DESARROLLO DE UN SISTEMA DE SEGURIDAD BASADO EN EL RECONOCIMIENTO FACIAL PARA LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ.

Autores:
Mendoza Víctor
Falcón Gabriel

Urb. Yuma II, calle Nº 3. Municipio San Diego
Teléfono: (0241) 8714240 (máster) – Fax: (0241) 8712394

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

DESARROLLO DE UN SISTEMA DE SEGURIDAD BASADO EN EL RECONOCIMIENTO FACIAL PARA LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ.

Trabajo de grado presentado como requisito para optar al título de

INGENIERO ELECTRÓNICO E INGENIERO EN TELECOMUNICACIONES

Autores: Mendoza Víctor
C.I: V-21.476.548
Falcón Gabriel
C.I: V-23.4096.614
Tutor: MSc. López H. Jetro R.
CI.: V-8.779.723

San Diego, agosto de 2018

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELECTRÓNICA

APROBACIÓN DEL TUTOR

Quien suscribe, MSc. JETRO LOPEZ portador de la cédula de identidad N° V-8.779.723, en mi carácter de tutor del trabajo de grado presentado por los ciudadanos, GABRIEL FALCON portador de la cédula de identidad N° V-23.4096.614 y VICTOR MENDOZA portador de la cédula de identidad N° V-21.476.548, titulado **DESARROLLO DE UN SISTEMA DE SEGURIDAD BASADO EN EL RECONOCIMIENTO FACIAL PARA LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ.**, acepto la tutoría del mencionado proyecto durante su etapa de desarrollo hasta su elaboración y evaluación; según las condiciones de la Coordinación de Pasantías y Trabajo de Grado de la Facultad de Ingeniería de la Universidad José Antonio Páez y sus correspondientes reglamentos.

En San Diego, a los 30 días del mes de Julio del año dos mil dieciocho.

_____
Jetro López
C.I.: V-8.779.723

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENERÍA ELECTRÓNICA

DEDICATORIA

Gabriel Falcón

Dedico este proyecto de tesis a Dios, a mis padres y hermanos. A Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar, a mis padres, porque creyeron en mí y porque me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ustedes, hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que sienten por mí, fue lo que me hizo ir hasta el final. Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí, es por ello que soy lo que soy ahora. A mis hermanos por siempre motivarme a seguir adelante y no detenerme. A mis hermanos, tíos, primos, abuelos y amigos. Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida. Mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles. A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENERÍA ELECTRÓNICA

DEDICATORIA

Víctor Mendoza

A mis padres por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo. A mis abuelos, por quererme y apoyarme siempre, esto también se lo debo a ustedes. Por ultimo agradezco a mi hermano Miguel por su apoyo durante todo este trabajo.

Todo este trabajo ha sido posible gracias a ellos.

ÍNDICE GENERAL

INDICE DE FIGURAS

INDICE DE GRÁFICO

INDICE DE TABLA

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERIA ELECTRONICA

DESARROLLO DE UN SISTEMA DE SEGURIDAD BASADO EN EL RECONOCIMIENTO FACIAL PARA LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ.

Autores: Mendoza Víctor, Falcón Gabriel
Tutor: Ing. Miguel González
Fecha: Agosto, 2018

RESUMEN

En el trabajo de investigación, se desarrolla un sistema de seguridad basado en el reconocimiento facial para la Universidad José Antonio Páez, con la finalidad de mantener un control de acceso a la población universitaria producto de la inseguridad. Dicho sistema se realizó con alternativas mediante el uso de la tecnología SBC, el reconocimiento facial y sistemas de base de datos, los cuales permitieron reconocer si el individuo pertenece o no a la institución, los algoritmos de reconocimiento facial que se utilizaron, fueron el Haar Cascade y LBPH para luego, llevar las muestras al archivo de entrenamiento hacia la base de datos. Adicionalmente se tuvo un registro de su entrada y salida de la misma, para esto se utilizaron cámaras, donde cada una posee un identificador que transmiten imágenes que viajan mediante la intranet hacia el servidor central en donde serán analizadas. La línea de investigación es Avances Tecnológicos de Información y Comunicación, bajo la modalidad de proyecto factible con un diseño experimental y nivel de investigación descriptiva. Se aplican técnicas de recolección de datos a la muestra de 60 estudiantes, con la intención de proporcionar respuestas y poder conseguir algunos resultados en función de las interrogantes de la investigación, se siguieron cada de una las fases metodológicas para la implementación del sistema. Al culminar el desarrollo, el sistema se sometió a diferentes pruebas para determinar su rendimiento, obteniendo que en condiciones ideales del lugar, el reconocimiento y la detección arrojaron verdaderos positivos mostrando un correcto funcionamiento del sistema, a diferencia de las condiciones no ideales, los cuales fueron muy variadas dificultando la obtención de los verdaderos positivos a raíz de las condiciones de iluminación que generan falsos positivos, variando así, la eficiencia del sistema .

Descriptores: Reconocimiento facial, Sistema de seguridad.

INTRODUCCIÓN

Desde que el ser humano ha existido le ha interesado descubrir los conceptos básicos que rigen su comportamiento y funcionalidades como individuo, en ese mismo sentido, se sabe que estos se reconocen los unos a los otros a través del rostro, por lo tanto a través del desarrollo evolutivo de la humanidad, la ciencia ha sido capaz de estudiar la mediciones en el rostro por medio de la biometría facial que consta con la automatización de procesos que extraen y comparan datos para verificar a la persona de la cual se extrajeron, todo esto por medio de sistemas computacionales capaces de ejecutar todos los cálculos correspondientes y dispositivos que capten las imágenes del rostro.

Después de las consideraciones anteriores se podría pensar que enseñarle a una computadora como reconocer diferentes rostros es una tarea sencilla, pero desafortunadamente esto no es así. El reconocimiento de rostros es un problema que fue considerado desde las primeras etapas de visión por computadora. Este problema ha sido estudiado más a fondo en los últimos años, gracias a los avances del poder computacional combinado con diferentes ramas de la matemática que han permitido diseñar algoritmos más complejos, utilizando diferentes técnicas como el uso de matrices y códigos binarios.

Los nuevos algoritmos diseñados con el pasar del tiempo en el área del reconocimiento facial, han surgido, haciendo posible así a una máquina de poder detectar una cara y analizar para su reconocimiento utilizando un entrenamiento previo del mismo. En la actualidad han surgido diversos dispositivos conocidos como SBC (*Single Board Computer*) que han permitido implementar nuevos diseños y prototipos que utilicen el reconocimiento facial ya sea en la robótica, domótica y otras aplicaciones.

En el primer capítulo, el problema, engloba el planteamiento del problema y de los objetivos, la justificación del trabajo realizado y el alcance del mismo. En el segundo

capítulo, marco teórico, se pretende contextualizar la investigación, a través de la exposición de los factores que delimitan el tema en cuestión. En el tercer capítulo, marco metodológico, se explica desde el punto de vista teórico la metodología de trabajo empleada para el desarrollo de la investigación en cuanto a actividades, productos esperados por cada etapa y técnicas empleadas. En el cuarto capítulo, se describen los procedimientos llevados a cabo para alcanzar los objetivos y cumplir con ellos, a través de las pruebas y los resultados del sistema y en el capítulo 5 se indican las conclusiones objetivo por objetivo y las recomendaciones.

CAPÍTULO I

EL PROBLEMA

1.1 Planteamiento del Problema

La inseguridad se encuentra actualmente en todo el mundo, y se ha convertido en un problema de salud pública, a consecuencias de su elevado crecimiento en los últimos años. En los países de América Latina la inseguridad es un reto compartido y un obstáculo para el desarrollo social y económico, desde este punto de vista las fuentes son multifacéticas y entre las principales se han destacado las provenientes de los procesos de globalización que han sido acompañados por la declinación de la protección del empleo, por resultados imprevistos de las crisis foráneas y por la volatilidad de los flujos de capital, e inestabilidades macroeconómicas, además de la debilidad de las instituciones para enfrentar esos riesgos. Sin duda alguna las causas más directas de la inseguridad se encuentran el elevado número de la violencia de diversos tipos y su expresión en la inseguridad creciente de la vida cotidiana de los ciudadanos.

Con lo descrito anteriormente se manifiesta que la seguridad ciudadana se expone con frecuencia, por ende, se debe atender a las causas estructurales que originan los problemas delincuenciales, sin embargo, los mismos ciudadanos buscan la forma de cuidarse por sí mismos. Algunos países han buscado la forma de incidir en algunas de las causas de la inseguridad a través de la recuperación de espacios públicos y recreativos, así como también un mejor alumbrado en las calles y programas de infraestructura urbana, que generará empleos para jóvenes en situación vulnerable.

En Venezuela, en referencia a las consideraciones anteriores, la delincuencia también ha venido incrementando de manera exponencial hasta llegar a creer que ningún sitio es seguro, esto incluye a la Universidad José Antonio Páez en donde la inseguridad dentro de la casa de estudios, ha llevado a que las autoridades responsables hayan implementado nuevas medidas las cuales no han dado los resultados esperados. Esto conlleva a que aún no se dispone de un sistema de seguridad viable, trayendo como consecuencia a la población estudiantil estar atentos y precavidos ante cualquier incidente, porque temen ser atracados dentro de su propia institución.

Las últimas medidas de seguridad que implementaron, fue la portación del carnet al momento de ingresar a las instalaciones y una revisión parcial del contenido de los bolsos en caso de no poseer vehículo, siendo esto innecesario debido a que situaciones pasadas, los mismos integrantes del cuerpo de seguridad se veían involucrados con eventualidades que ocurrirán. Gracias a los avances tecnológicos existen sistemas de seguridad que hacen uso del reconocimiento facial y otras áreas del procesamiento digital de imágenes para disminuir estos problemas, deben ser pocas las empresas en el país que lo utilicen para autenticar a algún usuario, siendo que esta herramienta es una de las más segura y con los años se ha ido disminuyendo la taza de error, pero debido a su gran costo de implementación, instalación y mantenimiento tanto en *software* y *hardware* no es una vía rentable, debido a la situación económica que atraviesa el país.

Estos sistemas de reconocimiento facial trabajan con el procesamiento digital de imágenes el cual utiliza diversas técnicas de Machine Learning en conjuntos con modelos matemáticos para realizar su función. Gracias a esto se han desarrollado distintos algoritmos de reconocimiento facial las cuales se han podido utilizar en diversos sistemas de seguridad ya que posee la libertar de reconocer diversos aspectos del rostro de una persona.

A su vez mediante el uso de los SBC (*Single BoardComputers*- Computadores de Placa Reducida) las cuales funcionan como computadoras, contenido en una

tarjeta de única placa el cual es comúnmente utilizado como controladores o interfaces de otros dispositivos, un ejemplo de estos dispositivos es el Raspberry Pi los cuales son de bajo costo y permiten realizar proyectos de investigación en diversas áreas. En la escuela de ingeniería electrónica y telecomunicaciones, pensando en la seguridad de los estudiantes y de sus laboratorios los cuales poseen equipos bastante costosos imposibilitando su recuperación en caso de hurto, plantea una propuesta de un sistema de seguridad, para esto se hará uso de un SBC, en nuestro caso un Raspberry Pi Modelo B junto a las técnicas desarrolladas en el área del procesamiento digital de imágenes que utilice el reconocimiento facial en las entradas que posea la Universidad José Antonio Páez.

1.2 Formulación del Problema.

Considerando la situación antes planteada, se formula la siguiente interrogante: ¿Cómo se puede solventar el problema de la inseguridad en la Universidad José Antonio Páez?

1.3 Objetivos de la Investigación.

1.3.1 Objetivo General.

Desarrollar un sistema de seguridad basado en el reconocimiento facial para la Universidad José Antonio Páez con la finalidad mantener un control de acceso de la población Universitaria.

