UNIVERSIDAD POLITÉCNICA SALESIANA DEL ECUADOR

SEDE GUAYAQUIL.

CARRERA DE INGENIERÍA ELECTRÓNICA.

Trabajo de titulación previo a la obtención del título de:
"INGENIERO ELECTRÓNICO"

TEMA:

"Diseño de un Prototipo de Control de Acceso a través de Reconocimiento Facial Mediante la Creación de un Algoritmo Basado en Software Libre Utilizando LattePanda"

AUTORES:

César Armando Nacipucha Nacipucha.

Jonathan Joel Frías Pérez.

TUTOR:

Lenin Cevallos Robalino, PhD

GUAYAQUIL, ECUADOR

2020

INFORMACIÓN DE LOS AUTORES.

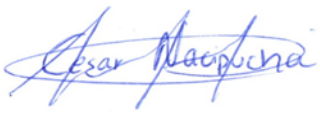
César Armando Nacipucha Nacipucha.
Estudiante de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana.
Email: cesar_nacipucha@hotmail.com, cnacipuchan@est.ups.edu.ec

Jonathan Joel Frías Pérez.
Estudiante de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana.
Email: Jofri-147@hotmail.com, jfrias@est.ups.edu.ec

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN.

Nosotros César Armando Nacipucha Nacipucha y Jonathan Joel Frías Pérez, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además, declaramos que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusividad de los autores.



César Armando Nacipucha Nacipucha

Cédula: 0952600427

Jonathan Joel Frías Pérez

Cédula: 0954691473

Diciembre 2020

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR.

Nosotros, César Armando Nacipucha Nacipucha con documento de identificación N°0952600427, y Jonathan Joel Frías Pérez con documento de identificación N°0954691473, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos los autores del trabajo de titulación, titulado: Diseño de un Prototipo de Control de Acceso a través de Reconocimiento Facial Mediante la Creación de un Algoritmo Basado en Software Libre Utilizando LattePanda. mismo que ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

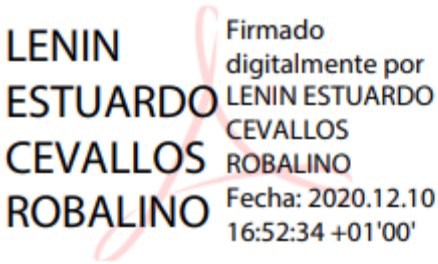
César Armando Nacipucha Nacipucha
Cédula: 0952600427

Jonathan Joel Frías Pérez
Cédula: 0954691473

Diciembre 2020.

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN SUSCRITO POR EL TUTOR

Certifico que el presente proyecto técnico fue desarrollado por los estudiantes Cesar Armando Nacipucha Nacipucha y Jonathan Joel Frías Pérez bajo mi supervisión; y, por lo tanto, se encuentran aptos para su correspondiente presentación.



Lenin Cevallos Robalino, PhD

Director de proyecto de titulación.

Universidad Politécnica Salesiana

Sede Guayaquil

Diciembre 2020

DEDICATORIA.

Dedico este proyecto de titulación a Dios, a mi padre César Nacipucha y a mi madre Julia Nacipucha, quienes en todo momento me han brindado su apoyo incondicional pese a las dificultades o circunstancias que hemos atravesado y por ser la pieza fundamental en mi vida e inspiración para poder cumplir mis objetivos.

Cesar Armando Nacipucha Nacipucha.

DEDICATORIA.

Dedico este proyecto a mi padre Gustavo Frías, quien como guía estuvo presente en el caminar de mi vida por su amor y apoyo incondicional dándome fuerzas para continuar con mis metas trazadas sin desfallecer a pesar de las difíciles circunstancias. A mi madre por su amor constante y dedicación en mi cuidado para lograr ser un hombre honesto y agradecido con Dios.

Jonathan Joel Frías Pérez.

AGRADECIMIENTO.

En primer lugar, doy gracias a Dios por todas las bendiciones que he recibido y toda la sabiduría que me ha brindado para poder llegar a este punto, agradezco a mis padres y hermanos que en todo momento me han sabido apoyar para seguir adelante, para ser una persona de bien y luchar por mis objetivos, por último, agradezco a todos los docentes con quien tuve el honor de recibir clases, adquiriendo diariamente los conocimientos con los que hoy he podido formarme de forma profesional.

Cesar Armando Nacipucha Nacipucha.

AGRADECIMINETO.

Quiero expresar mi gratitud a Dios, quien con su bendición llena siempre mi vida y por permitirme llegar hasta este momento tan importante de mi formación profesional. A mis padres por estar siempre presentes y han sabido darme su ejemplo de trabajo y honradez.

Mi profundo agradecimiento a mi tía Patricia Pérez a quien quiero como una madre, por su apoyo constante durante toda mi etapa universitaria. Así mismo, deseo expresar mi reconocimiento a mis maestros por compartir sus conocimientos.

Jonathan Joel Frías Pérez.

RESUMEN.

El presente trabajo se realizó para crear un prototipo para el control de accesos utilizando un sistema de reconocimiento facial, con esto se garantizará un nivel alto de seguridad para que los usuarios puedan acceder a un determinado sitio en cualquier momento, relacionando el uso de los sistemas biométricos convencionales y prevenir cualquier tipo de vulneración.

Para ello se pretende diseñar un sistema que haga uso del reconocimiento de rostros de los usuarios empleando algoritmos inteligentes, los cuales se verán implementados en un software que trabajará en conjunto con el dispositivo LattePanda, en vista a que es una tarjeta de alto rendimiento el cual soportará todo tipo de procesos a emplearse.

Para la administración de la plataforma se ha creado una base de datos a través de un servidor en la nube. De esta forma se puede acceder en cualquier momento y lugar y permite monitorear los datos generados por los dispositivos. Permitiendo como resultado final un dispositivo con un aplicativo de fácil uso que otorgue a sus usuarios una mayor seguridad en la autenticación.

ABSTRACT.

The present work was carried out to create a prototype for access control using a facial recognition system; this will guarantee a high level of security so that the users can access a certain site anytime they want, relating the use of conventional biometric systems and preventing any type of violation.

For this, it is intended to design a system that makes use of the users' face recognition using intelligent algorithms, which will be implemented in software that will work in conjunction with the LattePanda device, given that it is a high-performance card that will support every type of processes to be used.

For the administration of the platform, a database has been created through a server in the cloud. In this way, anyone can access at any time and place, and allows monitoring the data generated by the devices. Allowing as a final result a device with an easy-to-use application that gives its users greater security in authentication.

ÍNDICE DE CONTENIDO.

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

CAPÍTULO I

INTRODUCCIÓN.

Actualmente, casi todas las instituciones cuentan con un sistema de seguridad para poder gestionar el acceso de su personal a un determinado sitio, así como también llevar un registro de los horarios y fechas en que estos acceden, con el fin de tener un mejor control y una mayor privacidad de los ingresos.

Sitios como industrias, bancos, data centers son lugares en los que ameritan el uso de sistemas de control de accesos más allá de los estándares convencionales, por lo regular, los sistemas más utilizados se basan en huellas dactilares o tarjetas magnéticas los cuales proveen un nivel de seguridad relativamente bueno, sin embargo, al tratarse de entornos cerrados los sistemas de reconocimiento facial se catalogan como una mejor opción al utilizar métodos eficaces y no invasivos.

Existen en el mercado un sin número de marcas de estos dispositivos enfocados en brindar seguridad sin embargo resultan ser de una limitada funcionalidad y sus costos son considerablemente altos en comparación al resto de tecnologías.

El objetivo del trabajo de titulación es crear un prototipo utilizando el reconocimiento facial para garantizar la funcionalidad y el control de accesos fundamentándose en el software libre. Además, se utiliza el dispositivo LattePanda, este es el elemento central junto a una base de datos, un aplicativo web y una aplicación propia con los estándares que garanticen el servicio de reconocimiento facial.

1.1. Descripción del problema.

Los sistemas de seguridad de acceso utilizados por la mayoría de las entidades no son los adecuados para ciertos objetivos, en los cuales se tiene como propósito salvaguardar el ingreso de personas no autorizadas hacia un determinado sitio. La gran parte de ellas emplean herramientas como las tarjetas magnéticas, códigos de seguridad o detección por huella dactilar, lo cual es beneficioso por su bajo costo de adquisición e implementación, sin embargo, son métodos que conllevan un alto riesgo de vulnerabilidad debido a que existen distintos factores que, en caso de darse, no se podría identificar que ha existido una violación a la seguridad.

Tomando como ejemplo las tarjetas magnéticas, la perdida de una de ellas podría significar un riesgo de gran nivel, ya que personas que no pertenecen a una empresa o a un determinado lugar podrían acceder a ella con total facilidad, ocasionando daños o afectaciones de toda índole. En el caso de las huellas dactilares, aunque es más difícil hoy en día ya existen métodos para clonar o duplicar el patrón que existe en dichas huellas y de esta forma poder vulnerar este tipo de sistemas.

Pese a todas estas formas de quebrantar la seguridad también existen sistemas que garantizan altos grados de fiabilidad tales como los sistemas biométricos por detección facial, los cuales son muy poco utilizados debido a que su forma de administrar conlleva un poco más de complejidad y sus costos en el mercado son considerablemente mayores en comparación al resto de sistemas, por estas razones estos dispositivos se ven opacados pese a ser una opción viable en lo que respecta la garantía de un buen nivel de seguridad.

En función de esta necesidad se realiza el planteamiento del proyecto con el fin de afrontar los parámetros expuestos con anterioridad, mediante la creación de un prototipo de reconocimiento facial de fácil funcionamiento y que garantice un nivel de seguridad optimo en el control de accesos.

1.2. Antecedentes.

Los sistemas de accesos biométrico son herramientas indispensables en el diario vivir y su objetivo principal es garantizar un alto grado de seguridad ante la necesidad de acceder a un determinado sitio.

El uso de los sistemas de reconocimiento de rostros para el registro de sus usuarios o control de accesos restringidos se conocen como herramientas seguras con trayectoria donde se destaca su tasa de error mínima al utilizar técnicas nuevas de algoritmos que permiten complementarse con otros sistemas.

Por lo tanto, la implementación del reconocimiento facial permite garantizar la seguridad y el poder de acceso sobre los sitios disponibles de las empresas para controlar las acciones realizadas como, por ejemplo, el ingreso del personal mediante su registro facial.

1.3. Importancia y alcance.

La importancia de la creación del prototipo es brindar un acceso seguro y garantizar ingresos de personas ya sea esto en una empresa, edificio de gobierno o lugares con acceso altamente restringido. Así se garantiza seguridad a los usuarios y a la empresa debido a que trabaja con reconocimiento facial mediante el procesamiento de imágenes

y algoritmos que permiten la detección de rasgos faciales de cada persona; por lo tanto, se busca la generación de un prototipo económico mediante la utilización del software libre y Lattepanda debido a que se busca su introducción en los sistemas de acceso para que las personas puedan conocer su valor y adquirirlos.

1.4. Delimitación del problema.

La delimitación del proyecto de titulación se lleva a cabo en términos concretos y se divide de la siguiente forma:

1.4.1 Delimitación temporal.

Este proyecto tiene una duración aproximada de 12 meses como plazo normal y en caso de una posible extensión a máximo 13 meses desde el momento de su aprobación por las entidades del Consejo de Carrera de Ingeniería Electrónica.

1.4.2. Delimitación académica.

Con este prototipo se busca exponer a los estudiantes la posibilidad de construcción de prototipos utilizando recursos accesibles como Lattepanda y software libre que permite el reconocimiento facial en colaboración con la plataforma y la base de datos que permite poner en práctica los conocimientos obtenidos durante la vida académica en la Universidad Politécnica Salesiana.

1.5. Objetivos del proyecto.

1.5.1. Objetivo General.

Diseñar un prototipo de control de acceso biométrico basado en reconocimiento facial mediante la creación de un algoritmo empleando la tarjeta LattePanda.

1.5.2. Objetivos Específicos.

Este proyecto tiene los siguientes objetivos específicos:

- Diseñar un prototipo para el control de acceso.
- Desarrollar un algoritmo para el procesamiento de imágenes mediante lenguaje de programación Python.
- Diseñar un sistema de base de datos para controlar y mantener un registro de usuarios.
- Desarrollar un sistema de emisión de alarmas para prevenir accesos no autorizados.

CAPÍTULO II

2. FUNDAMENTOS TEÓRICOS.

2.1. LattePanda

El dispositivo LattePanda es un microcomputador que dispone de un hardware robusto en comparación al resto de tarjetas de su misma clase, la placa es capaz de trabajar con sistemas operativos al nivel de Windows 10 y ejecutar cualquier tipo de programas gracias al uso de potentes procesadores como lo es el Chipset Intel Atom – Cherry Trail que garantiza una buena calidad al momento de ejecutar varias tareas de forma simultánea., además de esto, el dispositivo incorpora una tarjeta Arduino dentro del mismo hardware haciendo uso de un coprocesador ATMMega32u4 e implementa varios puertos como lo es la salida HDMI, Ethernet, Jack 3.5 mm, Bluetooth 4.0, WiFi, USB 2.0 – 3.0, tarjeta microSD, salida para pantalla táctil, entre otros (LattePanda Team 2017).

Figura 1. Logo de LattePanda.

Fuente: LattePanda (2017)

LattePanda tiene una amplia variedad de aplicaciones en distintos campos tales como procesamiento de señales, desarrollo de aplicaciones para el control y automatización industrial, diseño de redes, etc. Esto se debe gracias a sus componentes internos y a la

compatibilidad existente con Arduino, logrando el uso eficiente de actuadores y sensores que garantizan a los usuarios el cumplimiento de las necesidades en dichas ramas.

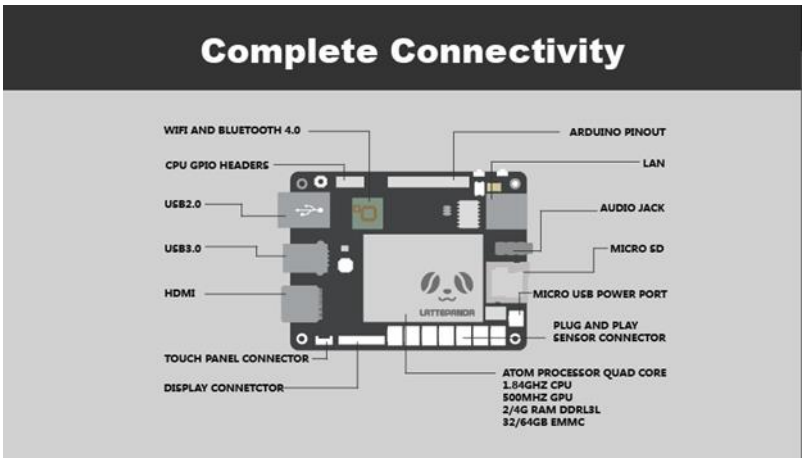


Figura 2. Conexiones de la tarjeta de LattePanda.

Fuente: (LattePanda Team, 2017)

2.1.1. Ventajas de utilizar LattePanda.

- El costo del hardware es totalmente accesible y su uso resulta completamente eficiente, debido a que se puede desarrollar una gran variedad de proyectos o mejoras de los ya existentes, dependiendo de la imaginación y creatividad del desarrollador.
- La compatibilidad con diferentes sistemas operativos, que le otorga una amplia gama de software a los usuarios, otorgando múltiples opciones para el desarrollo de proyectos.
- La incorporación de Arduino dentro de la tarjeta evita la necesidad de tener que adquirir el dispositivo por separado, permitiendo el uso de sensores y actuadores dentro del hardware.

- El uso de software libre brinda a los estudiantes la oportunidad de desarrollar sus proyectos sin la obligación de adquirir licencias.

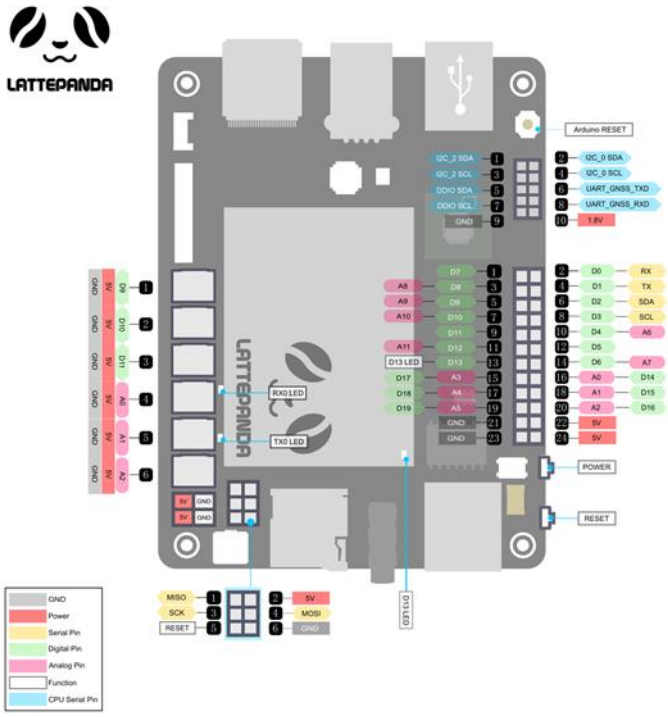


Figura 3. Diagrama de pines GPIO.

Fuente: (LattePanda Team, 2017)

2.1.2. Versiones de LattePanda.

LattePanda cuenta con una variedad de versiones en la cual las características del hardware varían, con la finalidad de poder brindar a sus usuarios opciones de acuerdo a lo que se necesite implementar. A continuación, una lista de los modelos disponibles y sus principales características.

LattePanda (2G/32G). Sus principales características son el uso de un procesador Intel Atom X5-Z8300 que brinda mayores prestaciones para el desarrollo de proyectos de hardware y software, trabaja con una frecuencia de 1,8 GHz, es capaz de soportar sistemas operativos como Windows 10 Home Edition, posee una memoria RAM de 2GB DDR3L y 32 GB de memoria Flash adicional a ello incluye una ranura de expansión para adicionar una memoria micro SD, posee conexión WIFI, Bluetooth 4.0, un conector LAN de 100 Mbps. También incorpora un coprocesador ATmega32U4 para el funcionamiento del Arduino haciendo uso de los conectores GPIO (20 conectores en total) los cuales mejoran el desempeño al momento de emplear sensores y actuadores (Androidpc 2015).

LattePanda (4G/64G). Sus características son similares a su antecesor el LattePanda (2G/32G), con la diferencia del procesador el cual integrara un Intel Cherry Trail Z8350 Quad Core con frecuencia de hasta 1.92 GHz, incluye una memoria RAM de 4GB DDR3L, cuya capacidad es de 64 GB de almacenamiento, con ranura de expansión y en cuanto a los conectores GPIO hará uso de 6 conectores adicionales para el uso de sensores de gravedad Plug and Play, el coprocesador del Arduino en este modelo es el mismo de su versión anterior el ATmega32u4 (Amazon 2019).

LattePanda Delta 432. Su principal característica es el uso de un procesador de Intel de 8va Generación Celeron N4100 de 4 núcleos con frecuencias de 1,1 a 2,4 GHz, integra un procesador para gráficos Intel UHD Graphics 600, 200-700 MHz, posee una memoria RAM de 4 GB LPDDR4 y una de almacenamiento de 32 GB, Wifi, Bluetooth 5.0, conexión LAN de 1 Gbps y coprocesador Arduino Leonardo sumando otras características como 2x 50p GPIO incluyendo I2S, I2C, TTL ,USB, UART, RTC, energía administrable, etc. (Ros 2019)

LattePanda Alpha 800s. Este modelo integra un procesador Intel 8th-8100Y Dual Core de 1,1 – 3,4 GHz, para la parte gráfica incorpora un procesador Intel HD Graphics 615, 300 – 900 MHz, posee una memoria RAM de 8 GB LPDDR3 pero sin memoria de almacenamiento interno, para este apartado se necesita de hacer uso de una memoria externa de tipo 1x M.2 M Key , PCLe 4x compatible con SSD SATA o NVMe, incorpora, Wifi, Bluetooth 4.2, posee un puerto LAN de 1 Gbps y al igual que el modelo Delta 432 pose el Arduino Leonardo de iguales características (Aliexpress 2019).

LattePanda Alpha 864s. Las características son casi en su totalidad similares al modelo Alpha 800s con la única diferencia de que este modelo incorpora una memoria de almacenamiento interno de 64 GB (XATAKA Móvil 2019).

2.1.3. Sistema operativo.

El microcomputador LattePanda se diseño con la capacidad necesaria para adoptar sistemas operativos, entre los más populares tenemos Windows y GNU/Linux. Linux a diferencia del resto de sistemas es Open Source es decir de código abierto, facilitando la personalizacion con el fin de volverse más asaptable al usuario (Pérez, Gonzalez, Vásquez y Pérez, 2016)

En la actualidad ya se encuentran desarrolladas varias distribuciones del sistema operativo Linux que pueden ser adoptadas para nuestra tarjeta, entre ellas tenemos Debian, Ubuntu, Fedira y Arcg Linux, todas estas correctamente gestionables debido a que no presentan problemas de compatibilidad.

2.2. Python.

Python es un lenguaje de programación, fue creado por Guido Van Rossum, este se mantiene en un proceso de desarrollo continuo y participativo debido a la comunidad de seguidores, por lo que, se permite una actualización aproximadamente cada seis meses; sin embargo, estos cambios no son radicales, sino que se busca enriquecer al programa y aportar compatibilidad con diversos programas (Marzal & Gracia, 2009).

Python tiene como ventaja a la gratuidad de su intérprete, este puede ser descargado desde la página web oficial http://www.python.org. Y cuenta con versiones para cualquier plataforma, en la que trabaje el usuario como Linux, Microsoft, Macintosh de Apple, etc (Marzal & Gracia, 2009).

Por lo tanto, Pyton es conocido como un lenguaje de programación de alto nivel que tiene una síntaxis sencilla y clara para la estructuración de sus programas, ya que cuenta con una lista amplia de librerías disponibles para se empleadas en el desarrollo de proyectos.

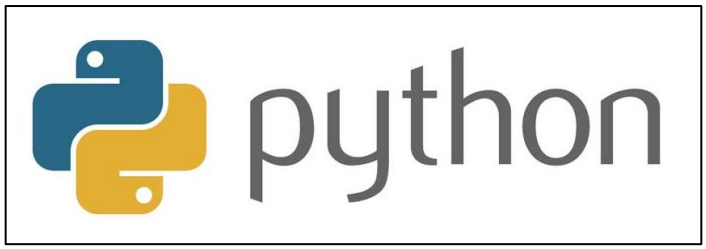


Figura 4. Logo de Python.

Fuente (Programación.net 2020)

2.2.1. Características de Python.

Python es un lenguaje que incluye múltiples ventajas las cuales lo hacen muy atractivo para los desarrolladores en el ámbito profesional, a continuación, mencionaremos las ventajas más destacadas:

- Programación orientada a objetos.
- Hace uso de una sintaxis clara y fácil de interpretar en comparación a los lenguajes de programación disponibles en el mercado.
- Es muy expresivo, por lo tanto, los programas llegan a resultar muy compactos en su estructura, gracias a ello es considerado como un lenguaje de alto nivel.
- Ofrece un entorno de ejecución el cual detecta errores de programación que los compiladores no logran detectar, ofreciendo una serie de información que nos ayuda a identificarlos y poder hacer la corrección necesaria.
- Posee una amplia selección de librerías que se pueden utilizar para poder tratar temas puntuales en un proyecto de programación.

2.3 Biometría.

Biometría proviene de las palabras bio, que significa vida y metría, que significa medida, por lo que, se puede decir, que mediante su utilización identifica las características de las personas, generalmente son fisiológicas como huellas digitales, reconocimiento de iris, geometría de la mano, entre otras para la verificación de su identidad (Tolosa Borja y Giz Bueno s.f.), Sin embargo, el uso o aplicación más frecuente ha sido en los sistemas de seguridad, con el fin de identificar y registrar el acceso de personas a un determinado sitio.

Figura 5. Biometría.

