gestión y protección

de los datos biométricos, los cuales deben ser manejados de manera segura y conforme a las normativas vigentes de privacidad.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un prototipo de un ecosistema de identidad de acceso basado en reconocimiento facial, mediante la implementación de Raspberry Pi como dispositivo principal para el procesamiento de datos biométricos y control de acceso en entornos físicos.

1.3.2 Objetivos Específicos

- Analizar los requisitos técnicos y funcionales, al igual que las limitaciones de Raspberry Pi, para implementar un sistema de acceso basado en reconocimiento facial.
- Diseñar la arquitectura del ecosistema, definiendo componentes de software y tecnologías IoT necesarias para la construcción del prototipo.
- Implementar un prototipo funcional que permita la autenticación biométrica y la recolección de datos de acceso.
- Evaluar la efectividad del sistema mediante la realización de pruebas funcionales y la corrección de defectos encontrados.

1.4 Alcance

En el presente trabajo de titulación se desarrolló un prototipo funcional de un sistema de control de accesos basado en reconocimiento facial, implementado sobre un Raspberry Pi 4. El sistema contó con la capacidad de identificar personas autorizadas mediante el análisis de imágenes capturadas por una cámara conectada al dispositivo, y registrar los accesos en una base de datos relacional.

El prototipo incluyó una interfaz web accesible desde una red local, que permitió visualizar la información recolectada en tiempo real, como los nombres de las personas reconocidas, las fechas y horas de detección, y una foto de la persona.

La comunicación entre los diferentes componentes se realizó mediante una API, que permite enviar y recibir datos entre el sistema de reconocimiento facial, la base de datos y la interfaz web. Este desarrollo estuvo enfocado en un entorno de pruebas controlado, con información básica, sin buscar una escalabilidad completa. No obstante, el prototipo sienta las bases para posibles ampliaciones futuras o adaptaciones en entornos reales.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2.1 Antecedentes

La seguridad y el control de acceso a instalaciones físicas y digitales han evolucionado como una prioridad crítica en un mundo hiperconectado. El incremento de amenazas cibernéticas ha revelado la insuficiencia de los sistemas de autenticación tradicionales. Métodos como códigos PIN, tarjetas NFC (Near Field Communication) y contraseñas, han demostrado ser vulnerables frente a técnicas como el phishing, clonación, ataques de fuerza bruta y el uso indebido de credenciales. Estas fallas no solo comprometen la seguridad física de las instalaciones, sino también la integridad de los datos.

En este sentido, los avances en la tecnología de reconocimiento facial han posibilitado nuevas vías para abordar estas deficiencias. Al identificar los rasgos distintivos e irreproducibles de los usuarios, los sistemas biométricos no solo mejoran la autenticación, sino que también ofrecen una opción más fiable y segura que las técnicas convencionales.

La integración de tecnologías IoT en sistemas de acceso plantea desafíos adicionales, como la complejidad técnica, los altos costos iniciales y la necesidad de manejar los datos biométricos cumpliendo con las normativas vigentes. El presente trabajo de titulación buscó ofrecer una solución prototipo que demuestre la vialidad de un ecosistema de identidad de acceso basado en reconocimiento facial, destacando su utilidad en entornos modernos donde la seguridad es esencial.

2.2 Internet de las Cosas (IoT)

El término IoT se utiliza para describir a los dispositivos integrados con conectividad a Internet, lo que les permite interactuar entre sí, con servicios y personas a escala global. Este nivel de conectividad puede aumentar la confiabilidad, la sostenibilidad y la eficiencia al mejorar el acceso a la información (Mukhopadhyay & Suryadevara, 2014).

La integración del reconocimiento facial con el paradigma del Internet de las Cosas (IoT) ha abierto nuevas oportunidades para la automatización y la seguridad en tiempo real. Por lo expuesto, Elordi et al. (2021) destacan que las plataformas IoT permiten combinar dispositivos con capacidades de procesamiento, censado y redes para crear ecosistemas interconectados.

Además, el uso de IoT en combinación con tecnologías de reconocimiento facial ha facilitado el desarrollo de soluciones para el control de acceso en hogares y oficinas, con

sistemas basados en Raspberry Pi. Este dispositivo, debido a su bajo costo y alta capacidad de procesamiento, se ha utilizado para implementar soluciones ligeras y eficientes en términos de consumo de energía (Syafeeza et al., 2020).

2.3 Sistemas de Control de Acceso basados en biometría

Como menciona el autor Sukhai (2004) el aseguramiento de la seguridad de los datos requiere una supervisión adecuada de cualquier acceso a la información. En este contexto, uno de los aspectos más cruciales es la intervención de las personas, quienes representan un "firewall humano" en el proceso. La base fundamental de un sistema de control de acceso se centra en proteger la confidencialidad, integridad y disponibilidad de los datos. Esto implica garantizar que la información esté protegida contra visualización no autorizada, que los datos se mantengan en su estado esperado en todo momento y que, al mismo tiempo, sean accesibles de manera eficiente, buscando un equilibrio entre la seguridad y la facilidad de acceso.

El componente principal de un sistema de control de acceso es la autenticación del individuo. Existen tres tipos principales de autenticación: aquella basada en el conocimiento, como contraseñas o preguntas de seguridad (algo que se sabe); aquella basada en objetos tangibles, como pasaportes, tarjetas de identificación o llaves (algo que se tiene); y finalmente, la que se basa en características físicas medibles del individuo, como huellas dactilares, firma, voz, iris, retina o geometría de la mano (algo que se es) (Sukhai, 2004)

Este último tipo de autenticación, basada en biometría, ha ganado cada vez más popularidad debido a su capacidad para identificar características físicas únicas de cada individuo. Si bien este enfoque aún está en una fase incipiente, se considera que la biometría desempeñará un papel crucial en los sistemas de seguridad del futuro (Sukhai, 2004). A lo largo de los años, se han evaluado diversas partes del cuerpo humano para determinar patrones únicos: desde las formas faciales y de las orejas, hasta la huella dactilar, el iris, la retina, la voz, el ADN, la marcha y las venas de las manos. Sin embargo, por razones de conveniencia, se han priorizado aquellas características físicas comúnmente visibles (Sukhai, 2004).

2.4 Reconocimiento facial en la seguridad

El reconocimiento facial es una de las modalidades biométricas más avanzadas y populares, utilizada tanto en sistemas de autenticación como en aplicaciones de seguridad y

control de acceso. Este método se ha convertido en una alternativa confiable frente a los sistemas tradicionales debido a su alta precisión y la dificultad para replicar rasgos únicos (Blanco-Gonzalo et al., 2018).

El reconocimiento facial presenta múltiples ventajas en comparación con otras tecnologías biométricas. Según Oloyede et al. (2020), esta técnica es más estable y menos susceptible a ser robada o clonada. Además, puede integrarse con tecnologías avanzadas, como redes neuronales convolucionales (CNN), que ofrecen alta precisión en condiciones de iluminación variable, poses no frontales y baja resolución (Syafeeza et al., 2020). El reconocimiento facial no requiere contacto físico, lo que lo convierte en una herramienta ideal para aplicaciones donde la higiene o la comodidad son críticas, como en dispositivos móviles o sistemas de control de acceso sin llave (Bagchi et al., 2022).

A pesar de sus ventajas, el reconocimiento facial enfrenta desafíos importantes, especialmente en entornos no controlados. Estos incluyen variaciones en la iluminación, expresiones faciales, envejecimiento y oclusiones parciales del rostro (Masud et al., 2020). Sin embargo, el uso de técnicas de aprendizaje profundo ha permitido superar estas limitaciones. Por ejemplo, las redes neuronales profundas (DNNs) y las técnicas de extracción de características automatizadas han desplazado a los métodos tradicionales, mejorando el rendimiento.

2.5 Machine Learning

El machine learning, deep learning y las redes neuronales son subcampos de la inteligencia artificial, sin embargo, las redes neuronales son en realidad un subcampo del machine learning, y el deep learning es un subcampo de las redes neuronales. La forma en que el deep learning y el machine learning difieren es en cómo aprende cada algoritmo. El "deep" machine learning puede utilizar conjuntos de datos etiquetados, también conocido como aprendizaje supervisado, para informar a su algoritmo, pero no requiere necesariamente un conjunto de datos etiquetados (IBM, 2021a).

2.6 Histogram of Oriented Gradients (HOG)

El Histograma de Gradientes Orientados (HOG) es una técnica ampliamente utilizada en visión por computadora e imagen digital para detectar objetos y reconocer patrones. La idea básica de HOG es capturar la estructura y la textura de una imagen, enfocándose particularmente en los gradientes, que representan los cambios en la intensidad de la imagen

(Jain, 2024). Esta técnica se utiliza principalmente en tareas como detección de objetos, detección de peatones y clasificación de imágenes, debido a su efectividad para describir las características locales de las imágenes de una manera robusta.

En el caso de HOG, los gradientes no se calculan en todas las direcciones posibles desde 0° hasta 360° para cada píxel. En cambio, los gradientes se calculan en direcciones específicas y luego se agrupan en contenedores de orientación, generalmente en un rango de 0° a 180° (Jain, 2024). El proceso comienza calculando los gradientes en cada píxel de la imagen, utilizando los gradientes en las direcciones Gx y Gy, que representan las variaciones en la intensidad de la imagen en los ejes horizontal y vertical, respectivamente. Estos gradientes se cuantifican en direcciones predefinidas y luego se almacenan en un histograma, lo que produce un descriptor numérico que representa las características de la imagen (Skillcate AI, 2022).

Para mejorar la precisión del descriptor HOG, la imagen se divide en pequeñas regiones denominadas celdas. En cada celda, se calculan los gradientes, y luego estos se normalizan para minimizar la influencia de variaciones en la iluminación de la imagen. Este proceso de normalización asegura que las características extraídas sean consistentes independientemente de las condiciones de iluminación, lo que hace que HOG sea resistente a los cambios en las condiciones ambientales de las imágenes (Ahamed et al., 2018). Posteriormente, las celdas se agrupan en bloques y los histogramas de orientación se combinan para formar un descriptor global que representa la imagen de manera efectiva.

Una de las ventajas más notables de HOG es su invariancia ante cambios de escala y rotación. Esto significa que el descriptor sigue siendo robusto incluso si la imagen se escala o rota, lo que lo convierte en una opción ideal para la detección y reconocimiento de objetos en condiciones diversas (Ahamed et al., 2018). Este enfoque ha demostrado ser muy eficaz en aplicaciones como el reconocimiento facial, donde las variaciones en la escala y la rotación de las caras son comunes.

2.7 Redes Neuronales Convolucionales (CNN)

Son modelos de deep learning diseñados para procesar datos con una estructura de tipo grid, como las imágenes. Estas redes aplican filtros o kernels sobre la imagen de entrada para extraer características relevantes, como bordes, texturas o formas (Chauhan et al., 2018)

Una CNN está compuesta principalmente por tres tipos de capas: convolucional, de agrupamiento (pooling) y completamente conectada; cada una cumple una función específica y, al combinarse, permiten que la red aprenda a identificar patrones en una imagen de manera progresiva. Al principio, detectan detalles simples como bordes o colores. A medida que se avanza por las capas, la red empieza a reconocer formas más complejas, hasta identificar objetos completos. En conjunto, estas capas permiten que una CNN pueda identificar patrones y reconocer objetos en imágenes con gran precisión (IBM, 2021b).

Capa convolucional: Es la capa principal de una CNN, usa filtros pequeños que se mueven por la imagen para detectar elementos básicos como bordes o colores. A medida que se agregan más capas, la red va reconociendo formas más complejas.

Capa de agrupamiento (Pooling): Esta capa reduce el tamaño de los datos y simplifica la información. Ayuda a que el modelo trabaje más rápido y a que se enfoque solo en los detalles más importantes de la imagen.

Capa completamente conectada: Es la última capa y se encarga de tomar una decisión, usa toda la información aprendida por las capas anteriores para clasificar lo que hay en la imagen, como si fuera una etiqueta final.

2.8 Raspberry Pi como plataforma de implementación

El Raspberry Pi es una computadora de placa única (SBC, por sus siglas en inglés) de bajo costo y tamaño compacto, desarrollada por la Fundación Raspberry Pi, diseñada originalmente para fomentar la enseñanza de la programación y la informática, su versatilidad y capacidad para ejecutar una variedad de sistemas operativos basados en Linux la han convertido en una herramienta popular para proyectos de electrónica, domótica e Internet de las Cosas (IoT) (Raspberry, 2021).

2.8.1 Ventajas

El Raspberry Pi destaca por ser un dispositivo económico y de fácil acceso a nivel mundial. Cuenta con un soporte periférico extenso gracias a sus 26 pines GPIO (General Purpose Input/Output), lo que permite la conexión sencilla de sensores y otros dispositivos externos, incluyendo la mayoría de los periféricos compatibles con Arduino. Además, tiene la capacidad de trabajar con múltiples sensores simultáneamente, lo que facilita el desarrollo de proyectos embebidos innovadores. Otro de sus puntos fuertes es la compatibilidad con una amplia variedad de lenguajes de programación, tales como C, C++, C#, Java, Ruby y

Python. Comparado con otras placas como Arduino, el Raspberry Pi incorpora un procesador más rápido, mejorando significativamente la velocidad y el tiempo de respuesta. Finalmente, al conectarle una pantalla, puede funcionar como una minicomputadora económica, con acceso a aplicaciones comunes como VLC o Google Chrome (Dipak Ghael et al., 2008).

2.8.2 Limitaciones

A pesar de sus múltiples ventajas, el Raspberry Pi también presenta algunas limitaciones. Una de ellas es la ausencia de un controlador multimedia integrado, ya que utiliza tarjetas SD como medio de almacenamiento, lo que implica velocidades de lectura y escritura más lentas y, en consecuencia, un tiempo de arranque más prolongado. Asimismo, no cuenta con un procesador gráfico dedicado, por lo que no es ideal para tareas que requieren un alto rendimiento visual, como la edición de video o los videojuegos. Otro inconveniente es la falta de sistemas de disipación térmica, como ventiladores o disipadores, ya que el dispositivo no los incorpora de forma predeterminada. No obstante, estos pueden ser añadidos externamente, lo cual permite mitigar el sobrecalentamiento que puede presentarse tras varias horas de uso continuo. Además, no es compatible con el sistema operativo Windows tradicional, lo que podría limitar su uso en ciertos entornos o con software específico. Por último, el Raspberry Pi carece de un Reloj en Tiempo Real (RTC) con batería integrada, lo que dificulta mantener la hora exacta cuando el dispositivo se apaga (Dipak Ghael et al., 2008).

2.9 Sistema Integrado

Este sistema se compone principalmente de una base de datos relacional, un framework web como Flask, y una interfaz de comunicación tipo API. Juntos, estos elementos forman un ecosistema que permite integrar módulos de reconocimiento facial con servicios web interactivos, visualización en tiempo real y registro seguro de accesos.

2.9.1 Base de Datos Relacional

Una base de datos relacional organiza los datos en tablas compuestas por filas y columnas, donde los datos están relacionados mediante claves primarias y foráneas. Esta estructura permite conectar múltiples tablas facilitando la integridad y consistencia de la información (IBM, 2024).

Este tipo de bases de datos se asocia comúnmente a sistemas transaccionales que cumplen con las propiedades ACID: atomicidad, coherencia, aislamiento y durabilidad.

Estas propiedades aseguran que las operaciones se realicen de forma segura y sin errores parciales, como en el caso de una transferencia bancaria (IBM, 2024).

2.9.2 Flask

Flask es un microframework web desarrollado en Python que permite crear aplicaciones web de forma simple y modular. Ofrece herramientas esenciales como enrutamiento de URLs, manejo de solicitudes del cliente, renderizado de páginas HTML y conexión con bases de datos, todo sin imponer una estructura rígida. Esto facilita el desarrollo rápido de sistemas personalizados y adaptables a distintos requerimientos (Jack Chan, 2019).

2.9.3 API (Interfaz de Programación de Aplicaciones)

Una API es un conjunto de definiciones y protocolos que permiten que distintas aplicaciones se comuniquen entre sí. Funciona como un contrato entre dos partes: una envía una solicitud con una estructura específica, y la otra responde conforme a esa estructura. El uso de APIs simplifica el diseño, mantenimiento e integración de aplicaciones, permitiendo flexibilidad y fomentando la reutilización del código (Red Hat, 2023).

CAPÍTULO III: DISEÑO DEL ECOSISTEMA

3.1 Diseño de la Arquitectura del Sistema

El desarrollo del sistema de control de acceso mediante reconocimiento facial implicó una arquitectura compuesta por distintos elementos tanto de hardware como de software. Dicha arquitectura debe garantizar un flujo eficiente de datos desde la captura de imágenes, hasta su procesamiento, almacenamiento y visualización. En este capítulo se define cómo se estructura y conecta cada uno de los componentes que forman parte del ecosistema.

El prototipo parte del uso de un Raspberry Pi como dispositivo de captura, con su módulo de cámara, y procesamiento primario, que se encarga de identificar rostros en tiempo real haciendo uso de modelos computacionales entrenados previamente. Estos datos luego se comunican con un servidor Flask que actúa como backend, el cual procesa peticiones, consulta la base de datos y responde a los clientes web. Finalmente, se cuenta con una interfaz web que permite la visualización de accesos en tiempo real.

El diseño modular permite que cada componente funcione de manera independiente, facilitando tareas como mantenimiento y monitoreo. Esta división favorece el análisis de cada parte del sistema, permitiendo realizar mejoras específicas sin comprometer todo el ecosistema (Universidad de Oviedo, 2006).

3.1.1 Diagrama General del Sistema

Figura 1: Arquitectura del Módulo de Entrenamiento.

Como se observa en la Figura 1, el proceso para el entrenamiento del modelo tiene su comienzo en el Raspberry Pi, específicamente su módulo para cámara. Utilizando la librería *OpenCV,* se capturan múltiples imágenes del rostro de la persona, las cuales son guardadas en un directorio con su nombre. Estas imágenes conforman el dataset de

entrenamiento personalizado, y son tomadas desde una interfaz que permite la captura de distintos ángulos y expresiones, mejorando la precisión del reconocimiento.

Una vez capturadas las imágenes, son procesadas con un script de Python. Dentro de este script se hace uso de la librería *face_recognition,* la cual nos permite detectar automáticamente los rostros y generar sus correspondientes codificaciones (Geitgey, 2022).

Finalmente, las codificaciones son serializadas mediante la librería *Pickle* y almacenadas en la base de datos haciendo uso de *psycopg2*. Las codificaciones son insertadas en la tabla **encodings,** asociándolos mediante una clave foránea con la tabla **personas.** Esto permite que, durante el reconocimiento, los vectores se consulten y comparen en tiempo real.

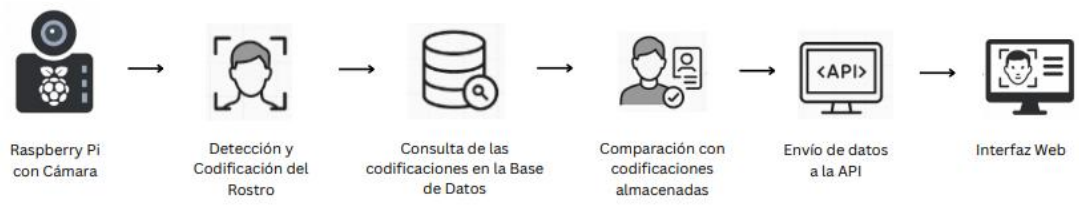


Figura 2: Arquitectura del Módulo de Detección y Registro

Por otro lado, la Figura 2 evidencia que el procedimiento para el reconocimiento involucra más aspectos. El módulo de cámara del Raspberry Pi realiza capturas en tiempo real haciendo uso de *OpenCV*. Cada imagen es procesada por la librería *face_recognition* que primero localiza los rostros y luego genera una codificación del rostro detectado.

Una vez codificado el rostro, el sistema consulta a la base de datos las codificaciones almacenadas y sus respectivos identificadores para cada persona. Es decir, almacena en un arreglo la codificación de cada imagen del dataset y su respectivo ID correspondiente a la persona a la que pertenece dicha codificación. Se extraen los vectores y se comparan uno a uno con la codificación actual para determinar la similitud. En caso de existir una coincidencia dentro de un umbral establecido, se identifica a la persona y se registra el acceso.

Tras la identificación, los datos de acceso, incluyendo el ID de la persona, se envían mediante una API construida en *Flask*, que actúa como puente entre el backend de reconocimiento y el frontend. La interfaz web desarrollada con HTML, CSS, JavaScript y

Bootstrap, muestra en tiempo real los accesos recientes, y se actualiza dinámicamente para reflejar la información de manera inmediata.

3.2 Componentes del Ecosistema

El sistema de control de acceso mediante reconocimiento facial se compone de dos grandes bloques: los componentes de hardware y los componentes de software, los cuales trabajan conjuntamente para garantizar la captura, procesamiento, comparación y visualización de los datos eficientemente. Esta arquitectura busca aprovechar la capacidad de procesamiento local de la microcomputadora, complementado por una interfaz web que facilita la supervisión del sistema en tiempo real.

Esta separación entre hardware y software no solo permite estructurar el sistema de forma clara y funcional, sino que también fomenta la modularidad, facilitando la implementación y mantenimiento. Esta arquitectura modular garantiza una operación fluida en cada etapa del proceso, desde la captura de datos hasta su visualización en la interfaz web.

3.2.1 Componentes de Hardware

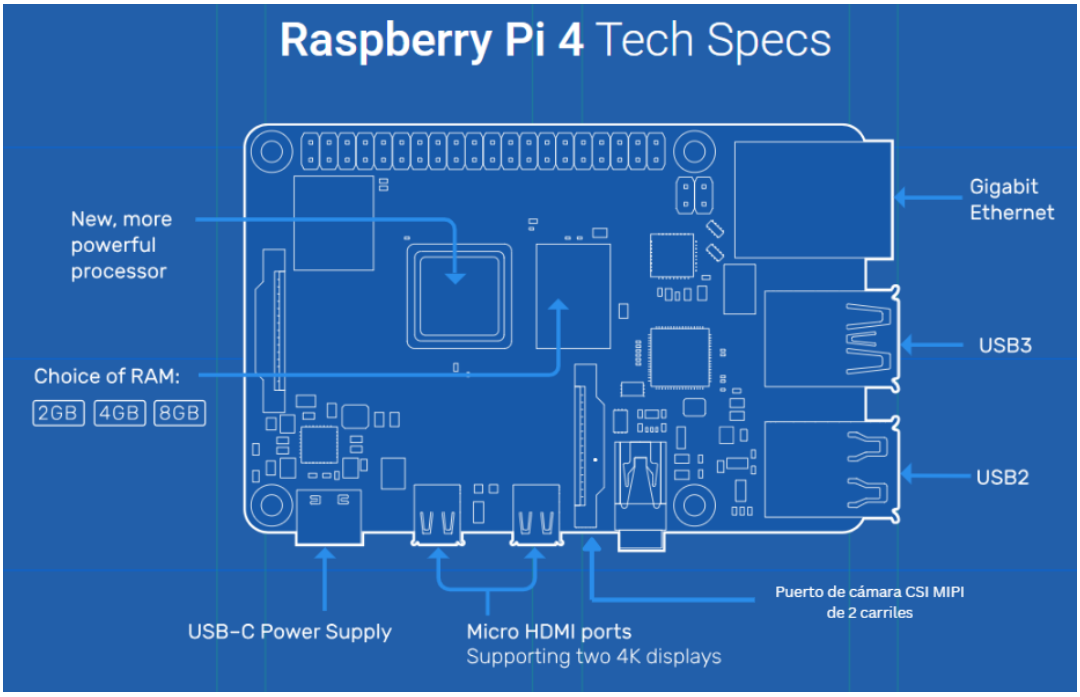


Figura 3: Esquemática del Raspberry Pi 4.

Nota: Adaptado de Raspberry Pi (2025).

Como se representa en la Figura 3, el sistema está construido sobre una microcomputadora Raspberry Pi 4 Modelo B con 8 GB de memoria RAM, seleccionada por

su equilibrio entre bajo consumo energético y capacidad de procesamiento suficiente para poder ejecutar el reconocimiento facial en tiempo real.

Esta microcomputadora actúa como el núcleo del sistema, controlando tanto la captura de imágenes como su procesamiento local. Además, su capacidad de conectividad (WiFi, Ethernet y USB 3.0), permite integrar el dispositivo fácilmente a una red para la comunicación con la base de datos (Raspberry Pi, 2025).