1.3.2 Objetivos Específicos.

- Determinar los requerimientos funcionales y no funcionales del sistema, mediante técnicas de recolección de datos.
- Diseñar el sistema de reconocimiento facial según la metodología XP.
- Desarrollar la estructura del sistema de seguridad con la ayuda de un Raspberry Pi y técnicas biométricas.

1.4 Justificación de la Investigación.

En las instituciones universitarias la seguridad es un tema crucial que deben ofrecer en sus instalaciones tanto a los estudiantes como a los equipos albergados en ella, en los últimos años la inseguridad dentro de las instalaciones se ha incrementado afectando tanto a los estudiantes como a la reputación de la propia universidad y el rápido incremento de la inflación del país se ha vuelto difícil implementar nuevas medidas de seguridad que satisfagan con los requisitos la situación se ha agravado al punto de ser preocupante.

Debido a que en el contexto actual del país asegurar un comportamiento normal no siempre es posible debido a que en ocasiones los cuerpos de seguridad se ven involucrados en los actos delictivos dentro de las instalaciones, los estudiantes de la escuela de ingeniería electrónica y telecomunicaciones propondrán un sistema de seguridad basado en el reconocimiento facial y una base de datos previamente cargada de la información de las personas autorizadas a permanecer en las instalaciones utilizando un SBC como es el Raspberry Pi.

Desde el punto de vista práctico, al implementar un sistema de seguridad con reconocimiento facial se obtendrá un control más estricto de las personas que ingresan a las instalaciones contando así con un nuevo registro donde se observó tanto el ingreso como la salida de la misma, además el fácil y rápido acceso a la universidad que beneficio a toda la población, sin necesidad de detenerse a mostrar el carnet u otro tipo de identificación.

1.5 Alcance.

La propuesta planteada va dirigida a todo tipo de organización que desee un sistema de seguridad eficiente utilizando el reconocimiento facial, el objetivo primario va orientado hacia la Universidad José Antonio Páez como una propuesta factible y económica a ser implementada en sus instalaciones contando así con una cámara encargada para la vigilancia, con la capacidad de utilizar múltiples cámaras conectadas a un módulo principal para un control más amplio obteniendo así un circuito de cámaras cerrado. El desarrollo de la investigación estará limitado a

realizarse en tiempo máximo de 3 meses. La investigación tendrá un presupuesto de aproximadamente cuatrocientos de BsS sujeto a cambios para los gastos en materiales y equipos el cual estará limitado a la disponibilidad de estos en el país.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes de la Investigación.

Para la elaboración de esta investigación, se hizo necesario revisar estudios anteriores que estuviesen relacionados con el tema tratado actualmente. Un antecedente, es todo aquel trabajo que precede a la investigación que se está realizando y contribuyen en cierta manera con la misma. Entonces, algunos de ellos son:

González M., Infante. M., (2017) en el trabajo titulado **"Construcción de un prototipo de robot social basado en un miniordenador raspberry pi 3 y arduino"** para optar por el título de Ingeniero Electrónico en la Universidad José Antonio Páez (Venezuela), donde presentan emular características de robots sociales ya elaborados por grandes empresas como Google, Microsoft y Amazon, con el fin de facilitar las labores cotidianas y repetitivas del usuario, siendo capaz de tomar decisiones sencillas como encender una bombilla mediante comandos de voz hasta hacer un recordatorio de las actividades planificadas en la agenda del usuario. En este trabajo se utilizaron lenguajes de programación como C++, phyton y JavaScript para finalmente realizar las pruebas de la implementación del robot. Dentro de las recomendaciones se hace referencia a la adaptación del reconocimiento facial de manera de generar interacción entre el usuario y el prototipo, tomando esto como iniciativa y referencia a nuestro sistema, puesto que esta tecnología ha transformado campos tales como la medicina, la seguridad, el marketing y otros, incorporando el procesamiento de imágenes como una valiosa fuente de información.

C**ajas M. y Viri P. (2017)** en el trabajo **titulado "Diseño e implementación de un sistema de seguridad vehicular mediante reconocimiento facial a través de visión artificial"** para optar por el título de Ingeniero Mecánico Automotriz realizado en la Universidad Politécnica Salesiana (Ecuador) donde presentan lograr evitar el

robo de vehículos, para brindar tranquilidad a los propietarios al saber que solo las personas autorizadas pueden hacer el uso del vehículo, a un bajo costo de su implementación. Se utilizaron materiales electrónicos, el lenguaje de programación phyton y las librerías del software libre OpenCV para el procesamiento de imágenes y visión computarizada a través de un computador de placa reducida Raspberry Pi 3, además utilizaron una cámara de tipo mini-webcam USB que mostro bueno resultados en solo en ambientes iluminados. Durante la implementación se realizaron distintas pruebas que dieron buenos resultados, apuntando a tener un fácil manejo para cualquier persona bajo una previa instrucción de su funcionamiento, un mejor control de utilización del automotor ya que fue eficiente y cumplió con las expectativas del propietario. Este trabajo se relaciona con la investigación planteada, ya que muestra cómo debe estructurarse un sistema de seguridad mediante reconocimiento facial y además indica que el algoritmo de Histogramas de Patrones Binarios (LBPH) es el que mejor tiene mayor rendimiento en detección debido a los ajustes en variaciones de iluminación.

En el trabajo de **Espinoza D. y Jorguera P. (2015)** en el trabajo titulado **"Reconocimiento facial"** para optar por el título de Ingenieros de Ejecución en Informática de la Universidad católica de Valparaíso (Chile), donde presentan Implementar un software de reconocimiento facial por medio de un lenguaje de programación utilizando librerías de Open CV, en dicho trabajo indican la comprensión del lenguaje Open CV en el ámbito de reconocimiento de patrones, algoritmos de reconocimiento facial y evaluación de detección de rostro para observar cual tiene mejor funcionamiento para implementarse. Utilizaron características de programación web como OpenCV y EmguCV para el desarrollo del sistema, posteriormente se explica el diseño y funcionamiento del reconocimiento facial. Esta investigación ayudó a conocer y comprender las ventajas y desventajas de utilizar Open CV, en el ámbito de reconocimiento de patrones y entender sus librerías para

aplicarlas en el uso del reconocimiento facial, trayendo como beneficios al proyecto de elegir cual se adaptaría mejor al software a implementar.

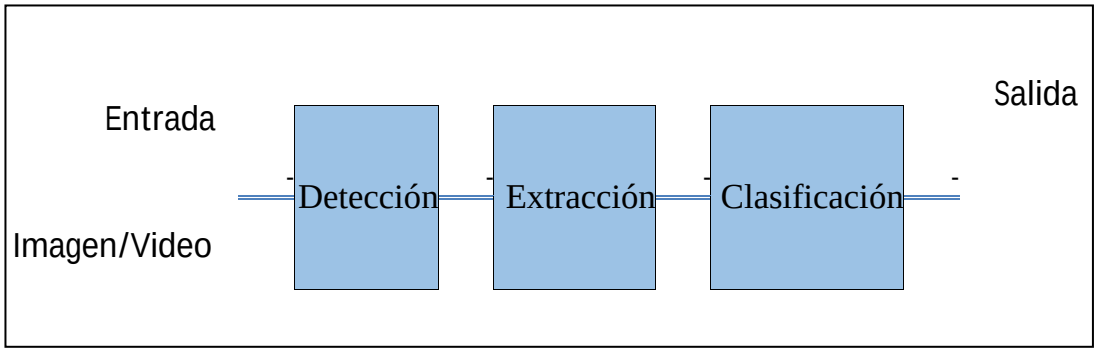
2.2 Bases teóricas.

En este capítulo se describió totalmente las bases teóricas sobre las cuales se fundamentó la investigación, de tal forma que se obtuvo una comprensión integra de cada una de las técnicas y herramientas utilizadas. Se puntualizó el proceso de reconocimiento facial desde un punto de vista general hasta un punto de vista mucho más específico, se consideró cada una de las etapas, técnicas y métodos del mismo que se utilizaron para la implementación. Luego se describió las características utilizadas en las herramientas de hardware y software.

2.2.1 Reconocimiento Facial.

Es una solución biométrica que emplea un algoritmo automático para verificar o reconocer la identidad de una persona en función de sus características fisiológicas extraídas de una imagen o de un fotograma clave de una fuente de video y luego son comparadas mediante una base de datos. Los sistemas de reconocimiento facial comprenden varios sub-problemas a resolver, como detección, extracción y clasificación, como se muestra en la Figura 1. Para que el proceso de reconocimiento sea realizado, es necesario pasar por una etapa de detección facial y éste es el primer paso para realizar el reconocimiento facial en una imagen o video.

Figura 1. Etapas del reconocimiento facial



Fuente: Falcón y Mendoza (2018).

Una vez que se obtienen los datos de la imagen o video que es tomado como entrada en el sistema se procede al proceso de detección en donde se extraen características de las distintas regiones del rostro mediante la captura de una o varias imágenes, luego vendría el proceso de extracción de características que se obtienen detalles faciales importantes de los datos de entrada así como el espaciado entre los ojos y el arco de las cejas y por último el proceso de clasificación donde se hace el uso de alguna base de datos para identificar a la persona. Estas etapas podrían ser manipuladas o mezcladas para así mejorar su factibilidad a la salida del sistema.

2.2.1.1 Reconocedores de rostros OpenCV.

Mediante OpenCV, la codificación del reconocimiento facial ahora es mucho más sencilla, puesto que se crearon tres reconocedores los cuales se pueden utilizar, simplemente cambiando una sola línea de código. Hay tres sencillos pasos para el reconocimiento facial de codificación de computadora, que son similares a los pasos que utilizan los seres humanos reconocer rostros. Estos pasos son:

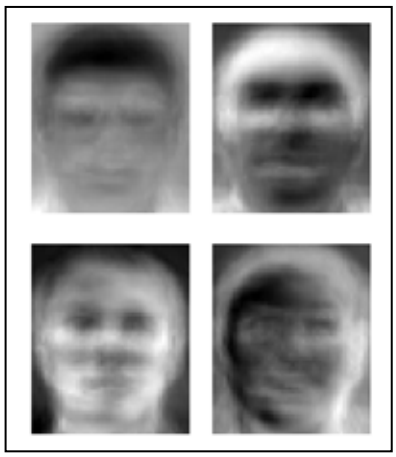
- Recolección de datos: Reúna los datos de la cara (imágenes de la cara en este caso) de las personas que desea identificar.
- Entrena al reconocedor: alimenta los datos de la cara y los nombres respectivos de cada cara al reconocedor para que pueda aprender.
- Reconocimiento: Alimente nuevas caras de esa gente y vea si el reconocedor facial que acaba de entrenar los reconoce.

2.2.1.1.1 Eigenfaces.

Este algoritmo considera el hecho de que no todas las partes de una cara son igualmente importantes o útiles para el reconocimiento facial. De hecho, cuando miras a alguien, reconoces a esa persona por sus rasgos distintivos, como los ojos, la nariz, las mejillas o la frente; y cómo varían el uno con el otro. En ese sentido, en las áreas de cambio máximo. Por ejemplo, desde los ojos hasta la nariz hay un cambio significativo, y lo mismo se aplica desde la nariz hasta la boca. Cuando miras varias

caras, las comparas mirando estas áreas, porque al captar la máxima variación entre caras, te ayudan a diferenciar una cara de la otra. De esta forma, es cómo funciona el reconocedor EigenFaces. Mira todas las imágenes de entrenamiento de todas las personas como un todo e intenta extraer los componentes que son relevantes y útiles y descarta el resto. Estas características importantes se llaman componentes principales como se muestran en la Figura 2.

Figura 2. Componentes principales de Eingenface.