Fuente: (Helmeyer Quevedo, 2017)

2.3.1. Sistemas biométricos.

El sistema biométrico es un conjunto de procedimientos que tiene como propósito de recolectar datos (características biológicas de un individuo) de forma automática, almacenarlos y compararlos de una forma precisa y confiable. En su gran mayoría son utilizados en los sistemas de acceso, en vista a que proporcionan un alto grado de seguridad debido a que se utilizan patrones únicos e irrepetibles de una persona. El reconocimiento de los patrones se realiza en tres etapas:

- La primera etapa permite el uso de los sensores para la toma de datos y la definición del patrón
- La segunda etapa se relaciona con la extracción de datos básicas para la creación de base de datos
- La tercera y última etapa permite la clasificación de los patrones de acuerdo con las clases.

2.3.2. Tipos de sistemas biométrico.

Los tipos de sistemas biométricos dependen de las características biológicas que se vallan a evaluar. Por lo tanto, las podemos dividir en dos grupos:

- Sistemas biométricos basados en características fisiológicas.
- Sistemas biométricos basados en el comportamiento.

Los sistemas biométricos basado en características fisiológicas o también denominadas estáticas son aquellos que hacen uso de las características fisiológicas, únicas e irrepetibles de un individuo como lo son la huella dactilar, la retina, el rostro, el iris, entre otras. Por otro lado, tenemos los sistemas biométricos basados en el comportamiento o también conocidas como dinámicas, son aquellas que evalúan las características producidas por una acción en particular como lo es la voz humana, los movimientos corporales, los gestos, entre otras. A continuación, se detallarán cada uno de los sistemas biométricos más utilizados en la actualidad.

2.3.2.1. Sistema biométrico por reconocimiento de huella dactilar.

El reconocimiento a través de la huella dactilar es la herramienta más utilizada en los sistemas biométricos, debido a su alto grado de seguridad ya que los patrones de las huellas son únicas e irrepetibles en cada persona. En la actualidad existen dos métodos para evaluar la coincidencia de muestras, el primero se basa en la determinación de minucias, que consiste en la comparación de formas sencillamente identificables y el segundo método se basa en la correlación de patrones, el cual analiza de manera global todos los rasgos que se encuentran contenidas dentro de la huella dactilar.

2.3.2.2. Sistema biométrico por reconocimiento del iris.

El reconocimiento mediante el patrón de iris corresponde al método eficiente y estable que tiene el mercado, no obstante, su proceso es uno de los más complejos debido a que se requiere de herramientas que logren garantizar precisión al momento de extraer patrones referentes al iris del individuo, para ello se emplean cámaras infrarrojas especializadas en retratar la imagen en alta resolución y haciendo uso de algoritmos de codificación esta imagen es convertida en un código único que la persona en particular.

2.3.2.3. Sistema biométrico por reconocimiento de retina.

La identificación por retina ocular se realiza a través del escaneo de patrones de los vasos sanguíneos, dichos datos son extraídos de la pupila de la persona y para ello se hace uso de un rayo infrarrojo de baja potencia con la finalidad de escanear la ruta que forman los vasos, puesto a que estos son más absorbentes a dicha luz y el patrón que resulta de este escaneo se convertirá en un código mediante el uso de algoritmos para luego ser utilizado en los sistemas de reconocimiento.

2.3.2.4. Sistema biométrico por reconocimiento de geometría de la mano.

Mientras que los sistemas que ejecutan sus operaciones por medio de la geometría de la mano son considerados los más rápidos entre las herramientas biométricas, con una tasa de error aceptable, para ello, se hace uso de lectores que en conjunto a unas guías dotadas de sensores realizan el escaneo de la forma de la mano (longitud, área, ancho, etc.), para luego computarizarlo mediante un algoritmo y llevarlo a un plano 3D para su respectivo almacenamiento en una base de datos.

2.3.2.5. Sistema biométrico por reconocimiento facial.

El reconocimiento facial es aquel sistema que nos permite efectuar la autenticación mediante la evaluación de patrones biométricos faciales de un determinado individuo, previo a ello es necesario adquirir la imagen real de la persona a la cual se le extraerán las características del rostro mediante el uso de un algoritmo que está ligado a un complejo proceso matemático, y posteriormente dichos datos son almacenados para efectuar la comparación y autenticación en tiempo real.

2.3.2.6. Sistema biométrico por reconocimiento de firmas.

El sistema de reconocimiento mediante firmas de una persona es aquella que determina la veracidad de una conducta, es decir estudia la relación del aspecto tales como lo es la velocidad, la concavidad, la ubicación (inicio y fin del trazo), el centro geométrico y el grado de inclinación de cada trazo que se efectúe. Las conductas son almacenadas para luego ser comparadas mediante el uso de algoritmos de coincidencia que evaluaran estas conductas

2.3.2.7. Sistema biométrico por reconocimiento de voz con redes neuronales.

El reconocimiento mediante voz es un sistema basado en la evaluación de las variaciones de vos, es decir que mediante el uso de un micrófono y algoritmos de reconocimiento usando redes neuronales se logra identificar estos patrones, de esta manera se puede evidenciar que es uno de los sistemas con menor índice de fiabilidad puesto a que posee un margen de error que se genera por factores externos como lo son el ruido, la interferencia, etc.

2.3.2.8. Sistema biométrico por dinámica del tecleo.

El sistema basado en la dinámica del tecleo hace uso de técnicas como el key-stroke, la duración de pulsaciones y el código morse, ya que en ellas se mantiene un patrón perenne y propio de las personas. Para este tipo de sistemas se necesita de un tecleado y de un algoritmo que capte la velocidad del tecleo con el fin de determinar a quién pertenecen dichas características. Sin embargo, este tipo de sistemas esta abiertos a un posible margen de error pues se debe considerar la existencia de factores externos que pueden alterar la captura de datos.

2.4. Detección de rostros.

Una de las grandes dificultades en la identificación de rostros es la detección de estos, por medio de imágenes o fotografías. Para que el algoritmo funcione perfectamente se debe hacer una detección precisa de la imagen. El algoritmo no solo debe detectar el rostro para la identificación o verificación de personas, sino que tiene que tomar en cuenta otros aspectos los cuales podrían dificultar el proceso de detección del rostro en una imagen, tales como: (López Pérez y Toro Agudelo 2012)

- Pose y ángulo del rostro.
- Tamaño del rostro.
- Presencia de lentes, barba, gorros, etc.
- Expresiones en la cara.
- Problemas de iluminación.
- Condiciones de la imagen.
- Cantidad desconocida de caras en la imagen

En la actualidad existen métodos que garantizan un óptimo reconocimiento, los métodos son basados en algoritmos de detección de rostros a través del análisis de datos y se pueden clasificar de la siguiente manera:

Métodos Basados en características Faciales: en este método se intenta hallar los rasgos presentes en el rostro de la persona, tales como son: labios, boca, ojos, cejas, etc.

Métodos Basados en Imágenes: en este método se hace uso de herramientas de reconocimiento de patrones para deducir un modelo a partir de un grupo de imágenes o fotografías de entrenamiento. Utilizan la imagen completa o una parte de ellas, motivo por el cual no se centran en rasgos faciales específicos.

2.4.1. Métodos basados en características faciales.

El desarrollo de esta área se da inicio de los años sesenta, su principal motivación fue el auge tecnológico y las ideas que involucraban a la representación de rostros con el fin de realizar identificaciones. Los estudios han evidenciado las relaciones geométricas entre puntos característicos que han permitido resolver las interrogantes sobre el empleo correcto de dicho método, para poder tener un desarrollo correcto de este método y distinguir las tres ramas relacionada con los métodos, como se indica a continuación:

- **Análisis de histogramas:** Esta técnica se enfoca directamente en el comportamiento de los pixeles, existen dos métodos desarrollados:

- **Método de análisis de regiones:** hace uso de las zonas que poseen una tonalidad más oscura en las regiones del rostro como lo son las pupilas, las cejas, los labios, etc. Se realiza una división de la imagen en regiones de las partes mencionadas y efectúan la comparación con el rostro del individuo para validar un resultado

- **Método de detección de bordes:** es uno de los campos más avanzados para el procesamiento de imágenes, se enfoca en identificar los bordes a partir de una imagen, los afinan, etiquetan y efectúa una comparación con otra imagen con la finalidad de encontrar estructuras similares de un rostro.

- **Análisis de rasgos faciales:** En vista a que el análisis de histogramas no valida completamente el rostro, es decir solo nos brinda información parcialmente correcta, se pude hacer uso de este método para encontrar minuciosamente los rasgos faciales. Se basan totalmente en las similitudes geométricas que coinciden en diferentes rasgos presentes en un rostro y para eso hacemos uso de estos dos criterios:

  - **Búsqueda de rasgos faciales:** el objetivo es localizar de manera ordenada los rasgos característicos del rostro del individuo, todo esto depende de la orientación que se tenga de la cara y las poses que se puedan asumir, por ejemplo, en una fotografía se podría empezar por determinar la frente, seguida de los ojos, nariz, boca, etc.

- **Búsqueda de constelaciones:** la hipótesis central se enfoca en explicar cómo las estructuras llegan a tener una geometría repetitiva y en conjunto con un modelo probabilístico se logra determinar como la estructura puede ir cambiando de forma. Para lograr eso, se hacen uso de imágenes de entrenamiento, un modelo probabilístico para dichas imágenes para luego poder efectuar la toma de decisión entre la imagen de entrenamiento y el rostro de la persona.

- **Análisis por medio de contornos activos:** para este tipo de análisis se emplea el uso de métodos que trabajan con modelos de contorno activos, es decir, tratan de hallar un modelo general de una característica o rasgo facial dentro de la imagen como lo es el contorno de la cara, los ojos, boca, etc. Para lograr emplear este método se hace uso de tres técnicas las cuales se describen a continuación.

  - **Deformación de los patrones:** Buscan adaptar de manera selectiva un modelo correspondiente a los parámetros de cada rasgo, con la finalidad de poder minimizar una función de energía para la adaptación.

  - **Contornos activos:** por lo general son mayormente utilizados para encontrar el contorno del rostro, al igual que el método anterior, tiene como finalidad minimizar la función de energía para adaptar el modelo.

  - **PDM (Modelos de distribución de puntos):** El objetivo de esta técnica es buscar representar las formas de un rostro de manera paramétrica. El proceso para llevar a cabo esto se basa en dividir el contorno PDM en

distintos puntos etiquetados y las posibles variantes que puedan tener dichos puntos se almacenan en un modelo estadístico que se efectúa cuidadosamente a partir de varias imágenes de entrenamiento

2.4.2. Métodos basados en imágenes.

Los métodos basados en imágenes brindan un grado mayor de fiabilidad con respecto a los mencionados anteriormente, debido a que los anteriores no contemplan los cambios que pueden darse en el ambiente, como lo son el cambio de iluminación, el cambio de fondo o simplemente la aparición de otros rostros. Para hacer frente a estos problemas se llegó a determinar las siguientes técnicas.

- **Redes neuronales**: Es una técnica usada con frecuencia para el reconocimiento de patrones de forma indistinta, con ello podemos distinguir si una imagen o fotografía posee uno o varios rostros. Todo esto se logra en varias fases, la primera de ellas denominada la fase de entrenamiento en la cual haremos uso de varias imágenes con rosto y otra cantidad sin ellas para poder entrenar la red neuronal y discernir si un objeto interfiere con la imagen de la cara. En la actualidad existen modelos de redes neuronales de distintos grados de complejidad con las cuales se puede obtener diferentes performances sin embargo todas ella se enfoca en el mismo fin que es reconocer al objeto en cuestión.

- **Análisis probabilístico:** Los métodos estadísticos se vinculan con el sistema de reconocimiento facial, este consiste en la determinación de patrones con relativas variaciones en aspectos como dimensiones y formas. Por lo tanto, métodos como

kullback-Leibler o el principio de verosimilitud tienen como finalidad hallar la distribución de una media, mediante el uso de histogramas para luego proceder con la compasión con el resto de los fotogramas aprendidos y poder darnos el resultado.

2.5. Reconocimiento de rostros.

Los métodos usados para el reconocimiento de rostros una vez una vez que estos ya hayan sido detectados se dividen en dos grupos, los métodos holísticos y los métodos basados en características locales.

Métodos holísticos: uso de modelos comparativos para el reconocimiento de rostros, uno de los más reconocidos es el template matching, donde se toma en cuenta cada píxel para su comparación y es un método imposible de utilizar en tiempo real debido a la cantidad de datos; por lo cual se han desarrollado otros métodos de correlación que toman en cuenta espacios grandes de píxeles que permiten aproximar el rostro y se conoce como subespacio facial, por ejemplo, PCA; LDA, FLD, etc.

Métodos basados en características faciales: son aquellos que se encargan de extraer las características principales del rostro como ojos, cejas, boca, nariz, entre otras. Con la finalidad de poder crear datos de entrada de acuerdo con su posición para luego ser comparados con el resto de las imágenes añadidas en una base de datos

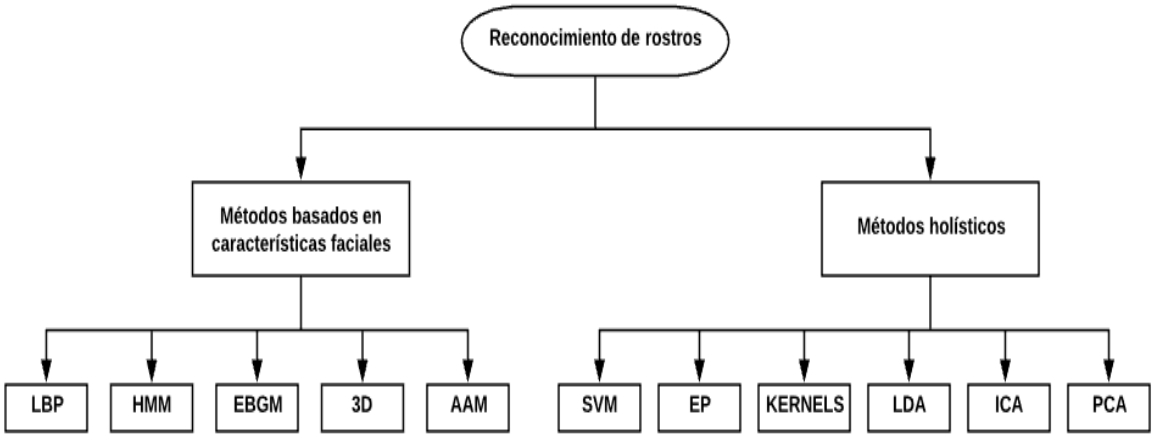


Figura 6. Técnicas para el reconocimiento de rostros.

Fuente: autores.

2.5.1. Métodos holísticos.

- **PCA (**Análisis de componentes principales), mediante esta técnica se realiza la reducción de las dimensiones relacionadas con los datos. Esta técnica hace uso de un mínimo número de variables partiendo de una gran cantidad de variables correlacionada, teniendo como principal objetivo representar una fotografía en términos de posiciones basados en coordenadas que reduzcan el número final de componentes, es decir PCA no ayuda con la obtención de vectores que mejor se ajustan al rostro de cada individuo de manera particular.

- **ICA** (Análisis de componentes independientes), como sus siglas lo indican, este método realiza un análisis de componentes independientes, minimiza las dependencias de segundo orden y de orden superior de los datos ingresados e intenta hallar la base a través de los datos que son estadísticamente

independientes. En base a esto se ha determinado 2 tipos de arquitecturas, la primera corresponde a imágenes de base estadísticamente independientes y la segunda compete a la representación del código factorial.

- **LDA** (Análisis discriminante lineal), corresponde a la búsqueda de vectores en el espacio subyacente que mejor distinga sus clases. Dentro de las muestras se tienen matrices de dispersión para cada clase. Su principal objetivo es maximizar A mientras se minimiza B, es decir maximizar la relación existente a través del uso de determinantes.

- **EP** (Evolutionary Pursuit), tiene un modelo adaptativo que vincula los ejes de proyección para mejorar el comportamiento, a través del tiempo y la capacidad de generalización del sistema. Debido a que la dimensión del espacio de solución de este problema es demasiado grande, se resuelve utilizando un algoritmo denominado Evolutionary Pursuit.

- **Kernel**, partiendo de que la variedad de rostros en el subespacio no necesita ser lineal, los métodos kernel son una generalización de métodos lineales, en donde se exploran los esquemas directos del colector no lineal con la finalidad de aprenderlo.

- **SVM (Maquina de vectores de soporte),** dado un conjunto de puntos pertenecientes a 2 clases, un SVM se encarga de hallar el hiperplano que separa la mayor fracción posible de puntos de la misma clase en el mismo lado mientras maximiza la distancia de cualquiera de las clases. El PCA se

utiliza primero para extraer características de imágenes faciales y luego las SVM aprende las funciones de discriminación entre cada par de imágenes.

2.5.2. Métodos basados en características faciales.

- **EBGM** (Coincidencia grafica de grupos elásticos), cada imagen comparte una estructura topológica similar, toma un aspecto gráfico con nodos en puntos referenciales, así cada nodo tiene 40 coeficientes de ondas de Gabor utilizando escalas y orientaciones conocidas como chorros. El reconocimiento se basa en gráficos etiquetados, estos tienen nodos para conectar bordes donde se utiliza distancias y etiquetados con chorros.

- **AAM (Modelo de apariencia activa),** modelo que combina un modelo de variación de apariencia donde se tiene un marco normalizado y un modelo estadístico que permite la forma y la apariencia del objeto para generalizarse a otros datos. Esto permite la detección de parámetros para manejar las diferencias entre imágenes y el modelo sintetizados en la imagen.

- **Reconocimiento facial 3D,** este enfoque es la capacidad permite la comparación de superficies independientemente de las deformaciones que son resultantes de las expresiones de cada persona. Para el proceso, se obtiene la imagen junto con la textura del rostro mientras que se elimina el cabello u otros aspectos para facilitar el proceso de reconocimiento. Y se realiza el cálculo canónico de la superficie facial; aunque no se toma en

cuenta las diferentes orientaciones que se pueden obtener de la cabeza y otras expresiones.

- **HMM (Modelos ocultos de Maknov),** pertenecen a un conjunto de modelos estadísticos que se usan para caracterizar las propiedades estadísticas de una señal. Se utilizan dos procesos reconocidos como cadena de Márkov donde se utiliza la matriz de probabilidad de transición y se utiliza las funciones para determinar el conjunto en base a la probabilidad de cada estado.

2.6. Sistemas de detección y reconocimiento facial.

El sistema de detección y reconocimiento facial utiliza un CPU (Unidad central de procesamiento), el cual hace uso de un algoritmo de procesamiento de características faciales para lograr identificar de forma automática al individuo en una imagen digital. Por lo tanto, se trata de una tecnología que en conjunto a las cámaras de video adquieren, analizan e interpretan imágenes con el fin de distinguir rostros humanos.

2.6.1. Ventajas de los sistemas de reconocimiento facial.

Los sistemas fueron creados con la capacidad de ofrecer ventajas que son percibidas por la población como los métodos de seguridad, entre ellos se tiene:

- Menores incidentes delictivos como robo o suplantación de identidad.
- Mayor precisión al momento de realizar la identificación de rasgos faciales.
- Acción rápida
- Menos posibilidad de contacto físico entre usuario – dispositivo.

- Mayor comodidad e higiene
- Hace uso de un método menos intrusivo
- No hace uso de otros dispositivos externos para la identificación

2.7. Base de datos.

También son conocidas como banco de datos y se refieren a un “instrumento que se ocupa de la recolección de datos, los clasifica de acuerdo con parámetros y los relaciona con la finalidad de poder brindar un rápido acceso a la información almacenada mediante el uso de ordenadores” (Fandos 2003).

En la actualidad esta herramienta también es utilizada para elaborar análisis de información y comportamiento de los datos, cuyos propósitos se ven favorecidos a la amplia capacidad que tienen dichos sistemas para almacenar y clasificar de acuerdo a las especificaciones técnicas que se requieran. Por lo tanto, si una persona solicita visualizar eventos ocurridos en un lapso considerablemente amplio, solo debe obtener acceso a estos bancos de información para poder cumplir su objetivo.

Figura 7. Gestores de bases de datos.

Fuente: (Marín 2019)

2.7.1. Ventajas de las bases de datos.

El uso de las bases de datos generan algunas ventajas, que se muestran a continuación:

- **Búsqueda rápida de datos:** Permite una búsqueda rápida de los datos que se encuentran alojados en las memorias, estos datos pueden ser añadidos, modificados y eliminados de una forma rápida y eficaz.

- **Prevenir el duplicado de información:** Esto permite conocer si los datos alojados en la memoria respectiva o existe una duplicidad, este proceso se realiza de forma instantánea para evitar el uso de datos con información repetida.

- **Ingreso de datos ilimitados:** A diferencia de tiempos anteriores donde la información se almacenaba en papel y el límite físico para poder almacenar datos ilimitados era imposible, la base de datos virtual no tiene dichos inconvenientes y se pueden almacenar de forma ilimitada a esto se le conoce como BIG DATA.

- **Mantenimiento eficaz y rápido:** Teniendo una base de datos digital es muy eficaz y rápido la depuración de información, es decir efectuar la clasificación y eliminar que ya no es necesaria en los registros.

- **Protección de datos:** Tiempo atrás cuando los registros de datos se manejaban en papel el único respaldo que se podía hacer era una copia escrita a mano de dicha información o una fotocopia lo cual tomaba mucho tiempo, pero

actualmente gracias a la base de datos virtual estos backups de datos se pueden realizar de una forma rápida y segura

2.7.2. Clasificación general de las bases de datos.

Las bases de datos se encuentran clasificadas de 2 formas, la primera de ellas denominada base de datos estadística y la siguiente llamada base de datos dinámicas, las cuales se detallan a continuación:

- **Bases de datos estadísticas:** Este tipo de base cuenta con leyendas para leer; además son considerados de gran importancia para decisiones que se puedan tomar a futuro. Por lo general, carecen de dinamismo y esto mejora las perspectivas pues se puede tomar en consideración los hechos del pasado y tener en cuenta cómo mejorar el futuro.

- **Bases de datos dinámicas:** Estas tienen gran movimiento, se actualizan constantemente para que la empresa pueda mantener renovados los datos y los procesos de consulta puedan ser realizados de forma exitosa. Estas bases son útiles para las tiendas que trabajan con productos de constantes cambios. También existe una subclasificación, las cuales se describe en la figura (8)

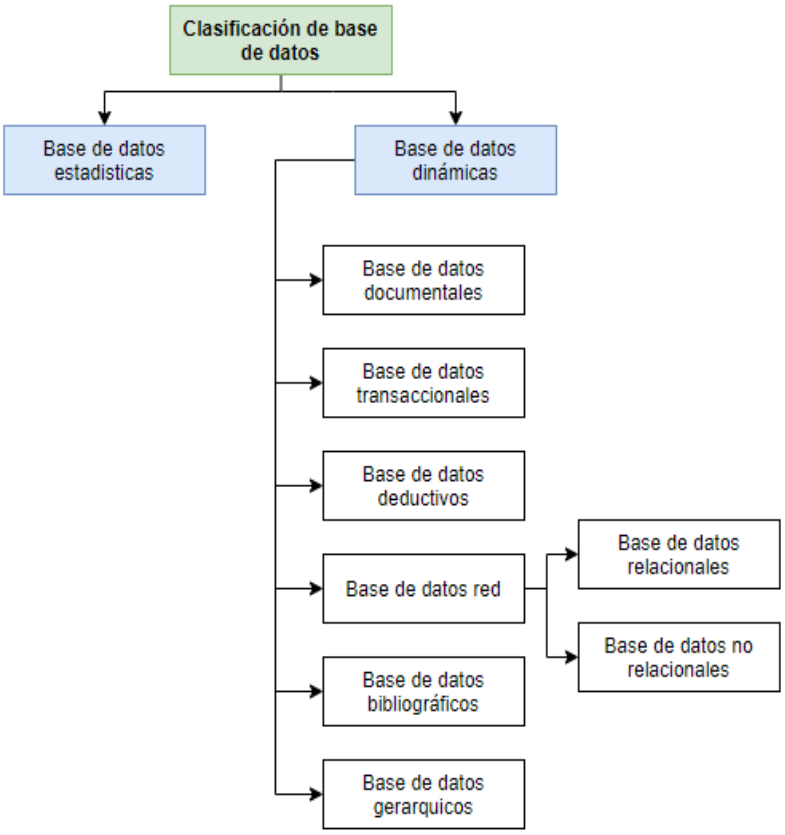


Figura 8. Clasificación de los tipos de base de datos según su contenido.