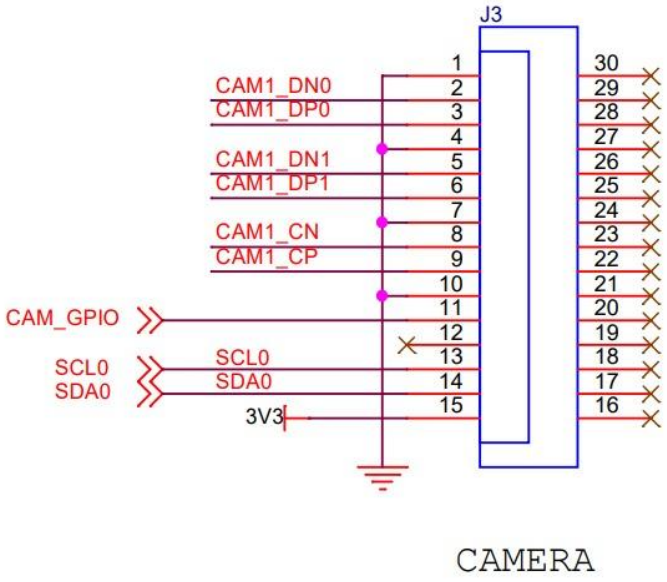


Figura 4: Puerto CSI MIPI para la Cámara.

Nota: Tomado de Adams (2019).

De acuerdo con lo representado en la Figura 4, para la captura de imágenes se utiliza una cámara conectada mediante la interfaz MIPI CSI (Camera Serial Interface), un estándar desarrollado por la Alianza MIPI para la conexión de sensores y periféricos a procesadores embebidos. Esta interfaz permite la transmisión eficiente de datos de imagen con bajo consumo energético (SINOSEEN, 2024).

La versión más utilizada, MIPI CSI-2, se caracteriza por su alto rendimiento y su compatibilidad con múltiples aplicaciones, como dispositivos móviles y del Internet de las Cosas (IoT). Esta versión soporta múltiples resoluciones que van desde 1080p hasta 8K, lo que lo convierte en una opción robusta para el reconocimiento facial (MIPI, 2025).

3.2.2 Componentes de Software

El ecosistema de software se basa principalmente en Python, un lenguaje de programación altamente utilizado en computación visual por su versatilidad, sintaxis clara y la amplia disponibilidad de librerías especializadas. Estas características lo convierten en una herramienta ideal para el desarrollo de sistemas de reconocimiento facial, ya que permite integrar de manera sencilla tanto el procesamiento de imágenes como la gestión de bases de datos e interfaces web. De igual manera, su comunidad activa y documentación accesible aceleran el desarrollo y la resolución de problemas durante la implementación.

El correcto funcionamiento del sistema depende en gran medida del uso coordinado de diversas librerías, que permiten desde la detección hasta la conexión con la base de datos. Estas herramientas trabajan en conjunto facilitando la integración de los distintos módulos. Algunas de las librerías más relevantes empleadas son:

- **face_recognition: e**sta librería permite detectar y reconocer rostros mediante modelos de Deep Learning basados en *dlib,* con una precisión del 99.38% en el benchmark *Labeled Faces in the Wild* (Geitgey, 2022). En el presente trabajo, esta librería se empleó para generar y comparar las codificaciones faciales, permitiendo identificar personas en tiempo real.
- **OpenCV (cv2):** es una librería open-source que contiene una amplia gama de algoritmos para computación visual altamente usada para el procesamiento de imágenes en tiempo real (OpenCV, 2025). En el presente trabajo, esta librería permitió capturar las imágenes para la construcción del dataset y detectar los rostros en tiempo real.
- **NumPy: e**sta librería proporciona objetos de arreglos multidimensionales y rutinas optimizadas para realizar operaciones matemáticas, lógicas, manipulación de formas, ordenamiento y más (NumPy, 2024). Se empleó esta librería para calcular y encontrar el índice de la mejor coincidencia entre un rostro detectado y las codificaciones almacenadas.
- **Psycopg2: l**ibrería que permite conectar el motor de base de datos PostgreSQL con Python. Es decir, funciona como un adaptador que permite realizar múltiples transacciones concurrentes (Psycopg, 2021).
- **Pickle: e**ste módulo implementa protocolos binarios para serializar y deserializar una estructura de objetos. Es decir, permite convertir objetos en un flujo de bytes (Python,

2025a). Pickle fue utilizado para guardar y cargar las codificaciones de las características faciales previamente entrenadas y guardadas en la base de datos.

- **Threading: e**sta librería permite ejecutar múltiples hilos de forma simultánea (Python, 2024). Se la utilizó para ejecutar el proceso de detección y envío de datos sin bloquear la interfaz principal.
- **Flask:** es un microframework web desarrollado en Python que permite crear aplicaciones web de forma simple y modular (Jack Chan, 2019). En este trabajo, se encarga de gestionar el backend del sistema, coordinando la comunicación entre el reconocimiento facial, la base de datos y la interfaz web.
- **Requests:** librería que permite realizar peticiones HTTP de manera sencilla y eficiente (PyPI, 2024). Se utiliza para enviar datos del sistema de reconocimiento facial al servidor Flask y para mostrar los datos en la interfaz web.

3.3 Selección del Motor de Base de Datos

Tabla 1: *Comparativa de DBMS Open Source*

Característica	PostgreSQL	MySQL	MongoDB
Licencia	PostgreSQL License	GPL (General Public License)	SSPL (Server-Side Public License)
Tipo	Relacional	Relacional	NoSQL
Tipos de Datos	Muy amplio: numéricos, texto, archivos JSON, objetos, arreglos de bytes, geoespaciales.	Básico: Numéricos, texto, fecha, hora	Flexible: JSON y BSON (Binary JSON)
Soporte ACID	Completo	Parcial	Parcial
Alta Disponibilidad	Soporte nativo para replicación, failover y clusterización	Posible mediante configuración adicional	Posible mediante la configuración de replicación
Rendimiento	Alto rendimiento en consultas complejas gracias a su optimización	Alto en escritura y muy variable en casos de lectura	Muy alto en escritura y lectura con grandes volúmenes de datos

Nota: La presente tabla compara las características de DBMS como PostgreSQL, MySQL y MongoDB. Información tomada de AWS (2024) y Deymar (2023).

En base a esta información, se ha decidido que para el presente trabajo se hará uso de PostgreSQL debido a su licencia de uso público y su gran rendimiento para consultas gracias a sus técnicas de indexación y optimización. Aunque MongoDB resulta atractivo por

su flexibilidad al manejar datos no estructurados, como las codificaciones de los rostros, PostgreSQL ofreció una solución más completa permitiendo almacenar dichas codificaciones en campos como BYTEA sin perder las ventajas de un sistema relacional. Esta ventaja fue fundamental para mantener relaciones claras y eficientes entre los datos biométricos y la información de cada persona.

3.4 Diseño de la Base de Datos

El modelado de datos contempla tres tablas principales: personas, encodings y accesos. La tabla personas almacena información básica sobre cada individuo registrado en el sistema, además de una foto obtenida al momento de crear el dataset. La tabla encodings guarda las codificaciones de las características faciales de cada persona. Finalmente, la tabla accesos registra cada detección, incluyendo la hora y la imagen del rostro detectado.

El diseño sigue un modelo relacional clásico, donde las tablas se relacionan mediante claves foráneas. Por ejemplo, cada registro en la tabla encodings hace referencia al identificador único de cada persona, lo que permite identificar rápidamente a quien pertenece una codificación determinada.

3.4.1 Diagrama Entidad Relación

El sistema utiliza un modelo entidad-relación (ER-D) claro y normalizado, diseñado para garantizar la integridad de los datos y facilitar consultas. De acuerdo con lo representado en la Figura 5, se presentan las tres entidades principales del sistema: personas, encodings y accesos, así como sus relaciones lógicas.

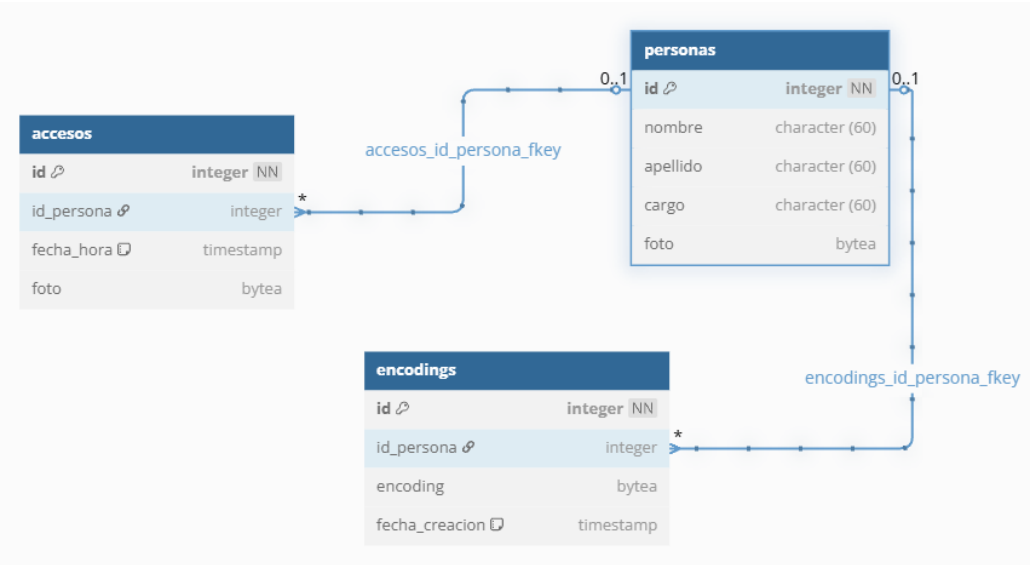


Figura 5: Diagrama Entidad Relación.

Cada entidad corresponde a una tabla en la base de datos PostgreSQL y está relacionada mediante claves foráneas como *id_persona* que definen relaciones uno a muchos. Este enfoque permite una estructura modular y escalable, ideal para aplicaciones donde la trazabilidad y consistencia son críticas. A continuación, se explican las relaciones:

- Una persona puede tener múltiples registros de accesos, lo cual permite almacenar un historial de detecciones con marca de tiempo y evidencia fotográfica en la tabla accesos.
- Una persona puede tener múltiples codificaciones faciales, generadas en distintos momentos para actualizar o mejorar la precisión del reconocimiento. Estas codificaciones se almacenan en la tabla encodings, en formato binario (BYTEA) junto con su fecha de creación.
- Recíprocamente, cada registro en accesos pertenece únicamente a una persona, asegurando que todo acceso esté vinculado a un usuario específico del sistema.
- De igual manera, cada codificación en encodings está asociada a una sola persona, evitando ambigüedades en la identificación biométrica.
- La entidad personas actúa como el núcleo del modelo, ya que está relacionada directamente con los registros de los accesos y las codificaciones, formando una estructura de relaciones uno-a-muchos desde personas hacia accesos y encodings, pero uno-a-uno en sentido inverso.

Este diseño relacional refleja un enfoque centrado en trazabilidad, coherencia y precisión de los datos, aspectos fundamentales en un sistema de control de acceso biométrico. La aplicación de reglas de integridad referencial impide, por ejemplo, la eliminación de personas con accesos registrados o codificaciones asociadas. Esto protege la coherencia del sistema ante errores, y garantiza la unicidad de cada codificación y acceso con su respectiva persona. |

3.5 Interacción entre Componentes

La interacción entre componentes del sistema es esencial para asegurar que la captura facial, la verificación de identidad, el registro de accesos y la visualización en la interfaz web ocurran de manera fluida. Dicho flujo de trabajo debe estar optimizado para garantizar una experiencia eficiente para los usuarios.

El diseño del sistema contempla varios componentes distribuidos que trabajan en conjunto: el hardware de captura (Raspberry Pi con cámara), el módulo de reconocimiento (modelo de reconocimiento facial), el backend desarrollado en Flask, el cual expone una API, la base de datos PostgreSQL y la interfaz web para la visualización de accesos. Esta segmentación permite una separación de responsabilidades y facilita el mantenimiento.

3.5.1 Flujo de Datos desde Captura hasta Visualización

Figura 6: Flujo de Datos para la Captura.

La Figura 6 representa el flujo de datos desde la detección de un rostro mediante la cámara conectada al Raspberry Pi. Una vez identificado el rostro, se realiza la captura de imágenes, que servirán para construir el dataset de entrenamiento para el modelo. Este conjunto de datos se organiza en carpetas, cada una etiquetada con el nombre de la persona correspondiente, lo que permite entrenar el modelo con múltiples individuos de manera estructurada.

Una vez construido el dataset, este se procesa localmente por el modelo de reconocimiento facial, el cual generará un vector de características (conocido como encoding) para cada imagen, mediante técnicas como HOG (Histogram of Oriented Gradients). Estas codificaciones se almacenan en la base de datos, asociándolas con el identificador único de la persona a la que pertenecen.

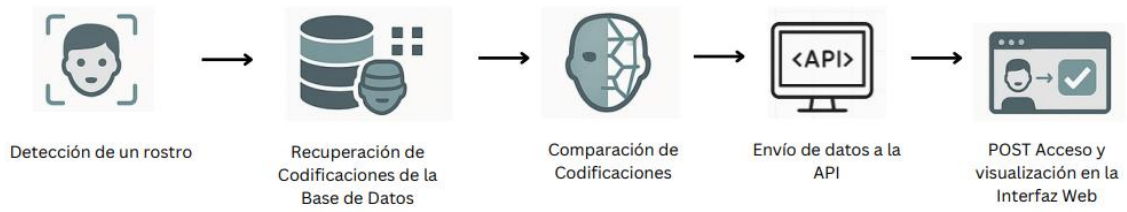


Figura 7: Flujo de Datos para el Reconocimiento.

Por otro lado, el flujo de datos para el reconocimiento inicia de manera idéntica al entrenamiento, detectando un rostro mediante la cámara conectada al Raspberry Pi como se observa en la Figura 7. En esta etapa, se capturan los fotogramas en tiempo real y se codifican

las características faciales presentes en la imagen. Esta detección da paso a la búsqueda de coincidencias con los datos almacenados.

Una vez detectado un rostro, el sistema realiza la recuperación de las codificaciones faciales desde la base de datos. Estas codificaciones, previamente generadas y almacenadas durante el entrenamiento, representan de manera matemática las características únicas del rostro de cada persona. Posteriormente, la codificación del rostro detectado es comparada con las codificaciones almacenadas, permitiendo detectar si existe alguna coincidencia.

En caso de encontrar una coincidencia, se procede con el envío de los datos a través de la API, incluyendo información como el identificador de la persona, la imagen capturada, en caso de ser alguien desconocido, y la hora de acceso. Esta información es posteada en la interfaz web, donde se la visualiza de manera clara y en tiempo real quién accedió.

3.5.2 Peticiones a la API y respuestas

La API se diseña para gestionar los accesos, proporcionando un punto central de interacción para registrar entradas y recuperar esta información de manera estructurada. Este enfoque facilita la integración y la gestión eficiente del sistema.

Tabla 2*:Peticiones a la API*

Ruta	Método	Descripción	Salida
/accesos	POST	Registra un nuevo registro de acceso con foto	Acceso Registrado
/accesos	GET	Devuelve los accesos registrados con las imágenes codificadas	Muestra de información de la persona en la interfaz
/personas	GET	Devuelve todos los registros de personas o los datos de la persona con el id especificado	Muestra la información de la persona desde la base de datos
/personas	POST	Registra una nueva persona en la base de datos	Persona Agregada

Nota: La presente tabla muestra las principales peticiones a la API y sus respectivas respuestas.

El sistema implementa una API RESTful como interfaz de comunicación entre el frontend y la base de datos, con el fin de proteger la integridad de los datos y evitar accesos directos no controlados. Esta API permite encapsular la lógica y asegurar que toda la interacción con la base de datos pase por una capa intermedia (Gupta, 2024).

Aunque la API está principalmente orientada a la gestión de los accesos, registrando entradas de personas detectadas por el sistema de reconocimiento facial y recuperando estos datos en formato JSON, también contempla operaciones básicas sobre la tabla **personas,** como agregar o listar individuos. Esto permite mantener un control centralizado de los usuarios autorizados, enlazándolos con sus respectivas codificaciones faciales y sus registros de accesos.

La Figura 8 ilustra el flujo de comunicación entre el cliente web, la API y la base de datos, mostrando cómo las peticiones se realizan y cómo se procesa la información para satisfacer las solicitudes.

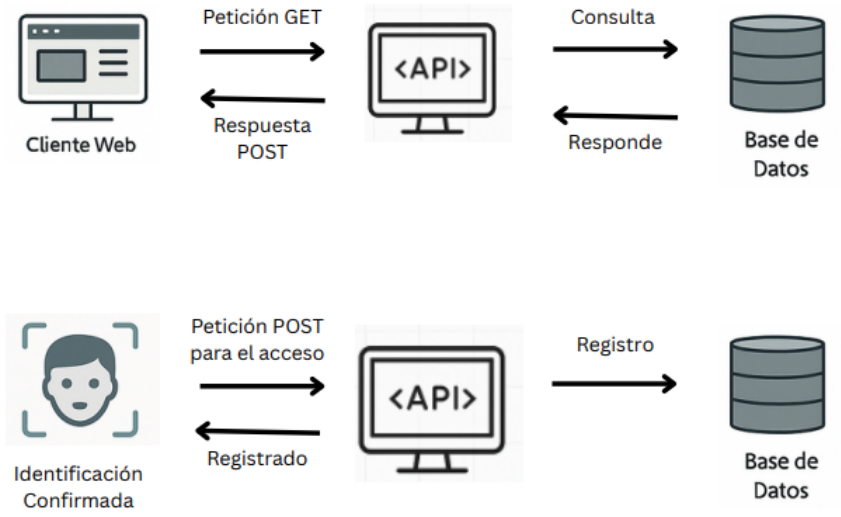


Figura 8: Peticiones a la API.

Esta estructura responde a una arquitectura cliente-servidor de tres niveles. La primera capa es la de presentación, representada por el frontend desarrollado en HTML, CSS, JavaScript y Bootstrap. La segunda capa corresponde al middleware, que incluye la API construida con Flask y la lógica de reconocimiento facial. Finalmente, la tercera capa es la base de datos PostgreSQL. Este diseño modular facilita la escalabilidad, mejora la seguridad y permite actualizar cada componente de forma independiente (Gupta, 2024). La Figura 9 presenta la arquitectura cliente-servidor de tres niveles.

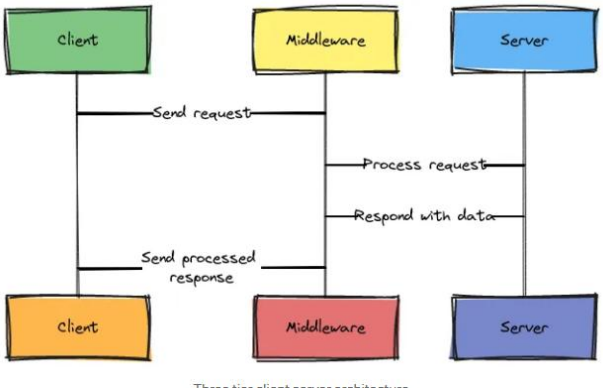


Figura 9: Arquitectura Cliente-Servidor de tres niveles.

Nota: Tomado de Gupta (2024).

CAPÍTULO IV: DESARROLLO DEL PROTOTIPO

4.1 Configuración del Entorno de Desarrollo

El desarrollo del prototipo comenzó con la preparación del entorno de trabajo, tanto a nivel de hardware como de software. Para garantizar la compatibilidad y el correcto funcionamiento de los distintos componentes, se optó por el uso de un Raspberry Pi 4 con 8 GB de memoria RAM como dispositivo central, en el cual se instaló el sistema operativo Bookworm. Este entorno se eligió debido a su ligereza, comunidad activa y documentación.

Desde el punto de vista del software, se optó por utilizar Python como lenguaje principal de desarrollo debido a su simplicidad, versatilidad y la alta disponibilidad de librerías especializadas en computación visual y procesamiento de imágenes como OpenCV y face_recognition (Batta, 2024).

En cuanto al entorno de desarrollo integrado (IDE) se optó por el uso de Thonny, una herramienta ligera y preinstalada en muchas distribuciones de Raspberry Pi OS. Thonny destaca por su interfaz intuitiva, ideal para desarrolladores que buscan simplicidad. Su integración directa con Python y su terminal embebida permitieron una depuración directa y ejecución eficiente del código dentro del mismo entorno físico del Raspberry Pi (Barnes, 2017).

4.1.1 Configuración del entorno de desarrollo integrado (IDE)

La configuración del entorno de desarrollo de desarrollo se llevó a cabo utilizando el editor Thonny, el cual viene preinstalado en Raspberry Pi OS. Este entorno fue escogido debido a su integración directa con Python 3.10. Una característica destacada de Thonny es su depurador visual por estructuras, que en lugar de mostrar el flujo del programa línea por línea, permite observar bloques completos. Además, su representación del llamado a funciones resulta muy didáctica, ya que abre una nueva ventana donde se visualizan las variables locales y un puntero al código en ejecución. Finalmente, Thonny permite el uso directo de entornos virtuales como intérprete, lo que mejora la organización del proyecto y evita conflictos con dependencias del sistema (Thonny, 2025).

Un ambiente virtual en Python es un entorno aislado que permite gestionar paquetes y dependencias de forma independiente para cada proyecto, evitando conflictos con otros entornos o el sistema operativo, como se representa en la Figura 10. Son creados dentro de

una carpeta (usualmente llamada venv o .venv) y contiene su propio intérprete de Python y herramientas como pip (Python, 2025b).

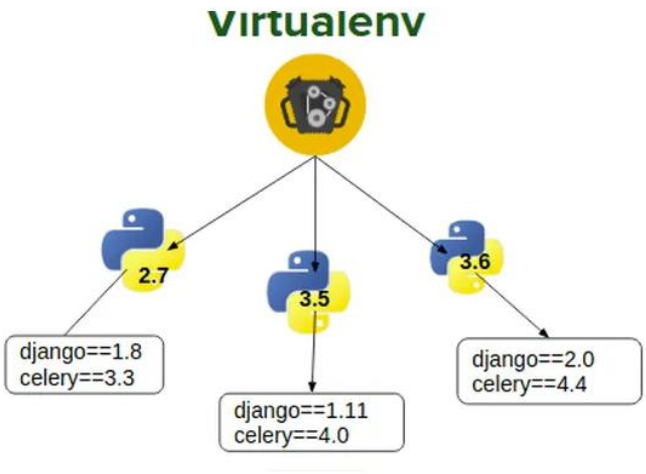


Figura 10: Entorno Virtual en Python.

Nota: Tomado de Burak (2021).

Una vez activado el entorno virtual desde la terminal del sistema, se procedió con la instalación de librerías necesarias mediante el gestor de paquetes pip. De esta manera, se aseguraron versiones compatibles de las dependencias requeridas, garantizando la estabilidad y funcionalidad del prototipo. Se priorizó la instalación de versiones estables y bien documentadas de bibliotecas clave como face_recognition, OpenCV y Flask.

Una vez listo el ambiente virtual en el que se desarrollarían los diferentes scripts del prototipo, se procedió a configurar el editor Thonny para hacer uso de dicho entorno como su intérprete principal. Esta acción permitió que Thonny reconociera automáticamente las librerías instaladas en el ambiente virtual, en lugar de las disponibles en el entorno global del sistema operativo.

Este enfoque estructurado facilitó además la creación de un archivo **requirements.txt**. Este archivo contiene un listado completo y detallado de todas las dependencias utilizadas en el prototipo. La utilidad de este archivo reside en que permite documentar el entorno de trabajo y facilita el despliegue del sistema en otros equipos, ya que basta con ejecutar una línea de comando para replicar el mismo conjunto de librerías, reduciendo de tal manera el margen de error por falta de dependencias en instalaciones futuras.

Tabla 3: *Librerías y herramientas utilizadas en el entorno de desarrollo*

Librería	Versión	Propósito
face_recognition	1.3.0	Reconocimiento facial
face_recognition_models	0.3.0	Modelos usados por la librería face_recognition
dlib	(por dependencia)	Algoritmos de Machine Learning (Base de la librería face_recognition)
opencv_python	4.11.0.86	Captura y procesamiento de imágenes
numpy	2.2.4	Cálculos numéricos y operaciones matriciales
types_psycopg2	2.9	Conexión con PostgreSQL
imutils	0.5.4	Funciones auxiliares para funciones básicas del procesamiento de imágenes
Flask	3.1.0	Servidor HTTP
flask_cors	5.0.1	Permitir las peticiones

Nota: La presente tabla muestras las librerías usadas para el desarrollo del prototipo y sus respectivas versiones.

4.1.2 Configuración del Raspberry Pi y sistema operativo Bookworm

La instalación inicial del sistema operativo se la hizo mediante Raspberry Pi Imager, una herramienta oficial que permite grabar imágenes de sistemas operativos en medios de almacenamiento. Se seleccionó la versión Raspberry Pi OS Bookworm con escritorio y controladores recomendados, lo que incluye una interfaz gráfica intuitiva y compatibilidad completa con los periféricos. Esta imagen fue quemada en una tarjeta microSD configurando previamente el nombre del host, las credenciales del usuario, la red Wi-Fi y la habilitación del acceso remoto vía SSH, tal como se representa en la Figura 11.