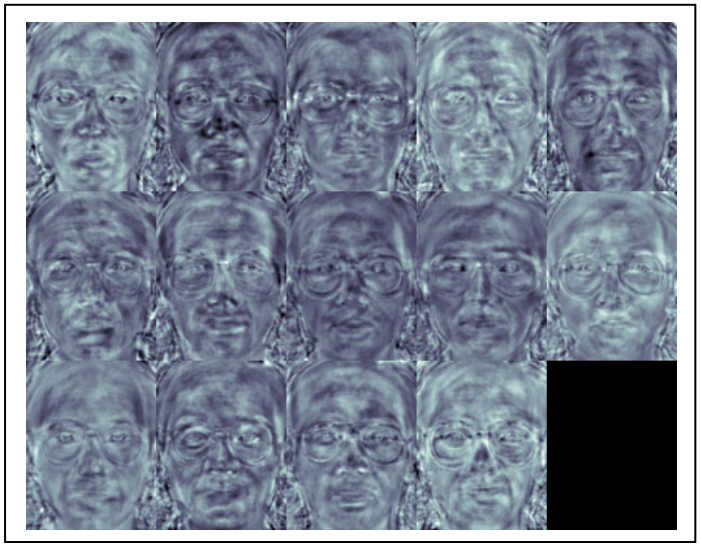


Fuente: https://docs.opencv.org/2.4/index.html (2011).

2.2.1.1.2 Fisherfaces.

Es una versión mejorada del último. Como se pudo ver, EigenFaces examina todas las caras de entrenamiento de todas las personas a la vez y encuentra los componentes principales de todas ellas combinadas. Al hacer eso, no se enfoca en las características que discriminan a un individuo de otro. En cambio, se concentra en los que representan todas las caras de todas las personas en los datos de entrenamiento, como un todo. El algoritmo reconocedor de caras de FisherFaces extrae los componentes principales que diferencian a una persona de las otras. En ese sentido, los componentes de un individuo no dominan sobre los demás. A continuación, se muestra la figura 3 los componentes principales utilizando el algoritmo de FisherFaces.

Figura 3. Componentes principales de Fisherfaces.



Fuente: https://docs.opencv.org/2.4/index.html (2011).

2.2.1.1.3 Histogramas de Patrones Binarios Locales (LBPH).

El proceso de este algoritmo viene dado en tomar una ventana de 3 × 3 y moverla a través de una imagen. En cada movimiento (cada parte local de la imagen), se compara el píxel en el centro, con sus píxeles circundantes. Denota a los vecinos con un valor de intensidad menor o igual que el píxel central en 1 y el resto en 0. Después de leer estos valores 0/1 debajo de la ventana 3 × 3 en el sentido de las agujas del reloj, tendrá un patrón binario como 11100011 que es local en un área particular de la imagen. Cuando termine de hacer esto en la imagen completa, tendrá una lista de patrones binarios locales.

Luego que se tiene una lista de patrones binarios locales, se convierte cada uno en un número decimal usando la conversión de binario a decimal y por último se realiza un histograma para cada cara en el conjunto de datos de entrenamiento. Esto significa que, si hubiera 100 imágenes en el conjunto de datos de entrenamiento, entonces LBPH extraerá 100 histogramas después del entrenamiento y los almacenará

para su posterior reconocimiento. El algoritmo también realiza un seguimiento de qué histograma pertenece a qué persona.

2.2.1.2 Detección Facial.

El proceso de detección abarca el método de análisis de características de la imagen con el propósito de identificar la localización de una cara o un conjunto de caras, por tanto, puede tornarse complicado debido a que depende de múltiples variables que no son sencillas de controlar, así como la expresión facial, posición e intensidad de luz, oclusión, orientación de la imagen, posición e inclinación de la cara. La detección de rostros por medio de computadoras puede ejecutarse de diferentes maneras y generan grandes complicaciones.

Existen aplicaciones de detección facial en donde no se requiere realizar el proceso de detección debido a que las imágenes almacenadas en la base de datos pueden estar normalizadas, esto quiere decir que poseen un formato de imagen de entrada estándar. Sin embargo, generalmente la entrada de los sistemas comunes de reconocimiento facial es una imagen que contiene una cantidad desconocida de rostros.

El mismo caso se presenta cuando se quiere desarrollar un sistema de seguimiento facial, donde se debe ubicar la posición de cada cara. Los métodos utilizados para este proceso cambian de acuerdo a su modo los cuales serían: autovalores y autovectores, redes neuronales, máquinas de soporte vectorial y análisis de las componentes principales. Hay factores los cuales son de suma importancia a la hora de la captura de imágenes que presentan algunas limitaciones o inconvenientes como pueden ser:

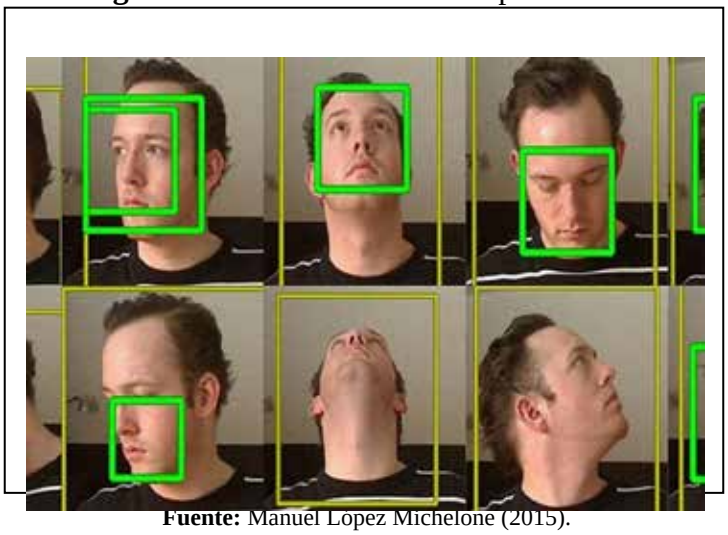
- Escala: En una imagen se puede mostrar un conjunto de diferentes escalas de rostros como se puede distinguir en la Figura 4, en el que el tamaño o escala de un rostro puede ser manejado por un proceso de cambio de tamaño por deformación. Este procedimiento de transformación requiere la ubicación de algunos puntos característicos como los ojos, la nariz y la boca.

Figura 4. Rostros en diferentes escalas.

Fuente: http://www.ociolaspalmas.com.(2017)

- Pose: El rendimiento de los sistemas de detección de rostro se ven afectados cuando hay variaciones en la posición de la cara (Figura 5) debido a que el escenario ideal para la detección es en el que sólo estuviesen involucradas imágenes de frente al sujeto ya que en su totalidad los trabajos están hechos para la detección de rostros de forma frontal.

Figura 5. Rostro con diferentes poses.



Fuente: Manuel López Michelone (2015).

- Iluminación: Los problemas de iluminación se observan en la Figura 6 en la cual el mismo rostro con la misma expresión facial y vista desde el mismo punto de vista, pueden afectar la calidad de la imagen de entrada, perjudicando la apariencia del rostro.

Figura 6. Rostros con diferente iluminación

Fuente: Sara Leblanc (2018).

- Expresión Facial: Las características faciales pueden variar drásticamente la geometría el rostro mediante diferentes gestos así como se muestra en la Figura 7.

Figura 7. Rostro con diferentes expresiones

Fuente: Josefina Léniz (2014).

- Oclusión: Es causada por la presencia de elementos como barba, lentes o sombreros, es decir que las caras pueden estar parcialmente cubiertas por objetos, lo que se

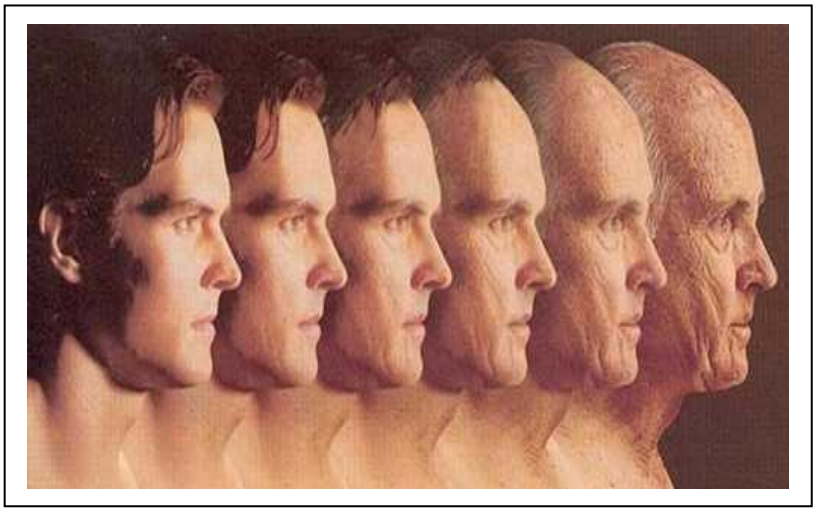
traduce en pérdida de características como se muestra en la Figura 8. Existen diferentes algoritmos para poder resolver este problema.

Figura 8. Rostros en Oclusión.

Fuente: https://www.lasexta.com (2018).

- Envejecimiento: A medida que pasan los años las personas tienen cambios significativos, así como se contempla en la Figura 9, para ello bastaría con una actualización periódica de la base de datos de la aplicación.

Figura 9. Rostro de la misma persona con edades diferentes.



Fuente: https://www.tispain.com/2015/04/sabias-que-nuestra-edad-esta-reflejada.html (2015).

2.2.1.2.1 Estructura del problema de detección facial.

El problema de la detección de manera global se divide para ser comprendido en distintas áreas del proceso. Dentro del sistema se encuentran la detección e ubicación de rostros y al mismo momento luego realizar el rastreo. De manera general este proceso se ejecuta primeramente con la realización de un tipo de reducción de dimensionalidad para alcanzar la respuesta en un tiempo considerable, adicionalmente se procede a extraer las muestras más relevantes de las regiones faciales para realizar las mediciones y sustraer las características. Para concluir se analizan todos los datos suministrados para la verificación respectiva de cada rostro para ser identificado.

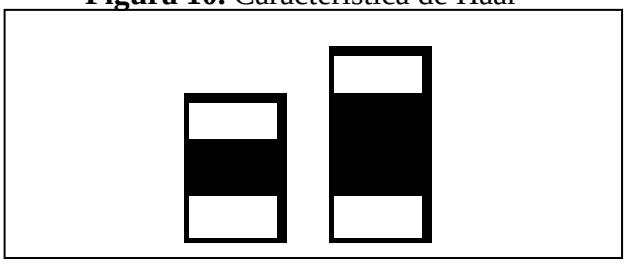
2.2.1.2.2 Clasificadores.