Fuente: autores.

2.7.3. Bases de datos no relacionales.

Las bases de datos NoSQL se relacionan con estructuras donde se almacenan diferente información sobre todo cuando las bases de datos tienen problemas como de escalabilidad y rendimiento; este tipo de situaciones suele ocurrir de forma diaria al introducir datos o realizar consultas. Un ejemplo es la base de datos conocida como MongoDb, esta base permite generar ventajas a los grandes proyectos.

2.7.4. MongoDB.

MongoDB corresponde a un sistema de banco de datos del tipo no relacional o NoSQL, por lo tanto, su uso se ve enfocado para esquemas de datos específicos con flexibilidad para el desarrollo de aplicaciones. Se encuentra escrito bajo el lenguaje C++ lo que proporciona un mejor rendimiento en ejecución y se encuentra licenciado como GNU AGPL 3, lo que lo convierte en un recurso open Source garantizando su funcionamiento en sistemas como Solaris, Linux, Windows. Etc.

Figura 9. Base de datos MongoDB.

Fuente: (New Horizons 2019)

2.7.4.1. Características de MongoDB.

Las características más relevantes de esta herramienta son detalladas a continuación.

- Alta Disponibilidad: Permite una alta disponibilidad de datos, garantizando una recuperación de los mismos de forma rápida e instantánea.

- Fragmentación: Permite el crecimiento de la base de datos otorgando la posibilidad de controlar los medios de almacenamiento

- Flexibilidad de esquema: Otorga la disponibilidad de efectuar cambios en los esquemas garantizando bases de datos más dinámicas.

- Alojamiento en la nube: Si todos los servicios están basados en la nube. MogoDB resulta más conveniente gracias a su completa escalabilidad.

2.8. Interfaz de programación de aplicaciones API.

API en sus siglas application programming interface es un conjunto de procedimientos o protocolos, que tienen como propósito comunicar el backend con el frontend,

2.8.1. Ventajas del uso de APIs.

- Independencia en la implementación de tecnologías y otros lenguajes
- Mejora del rendimiento a través de la minimización de uso de recursos.
- División entre cliente y servidor, ambos sistemas son independientes y su convergencia se realiza mediante la comunicación a través del lenguaje de intercambio JSON.
- Fomenta la construcción de sistemas escalables, fiables y flexibles.

2.8.2. API Rest.

Las interfaces de programación de aplicaciones respaldadas por el estilo arquitectónico del sistema arquitectónico REST se denominan API REST. Los servicios web, de bases de datos y los sistemas informáticos que cumplen con la API REST permiten que los sistemas solicitantes obtengan un acceso robusto y redefinan las representaciones de los recursos basados en la web mediante la implementación de un conjunto predefinido de protocolos sin estado y operaciones estándar (Richards, 2015).

2.9. Nginx Server.

Nginx en simples palabras es un servidor web, proxy inverso y balanceador de carga open source que hoy por hoy soporta a millones de páginas web en todo el mundo.

Nginx funciona de una forma no tradicional en comparación a los servidores web alternativos como Apache. Su arquitectura se enfoca más que nada de forma asincrónica, basándose enteramente en eventos, donde cada petición que viene desde un navegador web se maneja utilizando un solo thread o hilo. Esto hace que use muy pocos recursos, sobre todo en cuanto a uso de memoria, gestándose así una plataforma para servir peticiones web liviana y muy rápida (Borges 2019).

Figura 10. Logotipo de NGINX.

Fuente: (NGINX 2019)

2.9.1. Características Nginx Server.

Entre las principales características tenemos las siguientes:

- Balanceo de carga inteligente
- Servidor de archivos estáticos y dinámicos
- Ofrece alta disponibilidad.
- Escalabilidad asegurada.

2.9.2. Nuxt.js.

Nuxt.js es un framework que está basado en Vue.js y escrito en JavaScript. Es totalmente modular, de forma que podemos empezar con un paquete muy sencillo y, según los requerimientos los requerimientos del proyecto, se puede instalar las librerías o paquetes que sean necesario. Promueve la solución a las configuraciones, que eran más tediosas, con Vue.js, ya que hace que este proceso sea realmente fácil y muy sencillo. Es OpenSource y dispone de una gran comunidad, en la cual podemos consultar o buscar información sobre problemas que hayamos tenido dentro del código (Jiménez 2019).

Figura 11. Logotipo de Nuxt.js.

Fuente: (NuxtJS 2019)

2.10. Modulo ESP32 de Arduino.

Módulo que corresponde a la familia de la empresa Espressif Systems. Su diseño inicial fue pensado para el celular debido a que a pesar de su tamaño permitía la conexión a redes WiFi y otras conexiones como TCP/IP. Son prestaciones de bajo costo y contienen incorporaciones actuales como su utilización en Arduino o en Visual Studio Code (WAREX 2019).

Este módulo ha sido creado como un tipo de solución de Wi-Fi/Bluetooth; pero se integra como un todo ya que no solo ofrece una opción, sino que permite la conexión con varios periféricos como SPI, I^2C, I^2C, Ethernet, Tarjetas SD, entre otras interfaces táctiles y capacitivas. El procesador tiene dos núcleos de procesamiento que van desde los 80 megahercios (MHz) y 240 MHz en frecuencias operativas. Hay varios módulos, pero el más popular corresponde a el ESP32-WROOM-32D, que puede alcanzar la frecuencia más alta, este incluye una antena de rastreo de placa de CI, aunque tiene a disposición varias opciones de antenas (Benigno 2020).

Figura 12. Módulo ESP32-WROOM-32D.

Fuente: (Benigno 2020)

Para su utilización desde Arduino IDE, se verifica que:

- Se conecte con la PC utilizando del cable MicroUsb, que debe estar de color rojo
- Se inicia el IDE Arduino
    - En archivo, se busca las preferencias para ello, se puede utilizar el gestor de urls: https://dl.espressif.com/dl/package_esp32_index.json
- En la barra de Herramientas, se busca el Gestor de tarjetas y se establece los parámetros de configuración requeridos.
- En el Administrador de dispositivos se verifica los puertos y su conexión
- En Arduino se verifica la conexión con el puerto
- Se sube el programa Blink para programar el módulo donde se utiliza el código de inicialización.

CAPÍTULO III

3. MARCO METODOLÓGICO.

3.1. Diagrama funcional del sistema.

En este punto se detalla el funcionamiento del sistema mediante el uso de diagramas de flujo, para ello se divide el esquema en tres etapas que son:

- Etapa de registro de usuarios en la base de datos.
- Etapa de control de accesos mediante reconocimiento facial.
- Etapa de visualización de registros.

3.1.1. Etapa de registro de usuarios en la base de datos.

El ingreso de información al sistema es manejado por el ente administrador el cual solicitará los datos correspondientes de cada usuario (Nombres, apellidos, correo electrónico y 5 fotografías del rostro del individuo). El registro de datos se realiza mediante el acceso a la aplicación web con la IP: 34.67.143.38 y confirmar las credenciales de acceso.

Una vez realizado el acceso de forma exitosa, el administrador seleccionará la opción de usuarios, continuando con el botón agregar. Realizado estos pasos, se procede a agregar los datos del nuevo usuario y el ingreso de 5 fotografías del rostro del individuo.

Para el proceso final, se utiliza el botón codificar caras, con el que se puede actualizar las imágenes incluidas en la base de datos, mientras que el proceso para mantener su funcionalidad de mejor manera se utiliza fotografías con 5 ángulos para generar una mejor precisión en el sistema.

Figura 13. Diagrama funcional para el registro de usuarios.

Fuente: autores.

3.1.2. Etapa de control de accesos mediante reconocimiento facial.

Esta etapa da inicio una vez que la persona se acerca al prototipo y se realiza la detección de un rostro, el cual activara un temporizador de 3 segundos dándole al

usuario un tiempo prudencial para estabilizarse frente a la cámara antes de efectuar la capturar de la fotografía con la que trabajara el sistema de reconocimiento facial.

Una vez capturada la imagen del rostro, la información será procesada por el sistema de reconocimiento facial el cual emitirá dos posibles resultados, el resultado negativo en caso de que el rostro no sea detectado provocara la activación de la alerta hacia el ente administrador mediante un mensaje al correo electrónico. Por otro lado, si se genera una respuesta positiva, se procederá con el registro del usuario y apertura de la puerta o medio de acceso siempre y cuando el usuario no tenga un registro inferior o igual a un minuto, esto para evitar que el sistema realice marcaciones de forma involuntaria.

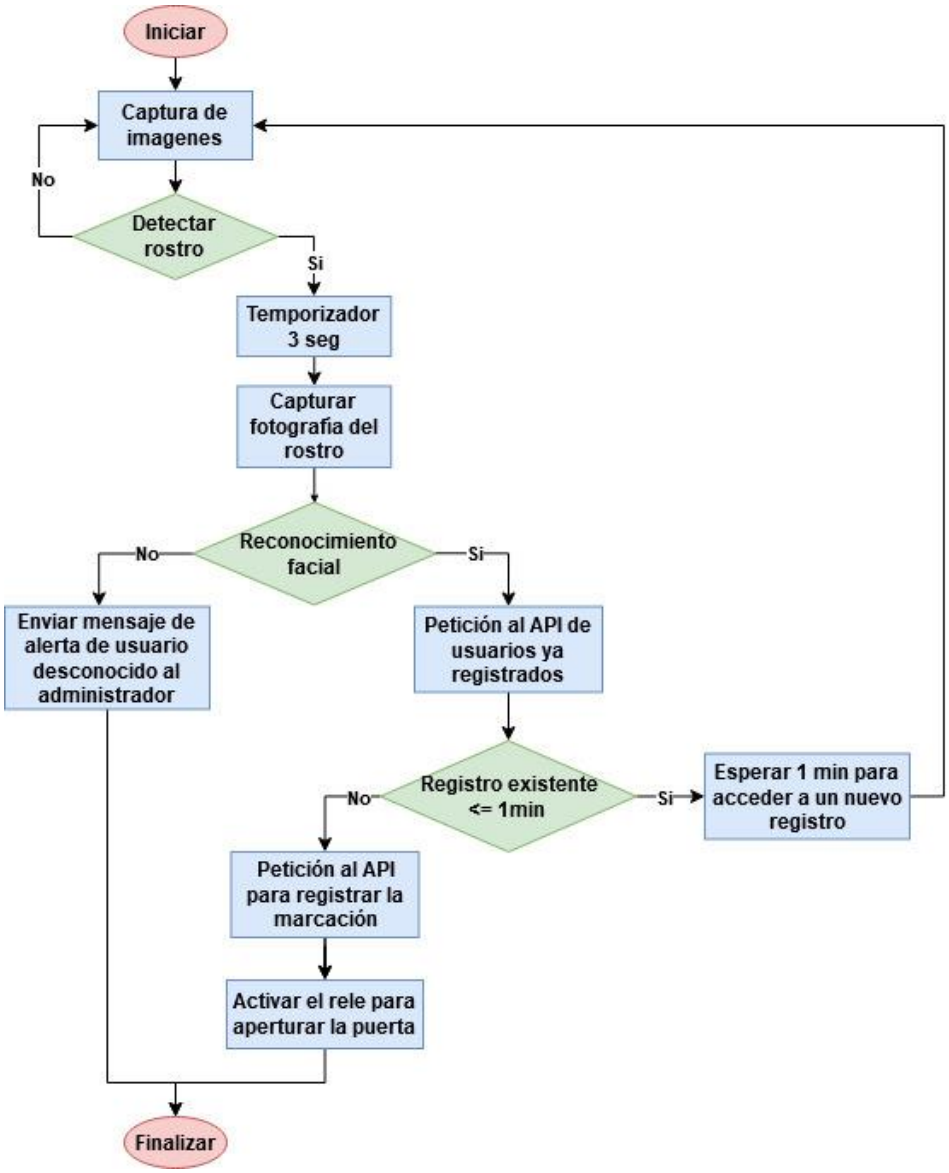


Figura 14. Diagrama funcional para el control de accesos por reconocimiento facial.

Fuente: autores.

3.1.3. Etapa de visualización de registros.

Para esta etapa se hace uso de las credenciales creada en el sistema y se utiliza la opción registros, el administrador tiene la función para poder ver a los usuarios registrados en el sistema.

Figura 15. Diagrama funcional para la visualización de registros.

Fuente: autores.

3.2. Diseño estructural del sistema de control de accesos.

A continuación, se describen los elementos y tecnologías utilizados para el diseño del sistema de verificación de accesos que se basa en el reconocimiento facial:

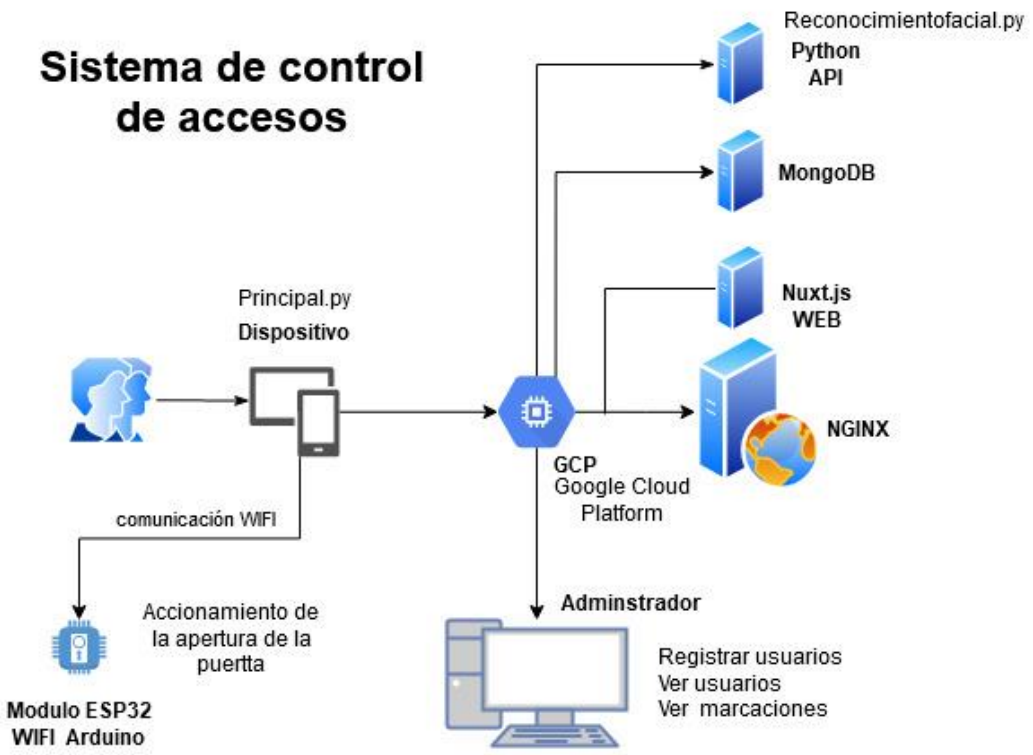


Figura 16. Diseño del sistema de control de acceso.

Fuente: autores.

- Dispositivo: se cuenta con una tarjeta LattePanda como controlador conectada con una cámara que realizara la entrada de datos, es decir, la toma de fotografías del usuario y las envía hacia la plataforma GCP en la cual serán procesadas.

- Plataforma GCP (Google Cloud Platform): se hace uso de la plataforma de Google, con almacenamiento en la nube, en la cual se tendrá alojados los servicios de la plataforma web, de la base de datos y del software de reconocimiento facial.

- Servidor NGINX: como servidor web/proxy encargado de direccionar las peticiones al resto de servicios que se encuentran alojados en la plataforma. Aquí llegara la fotografía enviada desde el dispositivo para luego ser enviada a la aplicación de reconocimiento facial para su respectivo procesamiento.

- Servicio web con Nuxt.js: el sitio web que se diseña para la visualización de registros hace uso del framework Vue.js para su desarrollo.

- Servicio API y aplicaciones basadas en Python: Este servicio corresponde a las aplicaciones de base de datos creada en MongoDB, al software de reconocimiento facial creado en Python y al API web encargado de la interacción de los servicios.

3.3. Construcción del prototipo para el control de acceso.

En este punto se detallan los elementos utilizados para la construcción del dispositivo, los cuales son definidos luego de realizar el planteamiento idóneo del proyecto.

3.3.1. Selección de componentes físicos.

LattePanda Versión 2G/32GB: El uso del Lattepanda como unidad central de procesamiento, se debe a sus principales características, entre ellas la capacidad de procesamiento que requiere el sistema para su optimo desempeño.

Tabla 1

Cuadro comparativo LattePanda con otras tecnologías

	Procesador	ROM	RAM	Interfaces	S.O Compatibles
Lattepanda 2G/32GB	1.8Ghz 64-Bit Quad-Core Intel Z8300	32 Gb Interno y expandible con Micro SD	2 Gb	1 USB 3.0, 2 USB 2.0, Wifi, Bluetooth 4.0, Ethernet, HDMI	Windows 10 / Linux
Raspberry PI 3	1.2 Ghz 64Bit Quad-Core ARMv8	Mediante Micro SD	1 Gb	4 USB 2.0, HDMI, Wifi, Bluetooth 4.1, Ethernet	Windows IoT / Linux
Odroid C2	1.5 Ghz Quad-Core ARM Cortex A53	Mediante Micro SD	2 Gb	4 USB 2.0, 1 USB OTG, HDMI, Ethernet	Linux /Android
Jaguar One	1.3 Ghz Quad-Core Intel Atom Z3735G/F	16 Gb Interno	1 Gb	3 USB 2.0, Ethernet, HDMI	Linux /Android / Windows 10
Pine A-64	1.2 Ghz 64Bit Quad-Core ARMv8	Mediante Micro SD	2 Gb	2 USB 2.0, HDMI, Ethernet	Windows IoT / Linux / Android

Fuente: autores.

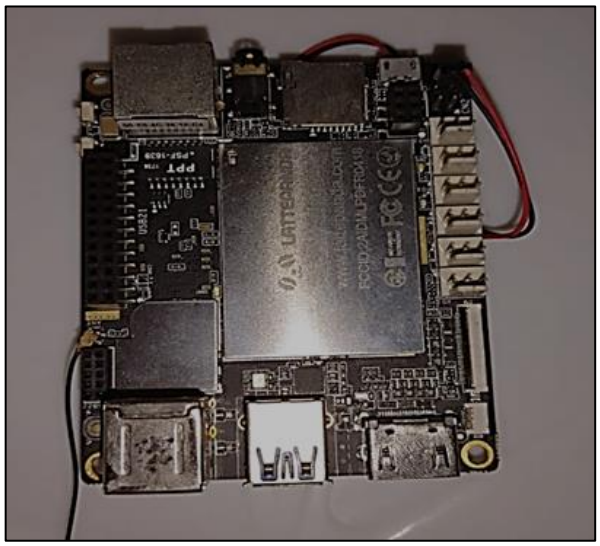


Figura 17. LattePanda 2G/32GB.

Fuente: autores.

Display 1024x600 IPS para Lattepanda: esto dispositivo se utiliza para mostrar la parte gráfica a los usuarios que han accedido al sistema

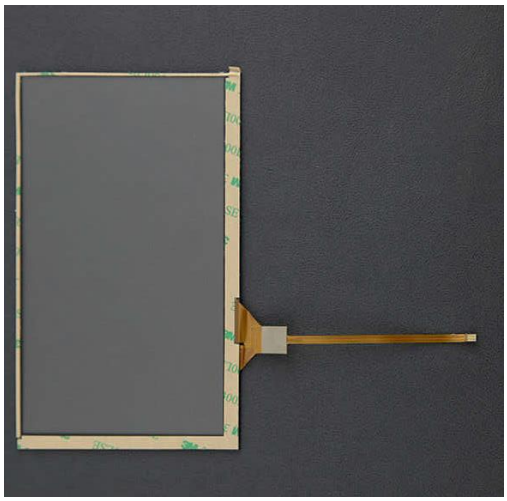


Figura 18. Display 1024x600 para LattePanda.

Fuente: autores.

Web Cam USB: El uso de la cámara web como dispositivo de entrada para la captura de imágenes debe ser de una resolución igual o menor a los 3 Megapíxeles debido al peso que manejan las fotografías y al procesado que se efectúan con ellas.

Figura 19. Cámara web.

Fuente: autores.

Adaptador de corriente 5V a 2A: Se hace uso del adaptador para la correcta funcionabilidad del equipo según las especificaciones técnicas del fabricante.

Figura 20. Adaptador de corriente 5V-2A.

Fuente: autores.

3.3.2. Diseño de case.

EL diseño del case se toma en cuenta para verificar que elementos pueden ir en la parte interior y cuales en la parte externa. Y para ello se ha dividido su diseño en 3 secciones:

- **Parte Trasera:** diseñada con una cavidad para enganche de pared, un orificio para la salida del aire de ventilación del LattePanda y una abertura para la entrada/salida de cables del dispositivo o conexiones a los pines GPIO

- **Parte delantera:** diseñada a con una estructura de soporte para los elementos como el display, la cámara y el LattePanda, en esta parte consta los respectivos soportes

- **Cubierta Frontal:** diseñada a manera de tapa en la cual consta de los respectivos espacios de salida tanto para la cámara como para el display.

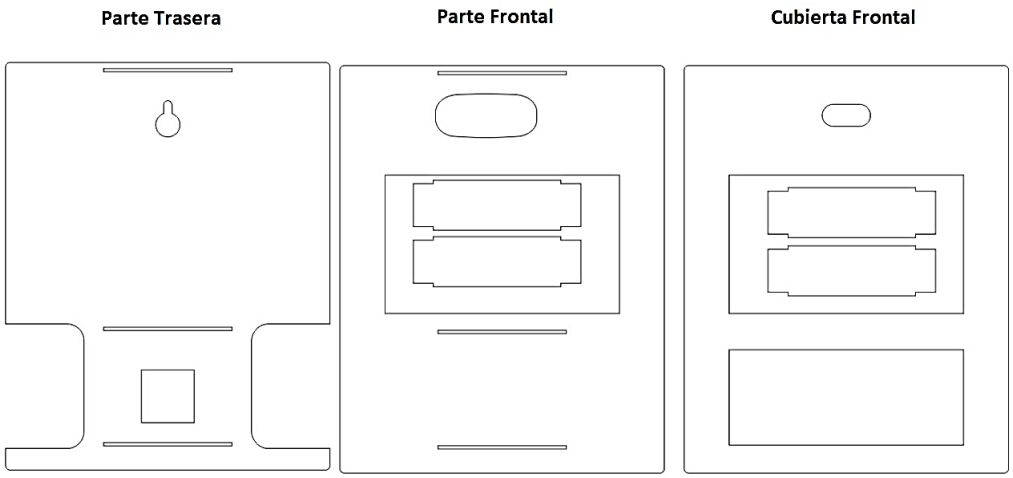


Figura 21. Diseño del case del prototipo.

Fuente: autores.