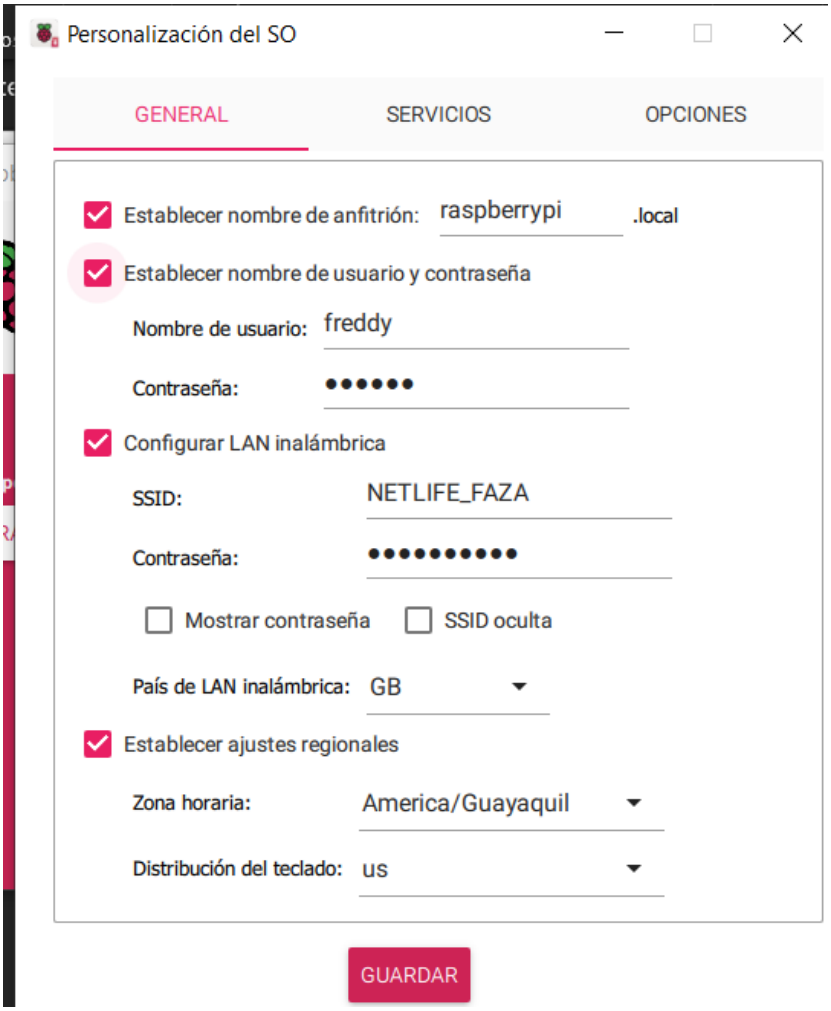


Figura 11: Personalización del Sistema Operativo

Una vez instalada la tarjeta microSD en el Raspberry Pi, el sistema fue iniciado y verificado para asegurar su correcto funcionamiento. Mediante el entorno gráfico se realizaron ajustes adicionales y pruebas de conectividad para validar la personalización previamente configurada.

Con el sistema base ya en funcionamiento, se procedió con la búsqueda e instalación de posibles actualizaciones del sistema operativo y de sus respectivos paquetes o dependencias. Esta tarea se la realizó desde la terminal mediante comandos, lo cual permitió mantener el sistema con ultimas mejoras de seguridad, estabilidad y compatibilidad.

Posteriormente, se realizó la conexión y configuración del módulo de cámara (PiCamera Module v1), el cual fue conectado al puerto CSI. A continuación, se presenta una tabla con las principales características técnicas del módulo:

Tabla 4: *Características técnicas PiCamera Module v1*

Especificación	PiCamera Module v1
Resolución de Imagen	5 megapíxeles
Modos de Video	1080p30
	720p60
	640x480p60/90
Sensor	OmniVision OV5647
Tamaño de Píxel	1.4 µm × 1.4 µm
Enfoque	Fijo
Profundidad	Aproximadamente 1m
Longitud Focal	3.60 mm +/- 0.01
Campo de Visión Horizontal	53.50 +/- 0.13 grados
Campo de Visión Vertical	41.41 +/- 0.11 grados

Nota: La presente tabla describe las principales características técnicas del módulo PiCamera v1 para el Raspberry Pi 4. con información tomada de Raspberry Pi (2025a).

El sistema operativo instalado, Bookworm, es la versión más reciente de Raspberry Pi OS, basada en Debian 12. Los cambios referentes a su versión anterior, Bullseye, representan un avance significativo en términos de rendimiento y seguridad. Entre las mejoras más notables se encuentra la migración del sistema gráfico X11 a Wayland, el uso del compositor Wayfire por defecto y la incorporación del nuevo sistema de audio PipeWire. Además, Bookworm incorpora NetworkManager como gestor de red y ofrece una versión optimizada del navegador Firefox para su arquitectura ARM.

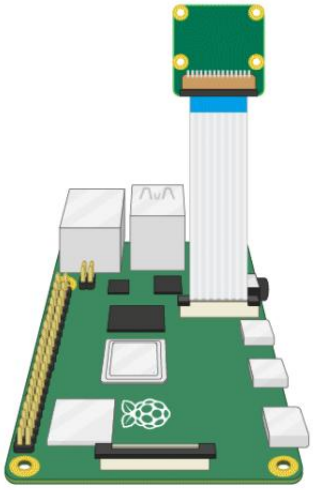


Figura 12: Raspberry Pi con su módulo cámara conectado.

Nota: La presenta Figura representa la configuración física final del prototipo posterior a la conexión del módulo cámara. Tomado de Raspberry Pi (2025b).

4.2 Implementación del Modelo de Reconocimiento Facial

En el presente trabajo de titulación no se diseñó ni entrenó completamente un modelo de reconocimiento facial desde cero. En su lugar, se optó por el uso de la librería **face_recognition,** que integra un modelo previamente entrenado, construido sobre **dlib.** Este modelo permite detectar y codificar rostros en imágenes mediante el uso de Histogramas de Gradientes Orientados (HOG), generando vectores numéricos (encodings) que representan características faciales únicas y que luego pueden ser comparadas entre sí.

Dlib es una biblioteca moderna de aprendizaje automático y visión por computadora desarrollada en C++ con enlace en Python. Se destaca por su enfoque modular y la integración de modelos estadísticos avanzados entrenados con algoritmos de machine learning. En el contexto del reconocimiento facial, Dlib proporciona herramientas de alto nivel para la detección, alineación y análisis geométrico del rostro, lo que la convierte en una de las bibliotecas más utilizadas en proyectos biométricos y sistemas de control de acceso (Dlib, 2022).

El papel del desarrollo consistió en alimentar dicho modelo preentrenado con datasets de personas autorizadas, generando sus correspondientes codificaciones y almacenándolas en la base de datos para su posterior comparación. La lógica general del sistema se basa en: capturar imágenes en tiempo real desde la cámara, detectar rostros y calcular sus codificaciones, comparar dichas codificaciones con las previamente registradas y determinar si existe una coincidencia.

4.2.1 Extracción de Características faciales mediante HOG y Dlib

La etapa de extracción de características faciales representa una de las fases más críticas dentro del sistema de reconocimiento facial, ya que permite transformar la imagen de un rostro en una representación matemática única que puede ser utilizada para su posterior identificación. En el presente prototipo, la tarea de extracción es ejecutada haciendo uso de la librería face_recognition que a rasgos generales y como lo describe Rosebrock (2021) integra las funciones de reconocimiento facial de **dlib** en una API sencilla y de fácil uso, junto con el modelo de detección de rostros basado en Histogramas de Gradientes Orientados (HOG) y un clasificador SVM (Máquina de Vectores de Soporte).

El método Histograma de Gradientes Orientados (HOG) es una técnica usada en la computación visual para poder detectar objetos en imágenes. Fue propuesto por Dalal y Triggs en 2005 como una técnica robusta para la detección de objetos, particularmente peatones. Su eficiencia lo ha convertido en una herramienta esencial en aplicaciones de reconocimiento facial. El principio fundamental de HOG es capturar la estructura/forma de un objeto, en este caso imágenes, describiendo los contornos y bordes a través de gradientes de intensidad (Singh, 2025). El proceso de extracción de características mediante HOG se podría describir en los siguientes pasos:

Preprocesamiento de Datos: Es necesario preprocesar la imagen para que tenga una relación de aspecto fija, específicamente 1:2, donde el tamaño preferido es 64 x 128 (Singh, 2025). Sin embargo, para este trabajo se redimensionaron las imágenes al 25% de su tamaño original para acelerar procesos y reducir tiempos de procesamiento al tener menos píxeles. De igual manera, es necesario transformar la imagen de formato BGR (azul, verde, rojo), usado por defecto por OpenCV, al formato RGB (rojo, verde, azul) ya que la librería face_recognition requiere este formato para funcionar correctamente, como se ejemplifica en la Figura 13. En caso de no estar en el formato correcto, los colores estarán mal interpretados y podría afectar al reconocimiento, de igual manera para que pueda funcionar la técnica HOG es necesario convertir la imagen a escala de grises (Geitgey, 2022).

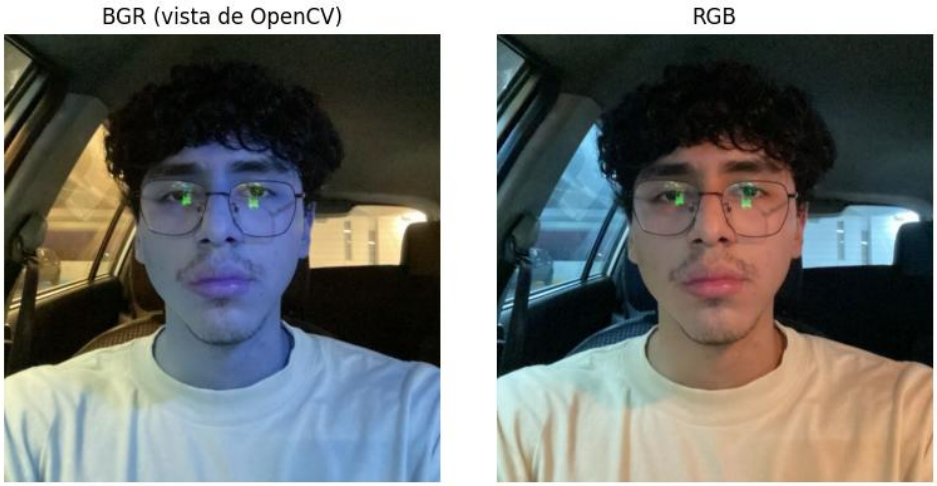


Figura 13: Comparativa entre RGB y BGR.

Cálculo de Gradientes: en el Histograma de Gradientes Orientados (HOG), se puede entender a las gradientes como el cambio de intensidad en los pixeles de una imagen. No se calculan en todas las direcciones de 0° a 360° para cada píxel, en su lugar se calculan

en direcciones específicas y luego se agrupan en “bins” que abarcan de 0° a 180° y usualmente suelen ser 9 bins, es decir de 20° cada uno (Jain, 2024).

Otro factor importante que tomar en cuenta para el cálculo de las gradientes es que el descriptor HOG está diseñado para trabajar principalmente con gradientes de intensidad en imágenes en escala de grises, por lo que no incorpora directamente información de color (Jain, 2024). Para este trabajo, el proceso de conversión a escala de grises de la imagen es realizado implícitamente por la librería dlib al momento de especificar el uso del modelo HOG (Dlib, 2022).

Regresando al cálculo de los gradientes, HOG hace uso del operador Sobel. El operador Sobel es una técnica ampliamente usada en procesamiento de imágenes para estimar el gradiente espacial bidimensional en imágenes en escala de grises. Su principal objetivo es resaltar regiones con altas variaciones de intensidad, comúnmente correspondiente a bordes. Este operador emplea dos kernels de 3x3, uno orientado en sentido horizontal (Gx) y el otro en sentido vertical (Gy), dichos kernels pueden ser observados en la Figura 14. Un kernel, es una pequeña matriz de valores numéricos que se aplica a la imagen mediante una operación de convolución, desplazándose píxel por píxel para calcular intensidades basadas en su vecindario local (Das et al., 2015).

The kernels are defined as:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Figura 14: Kernels de Sobel.

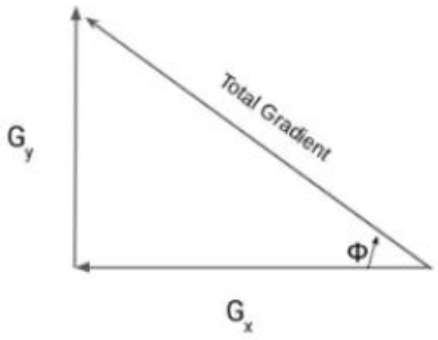
Nota: Tomado de Jain (2024).

Esta técnica se aplica para cada píxel de nuestra imagen, sin embargo, en el caso de bordes y esquinas se complica un poco su aplicación debido a que esta técnica hace uso de un vecindario local, es decir del píxel seleccionado, se selecciona una matriz de 3x3 de píxeles vecinos y se aplican los kernels. Por este motivo, se aplica reflect padding o relleno por reflexión, la cual es una técnica que amplía el tamaño de una imagen creando un borde simétrico reflejado a partir de los píxeles de sus bordes. Esta técnica consiste en duplicar los valores de los píxeles de los bordes hacia el exterior de la imagen como si fuera el reflejo de

un espejo. Es especialmente útil al aplicar convoluciones, ya que garantiza que el filtro tenga suficientes datos incluso en zonas periféricas (Vemulapati, 2023). La Figura 15 ilustra este procedimiento.

Figura 15: Relleno por reflexión.

Una vez calculada la gradiente en X y Y para cada píxel, el siguiente paso es calcular la magnitud de la gradiente y la orientación de esta. Para este cálculo, se hace uso del Teorema de Pitágoras, donde Gx es el cateto adyacente y Gy es el cateto opuesto. Esto nos permitirá obtener la magnitud de la gradiente y para el caso de orientación se hace uso de la función trigonométrica arco tangente, la cual devuelve el ángulo cuyo valor es la tangente (Singh, 2025), como se muestra en la Figura 16.



$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2}, \quad \text{Orientation} = \arctan \frac{G_y}{G_x}$$

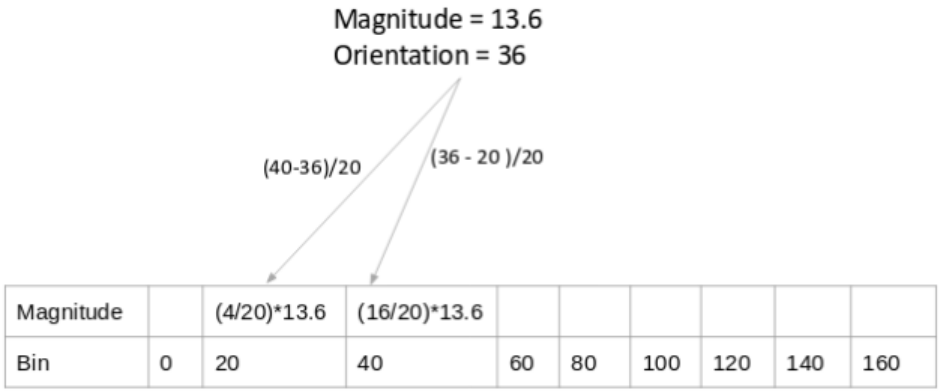
Figura 16: Pitágoras para el cálculo de la magnitud y orientación.

Nota: Adaptado de Jain (2024) y Singh (2025).

División en celdas: es necesario dividir la imagen en pequeñas regiones llamadas celdas, generalmente de 8x8, dentro de las cuales se calcula el histograma de gradientes basado en la orientación y magnitud de los gradientes de los pixeles. Al hacer esta división,

se obtiene el histograma de las celdas más pequeñas, que a su vez representan la imagen completa (Singh, 2025).

Una vez dividida en celdas de 8x8, procedemos con el cálculo del histograma. Cada celda calcula independientemente su histograma y las orientaciones son cuantificadas en bins, usualmente 9 bins representando los ángulos desde 0° a 180° (Jain, 2024). Un método comúnmente usado para el cálculo del histograma es la interpolación de magnitudes en bins vecinos, en donde la contribución de la gradiente de un píxel se añade a ambos lados del gradiente, donde la mayor contribución debe corresponder al valor del bin más cercano a la orientación (Singh, 2025). Un ejemplo para poder entender mejor esto es el siguiente:



Magnitude		(4/20)*13.6	(16/20)*13.6						
Bin	0	20	40	60	80	100	120	140	160

Figura 17: Interpolación de Magnitudes.

Nota: Tomado de Singh (2025).

En el ejemplo representado en la Figura 17, podemos notar que contamos con los datos calculados anteriormente, la magnitud (13.6) y la orientación (36°). El primer paso es identificar los bins más cercanos a la orientación, en este caso son 20 y 40, siendo el bin inferior y superior respectivamente. Como 40 es el bin más cercano a la orientación, es el bin que más peso debe tener. Se calcula la distancia al bin inferior (ángulo – bin inferior) que nos da 16 y hacemos lo mismo para la distancia al bin superior (bin superior – ángulo) que nos da 4. Calculamos los pesos dividiendo la distancia calculada para el peso del bin (20) y a este resultado lo multiplicamos por la magnitud, para de esa forma poder calcular el histograma.

Agrupación en bloques: una vez obtenidos los histogramas de orientación de gradientes por cada celda de 8x8 pixeles, estos se agrupan en bloques de 2x2 celdas, formando un bloque de 16x16. Cada celda aporta un vector de 9 valores correspondientes a los bins de orientación, lo que genera un vector conjunto de 36 elementos por cada bloque.

Para poder garantizar la robustez del descriptor ante los cambios de iluminación y variaciones de contraste, los vectores obtenidos por bloque son normalizados. Por ejemplo, una disminución en el brillo de la imagen provocaría que tanto las magnitudes como los valores de los histogramas disminuyeran proporcionalmente, afectando la precisión del reconocimiento (Jain, 2024).

Como menciona Singh (2025) no podemos eliminar por completo este problema, pero si podemos reducirlo mediante la normalización del histograma dentro de bloques superpuestos. Esta normalización se realiza calculando la magnitud del vector de 36 elementos y dividiendo cada uno de sus componentes por dicha magnitud, como se representa en la Figura 18.

$$V = [a_1, a_2, a_3, \ldots, a_{36}]$$

$$k = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_{36}^2}$$

$$V_{\text{normalized}} = \left[\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \ldots, \frac{a_{36}}{k}\right]$$

Figura 18: Normalización de bloques.

Nota: Adaptado de Singh (2025).

Concatenación del Descriptor: los histogramas normalizados de todos los bloques son concatenados formando un solo vector de características único que representa la imagen completa. Posteriormente, este vector es utilizado como entrada para clasificadores como SVM (Support Vector Machine) para la detección o identificación (Jain, 2024).

Este vector no representa una imagen visual como tal, sino una codificación simbólica de los bordes que define la estructura facial. Por ejemplo, los contornos de los ojos, cejas, nariz y boca quedan representados como orientaciones específicas que distinguen un rostro de otro, tal como lo ilustra la Figura 19.

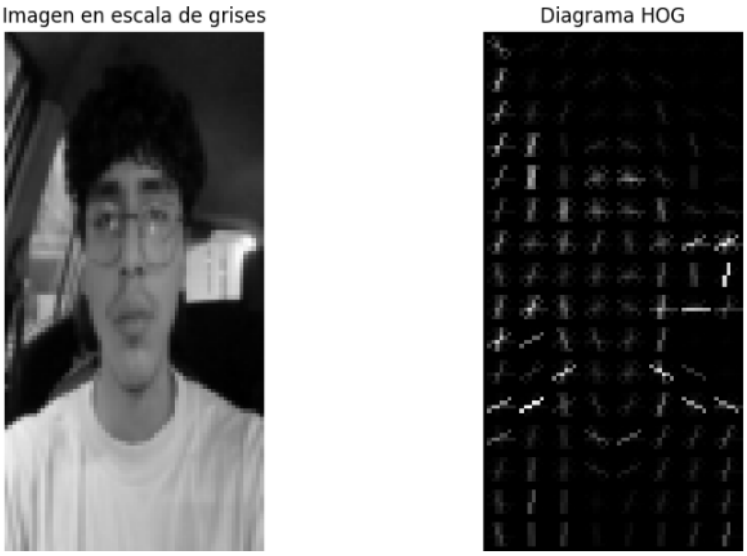


Figura 19: Diagrama HOG.

4.2.2 Detección facial con HOG y clasificación con SVM

Para el presente trabajo se hace uso principalmente de la librería face_recognition, la cual se basa internamente en la biblioteca dlib. Esta última implementa un detector frontal de rostros que combina el modelo HOG con un clasificador SVM (Support Vector Machine). Este detector identifica regiones de la imagen que probablemente contengan un rostro, basándose en los vectores generados por HOG (Dlib, 2022).

Posteriormente, estos vectores son evaluados por un clasificador SVM (Support Vector Machine). Un SVM es un algoritmo de machine learning supervisado que permite clasificar datos mediante la búsqueda de un hiperplano óptimo que maximice la distancia entre cada clase en un espacio N-dimensional (IBM, 2023). En este contexto, el SVM permitiría clasificar regiones de la imagen como un rostro o no. A diferencia de las redes neuronales profundas (DNN), SVM ofrece excelente rendimiento en dispositivos como Raspberry sin sacrificar precisión (Dlib, 2022).

Una vez detectado un rostro, se aplica un modelo de predicción de puntos clave (landmarks), el cual identifica 68 posiciones clave en el rostro incluyendo ojos, labios, cejas y mandíbula. Para esta predicción dlib hace uso de un modelo preentrenado para estimar las

posiciones de estos landmarks en X y Y, sin embargo, es fundamental mencionar que dlib no incluye la frente en estos puntos clave (Elmahmudi & Ugail, 2021). La Figura 20 presenta esta distribución de los 68 landmarks en un rostro.

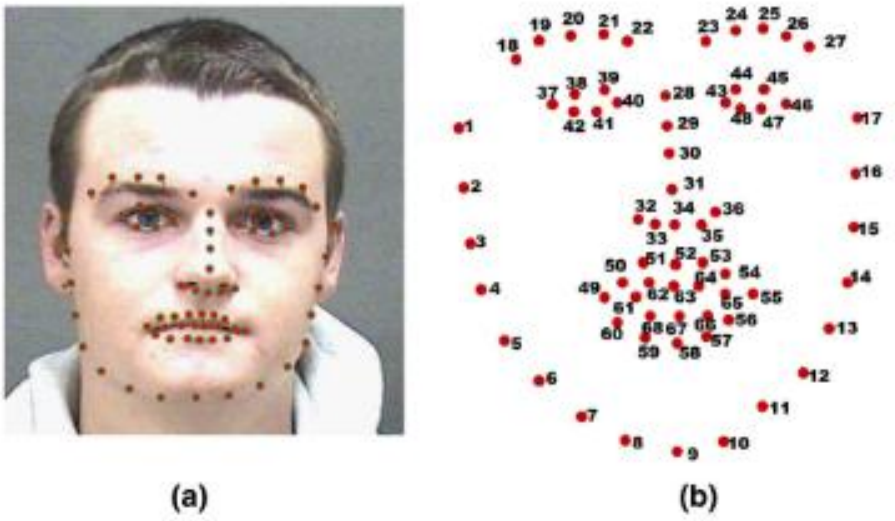


Figura 20: 68 landmarks en un rostro.

Nota: Tomado de Elmahmudi & Ugail (2021).

Este proceso de detección y análisis mediante HOG, SVM y predicción de landmarks representa la base técnica sobre la cual se construyen las funciones de la librería face_recognition. Dicha biblioteca, al estar desarrollada sobre dlib, implementa estos mecanismos, pero los encapsula en funciones que facilitan su implementación. Por lo tanto, comprender esta etapa previa resulta fundamental para entender cómo face_recognition logra identificar, codificar, comparar y reconocer rostros.

4.2.3 Codificación de Rostros con face_recognition

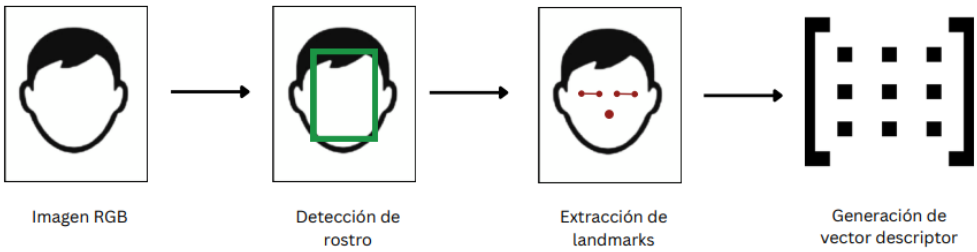


Figura 21: Diagrama general para la codificación.

De acuerdo con lo representado en la Figura 21, el proceso de codificación facial en el presente trabajo se inicia con la detección de las posiciones de las caras en una imagen. La imagen en formato RGB puede ser sometida a dos modelos de detección: HOG (Histograma de Gradientes Orientados) o CNN (Redes Neuronales Convolucionales). En este caso, se hace uso de HOG debido a su rapidez y bajo costo computacional, a pesar de que perdemos un poco de precisión. Esta detección devuelve una lista de coordenadas que delimitan cada rostro encontrado, representadas en formato CSS (top, right, bottom, left), las cuales se ajustan para no superar los límites de la imagen (Geitgey, 2022).