Un programa de computadora que decide si una imagen es positiva (imagen de la cara) o negativa (imagen sin la cara) recibe el nombre de clasificador. Un clasificador está capacitado en cientos de miles de imágenes faciales y no faciales para aprender a clasificar correctamente una nueva imagen. OpenCV nos proporciona dos clasificadores preformados y listos para ser utilizados para la detección de rostros:

- Clasificador Haar: Es un enfoque basado en el aprendizaje automático, un algoritmo creado por Paul Viola y Michael Jones; que (como se mencionó anteriormente) están entrenados a partir de muchas imágenes positivas (con caras) e imágenes negativas (sin caras). El algoritmo necesita extraer características, denominadas HaarFeatures, a partir de un valor obtenido por la diferencia de la suma de los píxeles de contraste claro con la de los píxeles con contraste oscuro. Todo esto es calculado utilizando una representación intermedia de la imagen, llamada imagen integral. Esto crea grupos de píxeles, que son usados para determinar las zonas relativamente claras y oscuras. El método define entonces una característica de Haar como dos o

tres grupos adyacentes con varianza de contraste relativamente alta entre ellos. En la Figura 10se observa una característica de Haar

Figura 10. Característica de Haar



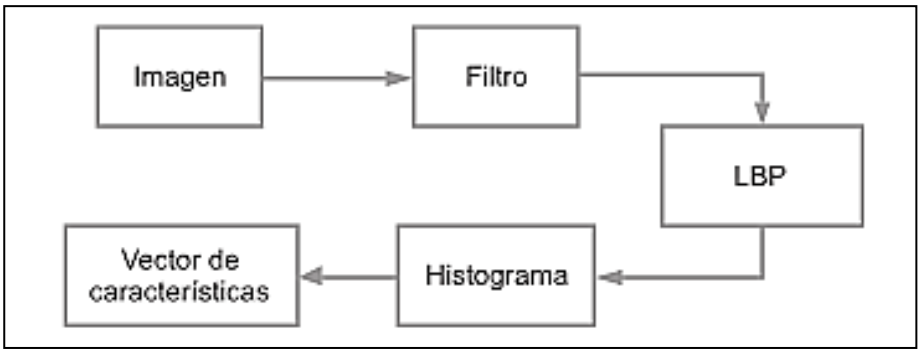
Fuente: https://docs.opencv.org/2.4/index.html (2011)

- LBP (Patrones Binarios Locales): Es un descriptor visual / de textura, y afortunadamente, nuestras caras también están compuestas de patrones micro visuales. Por lo tanto, las características de LBP se extraen para formar un vector de características que clasifica una cara de una que no es cara. El funcionamiento de LBP en que cada imagen de entrenamiento se divide en algunos bloques donde para cada bloque, LBP mira a 9 píxeles (ventana de 3 × 3) a la vez, y con un interés particular en el píxel ubicado en el centro de la ventana. Luego, compara el valor de píxel central con el valor de píxel de cada vecino debajo de la ventana de 3×3. Para cada píxel vecino que sea mayor o igual que el píxel central, establece su valor en 1 y para los demás los establece en 0.

Después de eso, lee los valores de píxel actualizados (que pueden ser 0 o 1) en el sentido de las agujas del reloj y forma un número binario para convertirlo en un número decimal, ese número decimal es el nuevo valor del píxel central y se hace para cada píxel en un bloque. Seguidamente se pasa a convertir cada valor de bloque en un histograma, por lo que se obtiene un histograma para cada bloque en una imagen y finalmente se concatena los histogramas de bloque para formar un vector de función para una imagen, que contiene todas las características que interesan, entonces es así como extraemos

las características de LBP de una imagen. A continuación, se visualiza en la Figura 11 el diagrama de flujo de LBP.

Figura 11. Diagrama de flujo de LBP



Fuente: https://www.researchgate.net (2010)

2.2.1.3 Extracción de características.

Los seres humanos tienen la capacidad de grabarse el rostro de las personas de diversas maneras, ya sea por algún rasgo que posea esa persona la cual se asocie para ser identificado. Aunque el reconocimiento de los rostros resulta natural para los seres humanos, dotar a una máquina de esta habilidad resulta bastante difícil. La extracción de característica consiste en la recolección de una o más fotografías para luego obtener información trascendental de estas.

Hay muchos algoritmos de extracción de características, que en su mayoría son usados en otras áreas distintas al reconocimiento de rostros. Para cumplir con sus objetivos, los investigadores especializados en computación visual han usado, modificado y adaptado muchos algoritmos y métodos ya existentes, como PCA (Principal ComponentsAnalysis - Análisis de Componentes Principales) el cual consiste en analizar una tabla de datos en la cual, las observaciones son descritas por muchas variables cuantitativas correlacionadas y dependientes [3], de esta forma se obtiene la información más relevante de la tabla. PCA puede hacer predicción, eliminación de redundancia, extracción de características, comprensión de la data.

2.2.1.3.1 Métodos para la extracción de rostros.

En la extracción de características de rostros existen los siguientes métodos:

- Métodos basados en el color.
- Técnicas basadas en planilla.
- Técnicas basadas en geometría.
- Aproximación basada en apariencia.

2.2.2 Herramientas para el Sistema.

2.2.2.1 Base de Datos.

Una base de datos (DB), en el sentido más general, es una colección organizada de datos. Más específicamente, una base de datos es un sistema electrónico que permite acceder, manipular y actualizar fácilmente los datos. En otras palabras, una organización utiliza una base de datos como método de almacenamiento, gestión y recuperación de información. Las bases de datos modernas se gestionan utilizando un sistema de gestión de bases de datos (DBMS). Los programadores de software conocen bien los conceptos de bases de datos a través de bases de datos relacionales como Oracle, SQL SERVER y MySQL, etc. Típicamente, una estructura de base de datos almacena datos en un formato tabular.

La arquitectura de la base de datos puede ser externa, interna o conceptual. El nivel externo especifica la forma en que cada tipo de usuario final comprende la organización de sus correspondientes datos relevantes en la base de datos. El nivel interno se ocupa del rendimiento, la escalabilidad, el costo y otros asuntos operativos. El nivel conceptual unifica perfectamente las diferentes vistas externas en una vista definida y totalmente global. Consiste en todos los datos genéricos requeridos por el usuario final.

2.2.2.1.1 Modelo Entidad Relación.

Es un modelo de datos que consiste en un conjunto de objetos básicos llamados Entidades y sus respectivas relaciones entre sí, sirve para diseñar el esquema de la base de datos antes de desarrollarla. Para construir un modelo entidad relación debemos identificar las entidades que harán parte de nuestro diagrama o modelo, con el fin de establecer los tipos de datos y las relaciones existentes entre ellas, veamos cuales son:

- Entidades: Representan cosas u objetos que incluso del mismo tipo son diferentes entre sí, refleja algo del mundo real (También puede ser abstracto).
- Relaciones: Las relaciones se representan con un Rombo, su finalidad es asociar una entidad con otra o consigo misma (Reflexiva)
- Relación reflexiva: Las relaciones reflexivas son aquellas que hacen referencia a sí mismas.
- Atributos: Se representan con una elipse y en la mayoría de los casos son propiedades de una entidad.
- Clave Primaria: Es aquel atributo que identifica de forma única a las entidades.
- Cardinalidades: Indican el número de entidades con las que se puede relacionar una entidad.

2.2.2.1.2 MySQL

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS) basado en Structured Query Language (SQL). MySQL se ejecuta en prácticamente todas las plataformas, incluidos Linux, UNIX y Windows. Aunque se puede utilizar en una amplia gama de aplicaciones, MySQL suele asociarse con aplicaciones basadas en web y publicaciones en línea, y es un componente importante de una pila empresarial de código abierto llamada LAMP. Las características de MySQL son las siguientes:

- Es un sistema de gestión de base de datos.

- Las bases de datos son relacionales.

- El software es de código abierto.

- El servidor de base de datos es muy rápido, confiable, escalable y fácil de usar.

- El servidor funciona en sistemas cliente / servidor o integrados.

2.2.2.1.3 PhpMyAdmin.

Es una herramienta de software libre escrita en PHP que está destinada a manejar la administración de un servidor de base de datos MySQL o MariaDB. Puede usar phpMyAdmin para realizar la mayoría de las tareas de administración, incluida la creación de una base de datos, la ejecución de consultas y la adición de cuentas de usuario.

2.2.2.2 Pip.

Es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Muchos paquetes pueden ser encontrados en el Python Package Index (PyPI). Python 2.7.9 y posteriores (en la serie Python2), Python 3.4 y posteriores incluyen pip (pip3 para Python3) por defecto.

2.2.2.3 Microframework.

Es un término usado para referirse al marco de aplicación web minimalista. Se contrasta con el marco full-stack también llamado frameworks empresariales. La idea es mantener el núcleo simple pero extensible. Carece de la mayor parte de la funcionalidad que es común esperar en un marco completo de aplicaciones web, como:

- Cuentas, autenticación, autorización, roles, etc.
- Abstracción de base de datos a través de un mapeo relacional de objetos
- Validación de entrada y saneamiento de entrada.
- Motor de plantilla web.

2.2.2.3.1 Flask.

Es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2 y tiene una licencia BSD.

2.2.3 Dispositivos Electrónicos.

2.2.3.1 Cámaras digitales.

Una cámara digital es un dispositivo electrónico generalmente portátil que contiene una superficie fotosensible que graba imágenes a través de un lente, para posteriormente ser visualizadas y/o procesadas en un computador. Por lo general las cámaras digitales permiten registrar no solo fotos, sino también videos y sonidos. La calidad de los documentos generados suele medirse de acuerdo a la cantidad de píxeles, aunque hay otros factores que deben tenerse en cuenta (como la sensibilidad y el tamaño del sensor).

2.2.3.2 Cámaras Web.

Una cámara web es una cámara de video que transmite cuadros en tiempo real a través de un computador hacia la red. Generalmente las cámaras web están conectadas a través de un cable USB. Utilizan sensores CMOS o CCD y proveen video en resoluciones que van desde VGA a 30 cuadros por segundo, aunque esta propiedad depende de la tecnología empleada en la cámara en cuestión.

Las cámaras compactas, bridge y réflex, si bien proveen una calidad de imagen bastante alta, no pueden ser utilizadas para el desarrollo de sistemas de vigilancia, ya que por naturaleza no están diseñadas para permanecer activas por demasiado tiempo, además de que en muchos casos no proveen una API para acceder a sus cuadros en tiempo real, por lo que se vuelven inútiles para llevar a cabo investigaciones de este tipo.

Por otra parte, las cámaras web (incluídas las IP) están diseñadas para transmitir video por largos períodos de tiempo y generalmente pueden hacerlo en tiempo real

hacia un computador. OpenCV es compatible principalmente con cámaras web con conexión USB, aunque también provee una interfaz para conectarse a cámaras IP a través de la red. Por estas razones, se ha decidido utilizar cámaras web e IP en la investigación actual.

Una cámara IP (Internet Protocol - Protocolo de Internet) es una cámara de video digital utilizada en sistemas de vigilancia, con la capacidad nativa de enviar y recibir data a través de una red de forma nativa. Pueden ser clasificadas como cámaras web, añadiendo la funcionalidad anteriormente descrita. La conexión se realiza generalmente utilizando el protocolo IEEE 802.3 o el IEEE 802.11 para que la data pueda ser transmitido en la red.

2.2.3.3 Raspberry Pi.

Es un miniordenador de placa reducida de bajo costo. Este puede ser utilizado en proyectos de electrónica, y para muchos de las cosas que hace el PC de escritorio, como hojas de cálculo, procesamiento de textos, navegación por internet, y jugar. También reproduce vídeo de alta definición. Por lo que podemos hacer notar que es un ordenador de tamaño bastante reducido, siendo esta una de sus mayores ventajas, debido a la comodidad que oferta a la hora de realizar un trabajo bastante compacto y presentarlo de forma adecuada, gracias a sus grandes características de procesamiento se pueden procesar datos y almacenarlos de manera bastante efectiva.

2.2.4 Lenguajes de Programación.

Arias M. (2008), define los lenguajes de programación como “aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucción, operadores y reglas de sintaxis, que pone a disposición del programador para que este pueda comunicarse con los dispositivos de hardware y software existentes”. Siendo el lenguaje de programación la herramienta utilizada.

2.2.4.1 Python.

Raúl González Duque (2009), define Python como “un lenguaje de programación creado por Guido Van Rossum a principios de los años 90. Trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado,

multiplataforma y orientado a objetos. Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados)". Es una herramienta de programación multiparadigma, ya que es capaz de soportar la programación orientada a objetos y la programación funcional.

2.2.4.2 OpenCV.

OpenCV viene de las siglas Open SourceComputerVision Library, es una librería abierta desarrollada por Intel en el año 1999, contiene alrededor de 500 funciones. Esta librería proporciona un alto nivel de funciones para el procesado de imágenes. Algunas de las características que permite OpenCV son operaciones básicas, procesamiento de imágenes, análisis estructural, análisis de movimiento, reconocimiento del modelo, reconstrucción 3D, calibración de cámara.