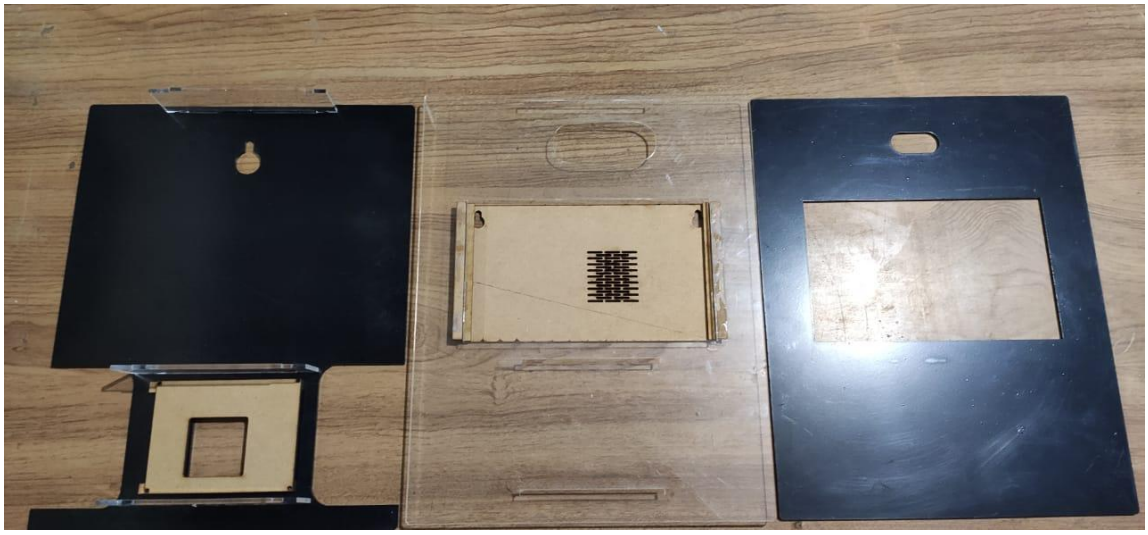


Figura 22. Construcción de estructura en acrílico.

Fuente: autores.

3.3.3. Conexiones físicas del dispositivo LattePanda.

El dispositivo tiene como controlador un Lattepanda representado en la parte central de la Figura 23, hará uso de una conexión a internet para acceder a la plataforma GCP, dicha conexión puede ser este mediante red LAN o Wifi, por otro lado hará uso de un display conectado mediante Flex en la cual se visualizara la imagen captada por la cámara y por último se conectada mediante uso del USB 3.0 para conectarse con la cámara de 2Mpxl el cual se encargara de la entrada de la información hacia el equipo.

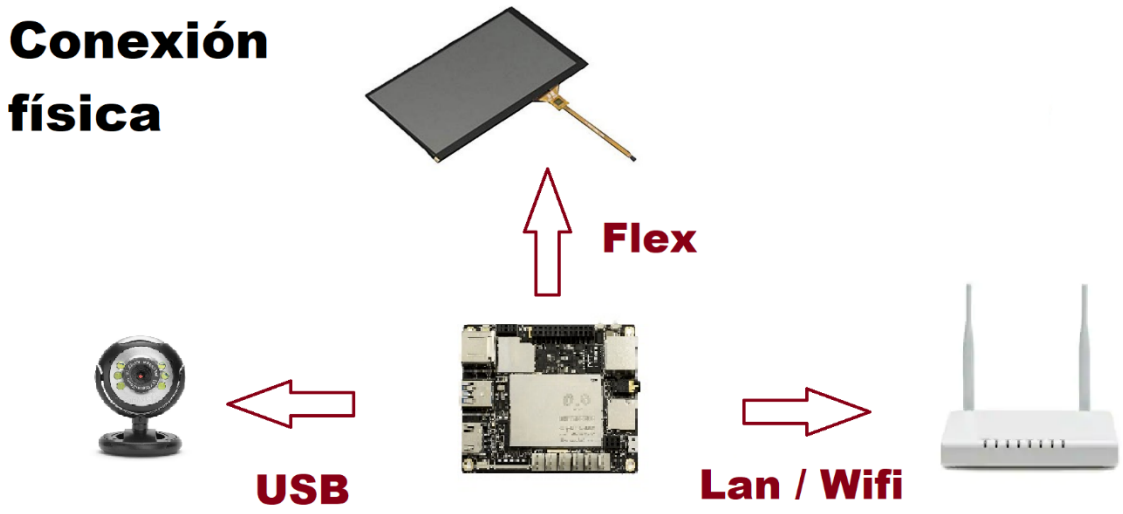


Figura 23. Conexiones físicas del dispositivo LattePanda.

Fuente: autores.

3.4. Plataforma virtual GCP.

El uso de GCP (Google Cloud Platform) como opción más conveniente para el desarrollo del proyecto, se debe a sus amplios beneficios donde se otorga una adaptable infraestructura multiplataforma con estándares de seguridad de alto nivel y alojamiento en la nube.

Una cuenta de correo electrónico de Gmail permite acceder a este servicio, para luego acceder al sitio web de Google Cloud, donde se procederá a hacer la adquisición

de este servicio de forma fácil y segura. Cabe indicar que para el buen desempeño del proyecto que se está realizando, se ha seleccionado el servicio con una capacidad de almacenamiento de 20GB en la nube y 4GB de memoria RAM.

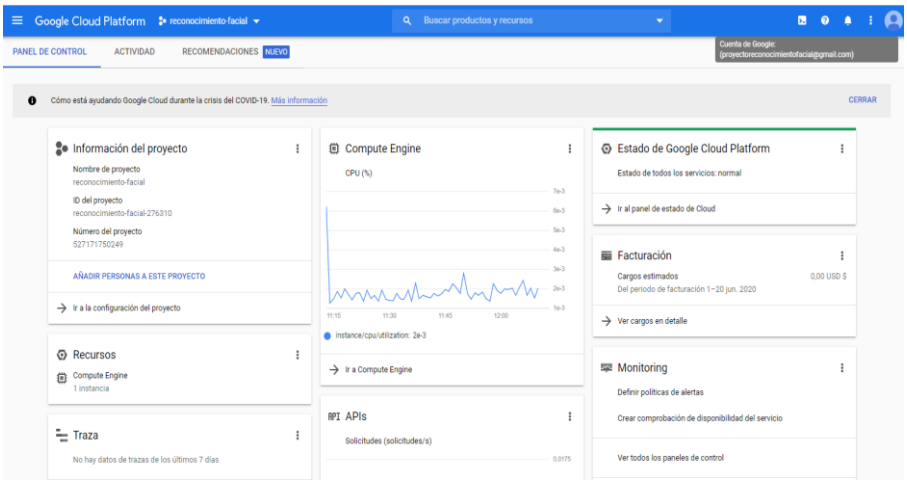


Figura 24. creación de plataforma GCP.

Fuente: autores.

3.5. Base de datos en Mongodb.

La Base de Mongodb es generada en tres secciones denominadas colecciones con el fin de poder ubicar la información para cada usuario y generar las marcaciones como se observa en la figura 25.

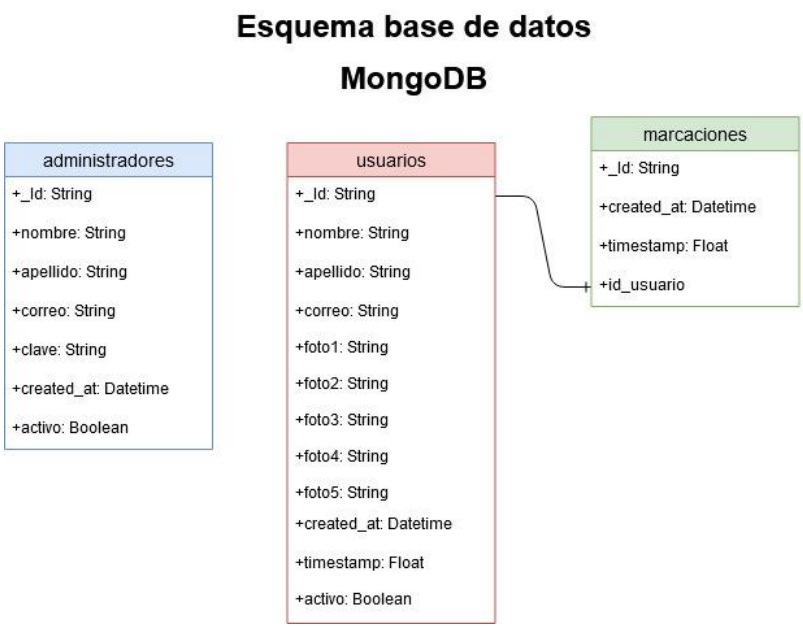


Figura 25. Esquematización de colecciones de la base de datos.

Fuente: autores.

3.5.1. Instalación de Mongodb en Ubuntu.

Para llevar a cabo la instalación de la base de datos dentro del sistema operativo Ubuntu lo recomendable en primer lugar es efectuar la actualización de repositorios y creación de una lista de origen, para este procedimiento se utiliza los comandos detallados en la figura 26.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 68818C72E52529D4

sudo echo "deb http://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list

sudo apt-get update
```

Figura 26. Líneas de comando para la actualización de repositorios.

Fuente: autores

Una vez finalizado el apartado anterior, se puede dar comienzo al proceso de instalación de Mongodb haciendo uso de la instrucción “apt-get install -y mongodb-org” en modo administrador, una vez culminada la parte de la instalación se creará de manera automática el archivo “mongod.service”, por lo cual no será necesario crear dicho archivo de forma manual. En este punto ya se puede dar inicio a mongos teniendo en cuenta que el puerto a utilizar es el 27017, el cual se puede constatar con el uso del terminal digitando la siguiente línea “sudo netstat -plntu”.

Figura 27. Líneas de comando para la actualización de registros.

Fuente: autores.

3.5.2. Configuración inicial de Mongodb.

Durante esta fase se pone en marcha Mongodb usando la línea “sudo service mongod start” y se ejecuta una serie de configuraciones básicas que se describen a continuación:

Para incrementar la seguridad en primer lugar se debe establecer un usuario con privilegios de administrador, para ello se debe hacer uso de la consola de Mongodb y ejecutar los comandos que se describen en la figura 28.