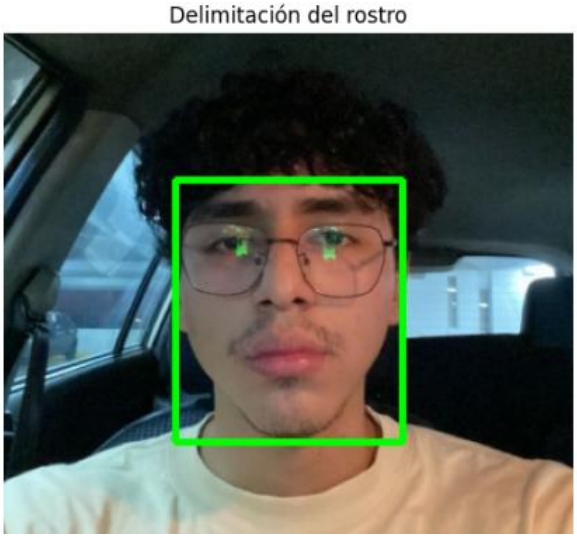


Figura 22: Delimitación del Rostro.

Posteriormente, con la imagen y las ubicaciones faciales detectadas se puede generar el vector descriptor de cada rostro. Internamente, se extraen los puntos clave faciales para alinear y normalizar cada rostro antes de la codificación. Es importante destacar que aquí se puede hacer uso de dos modelos: large y small. La diferencia entre estos modelos es la cantidad de landmarks que cada uno calcula, siendo large el que calcula los 68 puntos, mientras que el modelo small calcula únicamente 5, centrándose en la nariz y ojos. En este trabajo se utiliza el modelo small, que acelera el procesamiento al reducir la complejidad, enfocándose en los puntos más representativos (Geitgey, 2022). La Figura 23 muestra el detector de puntos de referencia de 5 puntos de dlib.

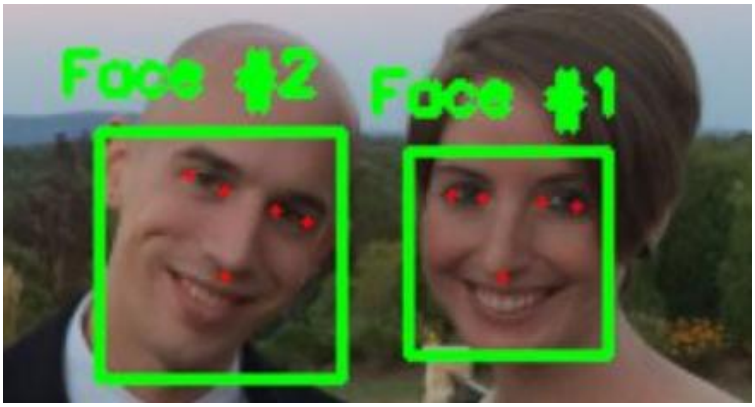


Figura 23: Detector de puntos de referencia facial de 5 puntos de Dlib.

Nota: Adaptado de Rosebrock (2018).

Finalmente, para cada conjunto de landmarks se hace uso nuevamente de dlib para poder generar un vector de 128 dimensiones, conocido como descriptor facial. Este vector representa de forma única las características del rostro, permitiendo su comparación posterior con otros vectores para reconocer o distinguir identidades. Cabe recalcar que, aunque se utilice el modelo small para la detección de landmarks, la dimensión del vector descriptor permanece constante en 128, pues es el tamaño fijo usado por la red neuronal aplicada en dlib (Dlib, 2022). La Figura 24 ejemplifica este descriptor facial.

```
[-0.16597123  0.11949642  0.10122138 -0.12254387 -0.11838274 -0.02759848
  0.01384261 -0.07288168  0.06527785 -0.13963827  0.17559825 -0.08519795
 -0.25020906 -0.0102928  -0.01079375  0.21948609 -0.21553843 -0.19072869
 -0.13214052 -0.10856642  0.01477439  0.07908501 -0.05431245  0.11583221
 -0.1715932  -0.23868942 -0.10927209 -0.01475592  0.01510147 -0.08837818
 -0.00252565  0.03029939 -0.17696191  0.00717495  0.04937807  0.12862331
 -0.01091681 -0.09640069  0.1934087   0.01389313 -0.23863317  0.01766105
  0.15954995  0.26412117  0.21739453 -0.00336392 -0.03702538 -0.05539587
  0.10163558 -0.28075284  0.05129996  0.20316276  0.02803321  0.16820049
 -0.02042791 -0.12237336 -0.00515839  0.22946236 -0.21929839  0.03012473
  0.10334508 -0.19385433  0.010464   -0.1046315   0.159848    0.14616895
 -0.19181257 -0.22495562  0.19514428 -0.23470432 -0.11755507  0.07461189
 -0.0839375  -0.09592333 -0.35081255  0.03915626  0.32692659  0.20959754
 -0.10889891 -0.01162777 -0.09144997 -0.02747694  0.00839157  0.15324235
 -0.02081387 -0.0425766  -0.06370837  0.0559925   0.19876704  0.04837478
  0.00898859  0.25267416  0.04914283  0.0009048   0.00353496  0.14037418
 -0.15166469 -0.07702    -0.10442869  0.03603458 -0.10455432 -0.05006842
  0.01399595  0.18746354 -0.10759938  0.24044091 -0.03882578 -0.01190661
 -0.12068785  0.03638472 -0.02094681 -0.02857181  0.10984001 -0.22919189
  0.14735889  0.20415246 -0.00772358  0.09734803  0.10852756  0.12523061
 -0.03349054 -0.08732443 -0.17586364 -0.09488015  0.04729999 -0.0654182
  0.10575557  0.07663471]
```

Figura 24: Ejemplo del descriptor facial.

Nota: Tomado de Smith (2019).

4.2.4 Almacenamiento de codificaciones en la base de datos

El reconocimiento facial depende de la capacidad del sistema para almacenar y recuperar eficazmente las características únicas de cada rostro. En el presente trabajo, las codificaciones faciales se generan como vectores de 128 dimensiones a partir del rostro

detectado como se explicó en el apartado 2.3. Para poder garantizar la persistencia de estas codificaciones y permitir su reutilización en procesos de identificación posteriores, se optó por almacenarlas en una base de datos relacional como PostgreSQL.

El proceso de almacenamiento inicia con la verificación de la existencia de la persona en la tabla personas, que contiene campos como nombre y apellido. Esto permite asociar la codificación que se va a almacenar con datos personales específicos. Posteriormente, se procede a serializar la codificación facial usando la biblioteca Pickle de Python, que convierte el vector NumPy en una secuencia binaria. Esta secuencia se almacena en la tabla encodings, específicamente en el campo con tipo de dato BYTEA, y se lo asocia con el identificador de la persona y un campo fecha que registra el momento del entrenamiento, como se detalla en la Figura 25.

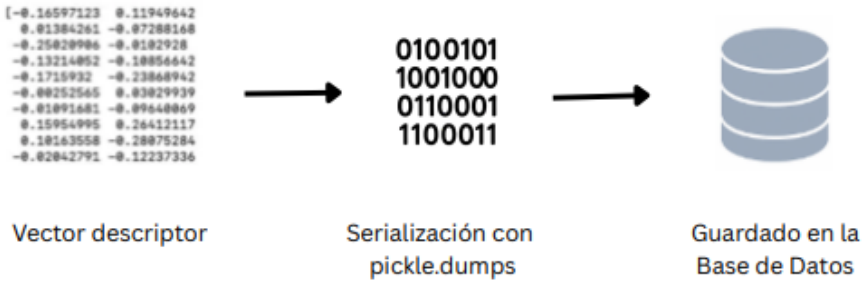


Figura 25: Proceso de Guardado en la base de datos.

Esta estructura permite que una persona tenga múltiples codificaciones, lo que es útil para registrar variaciones faciales naturales como ángulos distintos o gestos faciales. Además, permite ir agregando nuevas muestras faciales para poder mejorar la precisión del reconocimiento.

4.2.5 Entrenamiento con imágenes de nuestro dataset

El entrenamiento para este trabajo consiste en alimentar con nuevas imágenes el modelo implementado por la librería face_recognition, a fin de generar y almacenar sus correspondientes representaciones faciales. Dicho proceso permite que el sistema identifique a cada persona de forma confiable durante la fase de reconocimiento en tiempo real. Para esta etapa se hizo uso de un script que recorre de manera sistemática la carpeta dataset, donde cada subdirectorio está nombrado según la persona representada y contiene varias imágenes de su rostro, como se puede observar en la Figura 26.

Figura 26: Ejemplo del Dataset.

El preprocesamiento de estas imágenes sigue una secuencia bien definida: primero se leen utilizando OpenCV y se convierten a formato RGB. Luego se localiza el rostro en la imagen y se genera su codificación facial, es decir un vector numérico que encapsula sus rasgos más distintivos como se especifica en la sección 2.2.

Cada vector generado, es serializado y almacenado en la base de datos como se explica en la sección 2.4. Esta operación se realiza múltiples veces por persona, idealmente a partir de imágenes capturadas en varios ángulos y expresiones faciales. De esta manera el sistema obtiene una base de datos robusta y flexible, capaz de tolerar variaciones naturales en los rostros al momento de la identificación. La Figura 27 presenta de forma gráfica el proceso detallado descrito anteriormente.

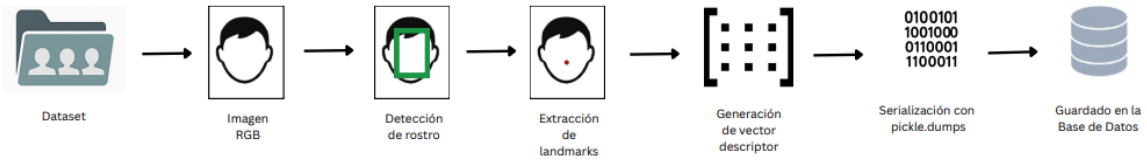


Figura 27: Entrenamiento con nuestro dataset.

La estructura modular de este procesa facilita su mantenimiento. Si se desea añadir a una nueva persona o reforzar la precisión de alguna ya registrada, basta con incorporar imágenes adicionales a la carpeta correspondiente y volver a ejecutar el script.

4.2.6 Reconocimiento facial en tiempo real

El reconocimiento en tiempo real constituye la fase más dinámica del sistema. En esta etapa se procesan imágenes capturadas en directo desde la cámara conectada al Raspberry Pi. Un script desarrollado en Python ejecuta un bucle continuo que captura

fotogramas mediante la librería OpenCV. Cada fotograma es transformado a formato RGB y a escala de grises para su correcto funcionamiento con la librería face_recognition.

Al detectar uno o más rostros en la imagen, el sistema genera para cada uno, una codificación facial, es decir, un vector de características que resume los rasgos biométricos del rostro detectado, específicamente nariz y ambos ojos. Estas codificaciones se consideran temporales, ya que son generadas en tiempo real exclusivamente para una comparación inmediata. Paralelamente, el sistema carga desde la base de datos las codificaciones previamente almacenadas, siendo los vectores únicos de las personas previamente registradas. Cabe recalcar que estos vectores han sido deserializados para que puedan ser interpretados como arreglos NumPy.

Una vez reunidas las codificaciones conocidas y las temporales, el sistema ejecuta un proceso de comparación basado en la distancia euclidiana (Geitgey, 2022). En el contexto del reconocimiento facial, la distancia euclidiana es una medida matemática que se utiliza para determinar el grado de similitud entre dos rostros representados como vectores de características. La distancia euclidiana, también conocida como métrica L2, calcula la raíz cuadrada de la suma de las diferencias al cuadrado entre los componentes correspondientes a ambos vectores. Mientras menor sea la distancia entre estos vectores, mayor será la similitud entre los rostros representados (Malkauthekar, 2013). La Figura 28 ilustra la fórmula para el cálculo de la distancia euclidiana.

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Figura 28: Fórmula para el cálculo de la distancia euclidiana.

Nota: Tomado de Malkauthekar (2013).

Esta comparación la realiza el sistema internamente en dos procesos: el primero calculando la distancia entre la codificación del rostro detectado y cada codificación conocida en la base de datos. El segundo proceso, evalúa si dicha distancia es menor o igual a un umbral determinado, usualmente 0.6, para considerarla una coincidencia válida. Estos

procesos permiten detectar de manera automática el nivel de similitud entre dos rostros y tomar una decisión binaria (coincide/no coincide).

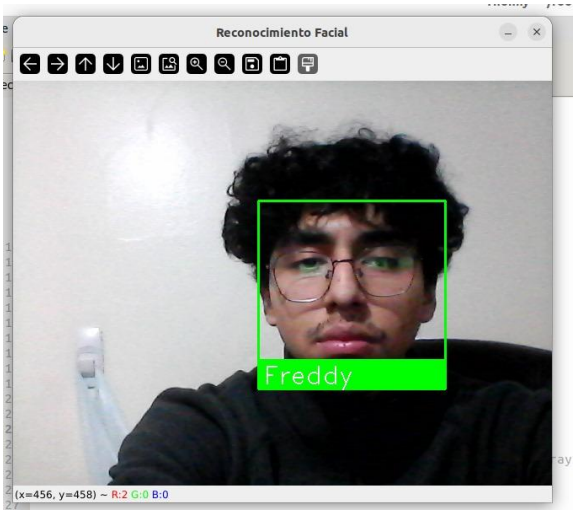


Figura 29: Ejemplo del Reconocimiento.

Cuando se identifica una coincidencia, el sistema obtiene el identificador de la persona asociada, muestra su nombre en la interfaz, y captura una imagen del rostro detectado como se observa en la Figura 29. Esta imagen se convierte a formato base64, y junto con el identificador (id_persona) y la marca de tiempo, se envía mediante una solicitud POST a una API desarrollada en Flask, la cual registra el evento en la tabla accesos. En caso de que el rostro no coincida con ninguna codificación registrada, el sistema realiza el mismo proceso, pero enviando un ID Null y lo clasifica como “Desconocido”. La Figura 30 presenta un ejemplo de esta situación.

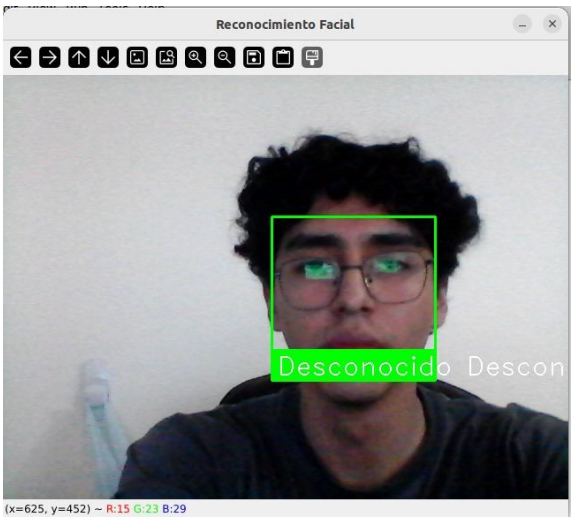


Figura 30: Ejemplo de Desconocido.

Tabla 5: *Comparación del comportamiento del sistema ante coincidencias y no coincidencias.*

Criterio	Coincidencia	No coincidencia
Umbral	Menor o igual a 0.6	Mayor a 0.6
Resultado	Coincide	No coincide
Identificación	Se asocia con un identificador único de la base de datos	Se clasifica como "Desconocido"
Visualización en Interfaz	Muestra el nombre de la persona identificada	Muestra el texto "Desconocido"
Registro en la API	Se envía id, una foto y la hora del acceso	Se envía a la API con id nulo, la foto y la hora de acceso

Nota: La presente tabla describe el comportamiento del sistema ante una coincidencia o no coincidencia al momento de detectar un rostro.

4.3 Construcción de la API RESTful

El desarrollo de la API en este sistema respondió a la necesidad de establecer una capa de comunicación eficiente, escalable y segura entre el módulo de reconocimiento facial, la base de datos y la interfaz web. Para lograrlo, se optó por una arquitectura RESTful (Representational State Transfer), ampliamente adoptada por su simplicidad, compatibilidad con HTTP y su facilidad para estructurar recursos mediante URLs (AWS, 2024b). Esta arquitectura permitió definir las operaciones sobre los datos utilizando métodos como GET, PUT, POST, DELETE.

La API fue implementada utilizando Flask, un microframework de Python que destaca por su ligereza, modularidad y bajo consumo de recursos. Flask permite la construcción de endpoints de forma rápida, facilitando el enrutamiento de las peticiones y la integración con librerías internas como psycopg2, para la conexión con la base de datos PostgreSQL, o face_recognition, para el reconocimiento facial.

Se optó por una estructura modular en el desarrollo de la API. Un ejemplo de esto es la conexión a la base de datos, la cual se encapsuló en un archivo externo que expone una función para obtener la conexión y poder ejecutar las consultas necesarias. De igual manera se manejaron funciones para la conversión de imágenes (base64 a binario y viceversa), lo que garantiza la correcta comunicación de datos cuando se transmiten desde el frontend.

Cabe destacar que se utilizaron convenciones semánticas para los nombres de las rutas para facilitar la comprensión y su uso desde el backend y frontend. Por ejemplo, /accesos gestiona los registros de entrada del sistema de reconocimiento facial.

Tabla 6: *Endpoints implementados en la API*

Ruta	Método	Parámetros	Respuesta	Código HTTP
/	GET	-	Página HTML de inicio	200
/crud	GET	-	Página HTML del CRUD	200
/personas	GET	-	Lista JSON con todas las personas	200
/personas	POST	Nombre, apellido, cargo	Persona agregada	201
/personas/<id>/foto	PUT	Foto (codificada en Base64)	Foto actualizada correctamente	200
/personas/<id>	DELETE	-	Persona eliminada	200
/accesos	GET	-	Lista JSON con los registros de los accesos	200
/accesos	POST	id_persona, foto	Acceso registrado	201

Nota: La presente tabla muestra los endpoints definidos en la API, donde cada ruta corresponde a una función específica del sistema, permitiendo la interacción con la base de datos PostgreSQL y la interfaz web. Además de los métodos básicos de consulta y registro de datos, se incluyen funciones para imágenes codificadas.

La API implementa validaciones previas a la inserción de datos en la base de datos. Por ejemplo, al agregar una persona, se verifica que los campos no estén vacíos y que no excedan una longitud máxima para evitar problemas con la base de datos. Este enfoque minimiza los errores de integridad y asegura datos limpios y coherentes. De igual manera, se emplean sentencias SQL parametrizadas para poder prevenir ataques de inyección SQL, como se muestra en la Figura 31.

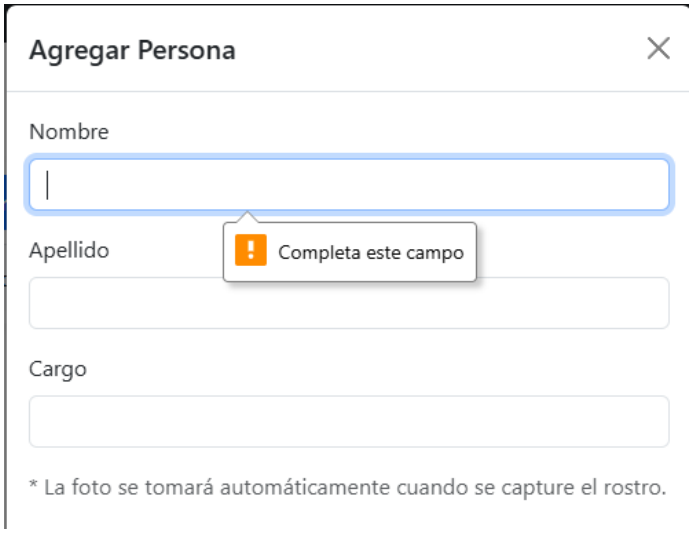


Figura 31: Ejemplo de validaciones de la API.

De igual manera, la API implementa una lógica específica para el manejo de imágenes, parte fundamental para el sistema de reconocimiento facial. Las imágenes son manejadas como cadenas codificadas en base64 para facilitar su transmisión en formato JSON. Luego, por el lado del servidor, estas cadenas son decodificadas a formato binario y se almacenan en la base de datos.

Esta lógica se aplica tanto al almacenar fotos de personas registradas como al guardar capturas de accesos detectados. En ambos casos, el sistema puede convertir los datos binarios nuevamente a base64 para ser enviados a la interfaz web, donde se renderizan como imágenes embebidas. De acuerdo con lo representado en las Figuras 32 y 33, este enfoque simplifica la arquitectura del sistema ya que evita el uso de rutas de almacenamiento físicas y mantiene toda la información centralizada en la base de datos.

Figura 32: Lógica para el guardado de imágenes.

Figura 33: Lógica para mostrar imágenes en la interfaz web.

4.4 Desarrollo de la Interfaz Web

El desarrollo de la interfaz web para el sistema de reconocimiento facial se basa principalmente en tecnologías estándar del frontend como HTML, CSS y JavaScript. HTML proporciona la estructura básica de la página, definiendo elementos como tablas, formularios, enlaces y botones, mientras que CSS se encarga del diseño visual, garantizando que la interfaz sea responsiva y agradable al usuario. Para poder lograr una experiencia moderna, se emplea Bootstrap, un framework que facilita la creación de componentes visuales adaptativos, como modales para la gestión de registros (CRUD), así como estilos para que las tablas y botones posean un diseño profesional (Bootstrap, 2025). Las Figuras 34 y 35 muestran ejemplos de la interfaz para los accesos sin datos y para la gestión de personas vacía, respectivamente.

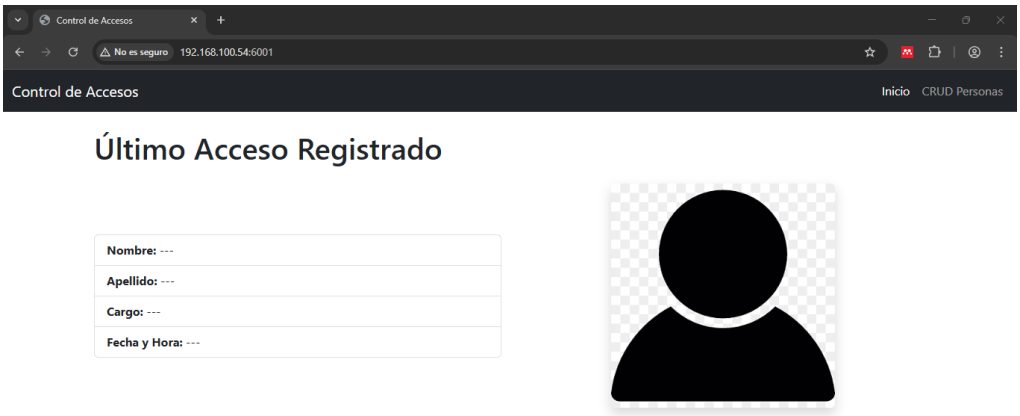


Figura 34: Interfaz para los accesos sin datos.

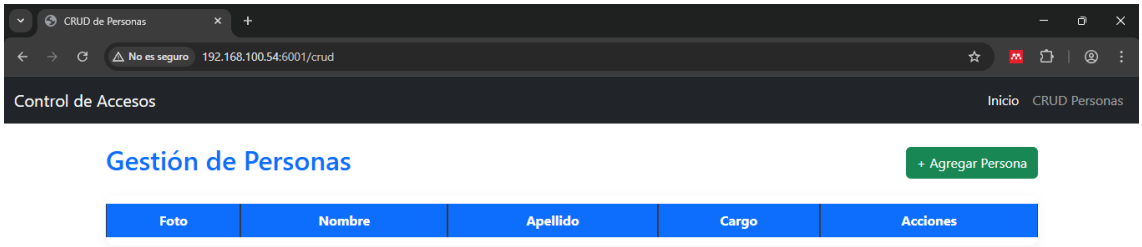


Figura 35: Interfaz para la gestión de personas vacía.

El eje operativo de la interfaz web reside en el manejo de JavaScript para la interacción dinámica con la API. En lugar de hacer uso de librería externas como JQuery, se utiliza la fetch API nativa de JavaScript la cual facilita la realización de solicitudes asíncronas para enviar y recibir información en formato JSON. Esta técnica posibilita la actualización de datos en tiempo real sin necesidad de recargar la página, mejorando la experiencia del usuario (MDN, 2025). Gracias a fetch, la interfaz es capaz de mostrar la lista de personas registradas, la tabla de accesos y manejar los eventos de creación o eliminación de registros.

La actualización dinámica de los datos en la interfaz se logra mediante la implementación de funciones JavaScript, que en intervalos definidos o al ocurrir ciertos eventos, hacen llamados a la API para obtener la información más reciente y modificar el DOM (Document Object Model). Un ejemplo concreto de esto es la interfaz de accesos, la cual se actualiza automáticamente cada 5 segundos, garantizando la visualización del último acceso registrado. Para optimizar esta operación, el sistema compara el último acceso recibido con el previamente mostrado, y solo actualiza la tabla si hay un nuevo dato. Esto evita actualizaciones innecesarias, reduciendo la carga de la API.