CAPÍTULO III
MARCO METODOLÓGICO

En el marco metodológico se definirán los pasos y procedimientos que habría que seguir con el fin de garantizar y cumplir con los objetivos establecidos en cada etapa de la investigación. Durante este capítulo se presentarán el tipo y diseño de investigación, técnicas utilizadas para la recolección de datos y todas las fases necesarias basadas en la metodología xp para el desarrollo del sistema.

3.1 Tipo de Investigación.

Esta investigación se clasificó como proyecto factible, ya que el objetivo principal era solventar el problema de la inseguridad dentro de la Universidad José Antonio Páez, para esto se propuso un sistema de seguridad basado en el reconocimiento facial haciendo uso de diversas tecnologías y programas, el cual fue puesto a prueba garantizando así su funcionamiento y efectividad.

Según de Moya, Renie (2004), un proyecto factible “consiste en un conjunto de actividades vinculadas entre sí, cuya ejecución permitirá el logro de objetivos previamente definidos en atención a las necesidades que pueda tener una institución o un grupo social en un momento determinado. Es decir, la finalidad del proyecto factible radica en el diseño de una propuesta de acción dirigida a resolver un problema o necesidad previamente detectada en el medio” (p. 89).

3.2 Diseño de la Investigación.

Según Palella, Santa y Martins, Feliberto (2010) el diseño experimental “es aquel según el cual el investigador manipula una variable experimental no comprobada, bajo condiciones estrictamente controladas. Su objetivo es describir de qué modo y por qué causa se produce o puede producirse un fenómeno. Busca predecir el futuro, elaborar pronósticos que una vez confirmados, se convierten en leyes y generalizaciones tendentes a incrementar el cúmulo de conocimientos pedagógicos el mejoramiento de la acción educativa” (p. 86). De acuerdo a la

definición antes planteada, el diseño de la investigación presente es de tipo experimental ya que se manipulará una variable para comprobar los efectos que producen en el fenómeno estudiado.

3.3. Nivel de la Investigación.

Según Arias, Fidias (2006) el nivel de investigación se refiere al "Grado de profundidad con que se aborda un objeto o fenómeno. Aquí se indicará si se trata de una investigación exploratoria, descriptiva o explicativa.". En este caso, el nivel de la investigación es descriptiva, y según Arias, Fidias (1999), "Los estudios descriptivos miden de forma independiente las variables, y aun cuando no se formulen hipótesis, las primeras aparecerán enunciadas en los objetivos de investigación.".

El nivel de la presente investigación se basa principalmente en describir la situación de inseguridad que ocurre dentro de las instalaciones de la Universidad José Antonio Páez indicando las razones por la cual este fenómeno se encuentra vigente así como las diversas herramientas que se han utilizado para dar intentar dar una solución al problema lo cual ayudó a obtener una relación de los acontecimientos obteniendo así las bases que se utilizaron para el diseño de un sistema que permita solventar o disminuir la problemática presentada.

3.4. Población y Muestra.

Según Arias, Fidias (2006) define población o población objetivo como "un conjunto finito o infinito de elementos con características comunes para los cuales serán extensivas conclusiones de la investigación. Esta queda determinada por el problema y por los objetivos del estudio" (p. 81). Por otro lado, según Egg, Andes, citado en Tamayo y Tamayo (1998) la muestra "es el conjunto de operaciones que se realizan para estudiar la distribución de determinados caracteres en la totalidad de una población universo o colectivo partiendo de la observación de una fracción de la población considerada" (p. 115).

Tomando en cuenta las definiciones anteriores en este trabajo se tomará como población a los alumnos de Ingeniería Electrónica y Telecomunicaciones del 8vo, 9no y 10mo semestre y como muestra tomaremos 60 personas de dicha población.

3.5 Técnicas e instrumentos de Recolección de Datos.

Según Sabino, Carlos (1996) “un instrumento de recolección de datos es en principio cualquier recurso de que pueda valerse el investigador para acercarse a los fenómenos y extraer de ellos información. De este modo el instrumento sintetiza en sí toda la labor previa de la investigación, resume los aportes del marco teórico al seleccionar datos que corresponden a los indicadores y, por lo tanto, a las variables o conceptos utilizados” (p. 149,150). Así mismo Arias, Fidias (1999) menciona que “las técnicas de recolección de datos son las distintas formas de obtener información” (p. 53). De acuerdo a estas definiciones se nombrarán las técnicas e instrumentos utilizados para la recolección de datos en la presente investigación.

3.5.1 Técnicas.

3.5.1.1 Encuesta.

Según Buendia (1998), define la encuesta como el “método de investigación capaz de dar respuestas a problemas tanto en términos descriptivos como de relación de variables, tras la recogida de información sistemática, según un diseño previamente establecido que asegure el rigor de la información obtenida” (p.120). Esta técnica se dará a través de un cuestionario realizada a una muestra de la población objetivo, a través de la encuesta se buscó obtener información sobre el conocimiento que posee los individuos de la inseguridad dentro de la universidad, de los tópicos relacionados a la propuesta a realizarse a través de este trabajo de investigación, del nivel de aceptación de las medidas que deberán implementarse, así como de la ética del mismo.

3.5.1.2 Observación.

Sierra Bravo (1984) define la técnica de la observación como “la inspección y estudio realizado por el investigador, mediante el empleo de sus propios sentidos, con o sin ayuda de aparatos técnicos, de las cosas o hechos de interés social, tal como son o tienen lugar espontáneamente”. Esta técnica será empleada para la recolección de las condiciones referentes donde el sistema será empleado conociendo así su alcance, limitaciones y eficiencia que este tendría al momento de su aplicación.

3.5.2 Instrumentos.

3.5.2.1 Cuestionarios.

Según Hurtado (2000) un cuestionario “es un instrumento que agrupa una serie de preguntas relativas a un evento, situación o temática particular, sobre el cual el investigador desea obtener información” (p. 469). De acuerdo a esto se realizará un cuestionario la cual consta de preguntas cerradas que servirán como instrumento principal para dar respaldo a la encuesta obteniendo así un mejor manejo de los datos que se obtengan de estas para dar un mejor aporte y enfoque a la investigación.

3.6 Fases Metodológicas.

El desarrollo de esta investigación consta de distintas fases las cuales guardan relación con los objetivos específicos mencionados anteriormente, para esto, se describirán brevemente cómo se logrará desarrollar cada una, así como de los resultados que se desean obtener en cada una de estas. Para llevar a cabo cada fase se emplean diversas metodologías aplicadas en la en el desarrollo de software para cubrir cada etapa de la investigación.

Fase I: Determinar los requerimientos funcionales y no funcionales del sistema, mediante técnicas de recolección de datos.

En esta fase, con la ayuda de las técnicas e instrumentos de recolección de datos se procederá a la determinación de los requerimientos que debe poseer el sistema para poseer un rendimiento óptimo, según la IEEE (Std 610.12-1900) los requerimientos son “una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal” (p. 62).

Según Pressman (2006) la “Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software” (p. 155).

De acuerdo a la definición presentada, se definirán tantos los requerimientos funcionales como no funcionales para llevar a cabo la investigación utilizando la Ingeniería de Requerimientos.

Fase II: Diseñar el sistema de reconocimiento facial según la metodología XP.

Es esta fase, se procede a realizar el software principal del sistema, siendo esta definida por Beck, Kent "Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.". Dado esto se seguirá una metodología ágil dentro de la ingeniería del software conocida como extreme programming (xp) para diseñar el sistema a ser utilizado en esta investigación.

Fase III: Desarrollar la estructura del sistema de seguridad con la ayuda de un Raspberry Pi y técnicas biométricas.

En este fase se utilizara el sistema diseñado en la fase II en conjunto con las técnicas biométricas utilizadas en la misma para dar vida al sistema de seguridad haciendo uso de un ordenador de placa reducida como lo es el raspberry pi haciendo uso del paradigma de programación orientada a objetos (OOP según siglas en inglés) la cual es definida por Booch, Grady como "un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan una instancia de alguna clase, y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones de herencia".

Fase IV: Realizar las pruebas necesarias para poder evaluar el rendimiento del sistema.

En esta fase se realizarán diversas pruebas al sistema desarrollado en la fase III para probar el rendimiento en diversas condiciones para corroborar que lo hecho previamente cumpla con lo especificado, en caso de existir errores y/o fallas deben corregirse. Por otra parte, se desea determinar la confiabilidad de los resultados

obtenidos con respecto a los esperados, así como de realizar una comparación de las diversas técnicas biométricas utilizaras para el desarrollo del sistema.

CAPÍTULO IV
RESULTADOS

A continuación, se describirán el desarrollo de cada fase las cuales describen los objetivos específicos planteados al inicio de este trabajo de grado.

4.1 Fase I. Determinar los requerimientos funcionales y no funcionales del sistema.

Con el fin de poder determinar la funcionalidad del sistema y analizar las opiniones y el nivel de familiarización que posee la muestra definida en el capítulo anterior, se llevará a cabo una observación directa al entorno de trabajo y una encuesta cerrada. Con la información obtenida de ambas técnicas empleadas se seleccionarán los requerimientos funcionales y no funcionales que deberá tener el sistema, así como las limitaciones del mismo.

4.1.1 Observación Directa.

Tomando en cuenta la muestra se observó el entorno en el cual esta se desenvuelve, para obtener información relevante para el funcionamiento del sistema la observación se realizó de manera pasiva, es decir, que solo se observaron los eventos sin aplicar ninguna interacción que pudiera afectar el funcionamiento del mismo. Se seleccionó este tipo de observación ya que la situación planteada en este proyecto, es la inseguridad y este es un evento que ocurre de manera inesperada.

En las observaciones el entorno de la muestra es la Universidad José Antonio Páez, se pudo observar y clasificar las observaciones en tres aspectos importantes, observación de puntos críticos, observación de exterior e interior.

4.1.1.1 Observación de Puntos Críticos.

- Aquí se observó el comportamiento de la muestra y de los posibles eventos que pudieran ocurrir en torno a los puntos de acceso principales de la universidad, los cuales son los accesos peatonales y el acceso vehicular. Al observar el entorno definido se pudieron observar los siguientes comportamientos y eventos:
- Tanto en los accesos peatonales como el acceso vehicular el usuario que está ingresando debe mostrar su identificación mediante el carnet universitario o el horario sellado por la administración de la universidad.
- En los accesos peatonales si el usuario posee algún tipo de bolso y/o maleta se debe revisar para poder acceder o salir del recinto.
- En los accesos peatonales si el usuario posee algún equipo electrónico como video beam, laptop u otro, este debe registrar su equipo al momento de entrar para luego verificar el mismo equipo a la salida.
- En el acceso vehicular solo se permite que entre y salga un usuario dentro del carro el cual deberá ser el dueño del mismo.
- Los usuarios que no pertenezcan a la institución deberán ser registrado dando detalles de la visita, así como del estudiante que lo hace pasar (en caso de haber).
- Si la cantidad de usuarios que están ingresando es alta esto ocasiona un retardo en el ingreso de los usuarios se vea disminuida y se ocasionen colas de espera para entrar al recinto.
- Las causas de los retardos ocasionalmente son debido a que el usuario no muestra el carnet en el momento debido, en el registro de los bolsos o aparatos electrónicos.
- Los registros de los accesos peatonales con respecto a los aparatos electrónicos que ingresan no se encuentran relacionados, cada acceso posee un registro distinto.

- En el acceso vehicular se pide abrir la maletera ya sea en la salida o entrada siendo la salida donde más veces suele ocurrir.
- A una determinada hora definida uno de los accesos peatonales se cierra dejando un solo acceso operativo.
- Los accesos peatonales como el acceso vehicular poseen módulos de vigilancia.