```
# mongo
> use admin
> db.createUser({user: “admin”, pwd: “admin123”, roles: [{ role: “root”, db: “admin” }] })
Successfully added user: {
      "user" : "admin",
      "roles" : [
            {
                  "role" : "root"
                  "db" : "admin"
            }
      ]
}
```

Figura 28. creación de usuario con privilegios de administrador en Mogodb.

Fuente: autores.

Luego de ejecutar los comandos anteriores se procede a la habilitación de la autenticación con las credenciales de administración creadas y por consiguiente efectuar la salida de mongo mediante la sintaxis “exit”, retornar a la línea de comandos y digitar las siguientes líneas “sudo nano /lib/systemd/system/mongod.service” y “ExecStart=/usr/bin/mongod --auth --config /etc/mongod.conf” las cuales cumplen la función de agregar la nueva opción de autenticación de acceso. Para comprobar el funcionamiento se debe ejecutar nuevamente mongos y validar que el registro sea correcto.

Una vez culminado la parte descrita anteriormente, se continúa con la habilitación de accesos y configuración del cortafuegos (Firewall de Ubuntu), con el propósito de permitir el ingreso de forma externa a Mongodb. Para eso primero se comprueba el estado del firewall mediante la línea “sudo ufw status”, por defecto estará inhabilitado y por ello, se ejecuta “sudo ufw allow ssh”, “sudo ufw enable” para logar habilitar el UFW y apertura el puerto SSH.

Para completar la configuración y permitir que una IP externa acceda a Mongodb, se hace uso del siguiente código “sudo ufw allow from (Dirección IP) to (Destino) port

(Número de puerto)", denotando así la sintaxis que permite a una IP externa acceder a la base de datos. Por último, cabe mencionar que para efectuar pruebas a nivel local se debe abrir el puerto 27017 con el propósito de que todos los sistemas de la red logren ingresar a Mongos, y para ello usamos la instrucción "sudo ufw allow 27017", para permitir la accesibilidad desde afuera, en caso de requerirla, se reconfigurará nuevamente en Mongodb el número de puerto y la dirección IP de su servidor.

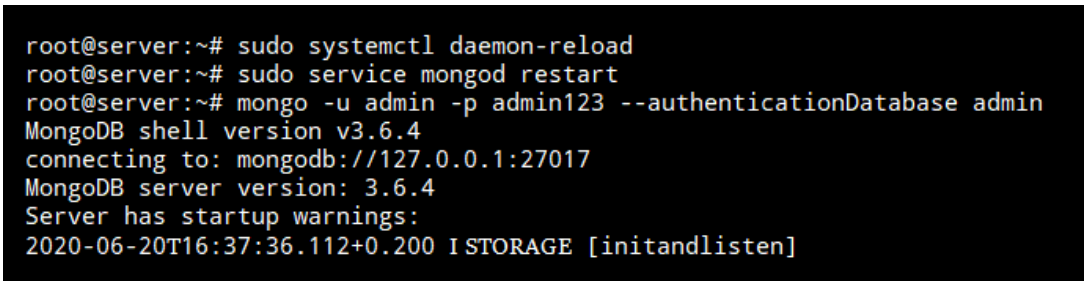


Figura 29. Figura Inicialización de servidor MongoDB.

Fuente: autores.

3.5.3. Creación de la base de datos y colecciones en MongoDB.

Aquí se detalla cómo se creó la base de datos con el fin de poder utilizar el reconocimiento facial y facilitar la utilización de MongoDB mediante la instrucción "use". De similar forma para generar colecciones se hace uso de la sintaxis "db.nombredelacolección.save ({Parámetro: 'dato'})".

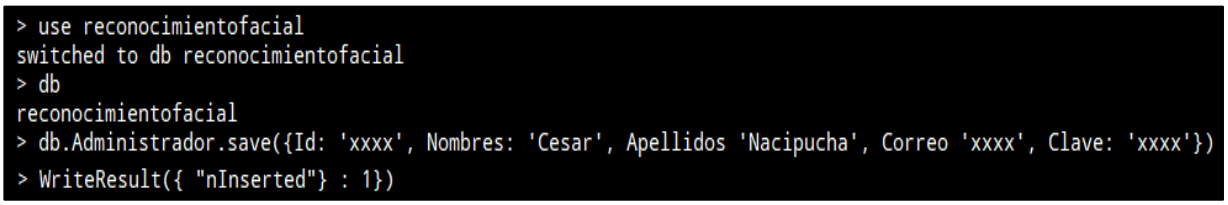


Figura 30. creación de colecciones mediante uso de comandos.

Fuente: autores.

Cuando se ha configurado la base de datos se realiza el proceso para utilizar el reconocimiento facial y los datos se guardarán en colecciones para su posterior visualización, a continuación, se muestran las 3 colecciones definidas:

- **Administradores:** En ella se lleva el registro de las personas que tendrán acceso a la información como administrador. Se muestra el uso de los datos:

Tabla 2

Datos de ingreso de la colección 'Administrados'

Dato a solicitar	Tipo de dato a ingresar
Número de identificación	String
Nombres	String
Apellidos	String
Correo Electrónico	String
Contraseña	String

Fuente: autores.

- **Usuarios:** En esta colección se detallan todos los usuarios o personas que harán uso del sistema de acceso, para cada usuario se registrarán los siguientes datos:

Tabla 3

Datos de ingreso de la colección 'Usuarios'

Dato a solicitar	Tipo de dato a ingresar
Número de identificación	String
Nombres	String
Apellidos	String
Foto del rostro # 1	String
Foto del rostro # 2	String
Foto del rostro # 3	String
Foto del rostro # 4	String

Foto del rostro # 5	String
Fechas de creación	Datetime
Registro de tiempo	Float
Contraseña	String

Fuente: autores.

- **Marcaciones:** En esta sección se reconocerá el registro de los usuarios mediante el reconocimiento facial y para realizar el proceso de consulta se utilizarán los datos explicados en la Tabla 4.

Tabla 4.

Datos de ingreso de la colección 'Consultas'

Dato a solicitar	Tipo de dato a ingresar
Número de identificación	String
Fechas de creación	Datetime
Registro de tiempo	Float
Usuario_Id	String

Fuente: autores.

3.5.4. MongoDB Compass.

Se utiliza el MongoDB Compass, para poder solicitar los accesos que permitan los diferentes procesos en el sistema. Los datos utilizados son los siguientes:

- Hostname: 34.67.143.38
- Puerto: 27017
- Tipo de autenticación: Usuername / Password
- Usuario: admin
- Contraseña: admin123
- Autenticación de la base de datos: admin

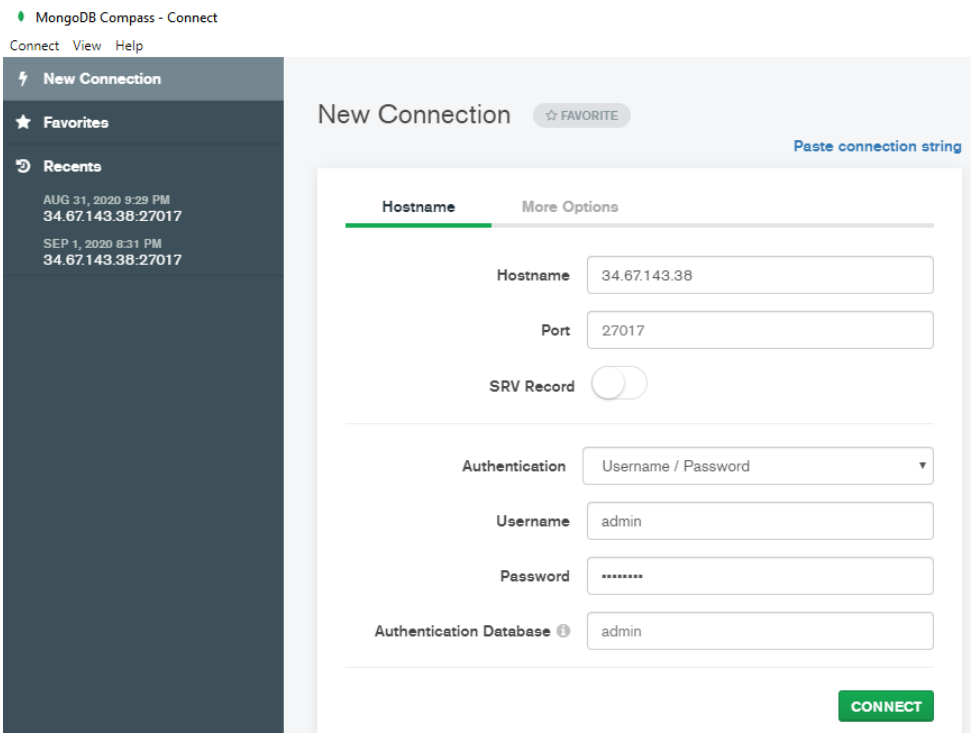


Figura 31. Interfaz de acceso de MongoDB Compass

Fuente: autores.

Una vez dentro de la base de datos, es posible visualizar el contenido, realizar modificaciones y crear las colecciones de forma sencilla y eficiente.

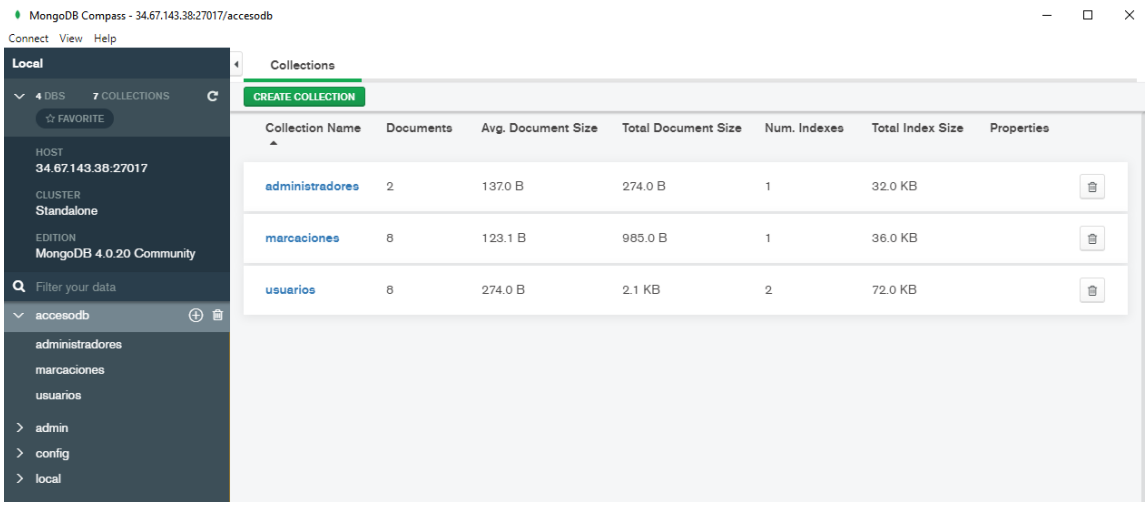


Figura 32. Colecciones creadas para el sistema de reconocimiento facial

Fuente: autores.

3.6. Interface API.

La interfaz de programación de aplicativos denominado API, se desarrolla bajo el lenguaje de programación Python haciendo uso de FaskAPI como web framework, encargándose de la comunicación entre 2 aplicaciones (Base de datos y aplicativo web) y para ello se emplean los métodos get, post, put y delete que trabajan con datos en formato JSON.

El procesamiento de solicitudes se maneja a través de la herramienta FaskAPI, la cual trabaja en conjunto con las cabeceras HTTP para controlar el mecanismo referente la entrega de paquetes de datos provenientes de la red mediante el uso de protocolo TCP/UDP, por ello se denomina al API como el motor de todo el back-end encargado de solventar las peticiones requeridas hacia el sistema. El código del programa realizado para la interfaz API, podrá ser encontrado en el apartado de anexos del presente trabajo de titulación.

Figura 33. Esquema de comunicación mediante el uso de API.

Fuente: autores.

3.6.1. Comunicación del API con la base de datos.

Mediante el uso de dirección: http://34.67.143.38:8000/docs#/ se logra acceder a la documentación del API, donde se podrá visualizar detalladamente los métodos utilizados en cada función.

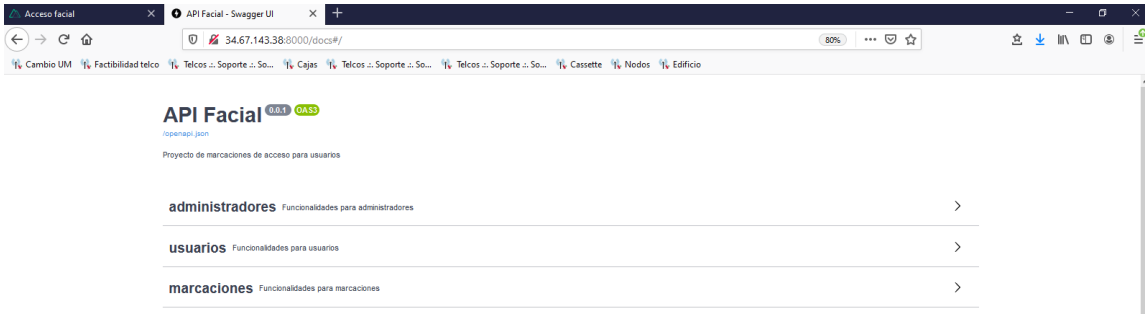


Figura 34 Estructura del API conforme a la base de datos.

Fuente: autores.

De acuerdo a lo mencionado en el apartado 3.5, se detallan 3 roles importantes en la estructura de la base de datos (clientes, administradores y marcaciones). Sobre dichos aspectos está basado la comunicación del servicio API, la cual se detallan a continuación:

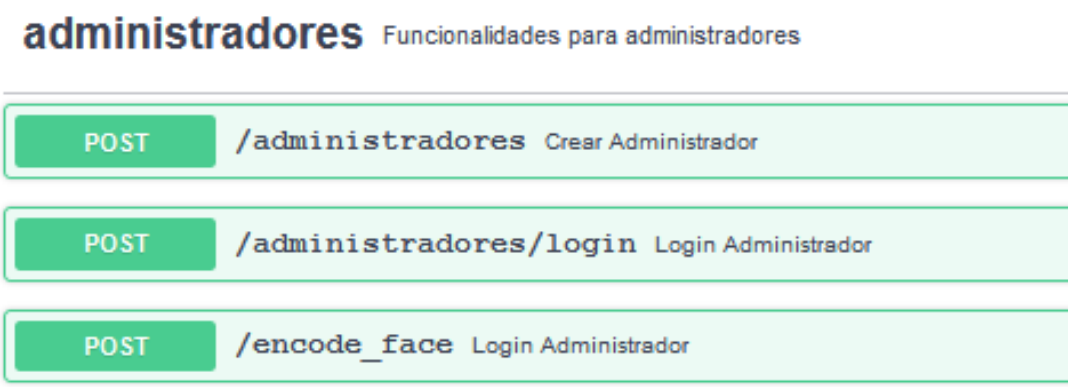


Figura 35. Métodos utilizados en las funciones de ‘administrador’.

Fuente: autores.

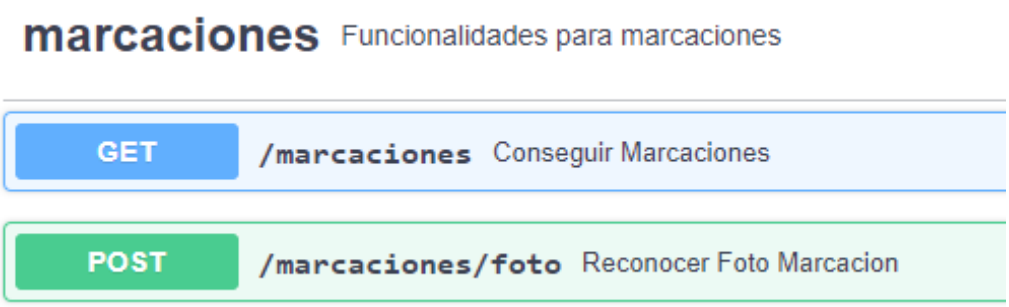


Figura 36. Métodos utilizados en las funciones de ‘marcaciones’.

Fuente: autores.

Figura 37. Métodos utilizados en las funciones de 'usuarios'.

Fuente: autores.

3.7. Aplicativo web para administrador (Front-End).

El aplicativo web se desarrolla mediante Nuxt,js en vista a que su desempeño abarca las expectativas del proyecto siendo de código abierto y sobre todo por su uso de fácil aprendizaje. El código del programa realizado para el aplicativo podrá ser encontrado en el apartado de anexos del presente trabajo de titulación.

3.7.1. Interfaces desarrolladas del aplicativo web.

Mediante el uso de la dirección ip 34.67.143.38 en un navegador podremos acceder al aplicativo web del administrador.

En el aplicativo web el ente administrador podrá visualizar, crear, modificar y eliminar usuarios. También podrá tener acceso a todos los registros de las marcaciones que se

hayan generado sin embargo por motivos de seguridad no tendrá privilegios de eliminar ni modificar marcaciones, solo cuenta con permisos de lectura.

Las interfaces que se encuentran en el aplicativo son las siguientes:

- Interfaz para inicio del administrador.
- Interfaz para la visualización de registros.
- Interfaz para creación / edición de usuario.
- Interfaz para la visualización de usuarios registrados.

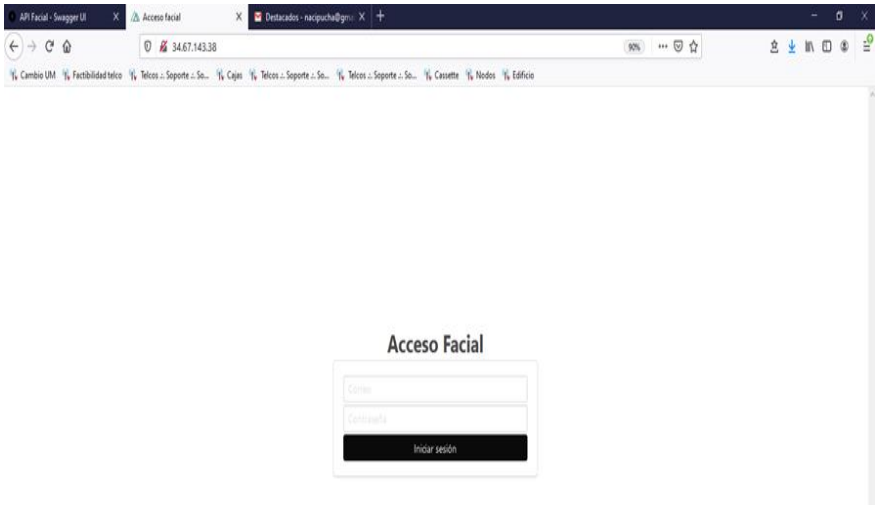


Figura 38. Interfaz para inicio de sesión del administrador.

Fuente: autores.

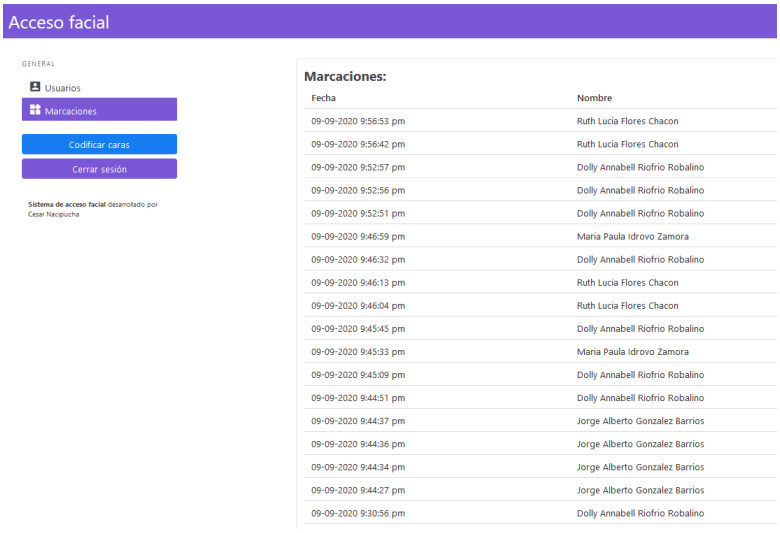


Figura 39. Interfaz para visualización de registros.

Fuente: autores.

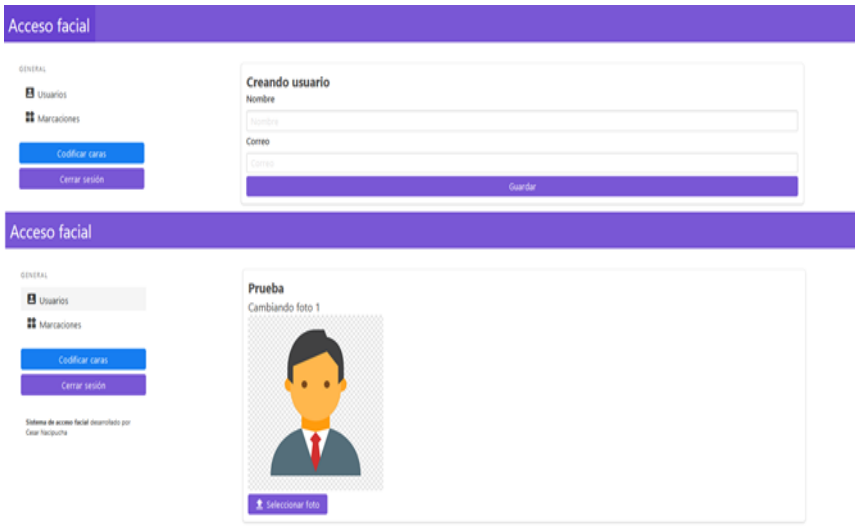


Figura 40. Interfaz de creación y edición de usuarios.

Fuente: autores.

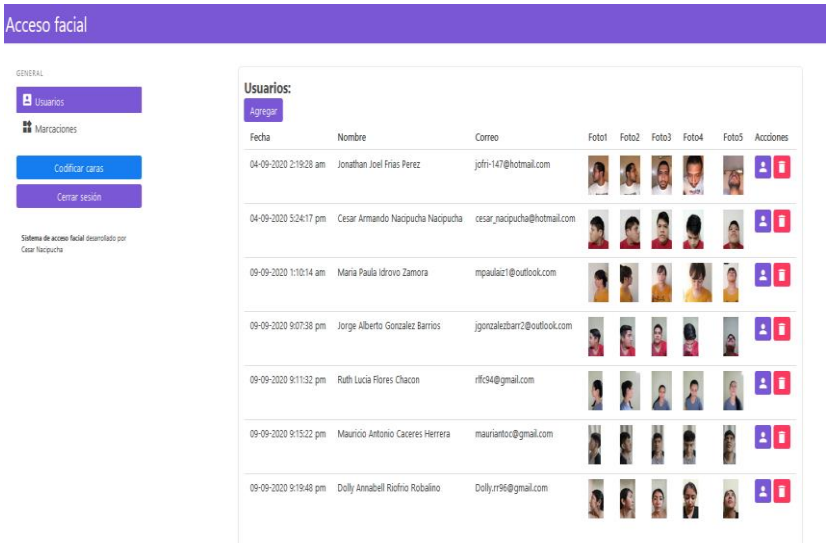


Figura 41. Interfaz de visualización de usuarios registrados.

Fuente: autores.

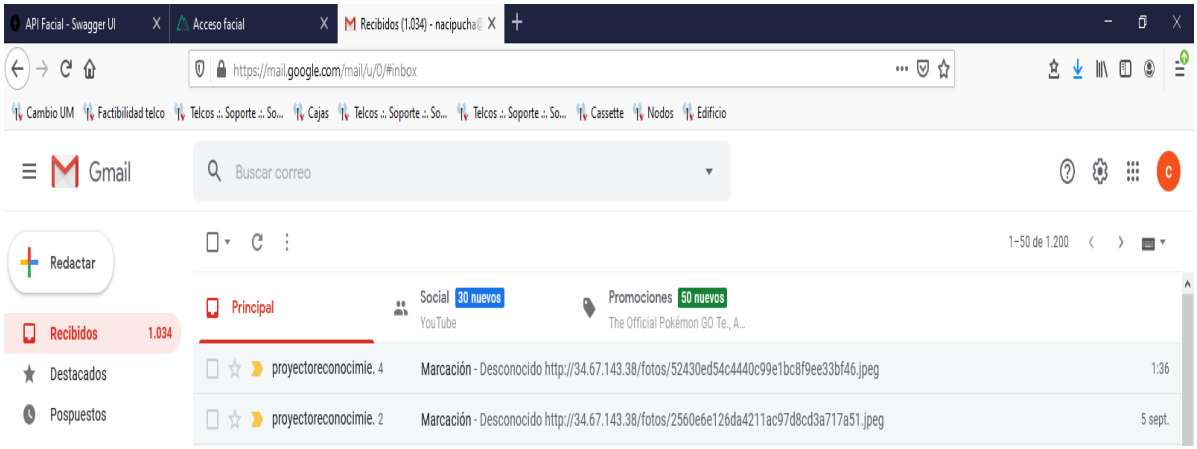


Figura 42. Alerta vía mail de usuario desconocido.

Fuente: autores.

3.8. Desarrollo de software para el reconocimiento facial.

Para la creación del software de reconocimiento facial se hace uso del lenguaje de programación Python por ser el lenguaje de código abierto con una mayor factibilidad al momento de implementar este tipo de aplicaciones, dicho programa se desarrolla en 3 etapas conformadas por una serie de procedimientos que tienen como propósito realizar el procesamiento de imágenes y obtener resultados de acuerdo con los métodos usados. En la figura 43 se puede visualizar el esquema de trabajo del software.

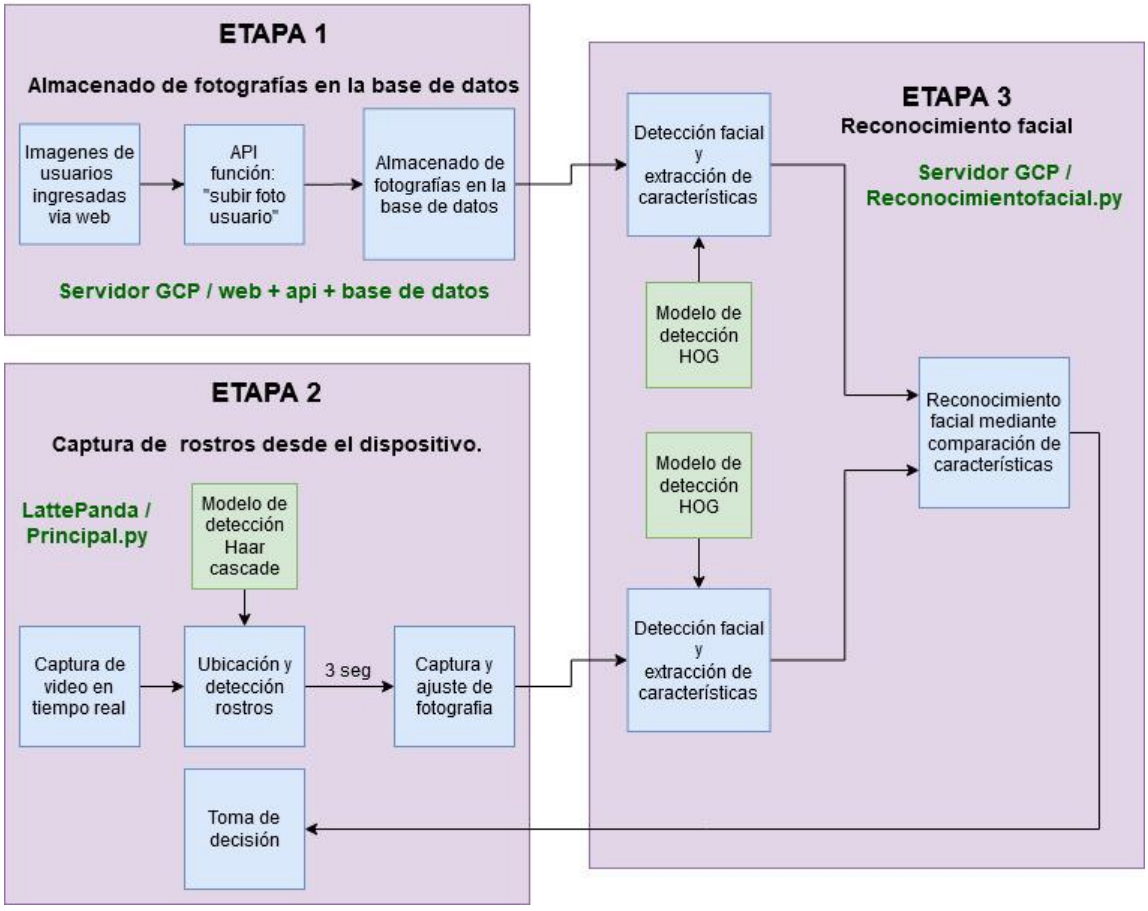


Figura 43. Esquema funcional del software para reconocimiento facial.

Fuente: autores.

3.8.1. Etapa 1: Almacenado de fotografías en la base de datos.

- **Imágenes de usuarios ingresadas vía web:** Hace referencia a las imágenes cargadas en los registros de cada usuario, mediante la plataforma web por parte del administrador.

Figura 44 Fotografías de usuarios ingresadas vía web.

Fuente: autores.

- **API función "Subir foto usuario":** La función a la que se hace mención tiene como propósito comunicar la plataforma web con la base de datos con el objetivo de poder almacenar todas las fotografías cargadas por parte del ente administrador.

Figura 45. Función Subir Foto Usuario.

Fuente: autores.

- **Almacenado de fotografías en la base de datos:** Las 5 fotografías de cada usuario son almacenadas en forma de arreglos dentro de la colección Usuarios del banco de datos para luego ser procesadas en la etapa 3 como se describirá en los siguientes apartados.

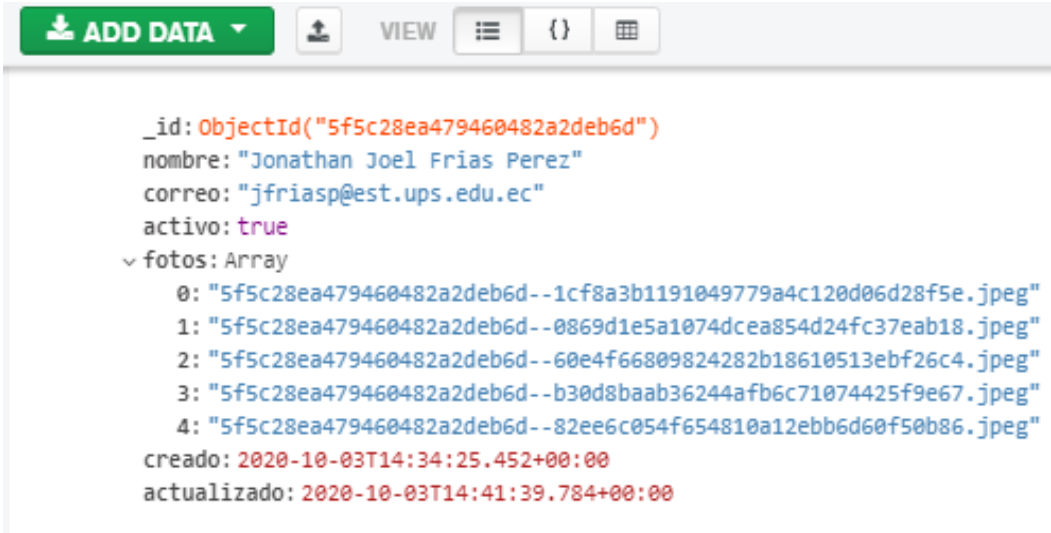


Figura 46 Almacenamiento de fotografías de usuarios en la base de datos.

Fuente: autores.

3.8.2. Etapa 2: Captura de rostros desde el dispositivo.

- **Captura de video en tiempo real:** empelando la función “VideoStream” se efectúa la captura continua de imágenes a través de la cámara incorporada en el prototipo. El frame capturado pasa por una conversión a escala de grises para un mejor procesamiento de manera interna.

- **Ubicación y detección rostros:** en este apartado se hace énfasis a 2 procesos que se ejecutan al mismo instante como lo son la ubicación centrada del rostro y la detección del rostro, que tienen como finalidad preparar al usuario para la captura fotográfica de la cara.

    - **Ubicación centrada del rostro:** Se cuenta con un sistema de procesamiento de imágenes con el objetivo de persuadir al usuario a que centre el rostro en el lente de la cámara y consiste en la

extracción de coordenadas de la ubicación del rostro (x, y, w, h) utilizados para el cálculo de centros, luego se realiza una condicional comparando los puntos centros obtenidos vs la zona céntrica de las dimensiones de imagen de pantalla dadas en pixeles con un rango de desfase de 30pixeles. Con este procedimiento se logra que el sistema pueda evaluar un solo rostro por captura.

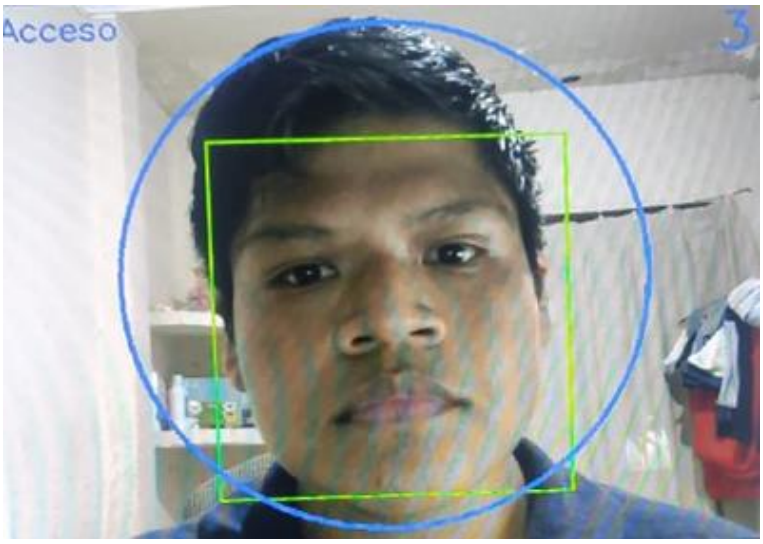


Figura 47. Detección y ubicación centrada del rostro.

Fuente: autores.

- **Detección del rostro:** la detección de rostros se realiza mediante el uso del modelo de detección “haar cascade” para rostros el cual se encuentra descrito en el siguiente apartado.

- **Modelo de detección “Haar cascade”:** Mediante este modelo se permite la combinación de clasificadores que sigue la estructura de una cascada, de igual manera la dificultad al momento de clasificar

incrementa conforme se avanza en cada etapa. Mediante la utilización de estos clasificadores se incrementa la posibilidad de encontrar un determinado objeto.

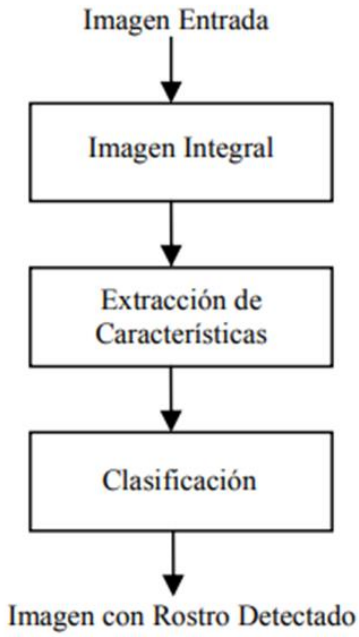


Figura 48. Algoritmo de Haar cascade.

Fuente: (Pardo 2017)

En la Figura 48., se muestra el proceso en tres etapas con el algoritmo cascade donde se expone una imagen de entrada como primer paso y esta se utiliza para la detección. De ahí continua con el proceso de inicialización de estructuras para este proceso se utiliza algún periférico o dispositivo como una cámara o escáner; y estos varían de acuerdo con la escena y el objetivo deseado. La Imagen integral extrae las características en diferentes escalas, cuando ha realizado todos los cálculos de los píxeles genera un valor medio. El proceso de extracción inicia con muchas imágenes para poder entrenar al clasificador. Y el

proceso de clasificación se introdujo el proceso de boosting a través de Adaboost, este permite asegurar un proceso rápido y excluyente debido a que solo se centra en las características críticas (Pardo 2017)

- **Captura y ajuste de fotografía:** Una vez el sistema haya detectado la existencia de un rostro y se encuentra centrado conforme a los parámetros establecidos, se emplea el uso de un Timer programado a 3 segundos con el fin de que el usuario se estabilice antes de que el sistema haga la toma de fotografía. Una vez tomada la foto se ejecuta un recorte ajustado de la cara del individuo empleando los puntos céntricos descritos en los apartados anteriores, y el software envía la captura recortada del rostro servidor hacia el servidor GCP que se encargara de efectuar el reconocimiento facial con el resto de las imágenes de usuarios ya registrados en la base de datos.

Figura 48. Imagen ajustada enviada al servidor GCP.

Fuente: autores.

- **Toma de decisión:** Este proceso se lleva a cabo una vez que se obtenga la respuesta por parte del programa desarrollado Reconocimientofacial.py que se encuentra alojado en el servidor GCP. La respuesta que llegara en forma de estados, es decir, se obtendrá un estado 200 en caso de que el reconocimiento sea exitoso y 400 en el caso de que el sistema no logre reconocer al individuo.

  Mediante el uso de condicionales, el sistema determina una decisión, si se obtiene un estado 200 el sistema enviará una petición vía wifi hacia el microcontrolador esp32 para activar el relé de la cerradura eléctrica y en el caso de obtener un estado 400 se enviará el mensaje de alerta vía email al administrador exponiendo la captura de imagen del individuo que intento acceder sin estar registrado.

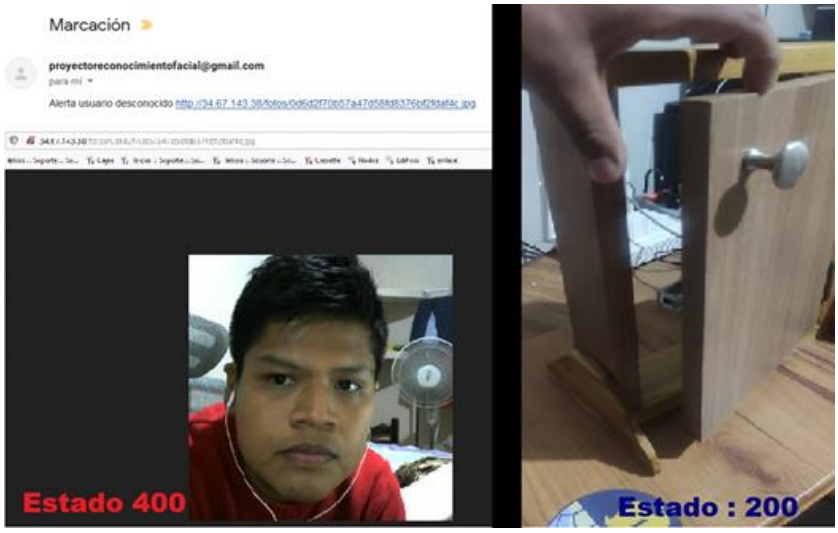


Figura 49. Decisión en estado 200 y estado 400.

Fuente: autores.

3.8.3. Etapa 3: Reconocimiento facial.

- **Detección facial y extracción de características:** El programa reconocimientofacial.py tiene como entrada la imagen que proviene del dispositivo y las imágenes almacenadas en la base de datos de todos los usuarios. Aunque las imágenes provengan de 2 lugares diferentes se aplicará el mismo método de detención facial y extracción de características HOG (Histograma de gradientes orientados) con el propósito de almacenar dichas características para posteriormente evaluarlas y obtener resultados.

- **Modelo de detección "HOG":**

  En cambio, este tipo de método es reconocido como Histogramas de Gradientes Orientados (HOG), este verifica que si la orientación se realiza con el fin de determinar áreas locales. Dentro de las características que ofrece se tiene:

  - Utiliza la información del gradiente, principalmente el contorno de los objetos.
  - Se puede agregar información, luego se puede recuperar y volver a agregar donde sea requerido (Martín 2016)

  Durante el proceso para generar los resultados correspondientes por HOG se utiliza un esquema piramidal donde se utiliza un solo vector HOG de una imagen, se la divide en varias subimágenes, y en cada una se extrae el vector correspondiente, este es individual, se repite el proceso hasta obtener un descriptor concatenado con todos los vectores