La gestión de eventos en la interfaz web se implementa principalmente mediante funciones JavaScript que responden a las acciones de los usuarios. Por ejemplo, al hacer click en el botón “Agregar Persona”, se muestra un modal Bootstrap con un formulario que

recoge los datos de nombre, apellido y cargo, como se observa en la Figura 36. Cuando se envía este formulario, se ejecuta una función que recoge los datos, construye un objeto JSON y se los envía a la API. Al ingresar exitosamente una persona, la interfaz muestra un mensaje de confirmación, como se ilustra en la Figura 37. Posteriormente, la gestión de personas refleja el nuevo ingreso, actualizando la tabla correspondiente como se observa en la Figura 38.

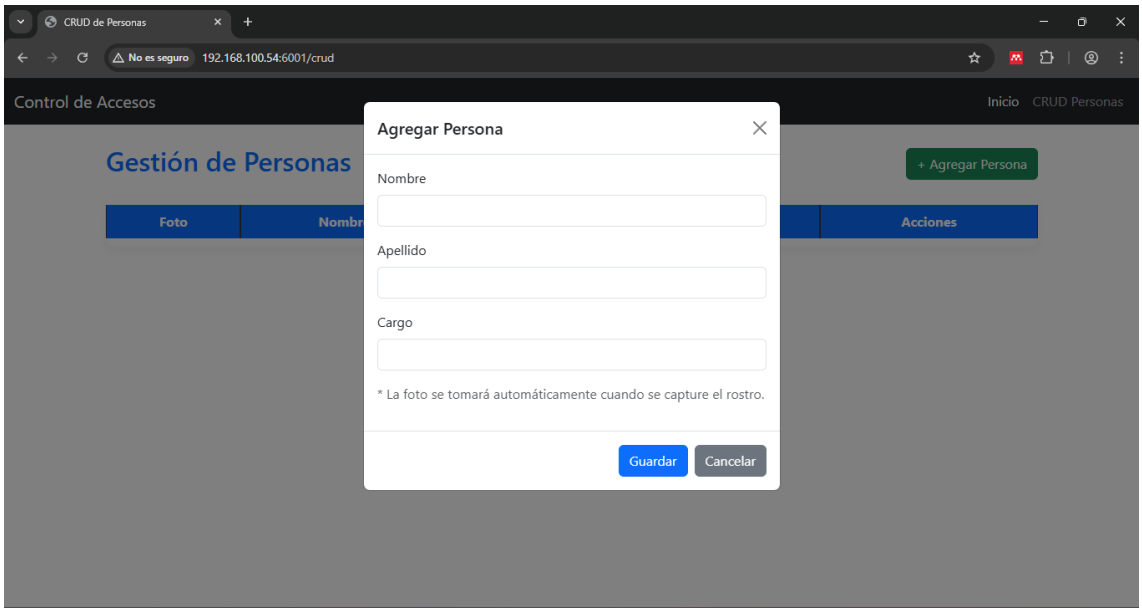


Figura 36: Modal para el ingreso de personas.

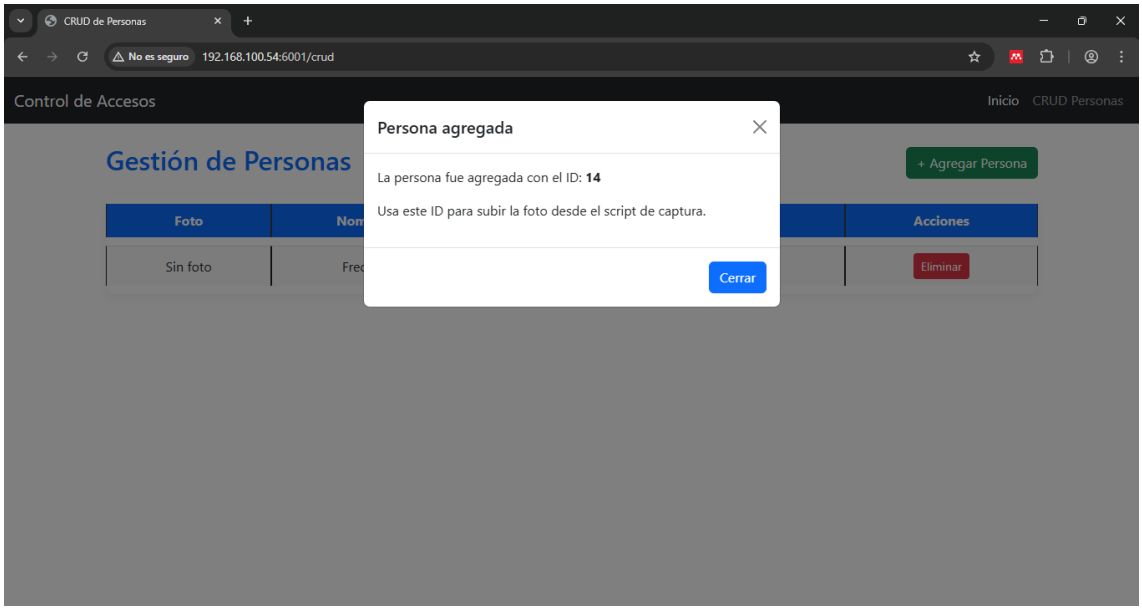


Figura 37: Ingreso exitoso de una persona.

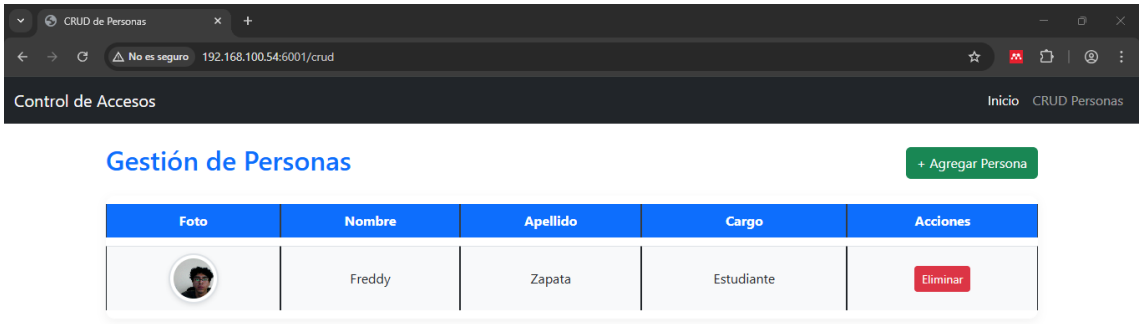


Figura 38: Interfaz para la gestión de personas tras un ingreso.

Por otro lado, el comportamiento visual de la interfaz cambia dinámicamente según los datos recibidos. En el caso de que se detecte una persona conocida mediante el sistema de reconocimiento facial, la interfaz muestra sus datos (nombre, cargo, hora de acceso) junto a la foto registrada al momento de capturar el dataset, es decir una foto de la base de datos. La Figura 39 presenta este comportamiento cuando se detecta a alguien conocido. Por el otro lado, si se detecta un desconocido, la interfaz muestra la imagen capturada durante la detección en tiempo real, con datos como “Desconocido” y la hora de la detección, como se aprecia en la Figura 40. Esta diferenciación no solo mejora la experiencia, sino que también facilita el monitoreo en tiempo real del sistema.

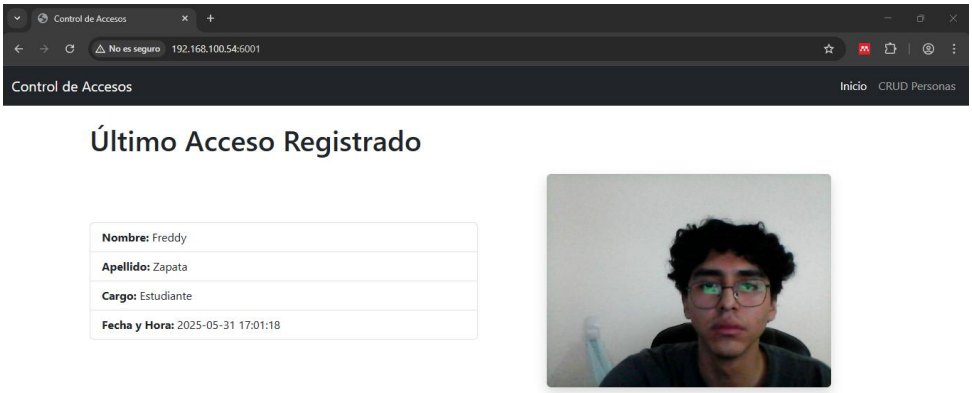


Figura 39: Comportamiento de la interfaz al detectar a alguien conocido.

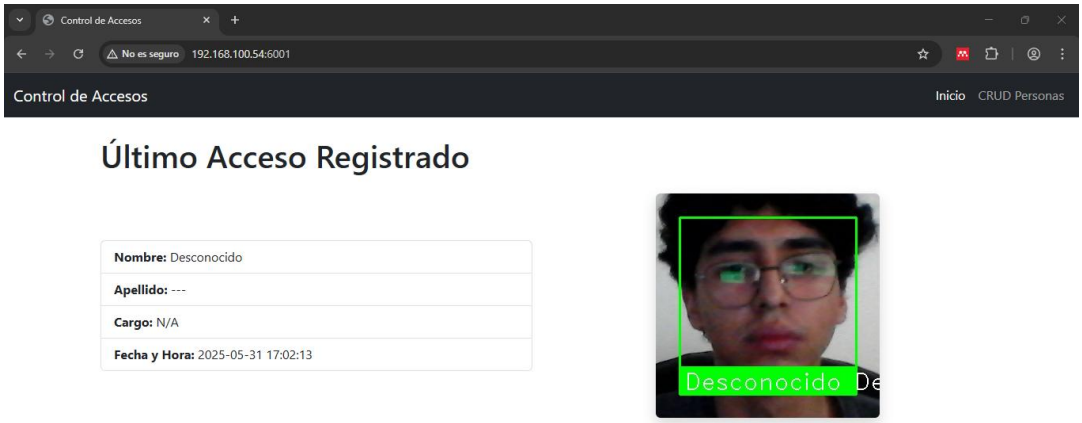


Figura 40: Comportamiento de la interfaz al detectar un desconocido.

Tabla 7: *Comparación del comportamiento visual de la interfaz web ante una detección*

Criterio	Persona conocida	Persona desconocida
Imagen mostrada	Foto del dataset	Imagen capturada en tiempo real
Texto mostrado	Nombre, cargo, hora de acceso	Texto "Desconocido", hora del intento
Registro en la tabla accesos	Se registra con el id de la persona	Se registra con id nulo y la hora del intento

Nota: La presente tabla resume el comportamiento visual de la interfaz web ante la detección de un rostro conocido o desconocido.

4.5 Integración del Prototipo

El funcionamiento completo del prototipo requiere la interacción coordinada de múltiples componentes desarrollados por separado. Si bien cada módulo, como la captura del dataset, el entrenamiento del modelo, el reconocimiento en tiempo real, la API RESTful y la interfaz web, cumple una función específica, es la integración lógica entre ellos lo que permite el correcto desempeño del prototipo. En esta sección se presenta una visión global del flujo de trabajo del sistema, ilustrando cómo se enlazan los diferentes scripts mediante una serie de diagramas de flujo.

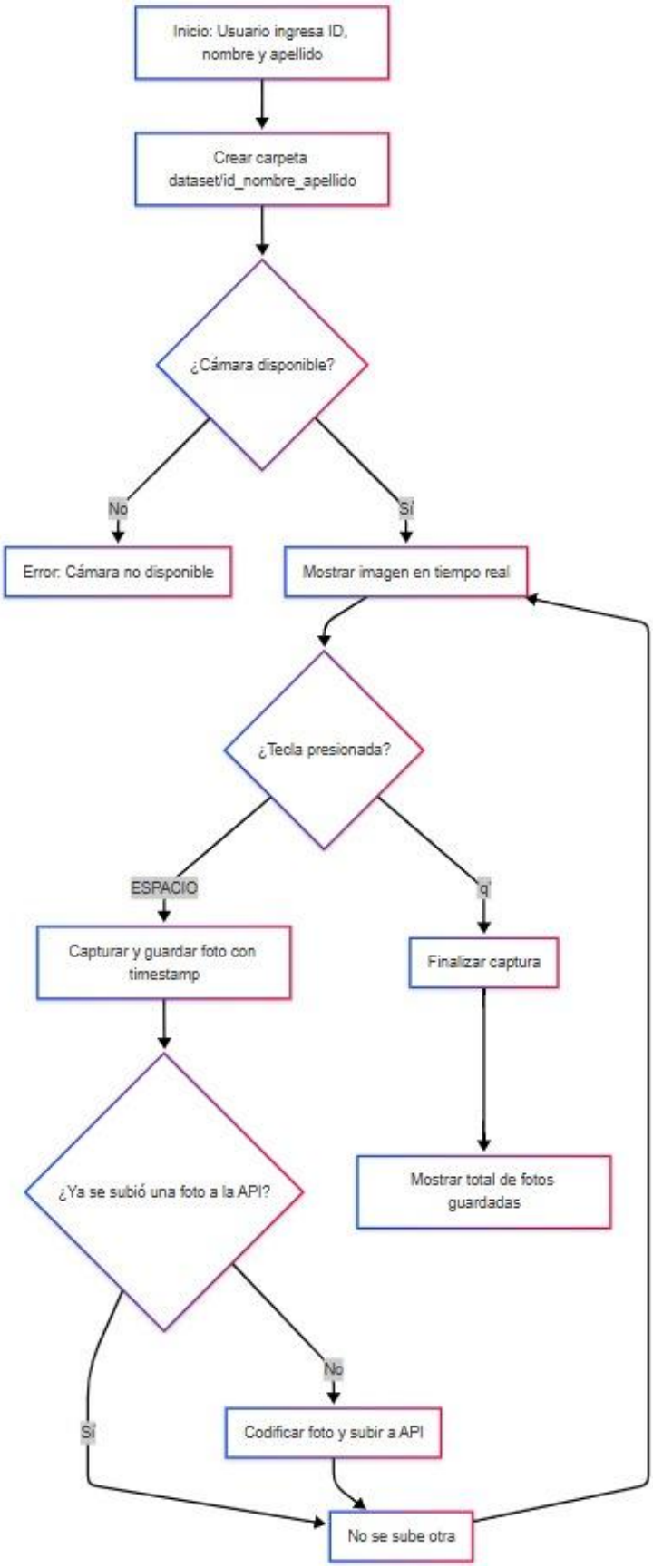


Figura 41: Diagrama de flujo del script para la creación del dataset.

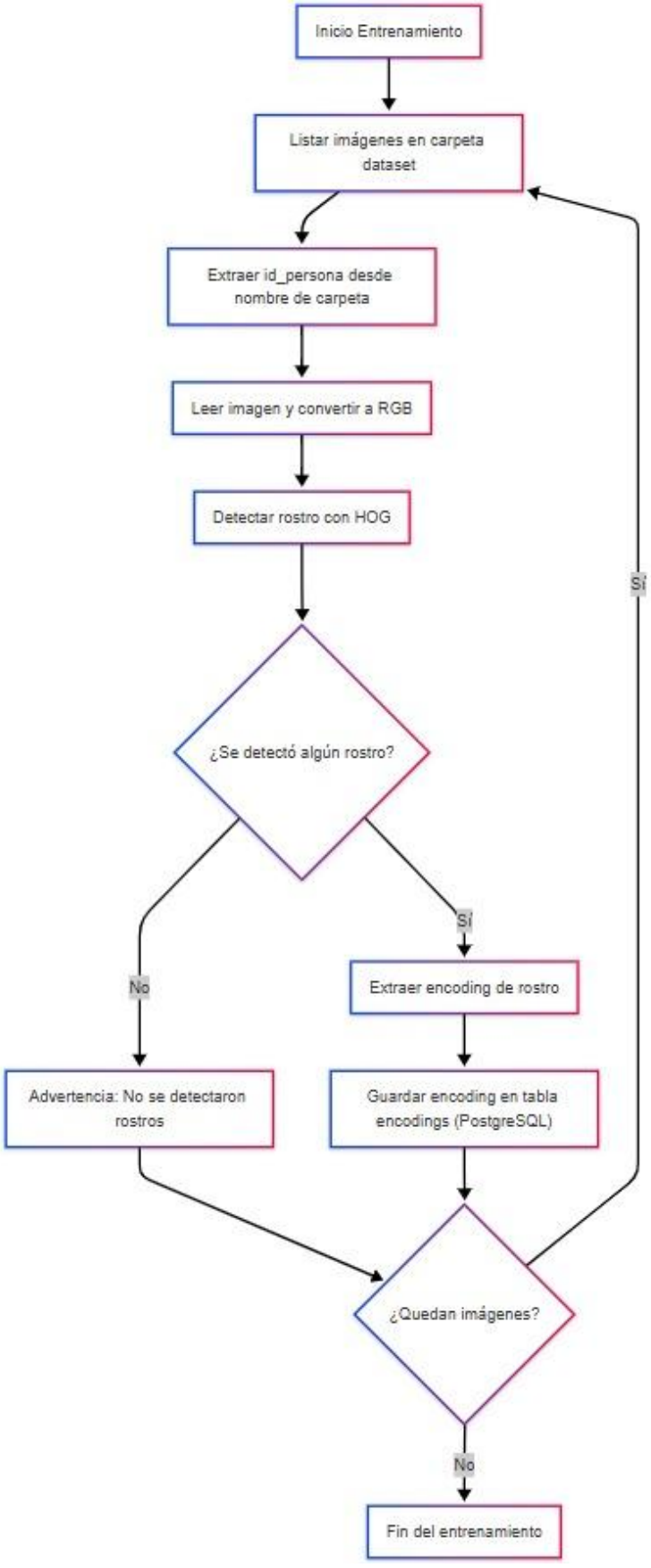


Figura 42: Diagrama de flujo del script para el entrenamiento.

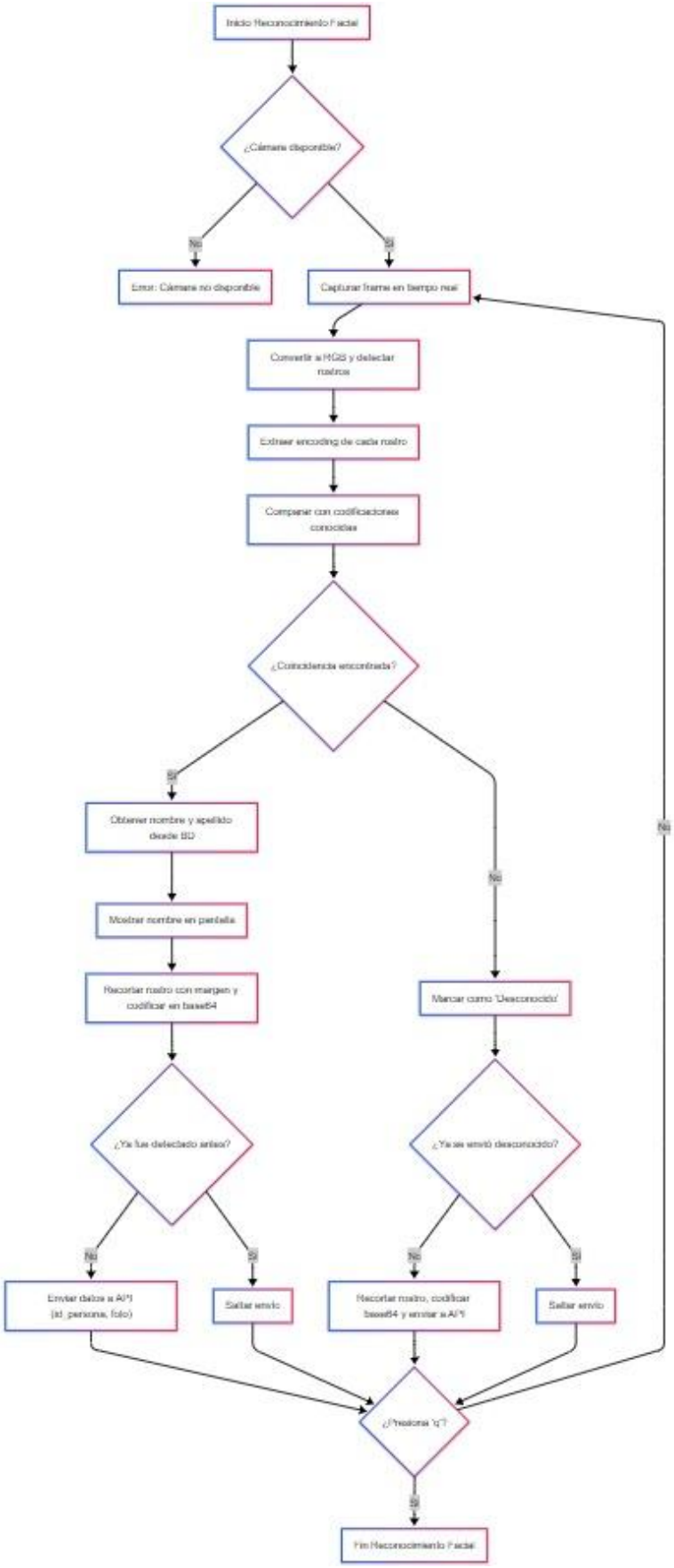


Figura 43: Diagrama de flujo del script de Reconocimiento facial.

CAPÍTULO V: Pruebas y Análisis de Resultados

5.1 Descripción del Entorno de Pruebas

El presente capítulo se enfoca en la validación funcional del sistema de reconocimiento facial mediante un conjunto de pruebas estructuradas. Estas pruebas están diseñadas para simular situaciones reales y evaluar el comportamiento del prototipo frente a variaciones en el ángulo del rostro, las condiciones de iluminación y los cambios temporales en los rasgos faciales, como accesorios o expresiones.

Las pruebas fueron realizadas en un entorno controlado, haciendo uso de un Raspberry Pi con su módulo cámara. El software utilizado fue el sistema de control de acceso basado en Flask, OpenCV y la librería face_recognition, conectado a una base de datos PostgreSQL. El sujeto de prueba en todos los escenarios fue el autor del presente trabajo, lo que garantiza consistencia en los datos de entrenamiento y en la evaluación del sistema.

5.2 Metodología de Evaluación

Las pruebas se llevaron a cabo en sesiones consecutivas, cada una correspondiente a un escenario específico. Para cada escenario se realizaron al menos 10 capturas distintas, registrando manualmente si el sistema reconocía correctamente al usuario o no. Las imágenes fueron procesadas en tiempo real por el sistema, y cada intento de reconocimiento fue almacenado con su respectiva codificación y timestamp. Las métricas consideradas incluyen tasa de reconocimiento, falsos negativos y tiempo aproximado de reconocimiento, dividido en dos: tiempo aproximado de detección y tiempo aproximado de comparación.

Cabe recalcar que las pruebas se centraron únicamente en un sujeto de prueba, haciendo uso de un conjunto de imágenes previamente entrenadas. Esto garantiza un enfoque controlado y repetible, aunque limitado en cuanto a generalización. Sin embargo, permite evaluar el desempeño técnico del sistema bajo condiciones variadas.

5.3 Escenarios de Pruebas

En esta sección se describen los diferentes escenarios planteados para evaluar el comportamiento del sistema de reconocimiento facial. Cada escenario busca simular condiciones reales que podrían afectar la precisión del reconocimiento.

La literatura sugiere que factores como la inclinación del rostro, las condiciones lumínicas y el uso de accesorios pueden interferir en la precisión de los sistemas biométricos faciales. Autores como Platero (2015) y Morcillo (2020) sugieren que estos factores son

críticos para validar el desempeño de sistemas similares. Por lo tanto, se consideran indispensables en esta evaluación. A continuación, se detallan los tres escenarios:

5.3.1 Escenario 1: Variaciones angulares del rostro

Este escenario tiene como objetivo evaluar el comportamiento del sistema cuando el rostro de una persona no se encuentra en una posición frontal directa. Este tipo de pruebas es crucial, ya que diversos autores coinciden en que la orientación del rostro influye considerablemente en la precisión del reconocimiento facial, afectando la precisión en la detección de puntos clave.

Inicialmente el sistema fue entrenado únicamente con imágenes frontales del rostro, sin presencia de ángulos o rotaciones notables. Con este entrenamiento base, se procedió a realizar pruebas con el rostro en cinco posiciones distintas, donde para cada una de estas posiciones se registraron 10 intentos de reconocimiento, totalizando 50 intentos:

- Perfil Izquierdo (90°)
- Perfil Derecho (90°)
- Inclinación hacia arriba (aproximadamente 45°)
- Inclinación hacia abajo (aproximadamente 45°)
- Ligera rotación diagonal (entre 20° y 30°)

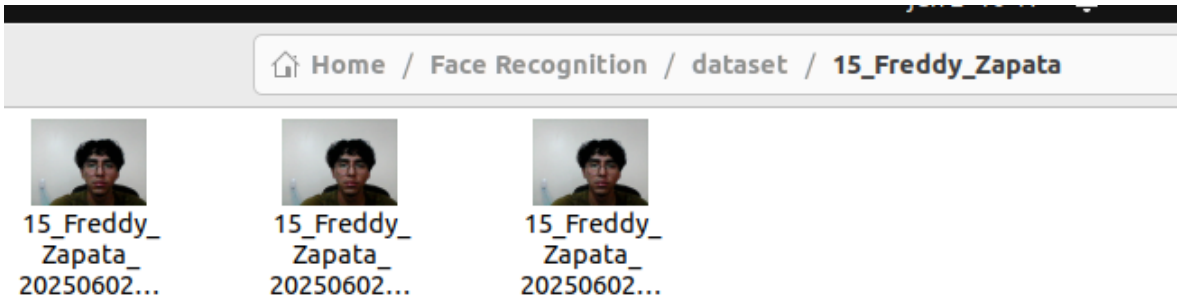


Figura 44: Dataset Inicial.