4.1.1.2 Observación de Exterior e Interior.

Aquí se observa el comportamiento de la muestra tanto dentro de las infraestructuras donde se imparten clase, así como de las áreas verdes, además se pudo observar como la muestra se mueve dentro de las dos regiones de los distintos roles que hacen vida dentro de la universidad. Estos roles generan subconjuntos dentro del conjunto de muestra que se toma obteniendo un conjunto de personal administrativo, personal obrero y estudiantes. Al observar el comportamiento de cada rol tanto en las áreas verdes como la infraestructura de estudios se pudo observar que:

- En los diferentes edificios de estudios se encuentran equipos esparcidos dentro de los mismos en el cual el profesor registra la hora de entrada y salida de cada clase respectiva.

- Cada edificio posee una zona en donde existe una mayor aglomerar de la muestra que en otras áreas:

* Edificio de Ingeniería: Piso de control de estudios.

* Edificio de Odontología: Planta baja.

* Edificio de Mecánica: Los alumnos suelen aglomerarse en la entrada del edificio.

* Edificio de Arquitectura: La muestra suele ubicarse en dos secciones principales, en la feria de comida y el área de arquitectura que se encuentra en el sótano del mismo edificio.

- Los vigilantes suelen rondar tanto por el edificio de aulas como en las zonas verdes.

- Los puntos de mayor riesgo en caso de pérdidas son los laboratorios ubicados en los diferentes edificios.

- La aglomeración de personas dentro de la universidad varía con el día y la hora en el que se esté observando.

- Hay un módulo de vigilancia entre el área verde que da hacia el estacionamiento y el estacionamiento mismo.

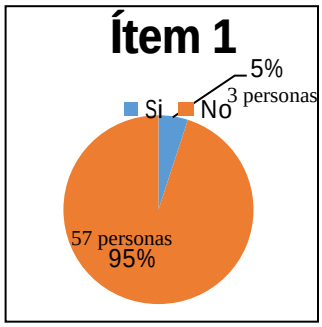
4.1.2 Encuesta Cerrada

En esta sección se aplicó una encuesta a la muestra seleccionada con el tipo de preguntas cerradas, es decir, una encuesta en donde cada pregunta tiene respuestas claramente definidas en el cual la muestra deberá seleccionar una o más respuestas de acuerdo sea el caso, esto se da con el fin de analizar las opiniones, aceptación y nivel de familiarización que posee la muestra con respecto al tema en cuestión.

1. ¿Cree usted que los sistemas de seguridad implementados en la universidad han dado resultados?

El 95% de los encuestados creen que los sistemas de seguridad implementados en la universidad no han dado resultados. (Figura 1)

Figura 1. Resultado ítem 1

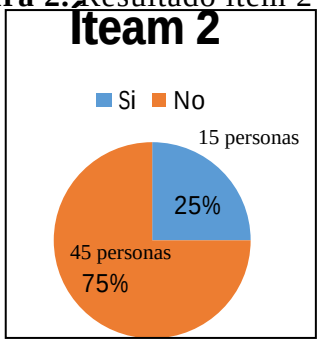


Fuente: Falcón y Mendoza (2018)

2. ¿Ha sido víctima de algún incidente dentro de las instalaciones de la universidad?

El 75% de los encuestados no han sido víctimas de algún incidente dentro de las instalaciones dentro de la universidad. (Figura 2)

Figura 2. Resultado ítem 2

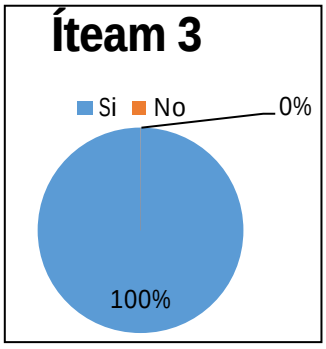


Fuente: Falcón y Mendoza (2018)

3. ¿Ha oído hablar del reconocimiento facial?

El 100% de la muestra considera ha oído hablar del reconocimiento facial. (Figura 3)

Figura 3. Resultado ítem 3

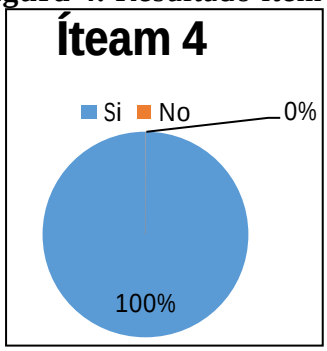


Fuente: Falcón y Mendoza (2018)

4. ¿Considera usted beneficioso implementar un sistema de seguridad usando el reconocimiento facial en la universidad?

El 100% de la muestra considera beneficioso la implementación de un sistema de seguridad a través del reconocimiento facial. (Figura 4)

Figura 4. Resultado ítem 4

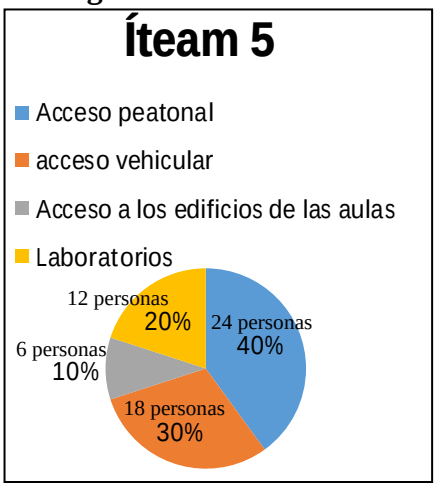


Fuente: Falcón y Mendoza (2018)

5. ¿Cuáles considera usted los puntos más importantes en donde debe colocar cámaras de reconocimiento facial?

El 40% de la muestra considera que el punto más importante en donde debe colocar cámaras de reconocimiento facial es en el acceso peatonal. (Figura 5)

Figura 5. Resultado ítem 5

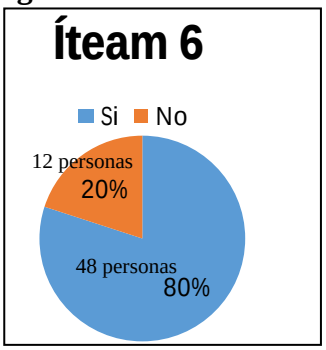


Fuente: Falcón y Mendoza (2018)

6. ¿Cree usted que un sistema basado en reconocimiento facial pueda tener otra implementación dentro de la universidad además de utilizarla como sistema de seguridad?

El 80% de la muestra cree el sistema basado en reconocimiento facial pudiera tener otra implementación dentro de la universidad además de utilizarla como sistema de seguridad. (Figura 6).

Figura 6. Resultado ítem 6

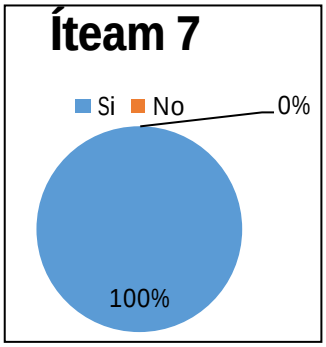


Fuente: Falcón y Mendoza (2018)

7. ¿Está usted de acuerdo de otorgarle a la institución un conjunto de imágenes las cuales serán utilizadas para el reconocimiento facial y entrenamiento del mismo?

El 100% de la muestra está de acuerdo de otorgarle a la institución un conjunto de imágenes las cuales serán utilizadas para el reconocimiento facial y entrenamiento del mismo. (Figura 7).

Figura 7. Resultado ítem 7

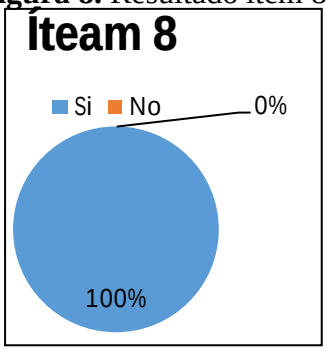


Fuente: Falcón y Mendoza (2018)

8. ¿Considera factible que el sistema registre entrada y salida del individuo por cualquier de los accesos a la universidad?

El 100% de la muestra considera factible que el sistema registre entrada y salida del individuo por cualquier de los accesos a la universidad. (Figura 8).

Figura 8. Resultado ítem 8

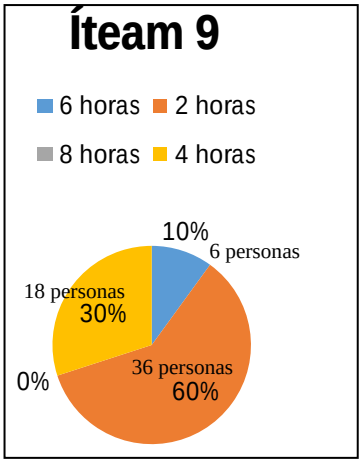


Fuente: : Falcón y Mendoza (2018)

9. ¿Cuánto tiempo considera que se debería otorgarle a un visitante que no posea la identificación necesaria para entrar a las instalaciones?

El 60% de la muestra considera que el tiempo que se debería otorgarle a un visitante que no posea la identificación necesaria para entrar a las instalaciones es de 2 horas. (Figura 9).

Figura 9. Resultado ítem 8

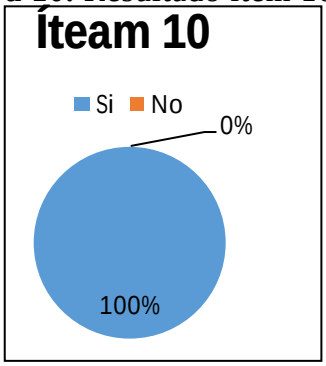


Fuente: Falcón y Mendoza (2018)

10. ¿Estaría usted de acuerdo con aceptar los cambios que deban implementarse para darle uso al sistema de manera apropiada?

El 100% está de acuerdo con los cambios para darle el uso apropiado al sistema. (Figura 10).

Figura 10. Resultado ítem 10



Fuente: Falcón y Mendoza (2018)

4.1.3 Levantamiento de Requerimientos

Analizando los datos obtenidos de las observaciones y las encuestas se pudo obtener un panorama claro de las funcionalidades básicas que debe poseer el sistema a desarrollar. Además, se tomó en cuenta el aspecto económico y el tiempo definido para la culminación de este proyecto para definir los requerimientos funcionales y no funcionales necesarios del sistema.

4.1.3.1 Requerimientos Funcionales

- Cada usuario debe ser correctamente identificado al momento de ingresar por los accesos peatonales principales.
- Se deben registrar los aparatos electrónicos que el usuario posea al momento de ingresar a las instalaciones para luego ser verificados al momento de la salida.
- Los usuarios que no se encuentren registrados en el sistema se les debe registrar por un usuario que posea solo posea la capacidad de indicar el motivo, horas permitidas y estudiante que le permitió la entrada (en caso de haber).
- Se debe registrar hora de entrada y salida.
- Debe haber un tipo de usuario que permita ingresar nuevos usuarios permitidos, así como de las imágenes.
- Los usuarios que poseen acceso a las instalaciones son el personal administrativo, obrero y estudiantes.
- Se deben registrar los videos de vigilancia, así como todos los registros realizados en una jornada.

4.1.3.2 Requerimientos No Funcionales

- Debe dar respuesta rápida
- Los datos deben estar disponibles en todo momento
- Los datos se deben encontrar centralizados
- No debe producir congestionamiento en los accesos peatonales
- Debe funcionar en todo momento durante su jornada

4.2 Fase II: Diseñar el Sistema de Reconocimiento Facial Según la Metodología XP

Conociendo los resultados obtenidos en la fase I se procederá a desarrollar el sistema de seguridad en dos fases, en la presente fase se implementará un sistema de reconocimiento facial el cual poseerá las características mínimas para que el reconocimiento facial se logre de manera satisfactoria. Para lograr esto se utilizará la programación extrema dividiendo el trabajo en tres secciones a ser implementadas:

- Diseño e implementación de la base de datos necesaria para la implementación del reconocimiento facial
- Diseño e implementación de la detección de rostros
- Diseño e implementación del reconocimiento de rostros

4.2.1 Instalación de Componentes Necesarios

Para lograr estos objetivos se utilizará el lenguaje de programación de alto nivel Python en su versión 3.6.4 bajo el sistema operativo Raspbian siendo esta una distribución linux en el cual se hablará más en la fase III, adicionalmente se utilizará la librería opencv-python la cual se encargará del procesamiento de imágenes necesarios para la detección y reconocimiento facial. Cabe destacar que la instalación del lenguaje de programación Python y la librería opencv-python son compatibles con diversos sistemas operativos.