individuales como si fuera solo uno; este proceso se puede visualizar en la Figura 51.

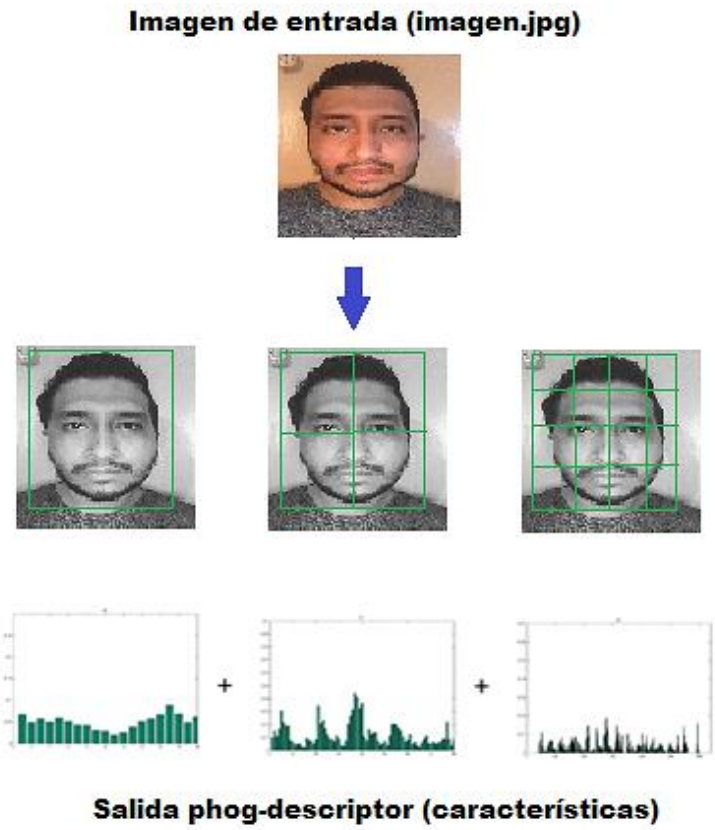


Figura 50. Proceso de HOG.

Fuente: Autores.

- **Reconocimiento facial mediante comparación de características:** Una vez aplicado el modelo de detección y extracción de puntos, las características correspondientes a las caras en las fotos de la base de datos estarán almacenadas en forma de arreglos dentro de la carpeta encodings. Y mediante el uso de condicionales se procede a evaluar las características encontradas en la foto del individuo con las ya alojadas dentro de la carpeta encondings. De esta manera dependiendo de la existencia de similitud se obtendrá 2 clases de respuestas, el estado 200

en caso de existir un rostro similar y de lo contrario un estado de error 400, esta respuesta es enviada al dispositivo de reconocimiento facial el cual se encargará de la toma de decisión en cuanto al acceso.

CAPÍTULO IV

4. ANALISIS Y RESULTADOS.

Una vez que el prototipo se ha construido y el sistema de acceso por reconocimiento facial se ha culminado en su totalidad, se proceden a efectuar pruebas del mismo para determinar el grado de aceptación que conlleva a implementar este tipo de sistemas, al igual que el correcto funcionamiento del prototipo.

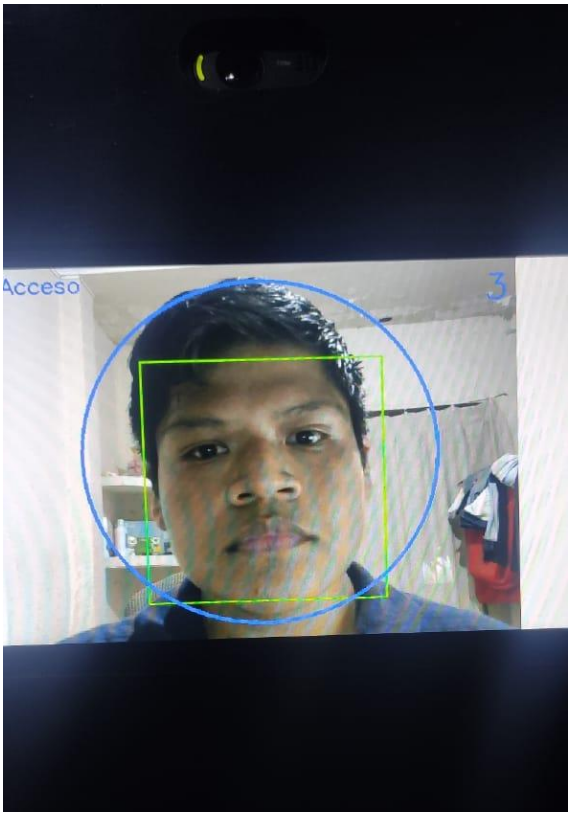


Figura 51.Prueba de sistema de control de acceso con el prototipo.

Fuente: autores.

4.1. Encuesta de aceptación para los sistemas de reconocimiento facial.

Para la implementación de los sistemas de reconocimiento facial se requiere conocer su grado de aceptación, para ello, se realizó una encuesta donde se obtuvieron los siguientes resultados.

Pregunta N° 1: ¿Qué tipo de tecnología de control de accesos ha utilizado?

Tabla 5.

Resultados primera pregunta.

Resultados a la pregunta N° 1	
Huella dactilar	55%
Tarjeta magnética	39%
Reconocimiento facial	6%
Ninguna	0%

Fuente: autores.

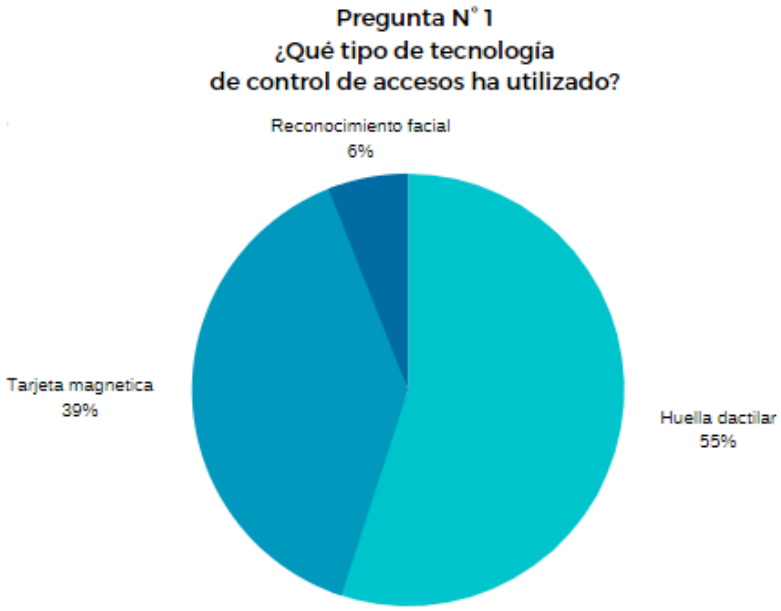


Figura 52. Resultados Primera pregunta.

Fuente: autores.

El principal objetivo de esta pregunta es conocer la prioridad de las diferentes tecnologías para los sistemas de control de acceso y que tanta frecuencia tiene el uso de los sistemas de reconocimiento facial, en la cual se determinó que apenas el 6% de las personas encuestadas ha hecho uso de este tipo de sistemas.

Pregunta N° 2: ¿Cree Ud. que los sistemas de reconocimiento facial ofrecen un mayor grado de seguridad con respecto al resto de tecnologías?

Tabla 6.

Resultados segunda pregunta.

Resultados a la pregunta N° 2	
Si	80%
No	20%

Fuente: autores.

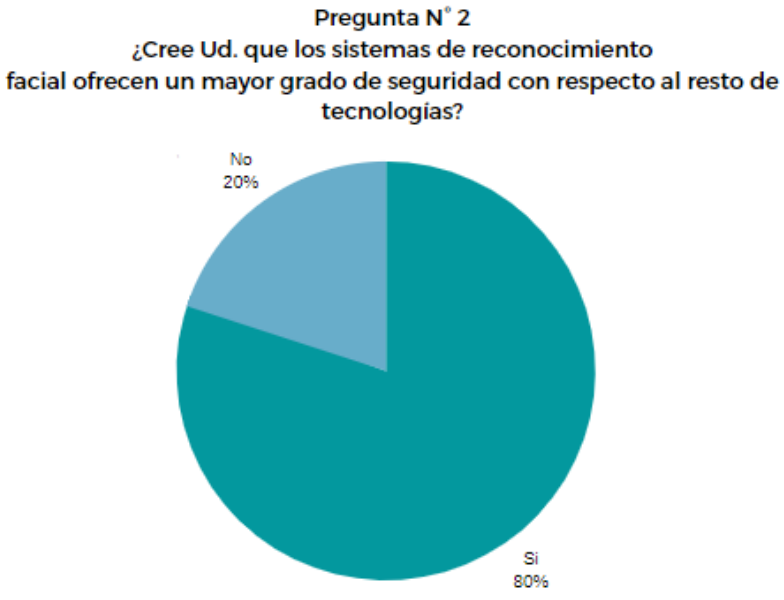


Figura 53. Resultados segunda pregunta.

Fuente: autores.

La pregunta se enfoca en el grado de seguridad que inspira este tipo de sistemas, en la cual se obtuvo un resultado positivo, debido a que 8 de cada 10 personas perciben este tipo de tecnologías como las más seguras con respecto al resto.

Pregunta N° 3: ¿Cree Ud. que los sistemas de reconocimiento son fáciles de utilizar?

Tabla 7.

Resultados tercera pregunta.

Resultados a la pregunta N° 3	
Si	70%
No	30%

Fuente: autores.

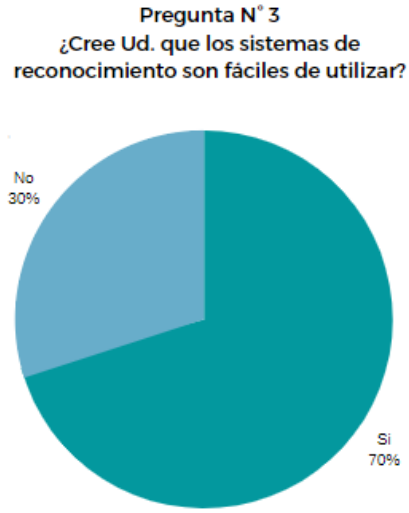


Figura 54. Resultados tercera pregunta.

Fuente: autores.

Se requiere determinar la perspectiva de las personas acerca de la facilidad de manipular un dispositivo de reconocimiento facial y se evidencia que la

mayoría de las personas determinan que la tecnología conlleva a un fácil uso del sistema.

Pregunta N° 4: Del 1 al 10, ¿Qué tan seguro cree Ud. de que los sistemas de acceso por reconocimiento facial mejoren el control de registros de accesos hacia un determinado sitio?

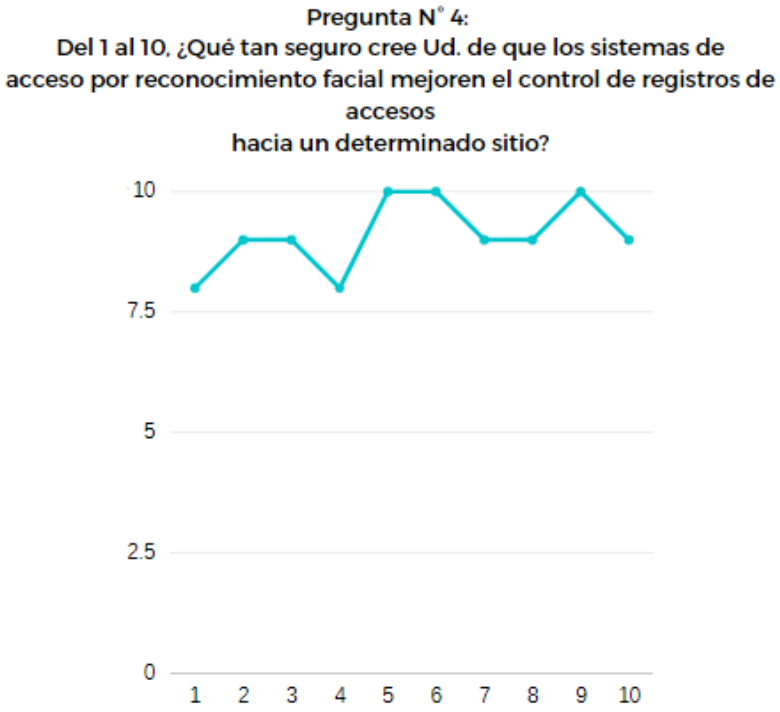


Figura 55 Resultados cuarta pregunta.

Fuente: autores.

Se observa que dicho tipo de sistemas pretenden una mejoría ante los registros de accesos con respecto a otros tipos de tecnología, reflejando resultados muy positivos en la encuesta realizada.

Pregunta N° 5: A su criterio, ¿Cuáles de las siguientes características es la más importante para seleccionar un sistema de reconocimiento facial?

Tabla 8.

Resultados quinta pregunta.

Resultados a la pregunta N° 5	
Calidad	30%
Seguridad	40%
Costo	20%
Fácil uso	10%

Fuente: autores.

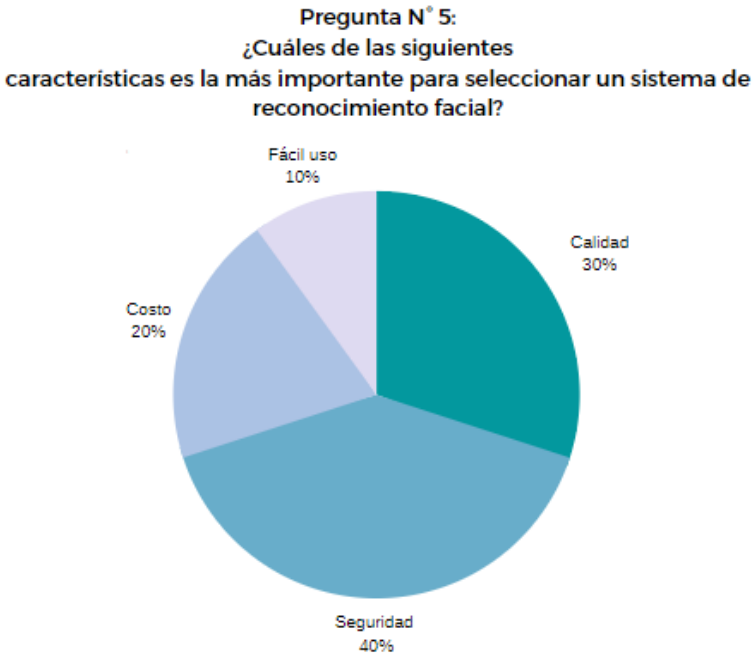


Figura 56. Resultados quinta pregunta.

Fuente: autores.

Se determina que un 40% están de acuerdo que la característica más importante de este sistema es la seguridad que puede llegar a brindar.

4.2. Pruebas del sistema de control de acceso.

Para la prueba del dispositivo y funcionamiento del sistema, se ha implementado de una cerradura eléctrica y un módulo wifi esp32 de Arduino que recibe señales de forma inalámbricas para la apertura de la puerta, con esto pretende simular el lugar al cual se desea acceder haciendo uso del sistema de reconocimiento facial.

Figura 57. Prueba del prototipo de control de acceso para reconocimiento facial.

Fuente: autores.

Figura 58. Puerta con cerradura eléctrica y módulo esp32 wifi Arduino para simulación de acceso mediante reconocimiento facial

Fuente: autores.

Figura 59. Estructura del prototipo para control de acceso.

Fuente: autores.

Se considera el registro de 5 usuarios diferentes a manera de prueba, con el fin de lograr determinar mediante 10 intentos, las cantidades de aciertos logrados por cada uno de los usuarios y obtener la medición de cuanta precisión otorga el equipo.

Tabla 9

Intentos registrados de 5 usuarios

Usuario (5)	Número de intentos (10)									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
# 1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
# 2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
# 3	OK	OK	X	OK	OK	OK	OK	OK	OK	OK
# 4	X	X	OK	OK	OK	OK	OK	OK	OK	OK
# 5	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

Fuente: autores

A través de los resultados obtenidos en las pruebas anteriores se determina las siguientes conclusiones

Tabla 10

Resultados obtenidos del uso del sistema de reconocimiento facial

Tipo de medición	Usuario				
	#1	**#2**	**#3**	**#4**	**#5**
Media	100%	100%	90%	80%	100%
Desviación típica	0	0	0.3	0.4	0

Fuente: autores.

Al explorar los resultados expuestos en la tabla 10, se puede determinar la cantidad de aciertos de cada individuo, el usuario #1, usuario #2 y usuario #5 obtuvieron la mayor cantidad de aciertos, con un total del 100% de intentos positivos, sin embargo, el usuario #4 obtuvo la menor cantidad de aciertos positivos con un porcentaje de 80%, en cambio el usuario #3 reflejo una del 90% de aciertos. Mediante estos resultados se deduce que la cantidad de aciertos en todos los casos son satisfactorios, arrojando un promedio general del 94% de efectividad al momento de usar el sistema de reconocimiento facial.

Con respecto a la desviación típica, se obtuvo la mayor precisión para los usuarios #1, usuario #2 y usuario #5, obteniendo el valor de cero en el resultado en su desviación y el peor de los casos fue para el usuario #4 con un valor de 0.4. Para concluir se verifican los resultados de las marcaciones y distinguir cuales son menos efectivos.

En los procesos se tiene factores que influyen, estos pueden ser la iluminación de la zona debido a que esto genera distorsiones en la imagen que capta la cámara al momento de realizar la fotografía y así no se puede reconocer fácilmente las características para detectar el rostro también se tiene a los recursos humanos que manipulan los dispositivos ya que pueden verificar el proceso de añadidura de datos a la base y modificarlos de acuerdo con los objetivos.

4.2.1. Pruebas de creación de usuarios y acceso a visualización de registros en plataforma web.

Para acceder a la creación de usuarios y a la visualización de marcaciones registradas se tiene que primero ingresar a la dirección http://34.67.143.38/ desde un navegador web y posteriormente realizar el login con las credenciales de administrador.

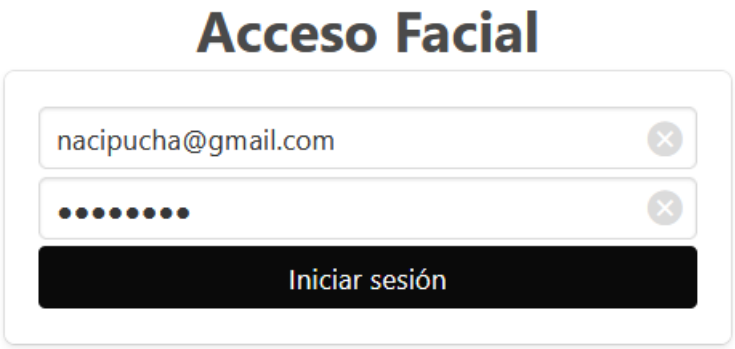


Figura 60. Ventana de inicio de sesión para el administrador.

Fuente: autores.

A continuación, se demuestra la manipulación para el registro de usuarios en el sistema y la visualización de logs correspondientes a los marcajes que ha realizado cada persona

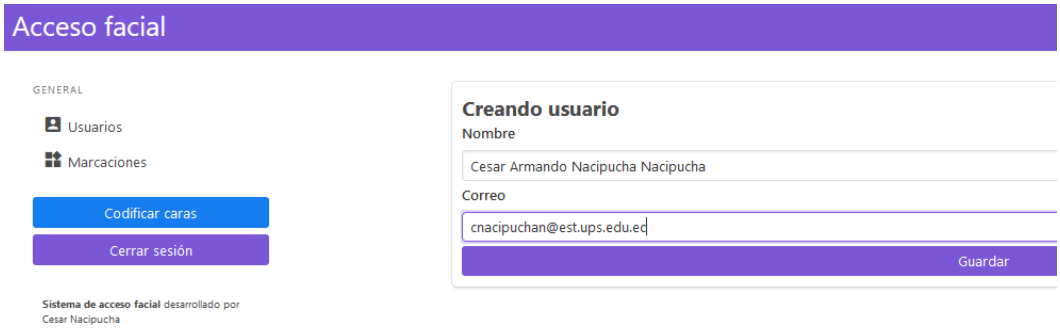


Figura 61. Creación de usuario, ingreso de datos en plataforma web.

Fuente: autores.

Otro de los campos a llenar son las fotografías de los rostros de cada usuario, el sistema permite ingresar una cantidad máxima de 5 fotos por persona

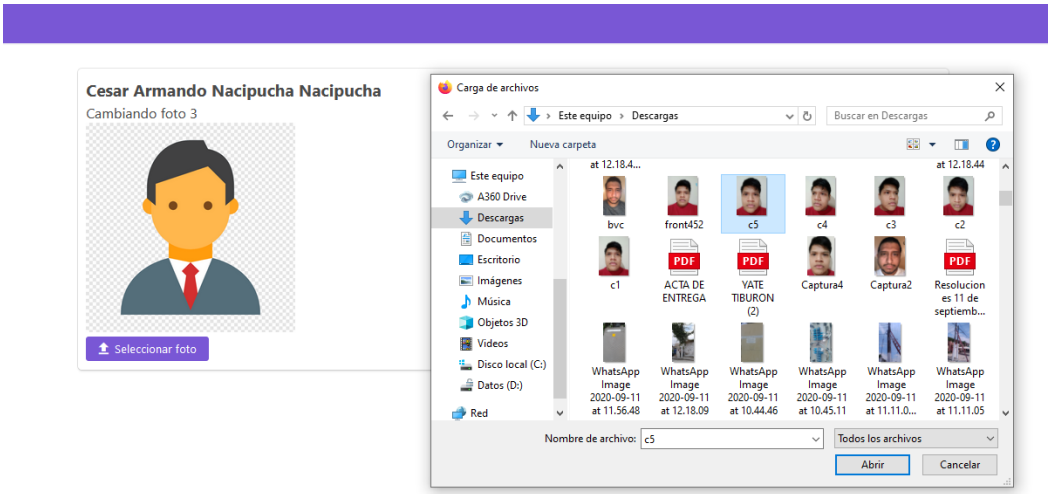


Figura 62. Asignación de fotos para el registro de usuarios.

Fuente: autores.

Una vez ingresado los usuarios, se puede visualizar una lista de las personas que han sido registradas, mostrando sus datos y sus fotografías.

Figura 63. Usuarios creados en la plataforma web.

Fuente: autores.

En la pestaña marcaciones podemos dar paso a la visualización completa de todos los registros que se han generado por el uso del sistema de reconocimiento facial

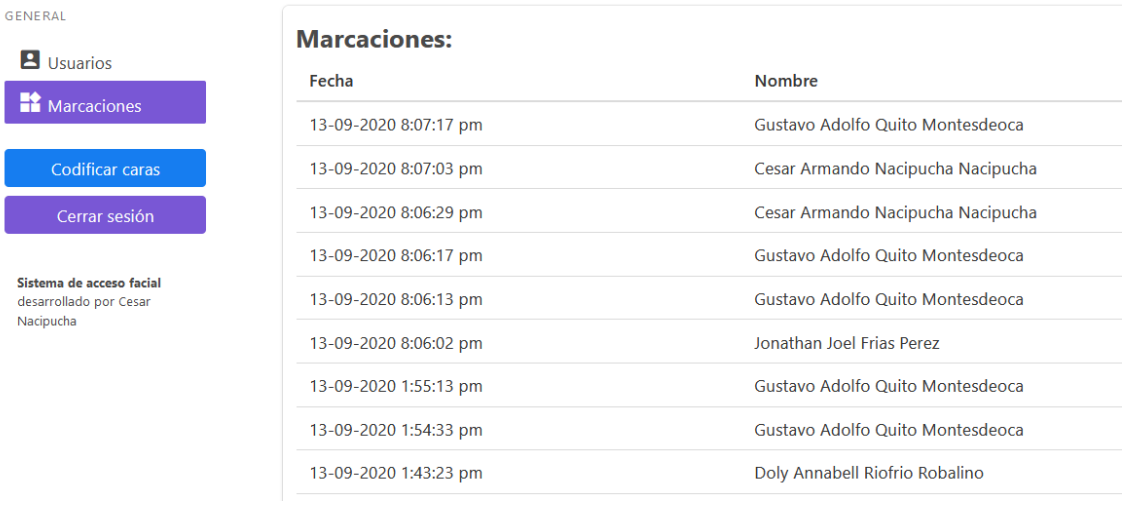


Figura 64. Visualización de registros de marcaciones de usuarios.

Fuente: autores.

4.2.2. Prueba de alerta vía correo electrónico para la detección de intrusos

La emisión de alertas se efectúa siempre que el sistema detecte un rostro desconocido, tomando como acción él envió de un mensaje al correo electrónico del administrador con la finalidad de prever y detectar usuarios intrusos que desean acceder sin previo registro o autorización.

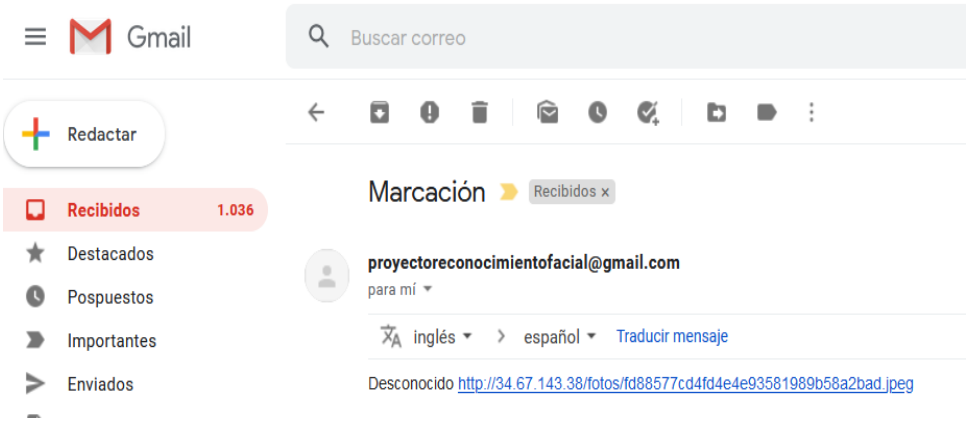


Figura 65. Correo de alerta enviado por el sistema hacia el administrador indicando usuario desconocido.

Fuente: autores.

Cabe mencionar que la fotografía llega al correo del administrador en forma de un enlace, el cual tendrá que ser aperturado para poder visualizar el rostro captado por la cámara de dispositivo

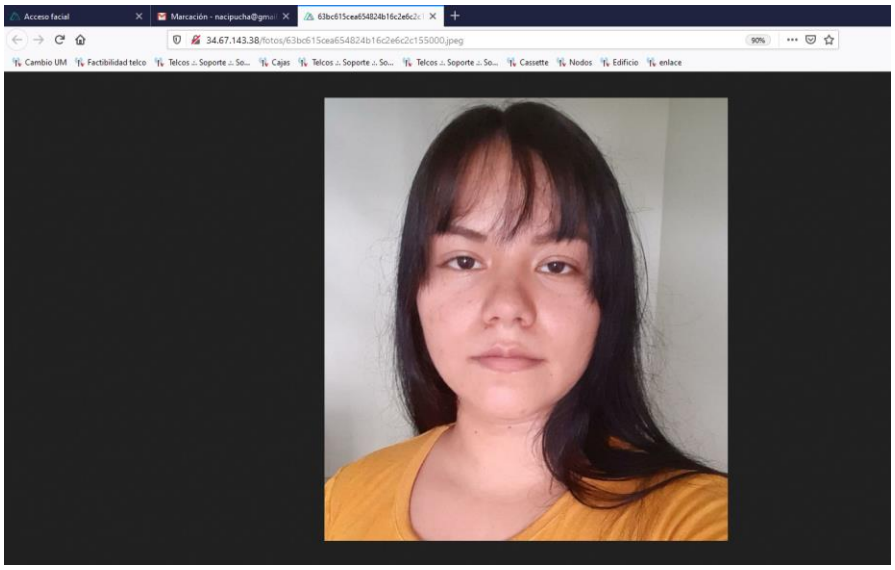


Figura 66. Mensaje enviado a administrador con el rostro del usuario desconocido.

Fuente: autores.

CONCLUSIONES.

Mediante la construcción del prototipo haciendo uso del LattePanda se comprobó que dicha tarjeta posee una amplia gama de prestaciones debido a su capacidad y robustez al momento de ejecutar aplicaciones que requieren un consumo exigente de recursos. En cuanto a las herramientas de software utilizadas tales como el servidor GCP, el API, la base de datos creada en MongoDB y el aplicativo web creado bajo Nuxt.js, se pudo constatar que su funcionalidad cubre las expectativas esperadas para el presente proyecto de titulación y en las pruebas realizadas con el sistema se puede confirmar su óptimo desempeño. De esta manera se puede concluir de forma general que la convergencia de tecnologías utilizadas para el desarrollo de este proyecto garantiza el cumplimiento de las exigencias requeridas y por consiguiente la culminación del proyecto técnico con resultados satisfactorios.

RECOMENDACIONES.

El dispositivo debe ubicarse en un lugar estratégico donde exista un grado de iluminación ideal con el fin de favorecer a una mejor captura de imágenes que dará como resultado una mayor precisión al momento de realizar reconocimiento facial.

Las imágenes correspondientes a los usuarios que serán cargadas al sistema por parte del administrador deben ser enfocadas al rostro del individuo y de diferentes ángulos (no mayores a 45°) con el objetivo de favorecer al sistema en la extracción de características faciales.

Se recomienda no realizar el desmontaje del prototipo debido a que en su interior se encuentran piezas electrónicas conectadas a la tarjeta LattePanda, que en caso de una mala manipulación pueden provocar fallos en el sistema o inconvenientes en el funcionamiento.

El prototipo debe estar conectado con un adaptador de corriente de 5V/3A que garantizan un suministro de energía ideal para su optimo desempeño.

No manipular el lente de la cámara con la mano u otros objetos debido a que puede perder su enfoque original.

REFERENCIAS.

[1] Aliexpress. 2019. LattePanda *AlphaPanda* . https://es.aliexpress.com/item/4000048323108.html.

[2] Amazon. 2019. *LattePanda 4G/64GB - a Win10 Development Board (without Win10 product key).* https://www.amazon.com.mx/LattePanda-4G-64GB-Development-without/dp/B01ID4HYE4?th=1.

[3] Androidpc. 2015. *LattePanda una placa de desarrollo con SoC Intel Atom x5-Z8300.* https://androidpc.es/lattepanda-una-placa-de-desarrollo-con-intel-atom-x5-z8300/.

[4] Benigno, J. 2020. *Cómo seleccionar y usar el módulo ESP32 con Wi-Fi/Bluetooth adecuado para una aplicación de IoT industria.* https://www.digikey.com/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module.

[5] Borges, S. 2019. *¿Qué es Nginx? Características, Ventajas e Instalación.* https://blog.infranetworking.com/que-es-nginx/.

[6] Fandos, M. 2003. «Formación basada en las Tecnologías de la Información y Comunicación.» https://www.tdx.cat/bitstream/handle/10803/8909/Etesis_1.pdf.

[7] Helmeyer Quevedo, Ramiro Francisco. 2017. «Tipos de sistemas Biométricos.» *ramirofranciscohelmeyerquevedo.info.* s.f. de s.f. https://ramirofranciscohelmeyerquevedo.info/ramiro-francisco-helmeyer-quevedo-sistemas-biometricos/.

[8] Jiménez, P. 2019. *Qué es Nuxt.js.* https://openwebinars.net/blog/que-es-nuxtjs-framework-de-vuejs/.

[9] LattePanda Team. 2017. *lattepanda-docs.* s.f. de s.f. http://www.lattepanda.com/lattepanda-docs.

[10] López Pérez, Nicolas, y Juan José Toro Agudelo. 2012. «TECNICAS DE BIOMETRIA BASADAS EN PATRONES FACIALES DEL SER HUMANO.» Universidad Tecnológica de Pereira. *(Proyecto de grado).* Pereira.

[11] Marín, R. 2019. *Los gestores de bases de datos más usados en la actualidad.* https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.

[12] Martín, M. 2016. *Detección de objetos por computador.* https://rdu.iua.edu.ar/bitstream/123456789/886/1/Versi%C3%B3n%2032%20del%20TFG%20Rodolfo%20Villanueva%20Detecci%C3%B3n%20de%20objetos%20por%20computador.pdf.

[13] Marzal, Andrés, y Isabel Gracia. 2009. «Introducción a la programación con Pyton.» En *Introducción a la programación con Pyton.*, de Andrés Marzal y Isabel Gracia, 17. Castellon de la Plana: Universitat Jaume I.

[14] New Horizons. 2019. *MongoDB.* https://www.newhorizonsmadrid.com/formaci%C3%B3n-y-certificaciones/cursos-t%C3%A9cnicos/big-data/mongodb.

[15] NGINX. 2019. *NGINX.* https://www.nginx.com/.

[16] NuxtJS . 2019. *NuxtJS Design.* https://nuxtjs.org/design/.

[17] Pérez, C, L González, L Vásquez, y L Pérez. 2016. *Sistema operativo modernos.*

http://carlosmanuelperezsolis2287.blogspot.com/2016/04/cuestionario-del-libro-sistema.html.

[18] Programación.net. 2020. *Librerías de Python que son sencillamente irresistibles.* https://programacion.net/articulo/20_librerias_de_python_que_son_sencillamente_irresistibles_1061.

[19] Ros, I. 2019. *LattePanda Delta 432, un mini PC Intel que cabe en la palma de tu mano.* https://www.muycomputer.com/2019/11/18/lattepanda-delta-432-mini-pc-intel/.

[20] Tolosa Borja, César, y Álvaro Giz Bueno. s.f. «DSI.» *Departamento de Sistemas Informáticos.* s.f. de s.f. https://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Documentacion/Trabajos/Biometria/Trabajo%20Biometria.pdf.

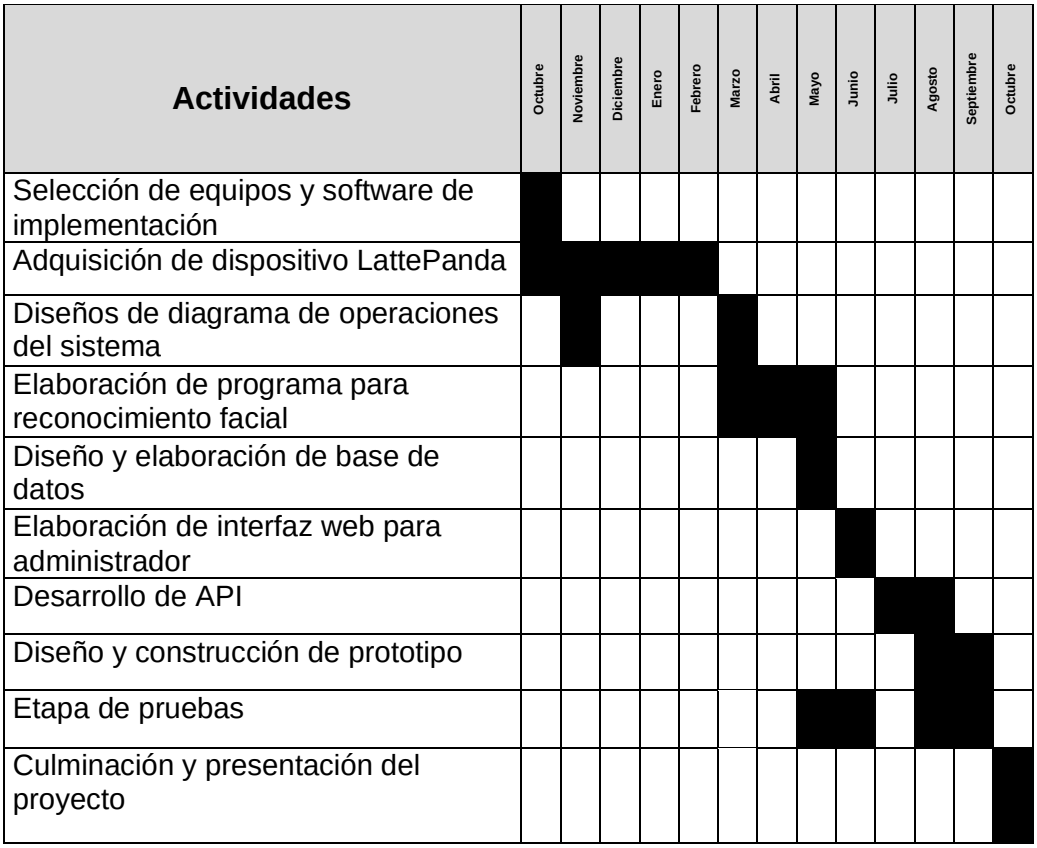
[21] WAREX. 2019. *ESP332.* https://warex.com.co/esp32/.

[22] XATAKA Móvil. 2019. *¿Qué podemos hacer con 16 GB de almacenamiento en un smartphone en la actualidad?* https://www.xatakamovil.com/movil-y-sociedad/que-podemos-hacer-con-16-gb-de-almacenamiento-en-un-smartphone-en-la-actualidad.

ANEXOS.

Materiales	Cantidad	Precio Unitario	Total
Lattepanda 2G/32G	1	$90	$90
Display 7 inch Touch	1	$40	$40
Web cam 3Mpxl	1	$45	$45
Estructura de acrílico	1	$25	$25
Patch cord UTP cat 5e	1	$3	$3
Cargador 5V-3A	1	$13	$13
Gastos varios	1	$20	$20
Módulo ESP 32 Arduino	1	$12	$12
		Total	$248

Presupuesto para la construcción de prototipo.



Cronograma de actividades del proyecto.