Posteriormente, se repitió el entrenamiento del sistema incluyendo en el dataset imágenes que representan estas variaciones angulares, es decir, se añadieron nuevas fotos con las mismas inclinaciones utilizadas en la fase anterior. Esta estrategia permitió evaluar si el sistema mejora su precisión al contar con ejemplos más diversos durante el aprendizaje. Una vez incorporadas las nuevas imágenes, se realizaron nuevamente las pruebas con los mismos cinco ángulos, manteniendo la misma cantidad de intentos por posición.

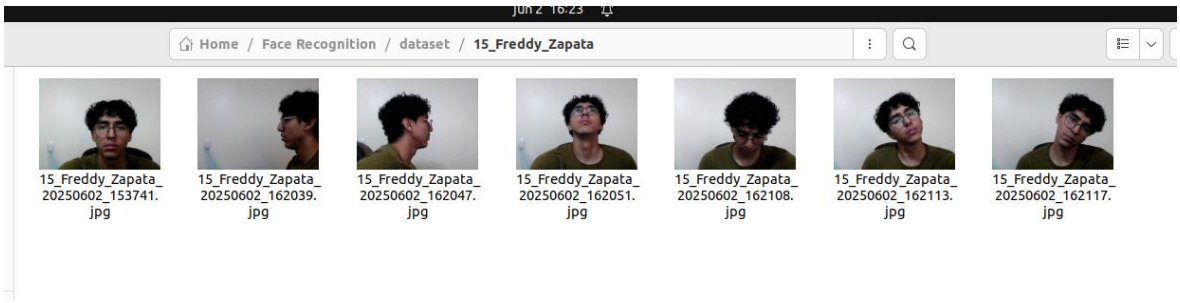


Figura 45: Dataset con fotos en varios ángulos.

Todas las pruebas se realizaron bajo condiciones de iluminación constante para aislar la variable angular y garantizar que los resultados reflejen exclusivamente el efecto del ángulo del rostro. Los datos recolectados incluyen tanto el número de aciertos como los fallos de reconocimiento por cada tipo de inclinación, lo que permite realizar un análisis comparativo entre el entrenamiento frontal y el entrenamiento con variaciones angulares.

5.3.2 Escenario 2: Condiciones de Iluminación Variables

Este escenario busca analizar la capacidad del sistema para reconocer rostros bajo diferentes condiciones de iluminación, dado que la luz es un factor determinante en la calidad de la imagen capturada y, por ende, en la precisión de los algoritmos de reconocimiento facial. Como se revisó en capítulos anteriores, técnicas como HOG, que trabajan con representaciones faciales basadas en gradientes, son especialmente sensibles a cambios drásticos en la iluminación.

Para estas pruebas se utilizó el modelo previamente entrenado con imágenes capturadas en condiciones de luz natural estable. Las pruebas consistieron en someter al sistema a cuatro condiciones lumínicas diferentes:

- Entorno con baja iluminación
- Entorno con exceso de luz
- Iluminación lateral
- Iluminación artificial directa

En cada caso se realizaron 10 intentos de reconocimiento, posicionando al sujeto de pruebas en el mismo ángulo frontal para aislar la variable lumínica.

Las pruebas de iluminación se realizaron en horas nocturnas para evitar la influencia de la luz natural. Esto permitió mantener un entorno controlado en el que solo se utilizaron fuentes de iluminación artificial ubicadas estratégicamente, según cada prueba definida para

este escenario. De esta manera, fue posible replicar condiciones específicas con mayor precisión, reduciendo inferencias externas, como atardeceres o nubes, que alteren la iluminación. Los resultados obtenidos permiten observar hasta qué punto el sistema es tolerante a variaciones en el brillo, contraste y uniformidad de la luz en el rostro.

Figura 46: Variaciones de la Iluminación.

5.3.3 Escenario 3: Accesorios y Expresiones Faciales

Este escenario pretende examinar cómo ciertos cambios faciales temporales o superficiales, como el uso de accesorios y la modificación de expresiones faciales comunes, afectan al desempeño del sistema. Estas variaciones representan situaciones reales que una persona puede presentar al momento de ser identificada por el sistema, y por tanto son esenciales para medir la robustez del prototipo.

Para las pruebas, se utilizó el modelo previamente entrenado con imágenes del rostro sin accesorios y con expresión neutra. En el caso particular del uso de lentes, se evaluó la capacidad del sistema para identificar correctamente a una persona que utilice lentes transparentes frente a lentes oscuros o de sol. Este último tipo de accesorio tiende a cubrir

una mayor superficie del rostro y oscurece la región ocular, lo cual puede dificultar el análisis de rasgos clave para el reconocimiento facial. En cada prueba se realizaron 10 intentos de reconocimiento, con el sujeto en posición frontal y bajo condiciones de iluminación constante, considerando las siguientes variables:

- Uso de gorra
- Uso de mascarilla
- Uso de gafas de sol
- Sujeto con una sonrisa amplia
- Sujeto con el rostro fruncido
- Sujeto con un ojo cerrado

Estas pruebas revelan las limitaciones del sistema frente a obstrucciones parciales del rostro o cambios en los patrones faciales, elementos comunes en entornos reales de control de acceso. Se registró si el sistema seguía identificando correctamente al sujeto o si los cambios generaban falsos negativos, especialmente en los casos donde una parte significativa del rostro estaba cubierta. Los resultados permitirán evaluar la necesidad de incorporar técnicas más avanzadas de preprocesamiento o modelos más robustos en implementaciones futuras.

Figura 47: Accesorios y Expresiones Faciales.

5.4 Resultados Obtenidos

En esta sección se presentan los resultados recolectados tras la ejecución de los distintos escenarios descritos anteriormente. Cada escenario permitió explorar una variable crítica para el rendimiento del sistema de reconocimiento facial, tales como el ángulo del rostro, las condiciones de iluminación y la presencia de accesorios o expresiones faciales que alteran los rasgos del sujeto. Los datos recolectados incluyen tanto los casos de identificación correcta como los errores de reconocimiento, así como observaciones relevantes sobre el comportamiento del sistema en situaciones particulares.

Para una valoración más exacta, los resultados han sido organizados por escenario, facilitando un análisis comparativo que evidencia las fortalezas y limitaciones del modelo frente a cada condición específica. Se incluyeron métricas como la tasa de reconocimiento, los falsos negativos (cuando el sistema no reconoce a una persona registrada) y el tiempo aproximado de reconocimiento dividido en dos: el tiempo que tarda en detectar un rostro y codificarlo y el tiempo que tarda en comparar la codificación con los registros de la base de datos.

5.4.1 Resultados del Escenario 1

En la primera configuración, modelo entrenado únicamente con imágenes frontales, los resultados evidenciaron limitaciones significativas. En particular, el sistema no fue capaz de identificar ningún rostro cuando se presentaba de perfil, ya sea izquierdo o derecho. Más aún, en estos casos, no solo falló en reconocer a la persona registrada, sino que ni siquiera detectó la presencia de un rostro. Este comportamiento coincide con lo reportado por Platero (2015), donde se indica que el modelo de reconocimiento presenta fallos consistentes en escenarios con vistas laterales.

Para las inclinaciones hacia arriba y hacia abajo, los resultados fueron irregulares. Se observó que en algunos intentos el sistema lograba identificar correctamente al usuario, mientras que en otros no detectaba el rostro, incluso hubo casos donde por unos instantes detectaba al usuario y enseguida dejaba de hacerlo y así repetidamente. Esta variabilidad ocurrió incluso en lapsos de pocos segundos, lo que sugiere una alta sensibilidad a condiciones externas como ligeros movimientos de la cabeza o cambios en la iluminación del entorno. En contraste, la rotación diagonal mostró un rendimiento óptimo, con una tasa de reconocimiento del 100%, lo que indica que esta forma de inclinación conserva suficientes características faciales visibles para una identificación exitosa.

Tabla 8: *Promedio de resultados obtenidos con dataset frontal*

Tipo de Prueba	Tasa de Reconocimiento (%)	Tiempo Promedio de Detección (s)	Tiempo Promedio de Comparación (s)	Tasa de Falsos Negativos (%)
Perfil Izquierdo	0	0	0	0
Perfil Derecho	0	0	0	0
Inclinación hacia Arriba	30	0,2979	0,0001	0
Inclinación hacia Abajo	30	0,3045	0,0001	0
Rotación Diagonal	100	0,26522	0,0002	0

Nota: La presente tabla recopila los valores promedio de las métricas registradas en las pruebas realizadas para cada variación angular, a partir de diez intentos por caso. Para consultar los resultados detallados de cada intento individual, dirigirse a las Tablas 12 a 16 en el apartado de Anexos.

Buscando mejorar los resultados obtenidos, especialmente las pruebas angulares con bajo rendimiento, se utilizó un dataset más variado, que incluía imágenes en distintas orientaciones, para entrenar al modelo. Sin embargo, durante el proceso de entrenamiento se observó que el script no lograba detectar rostros en algunas imágenes anguladas, específicamente en los perfiles izquierdo y derecho, y en la inclinación hacia abajo.

```
>>> %Run entreno.py
  [INFO] Procesando Imágenes...
  [INFO] Procesando imagen 1/7
  [WARNING] No se detectaron caras en la imagen: dataset/15_Freddy_Zapata/15_Freddy_Zapata_20250602_162039.jpg
  [INFO] Procesando imagen 2/7
  [INFO] Procesando imagen 3/7
  [WARNING] No se detectaron caras en la imagen: dataset/15_Freddy_Zapata/15_Freddy_Zapata_20250602_162108.jpg
  [INFO] Procesando imagen 4/7
  [INFO] Procesando imagen 5/7
  [WARNING] No se detectaron caras en la imagen: dataset/15_Freddy_Zapata/15_Freddy_Zapata_20250602_162047.jpg
  [INFO] Procesando imagen 6/7
  [INFO] Procesando imagen 7/7
  [INFO] Entrenamiento completado y datos guardados en PostgreSQL.

>>>
```

Figura 48: Resultados del script de entrenamiento con dataset variado.

Tabla 9: *Promedio de resultados obtenidos con dataset variado*

Tipo de Prueba	Tasa de Reconocimiento (%)	Tiempo Promedio de Detección (s)	Tiempo Promedio de Comparación (s)	Tasa de Falsos Negativos (%)
Perfil Izquierdo	0	0	0	0
Perfil Derecho	0	0	0	0
Inclinación hacia Arriba	50	0,27456	0,0002	0
Inclinación hacia Abajo	10	0,2728	0,0002	0
Rotación Diagonal	100	0,26048	0,0002	0

Nota: La presente tabla recopila los valores promedio de las métricas registradas en las pruebas de variación angular con un dataset variado. Para consultar los resultados detallados de cada intento individual, dirigirse a las Tablas 17 a 21 en el apartado de Anexos.

5.4.2 Resultados del Escenario 2

Durante la ejecución de este escenario se identificaron comportamientos críticos del sistema ante variaciones en las condiciones de iluminación. Aunque en ambientes con luz frontal directa o iluminación uniforme el sistema mantuvo un rendimiento aceptable, se detectaron fallos importantes bajo ciertas configuraciones lumínicas, especialmente cuando la fuente de luz provenía lateralmente.

Los resultados muestran un comportamiento variable. En condiciones de baja iluminación y con luz artificial directa, el sistema mantuvo un rendimiento óptimo, alcanzando una tasa de reconocimiento del 100% en ambos casos. De igual manera, los tiempos de detección y comparación permanecieron estables y dentro de rangos aceptables, sin presencia de falsos negativos. En contraste, ante un exceso de luz, se observó una disminución en la tasa de reconocimiento al 60%.

La condición más desfavorable fue la de la iluminación lateral, donde el sistema alcanzó apenas un 30% de reconocimiento. En algunas repeticiones, no logró detectar ningún rostro a pesar de la presencia del usuario frente a la cámara, comportamiento similar al observado en las pruebas de perfiles. No obstante, se registraron mejoras parciales al reducir la distancia entre el rostro y la cámara, lo cual sugiere sensibilidad a factores de posicionamiento en combinación con la iluminación.

Tabla 10: *Promedio de resultados obtenidos con variación en la Iluminación*

Tipo de Prueba	Tasa de Reconocimiento (%)	Tiempo Promedio de Detección (s)	Tiempo Promedio de Comparación (s)	Tasa de Falsos Negativos (%)
Baja Iluminación	100	0,27649	0,00022	0
Exceso de Luz	60	0,267266667	0,0002	0
Iluminación Lateral	30	0,2936	0,0002	0
Iluminación Artificial Directa	100	0,28249	0,0002	0

Nota: La presente tabla recopila los valores promedio de las métricas registradas en las pruebas de iluminación variada. Para consultar los resultados detallados de cada intento individual, dirigirse a las Tablas 22 a 25 en el apartado de Anexos.

5.4.3 Resultados del Escenario 3

Durante este escenario se evaluaron las capacidades del sistema ante condiciones que implican modificaciones físicas y expresiones faciales comunes, las cuales podrían afectar temporalmente la estructura visual del rostro. Las pruebas incluyeron el uso de gorra, mascarilla, gafas de sol, así como distintas expresiones faciales.

Los resultados evidencian que el sistema mantuvo un comportamiento altamente confiable frente a las expresiones faciales. Tanto la sonrisa, el fruncido y los ojos cerrados, no afectaron el reconocimiento, obteniendo en estos tres casos una tasa del 100% y sin falsos negativos registrados. Asimismo, los tiempos de detección y comparación se mantuvieron dentro de rangos aceptables.

Respecto al uso de accesorios, las gafas de sol y la mascarilla provocaron una ligera reducción en la tasa de reconocimiento, alcanzando el 80% en ambos casos. Estos resultados coinciden con lo reportado por Morcillo (2020), quien también evidenció que el uso de mascarilla puede ocasionar detecciones intermitentes debido a la obstrucción de rasgos clave del rostro.

Tabla 11: *Promedio de resultados obtenidos con accesorios y expresiones faciales*

Tipo de Prueba	Tasa de Reconocimiento (%)	Tiempo Promedio de Detección (s)	Tiempo Promedio de Comparación (s)	Tasa de Falsos Negativos (%)
Uso de Gorra	100	0,29926	0,0002	0
Uso de Mascarilla	80	0,3107875	0,0002	0
Uso de Gafas de Sol	80	0,32035	0,0002	0
Sonrisa	100	0,28941	0,0002	0
Fruncido	100	0,28356	0,0002	0
Ojo cerrado	100	0,28783	0,0002	0

Nota: La presente tabla recopila los valores promedio de las métricas registradas en las pruebas de accesorios y expresiones faciales. Para consultar los resultados detallados de cada intento individual, dirigirse a las Tablas 26 a 31 en el apartado de Anexos.

5.5 Análisis de Resultados

El propósito de este análisis es interpretar y valorar las métricas fundamentales que facilitan la determinación de la eficacia y eficiencia del sistema de reconocimiento facial. Se analizarán indicadores como la tasa de reconocimiento, el tiempo promedio de detección y comparación, así como la tasa de falsos negativos. Es crucial analizar cada una de estas métricas para comprender el desempeño del prototipo bajo diferentes condiciones e identificar oportunidades de mejora en su funcionamiento.

5.5.1 Tasa de Reconocimiento

La tasa de reconocimiento es una métrica que indica el porcentaje de rostros correctamente identificados por el sistema en relación con el total de pruebas realizadas. Esta métrica es crucial para determinar la precisión y confiabilidad del prototipo en escenarios de control de acceso. Un valor alto en esta tasa refleja un buen desempeño del modelo utilizado y una buena calidad en la base de datos de entrenamiento.

La Figura 49 muestra el comportamiento del sistema frente a variaciones angulares comparando un modelo entrenado con dos tipos de datasets. Se observa que, en ambos casos, el reconocimiento falla cuando el usuario está completamente de perfil, sea izquierdo o derecho. Esta limitación se debe a que el modelo utilizado requiere de la detección de cinco puntos clave del rostro siendo estos dos para cada ojo y uno para la nariz. Al momento de estar colocado en una posición de perfil únicamente se puede identificar como máximo dos

puntos, uno para el ojo correspondiente y otro para la nariz, que se consideran insuficientes para que lo reconozca como un rostro.

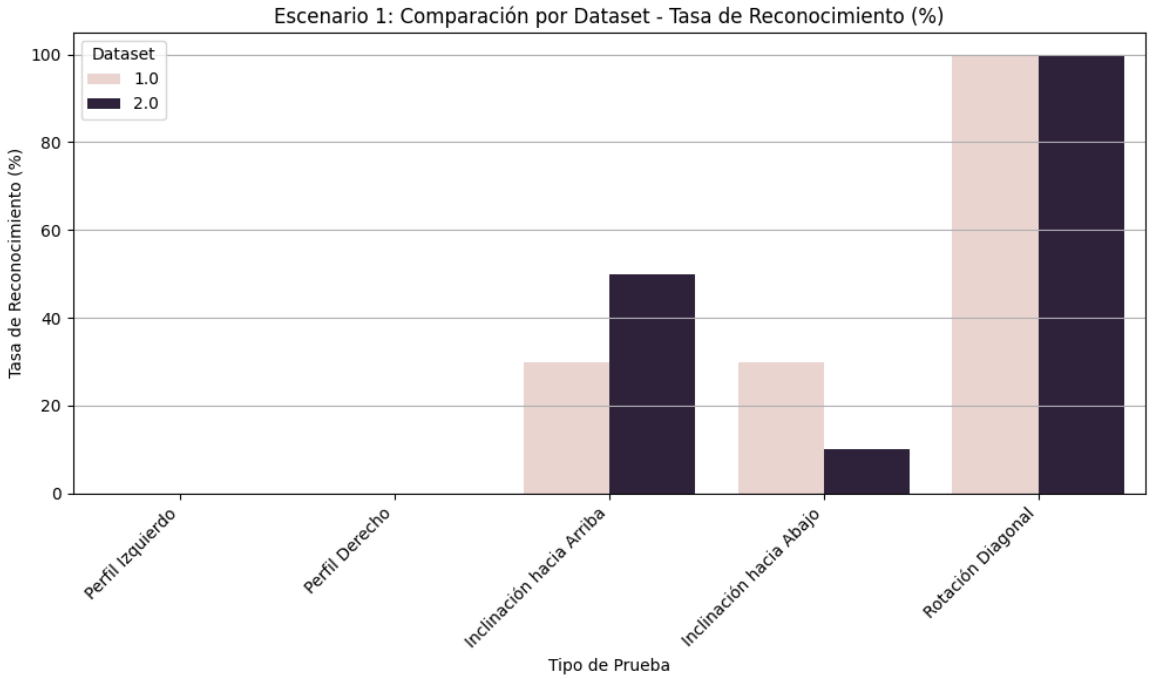


Figura 49: Comparativa por dataset - Tasa de Reconocimiento.

Respecto a las pruebas realizadas con una inclinación vertical, específicamente hacia abajo, podemos notar que el comportamiento fue similar para ambos datasets, teniendo una tasa de reconocimiento del 30%. Esto se debe, a que, como se mencionó anteriormente, el sistema cambiaba de estado de detección a no detectado en cortos intervalos de segundos. Es importante añadir que el reconocimiento tuvo una mejora al momento de elevar la mirada hacia la cámara.

Por el contrario, el caso del reconocimiento al estar en una inclinación hacia arriba, podemos notar que el uso del dataset con imágenes únicamente frontales tuvo un mejor desempeño que el uso de un dataset con imágenes variadas, específicamente de un 40% mejor. Como se mencionó en el apartado de resultados, al momento de entrenar el modelo con el dataset variado, el script no logró identificar rostros en tres imágenes que formaban parte de este, por lo que el bajo desempeño se debió a un modelo con calidad deficiente en su entrenamiento.

Por último, en los casos de rotación diagonal del rostro, el modelo, independientemente del dataset utilizado, logró un rendimiento óptimo, alcanzando una tasa

de reconocimiento del 100%, lo cual demuestra una adecuada generalización del modelo en esta condición.

La Figura 50 presenta una comparación global del desempeño del prototipo en términos de tasa de reconocimiento, considerando cada uno de los escenarios definidos y sus respectivas pruebas. Para esto, se realizó un promedio de la tasa de reconocimiento en cada prueba y se lo generalizó como el comportamiento del escenario para su análisis.

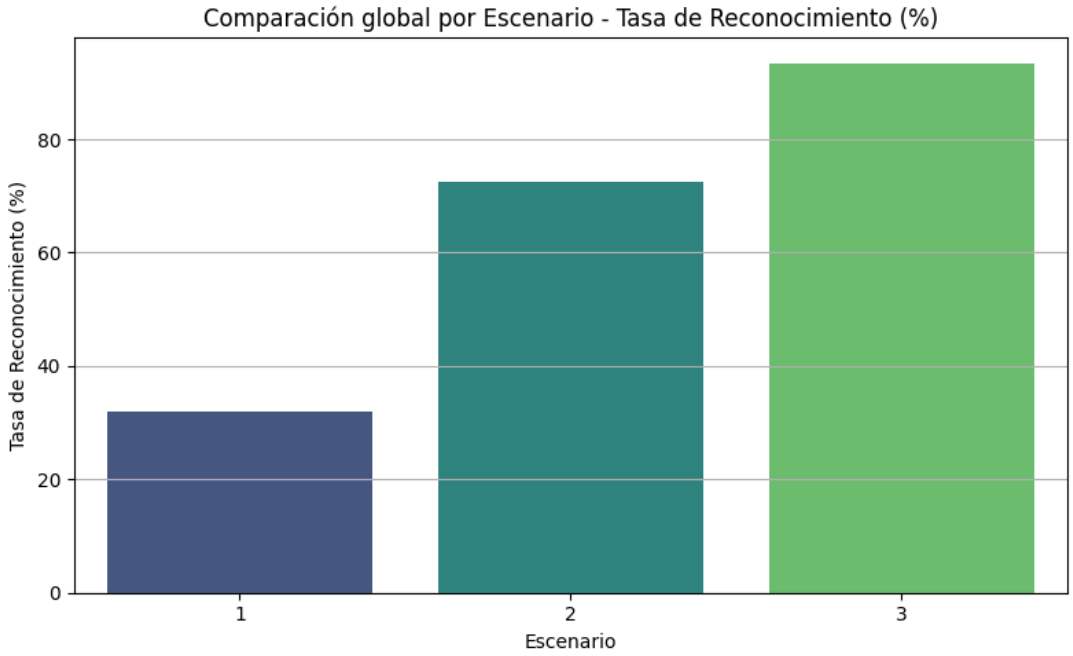


Figura 50: Comparación por escenario - Tasa de Reconocimiento

Se observa que la tasa de reconocimiento nunca desciende por debajo del 30%, con un 32% en el Escenario 1, lo cual puede considerarse como un punto de partida aceptable, especialmente al tener en cuenta las condiciones adversas bajo las que se realizaron algunas pruebas. En el Escenario 2, se logró una mejora considerable, alcanzando una tasa de reconocimiento del 72.5%, sufriendo bastante ante la condición de iluminación lateral donde la tasa de reconocimiento específica para esta prueba fue del 30% lo que influyó bastante en el promedio para esta métrica.

Finalmente, el Escenario 3 presentó el mejor desempeño, con una tasa de reconocimiento del 93.33%, lo cual evidencia una alta precisión del modelo ante accesorios y expresiones faciales. Este resultado resalta la capacidad de adaptación y efectividad del sistema, especialmente en condiciones óptimas o cuando el modelo ha sido mejor entrenado.

5.5.2 Tiempo promedio de detección

El tiempo promedio de detección mide el intervalo temporal que tarda el sistema en ubicar e identificar el rostro de una persona al momento de aparecer en cámara. Este intervalo incluye tanto la detección del rostro como la obtención de su codificación, que luego se compara con los registros de la base de datos. Un tiempo reducido indica eficiencia en el procesamiento, mientras que un tiempo elevado podría afectar la fluidez del sistema.

La Figura 51 muestra el tiempo que le toma al sistema ubicar un rostro y obtener su codificación al momento de ejecutar el script para el reconocimiento facial. Como podemos notar, las pruebas de perfil izquierdo y derecho no presentan valores y esto se debe a que el sistema no lograba detectar ningún rostro cuando el usuario se encontraba completamente de perfil, comportamiento idéntico al evaluado en la métrica anterior.