4.2.1.1 Python

La instalación de Python se hará bajo distribuciones Linux, más específicamente *Raspbian* perteneciendo esta distribución a la SBC conocida como *Raspberry Pi.* Comúnmente Python viene pre instalado en la mayoría de las distribuciones linux, en caso de que no, se recomienda que antes de instalar python se actualicen los paquetes de la distribución con los siguientes comandos:

- sudo apt-get update –y
- sudo apt-get upgrade -y

Una vez actualizado los paquetes se procede a instalar Python en el sistema con el siguiente comando:

- sudo apt-get install python3

4.2.1.2 Instalación de Pip

Pip (Pip Install Packages) es un sistema de gestión de paquetes utilizado para instalar y gestionar paquetes de software escritos en *Python.* A partir de la versión *Python* 2.7.9 (serie *python2)* y *Python 3.4 (*serie *python3) pip* viene incorporado cuando se instala *python* por defecto. En caso de que no se encuentre instalado, se puede instalar bajo el siguiente comando en la terminal del sistema operativo *Raspbian*:

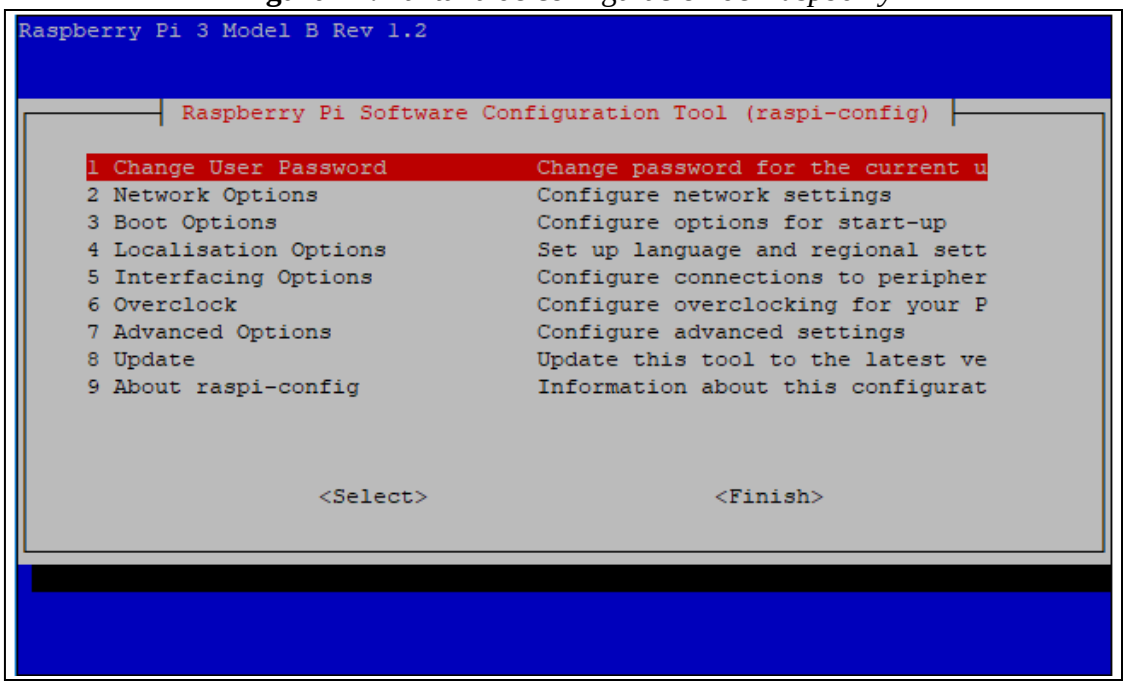
- sudo apt-get install python3-pip

4.2.1.3 Instalación de Opencv-Python

Opencv (*Open Source Computer Vision Library)* como lo indica sus siglas en inglés, es una librería de código abierto enfocada en la computación visual, esta se encuentra principalmente diseñada para la eficiencia computacional teniendo un enfoque en aplicaciones de tiempo real, la optimización se encuentra realizada en los lenguajes *C/C++* permitiendo así poder aprovechar el procesamiento multi-core. *Opencv-python* es la librería funcional bajo el lenguaje *Python* que se utilizara, el proceso de instalación de la librería dentro del sistema operativo *Raspbian* se puede llevar a cabo de una conexión física en el cual se pueda abrir la terminal y ejecutar comando o a través de una conexión *SSH***.** Para habilitar la conexión *SSH* se debe llevar a cabo la conexión física y ejecutar los siguientes comandos:

- sudo raspi-config

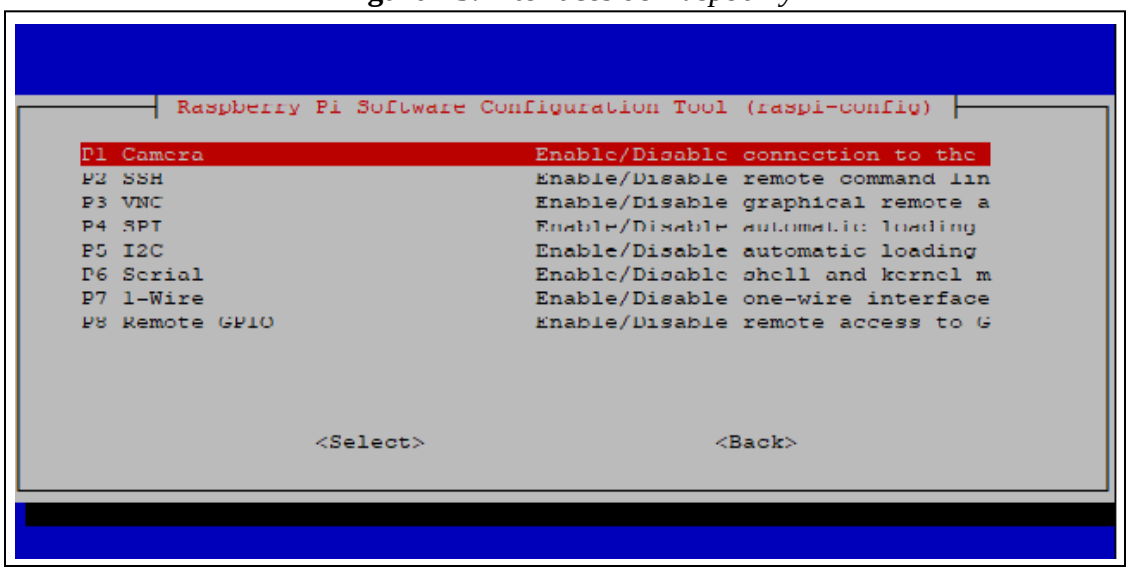
Figura 12. Pantalla de configuración del *raspberry*



Fuente: Falcón y Mendoza (2018).

Una vez dentro de la configuración se debe seleccionar la opción *Interfacing Options*, donde se observarán todas las interfaces disponibles que trae la *raspberry:*

Figura 13. Interfaces del *raspberry*



Fuente: Falcón y Mendoza (2018).

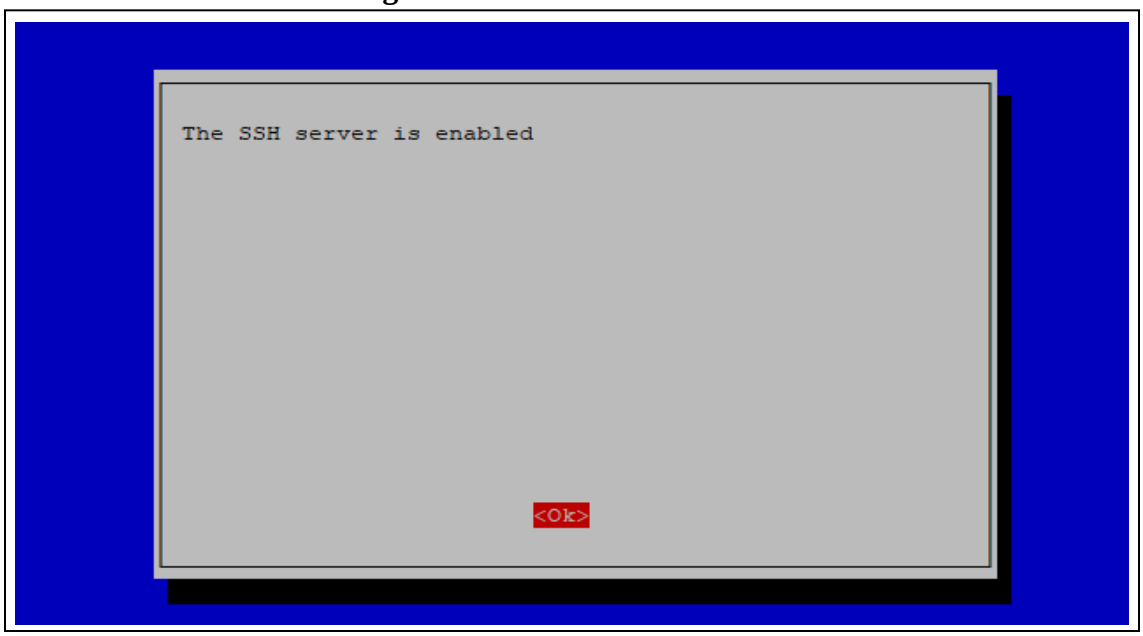
Una vez dentro de las opciones de las interfaces se selecciona la interfaz *SSH:*

Figura 14. Diálogo de confirmación de la interfaz SSH

Fuente: Falcón y Mendoza (2018).

Al seleccionar la opción *Yes* en el diálogo de confirmación de la activación de la interfaz *SSH* aparece un diálogo indicando que la interfaz *SSH* se ha activado:

Figura 15. Interfaz *SSH* activada

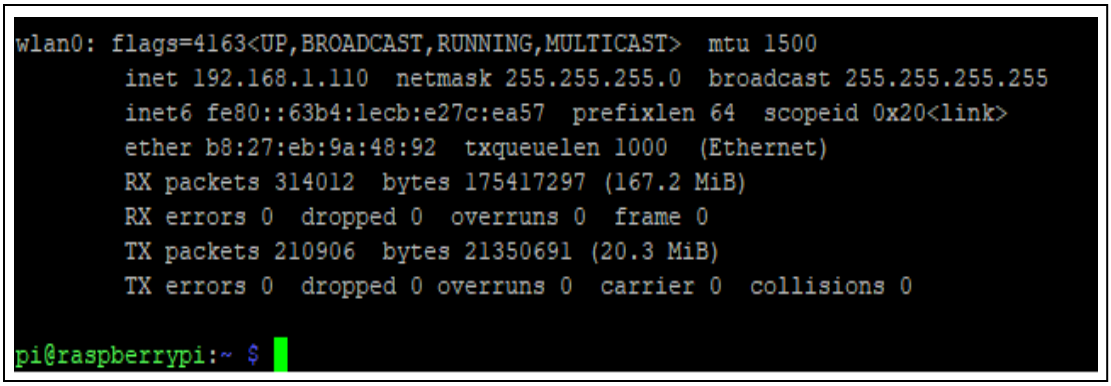


Fuente: Falcón y Mendoza (2018).