```python
from imutils import paths
import face_recognition
import pickle
import wget
import cv2
import os

class ReconocimientoFacial:

    def __init__(self):
        self.path = "/home/proyectoreconocimientofacial/sistema/facerecognition/"
        self.detectionMethod = 'hog'
        self.encodings = self.path+"encodings.pickle"
        self.dataset = self.path+"dataset"
        self.leerPickle()

    def leerPickle(self):
        self.data = pickle.loads(open(self.encodings, "rb").read())

    def codificarCaras(self):

        imagePaths = list(paths.list_images(self.dataset))

        knownEncodings = []
        knownNames = []

        for (i, imagePath) in enumerate(imagePaths):
            print("Codificando "+str(i)+"/"+str(len(imagePaths)))
            name = imagePath.split(os.path.sep)[-2]

            image = cv2.imread(imagePath)
            rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

            boxes = face_recognition.face_locations(rgb,
                                                    model=self.detectionMethod)

            encodings = face_recognition.face_encodings(rgb, boxes)
            for encoding in encodings:
                knownEncodings.append(encoding)
                knownNames.append(name)

        data = {"encodings": knownEncodings, "names": knownNames}
        f = open(self.encodings, "wb")
        f.write(pickle.dumps(data))
        f.close()

    def reconocer(self, foto):

        filename = wget.download(foto)
        image = cv2.imread(filename)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        boxes = face_recognition.face_locations(rgb, model=self.detectionMethod)
        encodings = face_recognition.face_encodings(rgb, boxes)
        names = []

        if len(encodings) > 1:
            return

        for encoding in encodings:
            matches = face_recognition.compare_faces(self.data["encodings"], encoding)
            name = "Desconocido"

            if True in matches:
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}

                for i in matchedIdxs:
                    name = self.data["names"][i]
                    counts[name] = counts.get(name, 0) + 1

                name = max(counts, key=counts.get)

            names.append(name)

        return names[0]
```

Figura A.1. Código de la aplicación para el reconocimiento facial.

Fuente: autores.

```
import numpy as np
import Image
cimport numpy as np
cimport cython

cdef extern from "math.h":
    double sqrt(double i)
    double fabs(double i)
    double floor(double i)

cpdef hog(im, int sbin = 8):
    """
    Computes a histogram of oriented gradient features.
    Adopted from Pedro Felzenszwalb's features.cc
    """
    cdef np.ndarray[np.double_t, ndim=3] data, feat
    cdef np.ndarray[np.double_t, ndim=1] hist, norm

    cdef int blocks0, blocks1
    cdef int out0, out1, out2
    cdef int visible0, visible1

    cdef double dy, dx, v
    cdef double dy2, dx2, v2
    cdef double dy3, dx3, v3
    cdef double best_dot, dot
    cdef int best_o

    cdef double xp, yp, vx0, vy0, vx1, vy1
    cdef int ixp, iyp
    cdef double n1, n2, n3, n4, t1, t2, t3, t4, h1, h2, h3, h4
    cdef int p

    cdef np.ndarray[np.double_t, ndim=1] uu
    uu = np.array([ 1.0000,  0.9397,  0.7660,  0.500,  0.1736,
                   -0.1736, -0.5000, -0.7660, -0.9397])
    cdef np.ndarray[np.double_t, ndim=1] vv
    vv = np.array([0.0000, 0.3420, 0.6428, 0.8660, 0.9848,
                   0.9848, 0.8660, 0.6428, 0.3420])

    cdef double eps = 0.0001 # to avoid division by 0
    cdef unsigned int cc0, cc1, cc2

    cdef int x, y, o, q
    cdef int dstptr, srcptr

    width, height = im.size
    blocks0 = height / sbin
    blocks1 = width / sbin

    out0 = blocks0 - 2
    out1 = blocks1 - 2
    out2 = 9 + 4

    visible0 = blocks0 * sbin
    visible1 = blocks1 * sbin

    data = np.asarray(im, dtype=np.double)

    cc0 = <unsigned int>(0)
    cc1 = <unsigned int>(1)
    cc2 = <unsigned int>(2)

    hist = np.zeros(shape=(blocks0 * blocks1 * 9), dtype=np.double)
    norm = np.zeros(shape=(blocks0 * blocks1), dtype=np.double)
    feat = np.zeros(shape=(out0, out1, out2), dtype=np.double)

    for x from 1 <= x < visible1 - 1:
        for y from 1 <= y < visible0 - 1:
            dy = data[y + 1, x, cc0] - data[y - 1, x, cc0]
            dx = data[y, x + 1, cc0] - data[y, x - 1, cc0]
            v = dx * dx + dy * dy

            dy2 = data[y + 1, x, cc1] - data[y - 1, x, cc1]
            dx2 = data[y, x + 1, cc1] - data[y, x - 1, cc1]
            v2 = dx2 * dx2 + dy2 * dy2

            dy3 = data[y + 1, x, cc2] - data[y - 1, x, cc2]
            dx3 = data[y, x + 1, cc2] - data[y, x - 1, cc2]
            v3 = dx3 * dx3 + dy3 * dy3
```

Figura A.2. Código de algoritmo de reconocimiento facial parte 1

Fuente: autores.

```
            if v2 > v: # pick channel with strongest gradient
                v = v2
                dx = dx2
                dy = dy2
            if v3 > v:
                v = v3
                dx = dx3
                dy = dy3

            # snap to one of 9 orientations
            best_dot = 0.
            best_o = 0
            for o from 0 <= o < 9:
                dot = fabs(uu[o] * dx + vv[o] * dy)
                if dot > best_dot:
                    best_dot = dot
                    best_o = o

            # add to 4 histograms around pixel using linear interpolation
            xp = (<double>(x) + 0.5) / <double>(sbin) - 0.5
            yp = (<double>(y) + 0.5) / <double>(sbin) - 0.5

            ixp = <int>floor(xp)
            iyp = <int>floor(yp)

            vx0 = xp - ixp
            vy0 = yp - iyp
            vx1 = 1.0 - vx0
            vy1 = 1.0 - vy0
            v = sqrt(v)

            if ixp >= 0 and iyp >= 0:
                hist[ixp * blocks0 + iyp + best_o*blocks0*blocks1] += vx1 * vy1 * v
            if ixp + 1 < blocks1 and iyp >= 0:
                hist[(ixp + 1) * blocks0 + iyp + best_o*blocks0*blocks1] += vx0 * vy1 * v
            if ixp >= 0 and iyp + 1 < blocks0:
                hist[ixp * blocks0 + (iyp + 1) + best_o*blocks0*blocks1] += vx1 * vy0 * v
            if ixp + 1 < blocks1 and iyp + 1 < blocks0:
                hist[(ixp + 1) * blocks0 + (iyp + 1) + best_o * blocks0 * blocks1] += vx0 * vy0 * v

    # compute energy in each block by summing over orientations
    for o from 0 <= o < 9:
        for q from 0 <= q < blocks0 * blocks1:
            norm[q] += hist[o * blocks0 * blocks1 + q] * hist[o * blocks0 * blocks1 + q]

    # compute normalized values
    for x from 0 <= x < out1:
        for y from 0 <= y < out0:
            p = (x+1) * blocks0 + y + 1
            n1 = 1.0 / sqrt(norm[p] + norm[p+1] + norm[p+blocks0] + norm[p+blocks0+1] + eps)
            p = (x+1) * blocks0 + y
            n2 = 1.0 / sqrt(norm[p] + norm[p+1] + norm[p+blocks0] + norm[p+blocks0+1] + eps)
            p = x * blocks0 + y + 1
            n3 = 1.0 / sqrt(norm[p] + norm[p+1] + norm[p+blocks0] + norm[p+blocks0+1] + eps)
            p = x * blocks0 + y
            n4 = 1.0 / sqrt(norm[p] + norm[p+1] + norm[p+blocks0] + norm[p+blocks0+1] + eps)

            t1 = 0
            t2 = 0
            t3 = 0
            t4 = 0

            srcptr = (x+1) * blocks0 + y + 1
            for o from 0 <= o < 9:
                h1 = hist[srcptr] * n1
                h2 = hist[srcptr] * n2
                h3 = hist[srcptr] * n3
                h4 = hist[srcptr] * n4
                # for some reason, gcc will not automatically inline
                # the min function here, so we just do it ourselves
                # for impressive speedups
                if h1 > 0.2:
                    h1 = 0.2
                if h2 > 0.2:
                    h2 = 0.2
                if h3 > 0.2:
                    h3 = 0.2
                if h4 > 0.2:
                    h4 = 0.2
```

Figura A.3. Código de algoritmo de reconocimiento facial parte 2.

Fuente: autores.