Se observa que los tiempos más altos registrados corresponden al modelo entrenado con el dataset conformado únicamente por imágenes frontales, teniendo valores de 0.2979 y 0.3045 segundos en las pruebas de inclinación hacia arriba y hacia abajo respectivamente. Este comportamiento se atribuye al cambio de estado de detección a no detección en cuestión de segundos, lo mismo que ocurrió en la métrica anterior. El sistema demoraba más en detectar un rostro ya que al estar con la mirada hacia abajo pierde cuatro de los cinco puntos clave del rostro, sin embargo, al elevar únicamente la mirada el sistema lograba reconocer el rostro, pero con tiempos elevados.

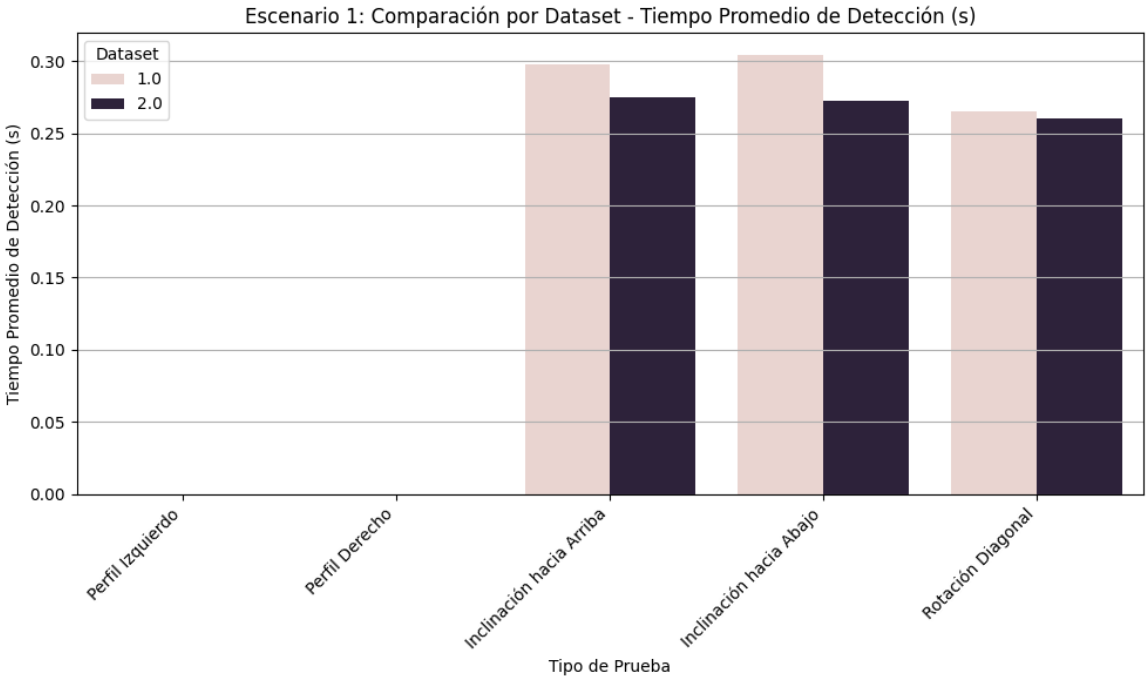


Figura 51: Comparativa por dataset - Tiempo promedio de Reconocimiento

Con respecto al modelo entrenado con el dataset formado con imágenes variadas, podemos notar que el tiempo promedio de detección es idéntico, con variaciones de milisegundos, para las pruebas de inclinación hacia arriba y hacia abajo. En la prueba de inclinación hacia arriba, se obtuvo un tiempo promedio de detección de 0.2745segundos y para la prueba de inclinación hacia abajo se obtuvo un tiempo promedio de 0.2728 segundos. Cabe destacar que, aunque el modelo no fue entrenado con gran cantidad de imágenes debido a la no detección de rostros por parte del script de entrenamiento, notamos una mejora de 0.03 segundos con respecto al primer dataset.

Finalmente, en las pruebas de rotación diagonal leve, podemos notar que el modelo presenta un comportamiento idéntico con ambos datasets. Al igual que las pruebas de inclinación vertical, podemos observar que la diferencia de tiempos es milimétrica, con valores de 0.2652 para el dataset con imágenes frontales y 0.2604 para el dataset con imágenes variadad.

Podemos concluir que el sistema presenta un rendimiento óptimo en condiciones donde el rostro no presenta inclinaciones agudas que impidan la detección de un rostro. De igual manera, podemos decir que la cantidad de imágenes del dataset no influye en el reconocimiento, sino la calidad y claridad de estas para detectar los cinco puntos clave.

La Figura 52 presenta una comparación del tiempo promedio de detección por escenario. Inicialmente se podría indicar que el escenario con mejor desempeño fue el Escenario 1 debido a que presenta un tiempo promedio de 0.2253, pero hay que tomar en cuenta que cuatro pruebas de este escenario no presentaron resultados, perfiles derecho e izquierdo por cada dataset, lo que influyó en el cálculo del promedio.

Sin embargo, podemos notar que el tiempo promedio de detección no desciende de los 0.22 segundos en ningún escenario lo que nos da una idea del comportamiento del sistema, donde en menos de medio segundo es capaz de detectar un rostro y obtener su codificación para poder compararlo y obtener una coincidencia o no.

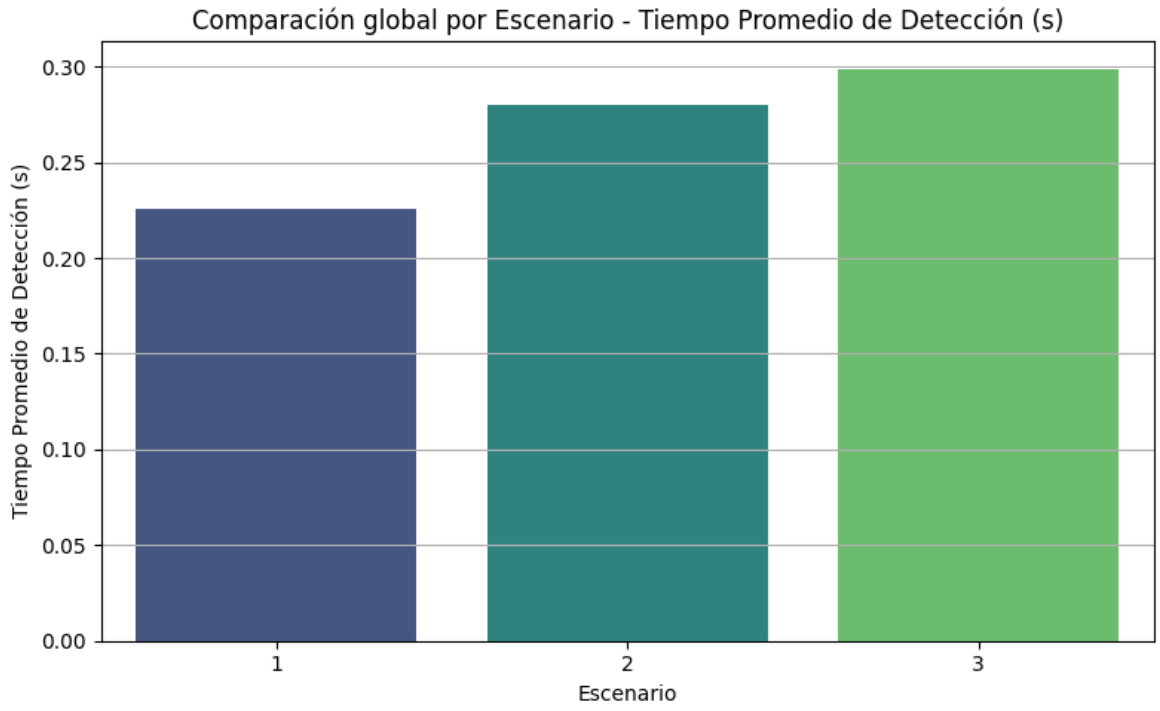


Figura 52: Comparación por Escenario - Tiempo Promedio de Detección

Respecto al Escenario 2, podemos notar que el tiempo promedio de detección tiene un incremento de aproximadamente cuatro centésimas, dado un valor de 0.2799 segundos. Este incremento en los tiempos de detección se debe a las pruebas individuales de este escenario donde tres de las cuatro pruebas mostraron resultados dentro de un rango de 0.27 a 0.29 segundos. Sin embargo, la prueba con menor tiempo fue la condición de exceso de luz donde se obtuvo un promedio de 0.2672 segundos para ubicar un rostro y conseguir su codificación.

Finalmente, el Escenario 3, presenta los tiempos de detección más altos registrados de todos los escenarios, con un valor promedio aproximado de 0.30 segundos. Este tiempo se debe en gran medida a dos pruebas correspondientes a este escenario, uso de mascarillas y uso de gafas de sol, donde los valores promedio fueron de 0.3108 y 0.3203 segundos respectivamente. Este comportamiento se debe a que el sistema cambiaba repentinamente de estado de detección a no detección sin movimiento alguno del usuario. Esto indica que una combinación entre el uso de accesorios y cambios no intencionales en la iluminación o el entorno influyen en la capacidad del sistema para poder detectar un rostro.

En general, se concluye que el sistema presenta un rendimiento adecuado, logrando detecciones y codificaciones en menos de medio segundo, siempre que las condiciones faciales sean parcialmente visibles y no se comprometan los puntos clave necesarios para el procesamiento.

5.5.3 Tiempo Promedio de comparación

El tiempo promedio de comparación hace referencia al intervalo de tiempo que tarda el sistema en comparar las codificaciones del rostro detectado en ese momento con las codificaciones almacenadas previamente en la base de datos. Esta etapa ocurre inmediatamente después de la detección y codificación del rostro, y representa un componente crítico del rendimiento general del sistema, ya que define que tan rápido puede determinarse si una persona es reconocida o no.

Un tiempo reducido en esta métrica asegura que el sistema puede operar en tiempo real, ofreciendo respuestas ágiles ante el ingreso de personas. Además, influye directamente en la fluidez del control de acceso y en la experiencia del usuario, especialmente cuando se tiene un volumen considerable de registros en la base de datos.

La Figura 53 muestra el tiempo que le toma al sistema comparar la codificación del rostro detectado en tiempo real con las codificaciones previamente almacenadas en la base de datos durante el entrenamiento del modelo. Como se observa, las pruebas en las que el usuario se encontraba completamente de perfil, sea izquierdo o derecho, no presentan datos. Este comportamiento, ya reportado en métricas anteriores, se debe a que en dichas posiciones el sistema no logró detectar ningún rostro, lo que imposibilitó la etapa de comparación.

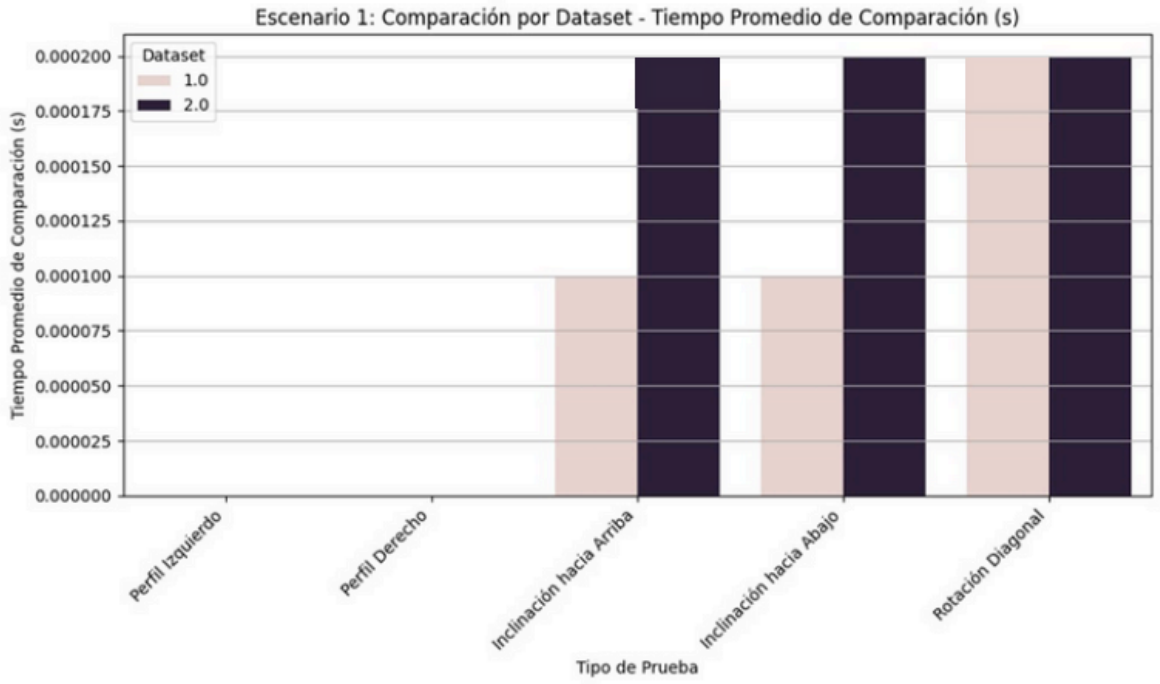


Figura 53: Comparativa por dataset - Tiempo promedio de Comparación

En general, los tiempos registrados para la comparación fueron extremadamente bajos, con valores constante de 0.0001 y 0.0002 segundos. El dataset entrenado únicamente con imágenes frontales alcanzó el mejor desempeño, registrando 0.0001 segundos en dos de sus pruebas, con un pico máximo en la rotación diagonal. Por su parte, el dataset entrenado con imágenes variadas, mantuvo un comportamiento uniforme en todas las pruebas, registrando constantemente un valor de 0.0002 segundos.

Este rendimiento se atribuye a la arquitectura del script de reconocimiento, el cual está diseñado para realizar una consulta inicial a la base de datos al momento de su ejecución, cargando todas las codificaciones en un arreglo. Posteriormente, la comparación se realiza iterativamente sobre dicho arreglo hasta encontrar una coincidencia. Esta estrategia de preprocesamiento e indexación reduce significativamente el tiempo necesario para cada comparación individual.

La Figura 54 presenta una comparación del tiempo promedio de comparación por escenario. Al igual que el comportamiento en el análisis del Escenario 1 por cada dataset, los tiempos promedio de comparación presentan únicamente dos valores: 0.0001 y 0.0002 segundos. El escenario con mejor rendimiento fue el Escenario 1, donde el promedio fue de 0.0001 y el resto de los escenarios fue de 0.0002 segundos.

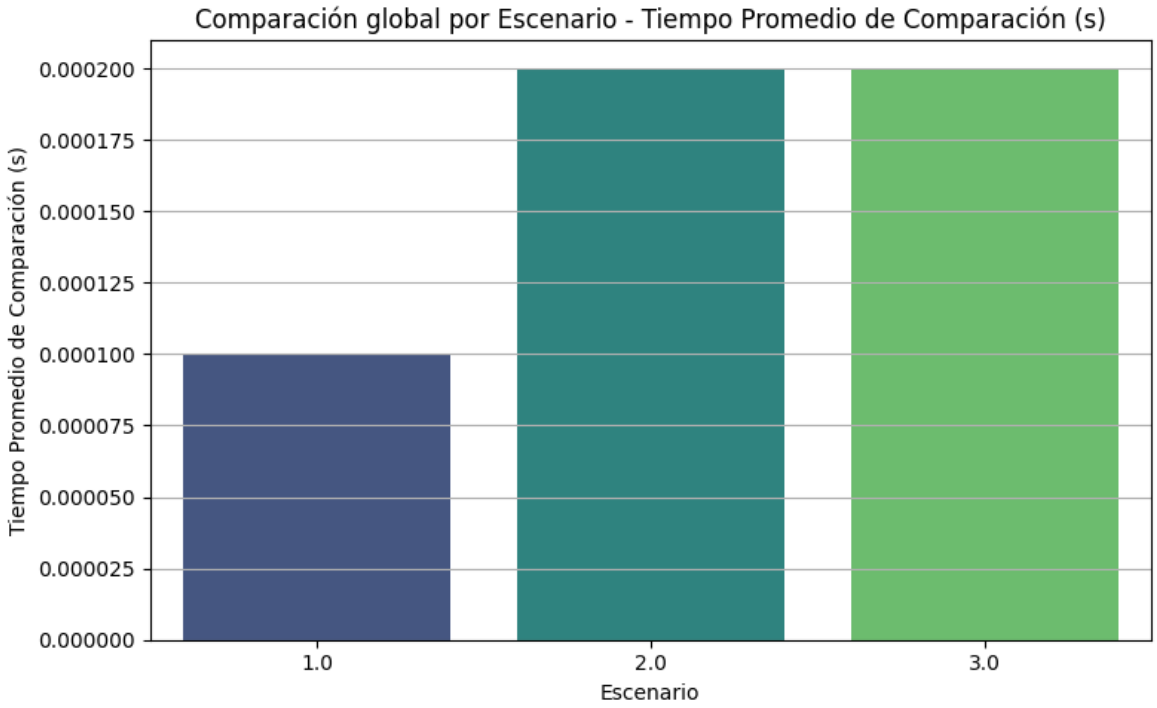


Figura 54: Comparación por Escenario - Tiempo promedio de Comparación

Es importante mencionar que el mayor tiempo de comparación detectado durante todas las pruebas fue de 0.0004, observado en el Escenario 2 bajo condiciones de baja iluminación. Este valor atípico se atribuye a una posible demora puntual en la respuesta de la base de datos, ya que en las siguientes iteraciones de la misma prueba el sistema retomó su comportamiento uniforme con tiempos de 0.0002 segundos.

5.5.4 Tasa de Falsos Negativos

La tasa de falsos negativos representa el porcentaje de casos en los que el sistema no reconoce a una persona autorizada, clasificándola incorrectamente como desconocida. Esta métrica es crítica en sistemas de seguridad, ya que un alto porcentaje puede generar inconvenientes y fallos en el control de acceso. Analizar esta tasa permite identificar debilidades en el algoritmo o en el dataset utilizado, además de orientar mejoras en el entrenamiento y ajustes del sistema.

En la totalidad de las pruebas realizadas, independiente del escenario, se observó un resultado constante del 0% en la tasa de falsos negativos. Esto significa que el sistema nunca identificó erróneamente a una persona registrada como desconocida. En todos los casos donde se logró detectar un rostro, la codificación fue correctamente comparada y asociada a su correspondiente identidad en la base de datos.

Este comportamiento refleja un desempeño sólido del modelo de reconocimiento facial utilizado, particularmente en cuanto a su capacidad para generar representaciones confiables, codificaciones, durante el entrenamiento y compararlas con precisión en tiempo real. Asimismo, denota una adecuada calidad de las imágenes utilizadas para la construcción del dataset, lo que favoreció un mapeo robusto de las características faciales relevantes de cada individuo.

Cabe mencionar que la tasa de falsos negativos no contempla los casos en los que el sistema no logró detectar un rostro, ya que esta métrica evalúa exclusivamente situaciones en las que sí se detecta un rostro, pero se clasifica erróneamente como desconocido. Por esta razón, durante el proceso de recolección de datos, aquellos casos en los que no se detectó un rostro fueron marcados como “n/a”, es decir, no aplica para evitar que influyan en el cálculo de esta tasa ni distorsionen los resultados.

La Figura 55 muestra la comparación por dataset de la tasa de falsos negativos, evidenciando un resultado del 0% en todos los casos. De igual manera, la Figura 56 presenta una comparación por escenario, donde también se mantiene un valor constante de 0%, confirmando la ausencia total de errores de este tipo a lo largo de las pruebas.

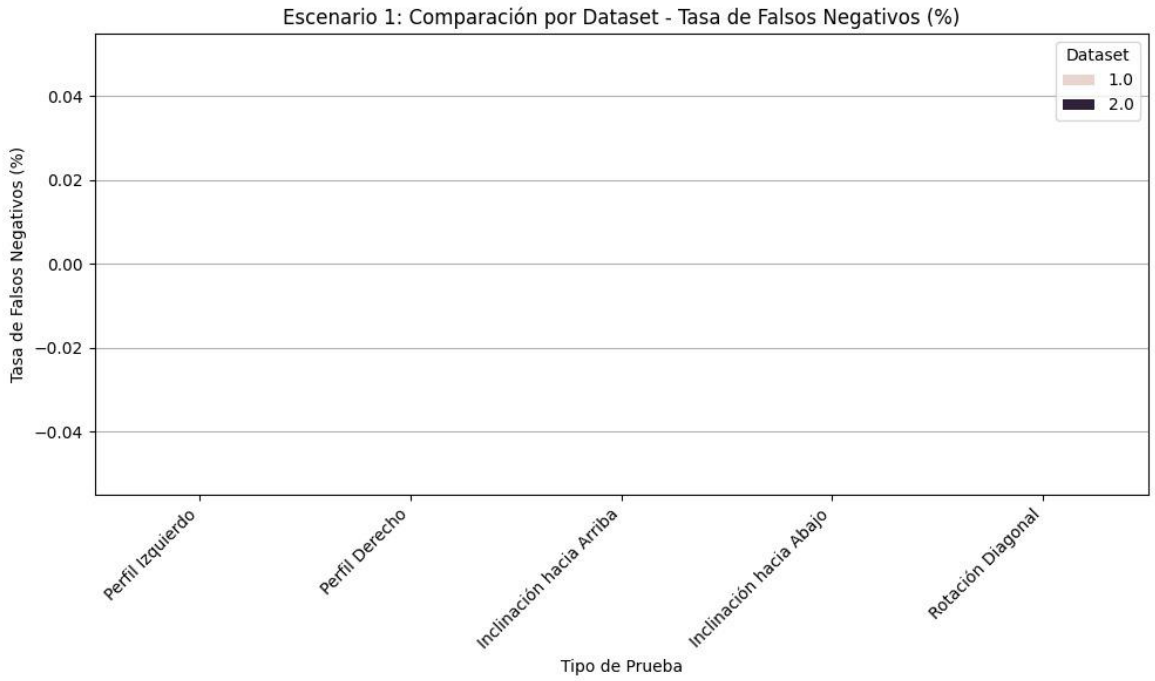


Figura 55: Comparativa por dataset - Tasa de Falsos Negativos

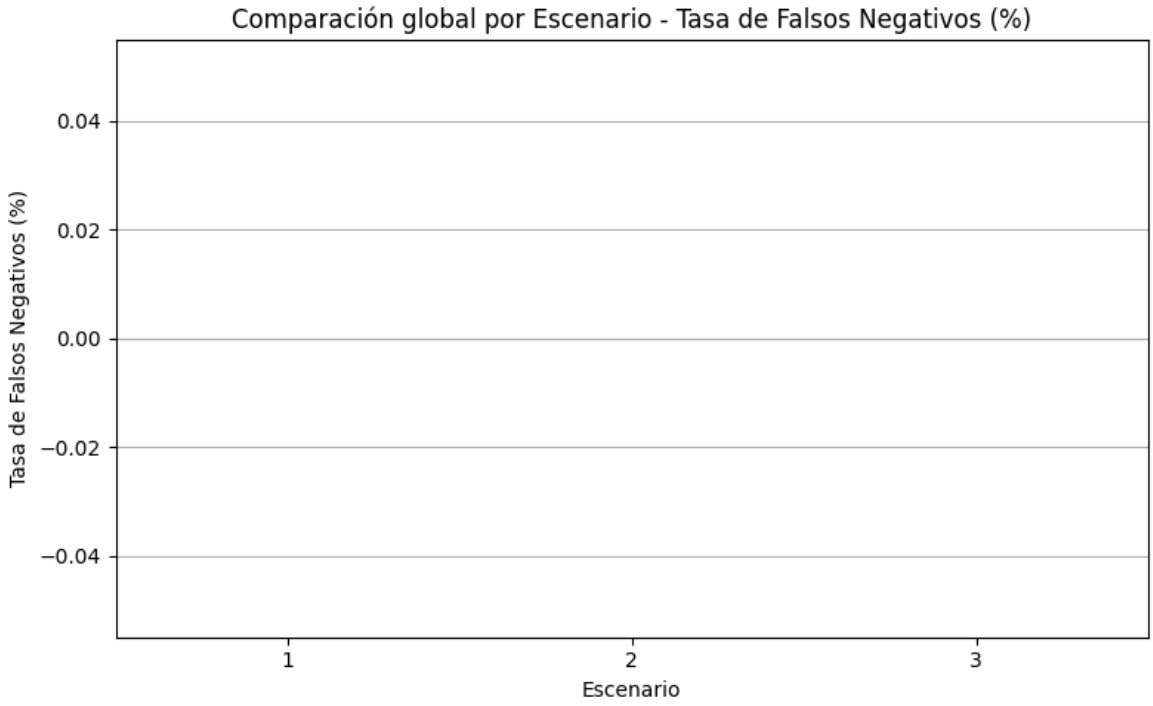


Figura 56: Comparación por Escenario - Tasa de Falsos Negativos

Estos resultados refuerzan la eficacia del sistema para reconocer a personas previamente registradas, incluso bajo condiciones variables. No obstante, aunque el 0% es un resultado ideal, se debe tomar en cuenta que la ausencia de falsos negativos podría estar influenciada también por el tamaño del dataset y del número limitado de personas evaluadas.

CAPÍTULO VI: Conclusiones y Recomendaciones

6.1 Conclusiones

El desarrollo del presente trabajo de titulación se enfocó en diseñar y construir un prototipo de un ecosistema de identidad de acceso basado en reconocimiento facial, empleando una Raspberry Pi como un núcleo para el procesamiento biométrico y el control de acceso en entornos físicos. Esta iniciativa buscó brindar una solución efectiva, en tiempo real y con bajo costo, capaz de identificar personas autorizadas mediante tecnologías de código abierto, contribuyendo así a mejorar la seguridad y eficiencia en contextos físicos.

En primer lugar, se llevó a cabo un análisis detallado de los requisitos técnicos y funcionales del sistema, teniendo en cuenta también las limitaciones inherentes al usar Raspberry Pi como dispositivo principal. Esta etapa permitió establecer los parámetros de diseño viables, comprender las capacidades de procesamiento, así como anticipar retos relacionados con la captura y análisis de imágenes en tiempo real. La revisión de estos requisitos fue crucial para orientar la elección de tecnologías y librerías compatibles, garantizando una base sólida para el desarrollo.

Respecto al diseño de la arquitectura del ecosistema, se establecieron de manera clara los componentes de software y hardware requeridos. La estructura final integró la Raspberry Pi junto a su módulo de cámara, una base de datos PostgreSQL para el almacenamiento de codificaciones faciales y los accesos, y un backend desarrollado en Flask que gestiona la lógica de control y permite la comunicación. El uso de tecnologías IoT, junto a una estructura modular del sistema, posibilitó la escalabilidad, mantenibilidad y una operación fluida.

Posteriormente, se implementó un prototipo funcional que integró la captura de imágenes, entrenamiento del modelo y reconocimiento facial en tiempo real. Además, se implementó una interfaz web desarrollada para gestionar personas y visualizar accesos registrados, lo que facilitó la validación del flujo completo de autenticación biométrica y recopilación automática de datos. La integración de estos elementos demostró la viabilidad de utilizar un sistema de bajo costo para tareas complejas de control de acceso.

Finalmente, la evaluación del sistema mediante pruebas funcionales permitió determinar el nivel de precisión alcanzado bajo diferentes condiciones. El sistema mostró un desempeño adecuado en escenarios con iluminación controlada y expresiones faciales neutras, mientras que, en condiciones más complejas como ángulos extremos, variaciones

en la luz o el uso de accesorios afectaron la precisión. No obstante, el sistema mantuvo un tiempo de respuesta bajo y no mostró falsos negativos en los casos donde se logró identificar exitosamente, lo que respalda su aplicabilidad en entornos reales.

En resumen, el trabajo de titulación cumplió con su objetivo de diseñar y desarrollar un prototipo de ecosistema de control de acceso basado en reconocimiento facial, integrando de manera coherente hardware, software y métodos de evaluación funcional. La experiencia obtenida y los resultados alcanzados demuestran que es posible construir soluciones IoT efectivas y asequibles para la verificación de identidad en entornos físicos, sentando las bases para desarrollos más robustos y escalables.

6.2 Recomendaciones

A partir de los resultados obtenidos en la implementación y evaluación del prototipo, se proponen a continuación una serie de recomendaciones enfocadas en mejorar el rendimiento del sistema de reconocimiento facial en futuras versiones. Estas recomendaciones buscan superar las limitaciones identificadas durante las pruebas funcionales, especialmente aquellas relacionadas con las condiciones adversas de iluminación, ángulos extremos del rostro y obstrucciones parciales.

En primer lugar, se recomienda reemplazar el modelo de detección y codificación basado en Histogram of Oriented Gradients (HOG) por un enfoque más avanzado mediante redes neuronales convolucionales (CNN). El uso de CNN permitiría una detección más precisa y tolerante a variaciones angulares, cambios en la iluminación y el uso de accesorios. Sin embargo, esta mejora implica un incremento en el consumo de recursos representando un reto si se mantiene el Raspberry Pi.

Adicionalmente, se recomienda ampliar y diversificar el dataset utilizado para el entrenamiento, incorporando imágenes con distintas expresiones, ángulos, iluminación y accesorios desde la etapa inicial. Esta variación puede potenciar la capacidad del modelo para generalizar y reconocer rostros en condiciones reales. Para ello, se podrían emplear técnicas como *data augmentation.* Esta técnica consiste en generar artificialmente nuevos datos a partir de los datos existentes (AWS, 2025), en el contexto del prototipo se la usaría para simular estas variaciones artificialmente, optimizando la eficiencia del entrenamiento sin requerir demasiadas capturas manuales.

Asimismo, se recomienda planificar pruebas en entornos reales de aplicación, como el registro de asistencia de trabajadores en pequeñas y medianas empresas (PYMEs). Esta perspectiva permitiría evaluar el desempeño del sistema en contextos no controlados, donde variables como la iluminación natural, cambios faciales espontáneos, distancias variables o condiciones del entorno inciden directamente en el comportamiento del sistema. Esta validación aportaría evidencia clave sobre la viabilidad del sistema para su implementación en escenarios reales y proporcionaría datos útiles para futuros ajustes.

Finalmente, se recomienda explorar la integración de modelos de reconocimiento híbridos que combinen la verificación facial con otras formas de autenticación, tales como tokens físicos, códigos temporales o reconocimiento dactilar, con el fin de aumentar la seguridad y reducir la tasa de errores en contextos críticos. Esto permitiría mantener un balance entre precisión, velocidad y seguridad.

Estas recomendaciones representan el punto de partida para una futura evolución del prototipo, enfocada no solo a mejorar la precisión técnica del reconocimiento facial, sino también a garantizar su sostenibilidad y adaptabilidad en escenarios reales de uso intenso y prolongado.

BIBLIOGRAFÍA

Adams, J. (2019, June 18). *Raspberry Pi4 Model B (REDUCED)*. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

Ahamed, H., Alam, I., & Islam, M. M. (2018). HOG-CNN Based Real Time Face Recognition. *2018 International Conference on Advancement in Electrical and Electronic Engineering, ICAEEE 2018*. https://doi.org/10.1109/ICAEEE.2018.8642989

AWS. (2024a). *MongoDB en comparación con PostgreSQL: diferencia entre bases de datos - AWS*. https://aws.amazon.com/es/compare/the-difference-between-mongodb-and-postgresql/

AWS. (2024b). *¿Qué es una API de RESTful? - Explicación de API de RESTful - AWS*. https://aws.amazon.com/es/what-is/restful-api/

AWS. (2025). *¿En qué consiste el aumento de datos?* https://aws.amazon.com/what-is/data-augmentation/

Bagchi, T., Mahapatra, A., Yadav, D., Mishra, D., Pandey, A., Chandrasekhar, P., & Kumar, A. (2022). Intelligent security system based on face recognition and IoT. *Materials Today: Proceedings*, *62*, 2133–2137. https://doi.org/10.1016/J.MATPR.2022.03.353

Barnes, R. (2017). *Thonny on a Raspberry Pi: using the new Python IDE in Raspbian* . https://magazine.raspberrypi.com/articles/thonny

Batta, V. M. B. (2024). Image Processing using Python. *International Journal of Advanced Research in Science, Communication and Technology*, 575–579. https://doi.org/10.48175/ijarsct-17499

Blanco-Gonzalo, R., Lunerti, C., Sanchez-Reillo, R., & Guest, R. (2018). Correction: Biometrics: Accessibility challenge or opportunity? *PLOS ONE*, *13*(4), e0196372. https://doi.org/10.1371/JOURNAL.PONE.0196372

Bootstrap. (2025). *Bootstrap · The most popular HTML, CSS, and JS library in the world.* https://getbootstrap.com/

Burak, D. (2021). *Overview: Creating Python Virtual Environment and Managing Dependency* . https://medium.com/analytics-vidhya/overview-creating-python-virtual-environment-and-managing-dependency-management-25023bb24cb3

Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018). Convolutional Neural Network (CNN) for Image Detection and Recognition. *ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications*, 278–282. https://doi.org/10.1109/ICSCCC.2018.8703316

Das, P. P., Jabid, T., & Shariar Mahamud, S. M. (2015). Single Image Face Recognition based on Gabor, Sobel and Local Ternary Pattern. In *International Journal of Computer Applications* (Vol. 132, Issue 16).

Deymar, A. (2023, May 5). *SQLite vs MySQL: Análisis A Detalle Para Tu Conveniencia*. https://www.hostinger.com/es/tutoriales/sqlite-vs-mysql-cual-es-la-diferencia#SQLite_vs_MySQL

Dipak Ghael, H., Solanki, L., Sahu, G., & Professor, A. (2008). A Review Paper on Raspberry Pi and its Applications. *International Journal of Advances in Engineering and Management (IJAEM*, *2*, 225. https://doi.org/10.35629/5252-0212225227

Dlib. (2022). *dlib C++ Library*. https://dlib.net/

Elmahmudi, A., & Ugail, H. (2021). A framework for facial age progression and regression using exemplar face templates. *Visual Computer*, *37*(7), 2023–2038. https://doi.org/10.1007/S00371-020-01960-Z

Elordi, U., Lunerti, C., Unzueta, L., Goenetxea, J., Aranjuelo, N., Bertelsen, A., & Arganda-Carreras, I. (2021). Designing Automated Deployment Strategies of Face Recognition Solutions in Heterogeneous IoT Platforms. *Information 2021, Vol. 12, Page 532*, *12*(12), 532. https://doi.org/10.3390/INFO12120532

Geitgey, A. (2022, June 10). *face_recognition/face_recognition/api.py at master · ageitgey/face_recognition · GitHub*. https://github.com/ageitgey/face_recognition/blob/master/face_recognition/api.py

Gupta, H. (2024, October 26). *Client-Server Architecture Explained with Examples, Diagrams, and Real-World Applications*. https://medium.com/nerd-for-tech/client-

server-architecture-explained-with-examples-diagrams-and-real-world-applications-407e9e04e2d1

IBM. (2021a, September 22). *What Is Machine Learning (ML)?* . Https://Www.Ibm.Com/Think/Topics/Machine-Learning. https://www.ibm.com/think/topics/machine-learning

IBM. (2021b, October 6). *What are Convolutional Neural Networks?* Https://Www.Ibm.Com/Think/Topics/Convolutional-Neural-Networks. https://www.ibm.com/think/topics/convolutional-neural-networks

IBM. (2023). *¿Qué es una máquina de vectores soporte?* https://www.ibm.com/es-es/think/topics/support-vector-machine

IBM. (2024, December 27). *¿Qué es una base de datos relacional?* Https://Www.Ibm.Com/Mx-Es/Topics/Relational-Databases. https://www.ibm.com/mx-es/topics/relational-databases

Jack Chan, R. C. (2019). *Python API Development Fundamentals: Develop a full-stack web application ... - Jack Chan, Ray Chung, Jack Huang - Google Libros*. https://books.google.es/books?hl=es&lr=&id=lhnADwAAQBAJ&oi=fnd&pg=PP1&dq=flask+python&ots=Jx0HdEBG3n&sig=eGd7LQOoMuk0Aa1bb_QPiiWe8-E#v=onepage&q=flask%20python&f=false

Jain, A. (2024, December 17). *All about HOG (Histogram of Oriented Gradients)* . Https://Medium.Com/@abhishekjainindore24/All-about-Hog-Histogram-of-Oriented-Gradients-869b5fab7bd5. https://medium.com/@abhishekjainindore24/all-about-hog-histogram-of-oriented-gradients-869b5fab7bd5

Malkauthekar, M. D. (2013). Analysis of Euclidean distance and Manhattan distance measure in face recognition. *IET Conference Publications*, *2013*(CP646), 503–507. https://doi.org/10.1049/CP.2013.2636

Masud, M., Muhammad, G., Alhumyani, H., Alshamrani, S. S., Cheikhrouhou, O., Ibrahim, S., & Hossain, M. S. (2020). Deep learning-based intelligent face recognition in IoT-cloud environment. *Computer Communications*, *152*, 215–222. https://doi.org/10.1016/J.COMCOM.2020.01.050

MDN. (2025). *Uso de Fetch - Referencia de la API Web* . https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

MIPI. (2025). *Camera Serial Interface 2 (MIPI CSI-2) | MIPI*. https://www.mipi.org/specifications/csi-2

Morcillo, F. (2020). *Desarrollo de un sistema de reconocimiento facial utilizando Deep Learning con OpenCV*.

Mukhopadhyay, S. C., & Suryadevara, N. K. (2014). Internet of Things: Challenges and Opportunities. *Smart Sensors, Measurement and Instrumentation*, *9*, 1–17. https://doi.org/10.1007/978-3-319-04223-7_1

NumPy. (2024). *NumPy documentation* . https://numpy.org/doc/stable/

Oloyede, M. O., Hancke, G. P., & Myburgh, H. C. (2020). A review on face recognition systems: recent approaches and challenges. *Multimedia Tools and Applications*, *79*(37–38), 27891–27922. https://doi.org/10.1007/S11042-020-09261-2/METRICS

OpenCV. (2025, January 8). *OpenCV: Introduction*. https://docs.opencv.org/4.11.0/d1/dfb/intro.html

Platero, D. C. (2015). *RECONOCIMIENTO DE IMÁGENES FACIALES ORIENTADO A CONTROLES.*

Psycopg. (2021). *Psycopg – PostgreSQL database adapter for Python — Psycopg 2.9.10 documentation*. https://www.psycopg.org/docs/

PyPI. (2024). *requests · PyPI*. https://pypi.org/project/requests/

Python. (2024). *threading — Paralelismo basado en hilos*. https://docs.python.org/es/3.8/library/threading.html

Python. (2025a). *pickle — Serialización de objetos en Python* . https://docs-python-org.translate.goog/3/library/pickle.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Python. (2025b). *venv — Creation of virtual environments* . https://docs.python.org/3/library/venv.html

Raspberry. (2021, February). *¿Que es Raspberry Pi? - Raspberry Pi*. Https://Raspberrypi.Cl/Que-Es-Raspberry/. https://raspberrypi.cl/que-es-raspberry/

Raspberry Pi. (2025a). *Camera - Raspberry Pi Documentation*. https://www.raspberrypi.com/documentation/accessories/camera.html

Raspberry Pi. (2025b). *Connect the Camera Module | Getting started with the Camera Module* . https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/2

Raspberry Pi. (2025c). *Raspberry Pi 4 Model B specifications – Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Red Hat. (2023, January 20). *¿Qué es una API?* . Https://Www.Redhat.Com/Es/Topics/Api/What-Are-Application-Programming-Interfaces. https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Rosebrock, A. (2018). *(Faster) Facial landmark detector with dlib*. https://pyimagesearch.com/2018/04/02/faster-facial-landmark-detector-with-dlib/

Rosebrock, A. (2021). *Face detection with dlib (HOG and CNN)* . https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/

Singh, A. (2025). *HOG Feature Descriptor: Feature Engineering for Images*. https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/#h-what-is-a-feature-descriptor

SINOSEEN. (2024, May 29). *Comprender la interfaz, el protocolo y las normas de mipi: una guía completa*. https://www.sinoseen.com/es/what-is-mipi-protocol-and-standards

Skillcate AI. (2022, August 13). *Histogram of Oriented Gradients (HOG) — Simplest Intuition* . Https://Medium.Com/@skillcate/Histogram-of-Oriented-Gradients-Hog-Simplest-Intuition-2392995f8010. https://medium.com/@skillcate/histogram-of-oriented-gradients-hog-simplest-intuition-2392995f8010

Smith, R. (2019). *A Beginners guide to Building your own Face Recognition System to creep out your Friends | by | TDS Archive | Medium*. https://medium.com/data-science/a-beginners-guide-to-building-your-own-face-recognition-system-to-creep-out-your-friends-df3f4c471d55

Sukhai, N. B. (2004). Access control & biometrics. *2004 Information Security Curriculum Development Conference, InfoSecCD 2004*, 124–127. https://doi.org/10.1145/1059524.1059552

Syafeeza, A. R., Mohd Fitri Alif, M. K., Nursyifaa Athirah, Y., Jaafar, A. S., Norihan, A. H., & Saleha, M. S. (2020). IoT based facial recognition door access control home security system using raspberry pi. *International Journal of Power Electronics and Drive Systems*, *11*(1), 417–424. https://doi.org/10.11591/ijpeds.v11.i1.pp417-424

Thonny. (2025). *Thonny, Python IDE for beginners*. https://thonny.org/

Universidad de Oviedo. (2006). *TEMA 1: COMPONENTES REUTILIZABLES DEL SOFTWARE. 1.1 DISEÑO MODULAR Y COMPONENTES SOFTWARE*.

Vemulapati, P. (2023). *Image Padding Techniques: Reflect Padding(part 2)* . https://medium.com/@Orca_Thunder/image-padding-techniques-reflect-padding-part-2-5a013cd96537

ANEXOS

Tabla 12: *Resultados del Reconocimiento de Perfil Izquierdo*

Reconocido	Tiempo de reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de rostro completamente de perfil izquierdo bajo el modelo entrado con imágenes frontales.

Tabla 13: *Resultados del Reconocimiento de Perfil Derecho*

Reconocido	Tiempo de reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de rostro completamente de perfil derecho bajo el modelo entrenado con imágenes frontales.

Tabla 14*: Resultados del Reconocimiento con Inclinación hacia arriba*

Reconocido	Tiempo de reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,3071	0,0002	No

No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
Si	0,3055	0,0001	No
Si	0,2811	0,0001	No
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de inclinación facial hacia arriba bajo el modelo entrenado con imágenes frontales.

Tabla 15: *Resultados del Reconocimiento con Inclinación hacia abajo*

Reconocido	Tiempo de reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,4264	0,0001	No
Si	0,2507	0,0001	No
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
Si	0,2365	0,0002	No
No	-	-	n/a

Nota: Datos correspondientes a la condición de inclinación facial hacia abajo usando el modelo entrenado con imágenes frontales.

Tabla 16: *Resultados del Reconocimiento con Inclinaciones diagonales leves*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2522	0,0002	No
Si	0,2780	0,0001	No
Si	0,2772	0,0002	No
Si	0,2569	0,0002	No
Si	0,2676	0,0002	No
Si	0,2567	0,0001	No
Si	0,2761	0,0002	No

Si	0,2499	0,0002	No
Si	0,2882	0,0002	No
Si	0,2494	0,0001	No

Nota: Datos correspondientes a la condición de inclinación diagonal leve bajo el modelo entrenado con imágenes frontales.

Tabla 17: *Resultados del Reconocimiento de Perfil Izquierdo (dataset variado)*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de rostro completamente de perfil izquierdo bajo el modelo entrado con imágenes en diferentes ángulos.

Tabla 18: *Resultados del Reconocimiento de Perfil Derecho (dataset variado)*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de rostro completamente de perfil derecho bajo el modelo entrado con imágenes en diferentes ángulos.

Tabla 19: *Resultados del Reconocimiento con Inclinación hacia arriba (dataset variado)*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2714	0,0002	No
Si	0,2610	0,0002	No
Si	0,2643	0,0002	No
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
Si	0,2795	0,0002	No
Si	0,2966	0,0001	No
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de inclinación facial hacia arriba usando el modelo entrenado con imágenes en diferentes ángulos.

Tabla 20: *Resultados del Reconocimiento con Inclinación hacia abajo (dataset variado)*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
Si	0,2728	0,0002	No
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de inclinación facial hacia abajo usando el modelo entrenado con imágenes en diferentes ángulos.

Tabla 21: *Resultados del Reconocimiento con Inclinaciones diagonales leves (dataset variado)*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2566	0,0002	No
Si	0,2450	0,0002	No
Si	0,2754	0,0002	No

Si	0,2611	0,0002	No
Si	0,2630	0,0002	No
Si	0,2684	0,0002	No
Si	0,2761	0,0002	No
Si	0,2533	0,0002	No
Si	0,2477	0,0002	No
Si	0,2582	0,0002	No

Nota: Datos correspondientes a la condición de inclinación diagonal leve usando el modelo entrenado con imágenes en diferentes ángulos.

Tabla 22: *Resultados del Reconocimiento con Baja Iluminación*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2648	0,0004	No
Si	0,2655	0,0002	No
Si	0,2874	0,0002	No
Si	0,2638	0,0002	No
Si	0,2532	0,0002	No
Si	0,2957	0,0002	No
Si	0,2886	0,0002	No
Si	0,2587	0,0002	No
Si	0,2927	0,0002	No
Si	0,2945	0,0002	No

Nota: Datos correspondientes a la condición de baja iluminación.

Tabla 23: *Resultados del Reconocimiento con Exceso de Luz*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2523	0,0002	No
Si	0,2788	0,0002	No
No	-	-	n/a
Si	0,2753	0,0002	No
No	-	-	n/a
Si	0,282	0,0002	No
Si	0,2609	0,0002	No
No	-	-	n/a
No	-	-	n/a
Si	0,2543	0,0002	No

Nota: Datos correspondientes a la condición de exceso de iluminación.

Tabla 24: *Resultados del Reconocimiento con Iluminación Lateral*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
Si	0,2761	0,0002	No
Si	0,2995	0,0002	No
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a
Si	0,3052	0,0002	No
No	-	-	n/a
No	-	-	n/a
No	-	-	n/a

Nota: Datos correspondientes a la condición de iluminación lateral.

Tabla 25: *Resultados del Reconocimiento con Iluminación Directa*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,269	0,0002	No
Si	0,2915	0,0002	No
Si	0,2748	0,0002	No
Si	0,2908	0,0002	No
Si	0,272	0,0002	No
Si	0,2933	0,0002	No
Si	0,2804	0,0002	No
Si	0,2892	0,0002	No
Si	0,2769	0,0002	No
Si	0,287	0,0002	No

Nota: Datos correspondientes a la condición de iluminación artificial directa.

Tabla 26: *Resultados del Reconocimiento con el uso de gorra*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2923	0,0002	No
Si	0,3109	0,0002	No
Si	0,319	0,0002	No
Si	0,2913	0,0002	No
Si	0,2992	0,0002	No
Si	0,2768	0,0002	No
Si	0,2883	0,0002	No
Si	0,2938	0,0002	No
Si	0,3279	0,0002	No
Si	0,2931	0,0002	No

Nota: Datos correspondientes a la condición de uso de gorra durante el proceso de reconocimiento.

Tabla 27: *Resultados del Reconocimiento con el uso de mascarilla*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
Si	0,2997	0,0002	No
Si	0,3027	0,0002	No
Si	0,2973	0,0002	No
Si	0,3176	0,0002	No
Si	0,3313	0,0002	No
Si	0,3047	0,0002	No
Si	0,3109	0,0002	No
Si	0,3221	0,0002	No
No	-	-	n/a

Nota: Datos correspondientes a la condición de uso de mascarilla durante el proceso de reconocimiento.

Tabla 28: *Resultados del Reconocimiento con el uso de gafas de sol*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
No	-	-	n/a
Si	0,3531	0,0002	No
Si	0,3008	0,0002	No

No	-	-	n/a
Si	0,3050	0,0002	No
Si	0,3051	0,0002	No
Si	0,3107	0,0002	No
Si	0,3154	0,0002	No
Si	0,3583	0,0002	No
Si	0,3144	0,0002	No

Nota: Datos correspondientes a la condición de uso de gafas de sol durante el proceso de reconocimiento.

Tabla 29: *Resultados del Reconocimiento durante la expresión de sonrisa*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2878	0,0002	No
Si	0,3038	0,0002	No
Si	0,3119	0,0002	No
Si	0,2764	0,0002	No
Si	0,2736	0,0002	No
Si	0,2836	0,0002	No
Si	0,2939	0,0002	No
Si	0,2785	0,0002	No
Si	0,2864	0,0002	No
Si	0,2982	0,0002	No

Nota: Datos correspondientes al desempeño del sistema de reconocimiento facial bajo la condición de sonrisa.

Tabla 30*: Resultados del Reconocimiento con expresión de fruncido*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2777	0,0002	No
Si	0,2892	0,0002	No
Si	0,2937	0,0002	No
Si	0,2803	0,0002	No
Si	0,2959	0,0002	No
Si	0,2947	0,0002	No
Si	0,2637	0,0002	No
Si	0,2779	0,0002	No
Si	0,2675	0,0002	No

Si	0,295	0,0002	No

Nota: Datos correspondientes al reconocimiento facial cuando el usuario presenta expresión de fruncido.

Tabla 31: *Resultados del Reconocimiento con un ojo cerrado*

Reconocido	Tiempo de Reconocimiento (s)	Tiempo de Comparación (s)	Falsos Negativos
Si	0,2954	0,0002	No
Si	0,3106	0,0002	No
Si	0,312	0,0002	No
Si	0,2729	0,0002	No
Si	0,2624	0,0002	No
Si	0,2789	0,0002	No
Si	0,2677	0,0002	No
Si	0,2823	0,0002	No
Si	0,2782	0,0002	No
Si	0,3179	0,0002	No

Nota: Datos correspondientes al reconocimiento facial cuando el usuario mantiene uno o ambos ojos cerrados.