```
                feat[y, x, o] = 0.5 * (h1 + h2 + h3 + h4)
                t1 += h1
                t2 += h2
                t3 += h3
                t4 += h4
                srcptr += blocks0 * blocks1

            feat[y, x, 9] = 0.2357 * t1
            dstptr += out0 * out1
            feat[y, x, 10] = 0.2357 * t2
            dstptr += out0 * out1
            feat[y, x, 11] = 0.2357 * t3
            dstptr += out0 * out1
            feat[y, x, 12] = 0.2357 * t4

    return feat

cpdef hogpad(np.ndarray[np.double_t, ndim=3] hog):
    cdef np.ndarray[np.double_t, ndim=3] out
    cdef int i, j, k
    cdef int w = hog.shape[0], h = hog.shape[1], z = hog.shape[2]
    out = np.zeros((w + 2, h + 2, z))

    for i in range(w):
        for j in range(h):
            for k in range(z):
                out[i+1, j+1, k] = hog[i, j, k]
    return out

cpdef rgbhist(im, int binsize = 8):
    """
    Computes an RGB color histogram with a binsize.
    """
    cdef int w = im.size[0], h = im.size[1]
    cdef np.ndarray[np.uint8_t, ndim=3] data = np.asarray(im)
    cdef np.ndarray[np.double_t, ndim=1] hist
    hist = np.zeros(binsize * binsize * binsize)
    for i from 0 <= i < w:
        for j from 0 <= j < h:
            bin  = (<int>data[j,i,0]) / (256/binsize)
            bin += (<int>data[j,i,1]) / (256/binsize) * binsize
            bin += (<int>data[j,i,2]) / (256/binsize) * binsize * binsize
            hist[bin] += 1
    return hist

cpdef rgbmean(im):
    """
    Computes mean and covariances of RGB colors.
    """
    cdef int w = im.size[0], h = im.size[1]
    cdef double r, g, b
    cdef np.ndarray[np.uint8_t, ndim=3] data = np.asarray(im)
    cdef np.ndarray[np.double_t, ndim=1] out = np.zeros(9)

    for i in range(w):
        for j in range(h):
            r = data[j, i, 0] / 255.
            g = data[j, i, 1] / 255.
            b = data[j, i, 2] / 255.

            out[0] += r
            out[1] += g
            out[2] += b

            out[3] += r * r
            out[4] += r * g
            out[5] += r * b

            out[6] += g * g
            out[7] += g * b

            out[8] += b * b
    return out  / (w * h)
```

Figura A.4. Código de algoritmo de reconocimiento facial parte 3.

Fuente: autores.

```python
from fastapi import Depends, FastAPI, Header, HTTPException
from fastapi.middleware.cors import CORSMiddleware

from routers.Administrador import CrearAdministrador, LoginAdministrador, CodificarCaras
from routers.Usuario import ConseguirUsuarios, CrearUsuario, ActualizarUsuario, BorrarUsuario, SubirFotoUsuario
from routers.Marcacion import ConseguirMarcaciones, ReconocerFoto

tags_metadata = [
    {
        "name": "administradores",
        "description": "Funcionalidades para administradores"
    },
    {
        "name": "usuarios",
        "description": "Funcionalidades para usuarios"
    },
    {
        "name": "marcaciones",
        "description": "Funcionalidades para marcaciones"
    }
]


app = FastAPI(
    title="API Facial",
    description="Proyecto de marcaciones de acceso para usuarios",
    version="0.0.1",
    openapi_tags=tags_metadata
)

app.add_middleware(
    CORSMiddleware,
    allow_origins="*",
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

async def get_token_header(x_token: str = Header(...)):
    if x_token != "fake-super-secret-token":
        raise HTTPException(status_code=400, detail="X-Token header invalid")


app.include_router(CrearAdministrador.router)
app.include_router(LoginAdministrador.router)
app.include_router(CodificarCaras.router)

app.include_router(ConseguirUsuarios.router)
app.include_router(CrearUsuario.router)
app.include_router(ActualizarUsuario.router)
app.include_router(BorrarUsuario.router)
app.include_router(SubirFotoUsuario.router)

app.include_router(ConseguirMarcaciones.router)
app.include_router(ReconocerFoto.router)
```

Figura A.5. Código principal del aplicativo API.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.AdministradorModel import AdministradorModel

router = APIRouter()

class Administrador(BaseModel):
    nombre: str
    correo: str
    clave: str

@router.post("/administradores",  tags=["administradores"], status_code=status.HTTP_201_CREATED)
def crear_administrador(data: Administrador, response: Response):
    """
    Registrar administrador
    """
    data = dict(data)

    administrador = AdministradorModel.buscar_por_correo(data["correo"])
    if administrador:
        response.status_code = status.HTTP_400_BAD_REQUEST
        return {"message": "El correo ingresado, ya se registro"}

    AdministradorModel.guardar(data)
    return {"message": "El administrador se guardo"}
```

Figura A.6. Código API para creación de administradores.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.AdministradorModel import AdministradorModel

router = APIRouter()


class AdministradorLogin(BaseModel):
    correo: str = ""
    clave: str = ""


@router.post("/administradores/login",  tags=["administradores"], status_code=status.HTTP_200_OK)
def login_administrador(data: AdministradorLogin, response: Response):
    """
    Iniciar sesión para el administrador
    """
    data = dict(data)

    administrador = AdministradorModel.buscar_por_correo(data["correo"])
    if administrador and administrador["clave"] == data["clave"]:
        administrador.pop("clave")
        return administrador

    response.status_code = status.HTTP_400_BAD_REQUEST
    return {"message": "No se encuentra el administrador"}
```

Figura A.7. Código API para Login de administrador.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel
from facerecognition.ReconocimientoFacial import ReconocimientoFacial
import os


reconocimientoFacial = ReconocimientoFacial()

router = APIRouter()


class AdministradorLogin(BaseModel):
    correo: str = ""
    clave: str = ""


@router.post("/encode_face",  tags=["administradores"], status_code=status.HTTP_200_OK)
def login_administrador(response: Response):
    """
    Codificar caras
    """
    path = "/home/proyectoreconocimientofacial/sistema/facerecognition/dataset/"
    path_fotos = "/home/proyectoreconocimientofacial/sistema/fotos/"
    os.system("rm -rf "+path+"*")

    usuarios = UsuarioModel.conseguir_todo()

    for usuario in usuarios:
        os.system("mkdir -p "+path+usuario["_id"])
        for foto in usuario["fotos"]:
            os.system("cp "+path_fotos+foto+" "+path+usuario["_id"])

    reconocimientoFacial.codificarCaras()

    return {"message": "Se ejecutó la codificación de caras"}
```

Figura A.8. Código API para codificación de caras.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel

router = APIRouter()


@router.get("/usuarios",  tags=["usuarios"], status_code=status.HTTP_200_OK)
def conseguir_usuarios(response: Response):
    """
    Conseguir todos los usuarios registrados
    """
    usuarios = UsuarioModel.conseguir_todo()
    return {"usuarios": usuarios}
```

Figura A.9. Código API para conseguir los usuarios registrados en la base de datos.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel

router = APIRouter()


class UsuarioCrear(BaseModel):
    nombre: str
    correo: str


@router.post("/usuarios",  tags=["usuarios"], status_code=status.HTTP_201_CREATED)
def crear_usuario(data: UsuarioCrear, response: Response):
    """
    Registrar usuario
    """
    data = dict(data)

    usuario = UsuarioModel.buscar_por_correo(data["correo"])
    if usuario:
        response.status_code = status.HTTP_400_BAD_REQUEST
        return {"message": "El correo ingresado, ya se registro"}

    UsuarioModel.guardar(data)
    return {"message": "El usuario se guardo"}
```

Figura A.10. Código API para conseguir para la creación de usuarios.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel

router = APIRouter()


class UsuarioActualizar(BaseModel):
    nombre: str
    correo: str


@router.put("/usuarios/{usuario_id}",  tags=["usuarios"], status_code=status.HTTP_200_OK)
def actualizar_usuario(data: UsuarioActualizar, usuario_id: str, response: Response):
    """
    Actualizar usuario
    """
    data = dict(data)

    usuario = UsuarioModel.conseguir_por_id(usuario_id)
    if usuario:
        UsuarioModel.actualizar(usuario_id, data)
        return {"message": "Usuario actualizado"}

    response.status_code = status.HTTP_400_BAD_REQUEST
    return {"message": "No existe usuario"}
```

Figura A.11. Código API para conseguir para editar usuarios.

Fuente: autores.

> Usuario > BorrarUsuario.py

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel

router = APIRouter()


@router.delete("/usuarios/{usuario_id}",  tags=["usuarios"], status_code=status.HTTP_200_OK)
def borrar_usuario(usuario_id: str, response: Response):
    """
    Borrar usuario
    """
    usuario = UsuarioModel.conseguir_por_id(usuario_id)
    if usuario:
        UsuarioModel.borrar(usuario_id)
        return {"message": "El usuario se elimino"}

    response.status_code = status.HTTP_400_BAD_REQUEST
    return {"message": "No se encuentra el usuario"}
```

Figura A.12. Código API para eliminar usuarios.

Fuente: autores.

```python
import os
import uuid
import shutil
from fastapi import APIRouter, Response, status, File, UploadFile
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel

router = APIRouter()

@router.post("/usuarios/{usuario_id}",  tags=["usuarios"], status_code=status.HTTP_200_OK)
async def subir_foto_usuario(usuario_id: str, foto_seleccionada: int, response: Response, file: UploadFile = File(...)):
    """
    Subir foto para usuario
    """
    usuario = UsuarioModel.conseguir_por_id(usuario_id)
    if not usuario:
        return {"message": "Usuario no encontrado"}

    upload_folder = "/home/proyectoreconocimientofacial/sistema/fotos/"
    file_object = file.file
    # create empty file to copy the file_object to
    filename = str(file.filename)
    extension = filename.split(".")[1]
    uuid_code = uuid.uuid4()
    uuid_hex = uuid_code.hex

    nombre = usuario_id + "--" + uuid_hex + "." + extension

    extensiones_permitidas = ["png", "jpg", "jpeg"]

    if not extension.lower() in extensiones_permitidas:
        response.status_code = status.HTTP_400_BAD_REQUEST
        return {"message": "La foto subida no coincide con la extensión permitida"}

    usuario["fotos"][foto_seleccionada] = nombre
    UsuarioModel.actualizar(usuario_id, {"fotos": usuario["fotos"]})
    upload_folder = open(os.path.join(upload_folder, nombre), 'wb+')
    shutil.copyfileobj(file_object, upload_folder)
    upload_folder.close()
    return {"message": "Foto subida"}
```

Figura A.13. Código API para cargar fotos a los usuarios.

Fuente: autores.

```python
import os
import uuid
import shutil
from fastapi_mail import FastMail
from fastapi import APIRouter, Response, status, File, UploadFile, BackgroundTasks
from pydantic import BaseModel
from Model.UsuarioModel import UsuarioModel
from Model.MarcacionModel import MarcacionModel
from Model.AdministradorModel import AdministradorModel
from facerecognition.ReconocimientoFacial import ReconocimientoFacial

reconocimientoFacial = ReconocimientoFacial()
router = APIRouter()

def enviarCorreos(background_tasks, filename):
    administradores = AdministradorModel.conseguir_todo()

    to = []
    for administrador in administradores:
        to.append(administrador["correo"])

    mail = FastMail(email="proyectoreconocimientofacial@gmail.com",
                    password="proyecto2020", tls=True, port="587", service="gmail")

    bulk = False
    if len(to) > 1:
        bulk = True
    background_tasks.add_task(mail.send_message, recipient=to, subject="Marcación",
                              body="Desconocido "+filename, text_format="plain", bulk=bulk)


@router.post("/marcaciones/foto",  tags=["marcaciones"], status_code=status.HTTP_200_OK)
async def reconocer_foto_marcacion(background_tasks: BackgroundTasks, response: Response, file: UploadFile = File(...)):
    """
    Reconocer foto para registrar una marcación
    """
    upload_folder = "/home/proyectoreconocimientofacial/sistema/fotos/"
    file_object = file.file
    # create empty file to copy the file_object to
    filename = str(file.filename)
    extension = filename.split(".")[1]
    uuid_code = uuid.uuid4()
    uuid_hex = uuid_code.hex

    nombre = uuid_hex + "." + extension

    extensiones_permitidas = ["png", "jpg", "jpeg"]

    if not extension.lower() in extensiones_permitidas:
        response.status_code = status.HTTP_400_BAD_REQUEST
        return {"message": "La foto subida no coincide con la extensión permitida"}

    if os.path.isfile('./fotos/'+nombre):
        os.remove(upload_folder+nombre)

    upload_folder = open(os.path.join(upload_folder, nombre), 'wb+')
    shutil.copyfileobj(file_object, upload_folder)
    upload_folder.close()

    reconocimientoFacial.leerPickle()
    usuario_id = reconocimientoFacial.reconocer("http://34.67.143.38/fotos/"+nombre)

    if usuario_id == "Desconocido":
        enviarCorreos(background_tasks, "http://34.67.143.38/fotos/"+nombre)
        response.status_code = status.HTTP_400_BAD_REQUEST
        return {"message": "Desconocido"}

    usuario = UsuarioModel.conseguir_por_id(usuario_id)
    if not usuario:
        response.status_code = status.HTTP_400_BAD_REQUEST
        return {"message": "Usuario no encontrado"}

    MarcacionModel.guardar(
        {"usuario_id": usuario["_id"], "nombre": usuario["nombre"]})

    return {"message": "Registro marcado", "usuario": usuario}
```

Figura A.14. Código API para ejecutar el reconocimiento facial.

Fuente: autores.

```python
from fastapi import APIRouter, Response, status
from pydantic import BaseModel
from Model.MarcacionModel import MarcacionModel

router = APIRouter()


@router.get("/marcaciones",  tags=["marcaciones"], status_code=status.HTTP_200_OK)
def conseguir_marcaciones(response: Response):
    """
    Conseguir todas las marcaciones registradas
    """
    marcaciones = MarcacionModel.conseguir_todo()
    return {"marcaciones": marcaciones}
```

Figura A.15. Código API para conseguir las marcaciones de la base de datos

Fuente: autores.

```python
from bson.objectid import ObjectId
from Database.MongoDB import MongoDB
from Utils.Datetime import Datetime
from Utils.JsonSerializer import JsonSerializer

db = MongoDB().db

class AdministradorModel:

    @staticmethod
    def conseguir_todo():
        query = db.administradores.find({"activo": True})
        query = list(query)
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def conseguir_por_id(_id):
        query = db.administradores.find_one(
            {"_id": ObjectId(_id), "activo": True})
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def buscar_por_correo(correo):
        query = db.administradores.find_one({'correo': correo})
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def guardar(data):
        data["activo"] = True
        data["creado"] = Datetime.get_current_datetime()
        data["actualizado"] = Datetime.get_current_datetime()
        inserted_id = db.administradores.insert_one(data).inserted_id
        return inserted_id

    @staticmethod
    def actualizar(_id, data):
        data["actualizado"] = Datetime.get_current_datetime()
        db.administradores.update_one(
            {"_id": ObjectId(_id)},
            {"$set": data}
        )

    @staticmethod
    def borrar(_id):
        data = {
            "activo": False,
            "creado": Datetime.get_current_datetime(),
            "actualizado": Datetime.get_current_datetime()
        }
        db.administradores.update_one(
            {"_id": ObjectId(_id)},
            {"$set": data}
        )
```

Figura A.16. Clase Administrador y conexión con base de datos.
Fuente: autores.

```python
import pymongo
from bson.objectid import ObjectId
from Database.MongoDB import MongoDB
from Utils.Datetime import Datetime
from Utils.JsonSerializer import JsonSerializer

db = MongoDB().db

class MarcacionModel:

    @staticmethod
    def conseguir_todo():
        query = db.marcaciones.find({"activo": True}, sort=[
                                    ('creado', pymongo.DESCENDING)])
        query = list(query)
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def conseguir_por_id(_id):
        query = db.marcaciones.find_one(
            {"_id": ObjectId(_id), "activo": True})
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def guardar(data):
        data["activo"] = True
        data["creado"] = Datetime.get_current_datetime()
        inserted_id = db.marcaciones.insert_one(data).inserted_id
        return inserted_id

    @staticmethod
    def borrar(_id):
        data = {
            "activo": False,
            "creado": Datetime.get_current_datetime(),
            "actualizado": Datetime.get_current_datetime()
        }
        db.marcaciones.update_one(
            {"_id": ObjectId(_id)},
            {"$set": data}
        )
```

Figura A.17. Clase Marcación y conexión con base de datos.

Fuente: autores.

```python
from bson.objectid import ObjectId
from Database.MongoDB import MongoDB
from Utils.Datetime import Datetime
from Utils.JsonSerializer import JsonSerializer

db = MongoDB().db

class UsuarioModel:

    @staticmethod
    def conseguir_todo():
        query = db.usuarios.find({"activo": True})
        query = list(query)
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def buscar_por_correo(correo):
        query = db.usuarios.find_one({'correo': correo})
        query = JsonSerializer.serialize(query)
        return query

    @staticmethod
    def conseguir_por_id(_id):
        query = db.usuarios.find_one(
            {"_id": ObjectId(_id), "activo": True})
        query = JsonSerializer.serialize(query)

    @staticmethod
    def actualizar(_id, data):
        data["actualizado"] = Datetime.get_current_datetime()
        db.usuarios.update_one(
            {"_id": ObjectId(_id)},
            {"$set": data}
        )

    @staticmethod
    def borrar(_id):
        data = {
            "activo": False,
            "creado": Datetime.get_current_datetime(),
            "actualizado": Datetime.get_current_datetime()
        }
        db.usuarios.update_one(
            {"_id": ObjectId(_id)},
            {"$set": data}
        )
```

Figura A.18. Clase Usuario y conexión con base de datos.

Fuente: autores.

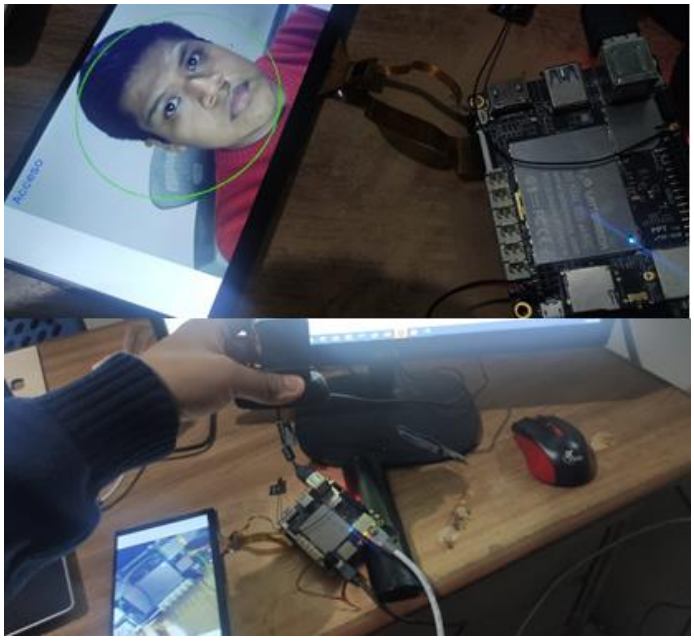


Figura A.19. Pruebas de reconocimiento facial con el dispositivo LattePanda.

Fuente: autores.

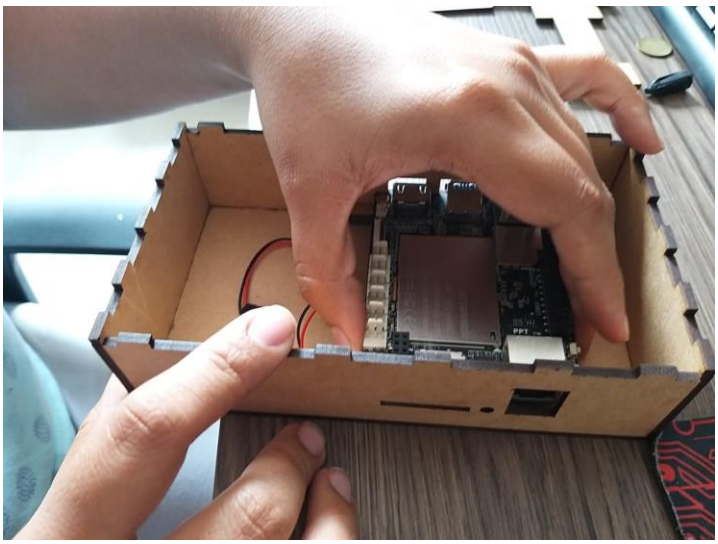


Figura A.20. Armado del case para el prototipo de reconocimiento facial.

Fuente: autores.

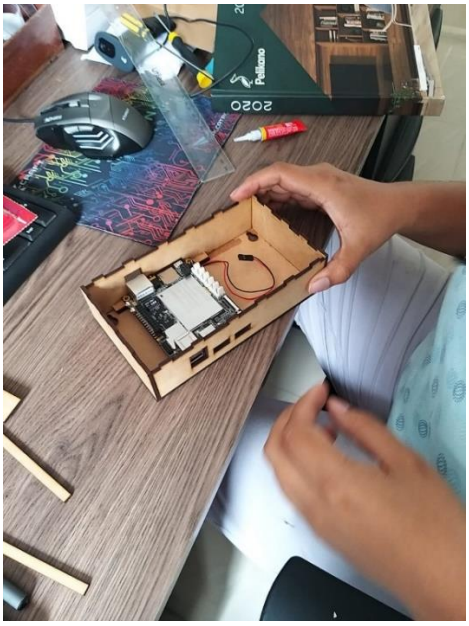


Figura A.21. Armado del case, soporte para el LattePanda.

Fuente: autores.

Figura A.22. Montaje de los componentes del prototipo.

Fuente: autores.

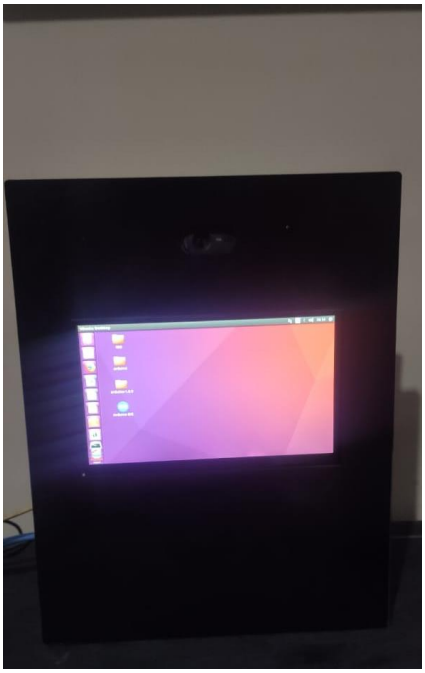


Figura A.23. Pruebas del dispositivo LattePanda una vez montado con el case.

Fuente: autores.

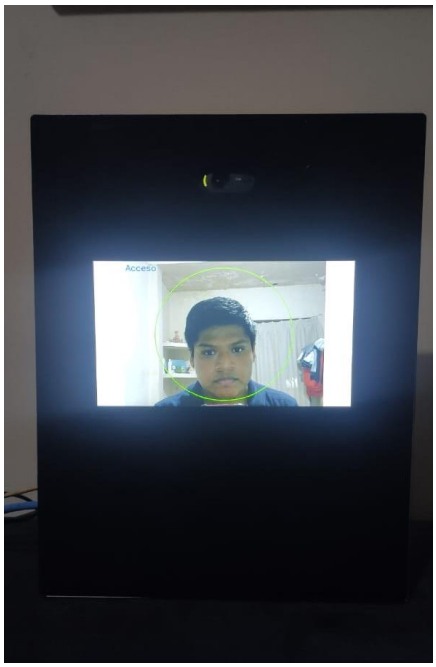


Figura A.24. Pruebas funcionales del prototipo.

Fuente: autores.