Una vez realizado esto ya se puede acceder al *raspberry* a través de la conexión *SSH*, para acceder a esta se requiere conocer la *IP* del dispositivo dentro de la red *WLAN*, para conocer la dirección *IP* se utilizará el siguiente comando:

- Ifconfig

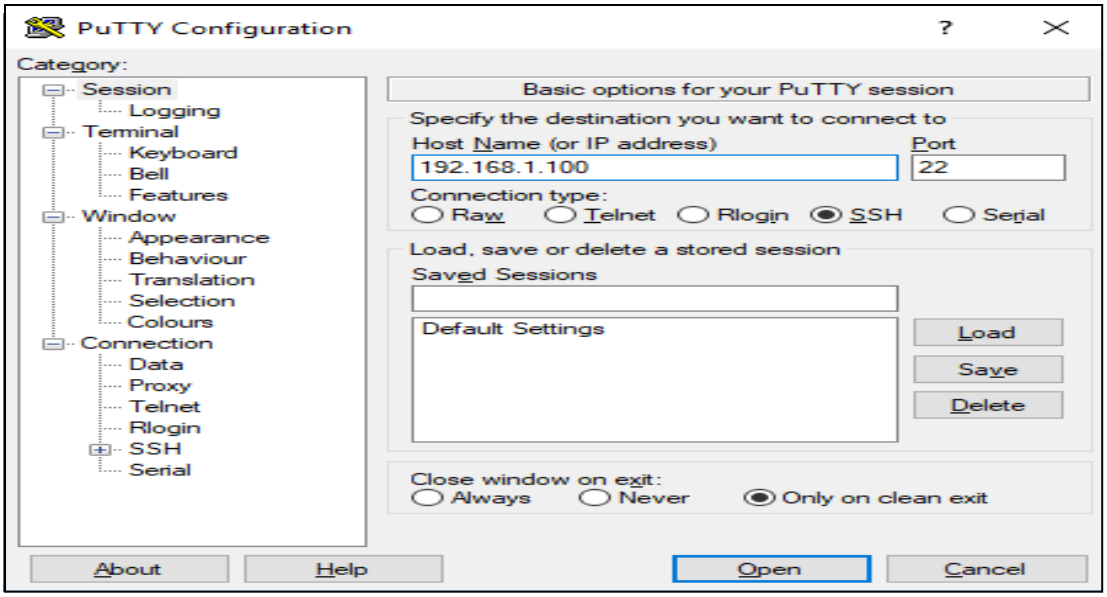
Figura 16. Interfaz *wlan0* y su *IP* correspondiente en el renglón *inet*



Fuente: Falcón y Mendoza (2018).

La instalación de *opencv* se llevará a través de una conexión *SSH* desde un computador con sistema operativo *Windows 10* utilizando el programa *Putty*. La instalación se procederá a través diversos pasos ya que se requieren realizar ciertas configuraciones para que la esta sea lo más sencilla posible evitando así que ocurra algún error durante la instalación. La interfaz del *Putty* es bastante intuitiva, se requiere colocar la *IP* en la sección *Host Name (or IP address)*, indicar el puerto que es nuestro caso es el 22 y establecer la conexión, una vez la conexión se haya establecido se pedirá el nombre de usuario con el cual se ingresará, en nuestro caso será *pi* y su correspondiente clave.

Figura 17. Estableciendo la conexión *SSH* a través de *Putty*

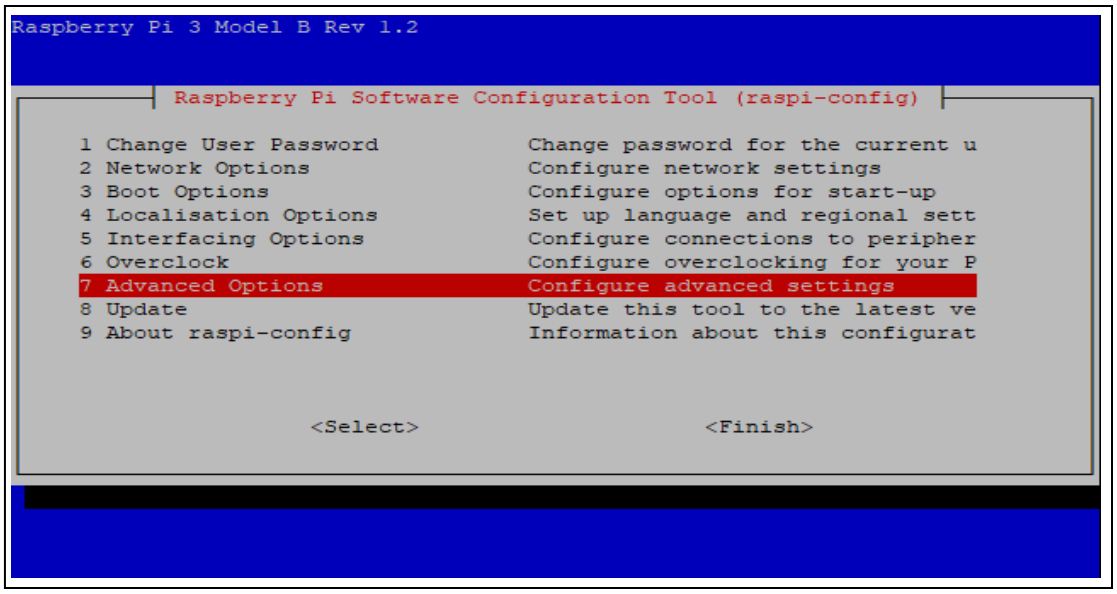


Fuente: Falcón y Mendoza (2018).

Si el sistema operativo *raspbian* está recién instalado lo primero que se debe hacer es expandir el sistema de archivos (*filesystem)* para incluir todo el espacio disponible en la memoria *micro-SD*, para esto se accederá a la configuración del raspberry seleccionando la opción *Advanced Options* (opciones avanzadas):

- sudo raspi-config

Figura 18. Opciones avanzadas del *rasperry*



Fuente: Falcón y Mendoza (2018)..

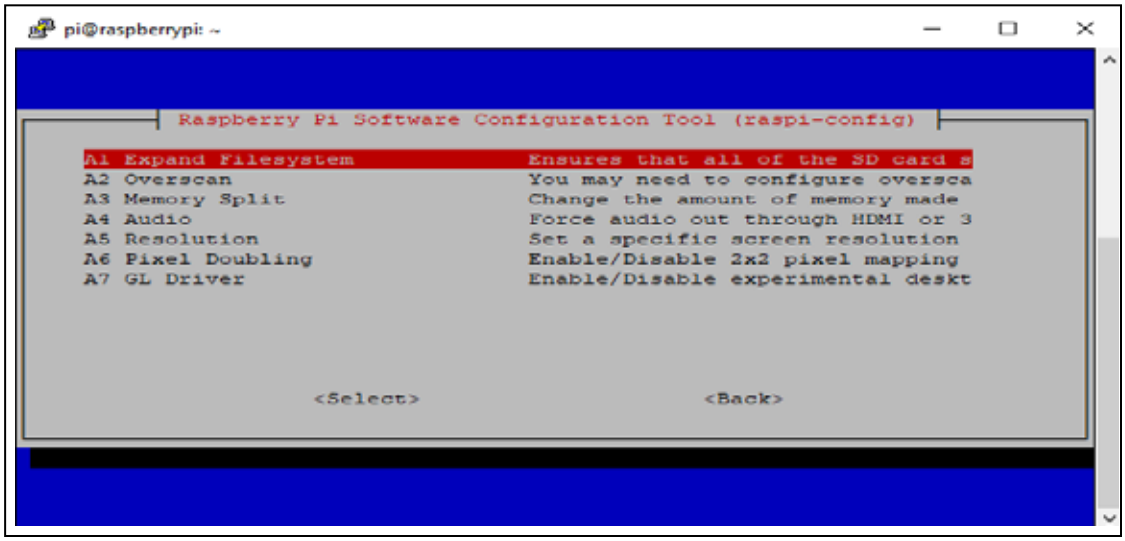
Lo siguiente es seleccionar la opción *A1 Expand Filesystem* en el menú de configuración y completar la configuración, una vez completada la expansión es recomendable reiniciar el dispositivo, para esto se puede utilizar el siguiente comando:

- Sudo reboot

Una vez reiniciado el dispositivo se comprueba que la expansión del sistema de archivos se haya realizado satisfactoriamente, para esto se verifica que el espacio disponible del sistema sea igual al espacio de la memoria *micro-SD*, para observar esto se procede con el siguiente comando:

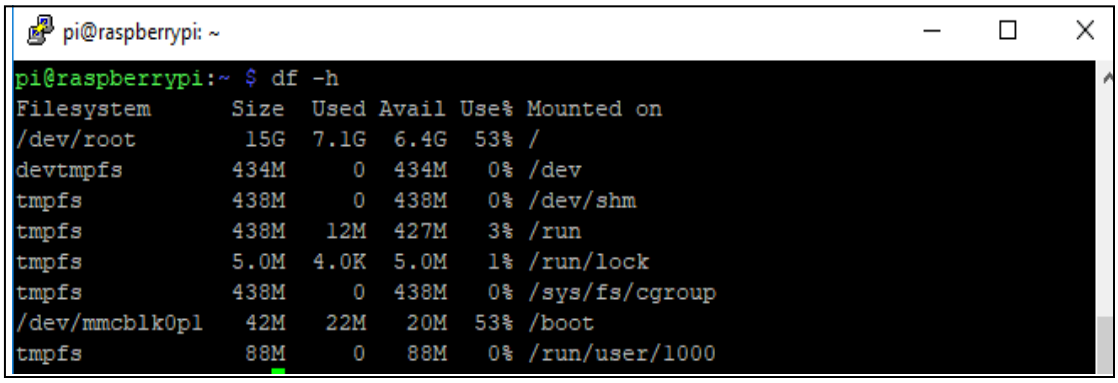
- df -h

Figura 19. Opción A1 *Expand Filesystem*



Fuente: Falcón y Mendoza (2018).

Figura 20. Verificación de expansión del sistema de archivos



Fuente: Falcón y Mendoza (2018).

A continuación, se deben instalar las dependencias necesarias, para esto debemos tener el sistema operativo actualizado, esto se realiza con el siguiente comando:

- sudo apt-get update && sudo apt-get upgrade

Una vez actualizado el sistema se pasará a instalar diversas herramientas de desarrollo en las cuales se incluye *CMake* el cual nos ayudará a configurar el proceso de instalación de *opencv*, esto se realizará con el siguiente comando:

- sudo apt-get install build-essential cmake pkg-config

En el próximo paso se instalan los paquetes *I/O* (*In/Out* - Entrada/Salida) para imágenes y videos, estos nos permitirán cargar los distintos formatos de imágenes y videos, para esto se realizarán los siguientes comandos:

- sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
- sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
- sudo apt-get install libxvidcore-dev libx264-dev

La librería *OpennCV* viene con un módulo llamado highgui el cual es usado para mostrar imágenes en nuestra pantalla y construir básicas *GUIs*. Para compilar el módulo highgui necesitamos instalar la librería de desarrollo *GTK* con el siguiente comando:

- sudo apt-get install libgtk2.0-dev libgtk-3-dev

Muchas operaciones dentro de OpenCV pueden ser optimizadas si se instalan las siguientes dependencias con el siguiente comando:

- sudo apt-get install libatlas-base-dev gfortran

Una vez se hayan instalado todas las dependencias se debe descargar el código fuente de *OpenCV* de su repositorio oficial, en esta sección se selecciona la versión 3.4.0 de opencv y se descargara usando los siguientes comandos:

- cd ~
- wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.4.0.zip
- unzip opencv.zip
- wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.4.0.zip
- unzip opencv_contrib.zip

Una vez realizada la descarga instalaremos la dependencia *NumPy* el cual es un paquete para *Python* usado para el procesamiento numérico, este se instalará usando el comando *Pip:*

- *pip install numpy*

Es momento de compilar e instalar *OpenCV* para esto nos debemos dirigir a la carpeta opencv-3.4.0 y ejecutar los siguientes comandos para configurar nuestro constructor utilizando *CMake*:

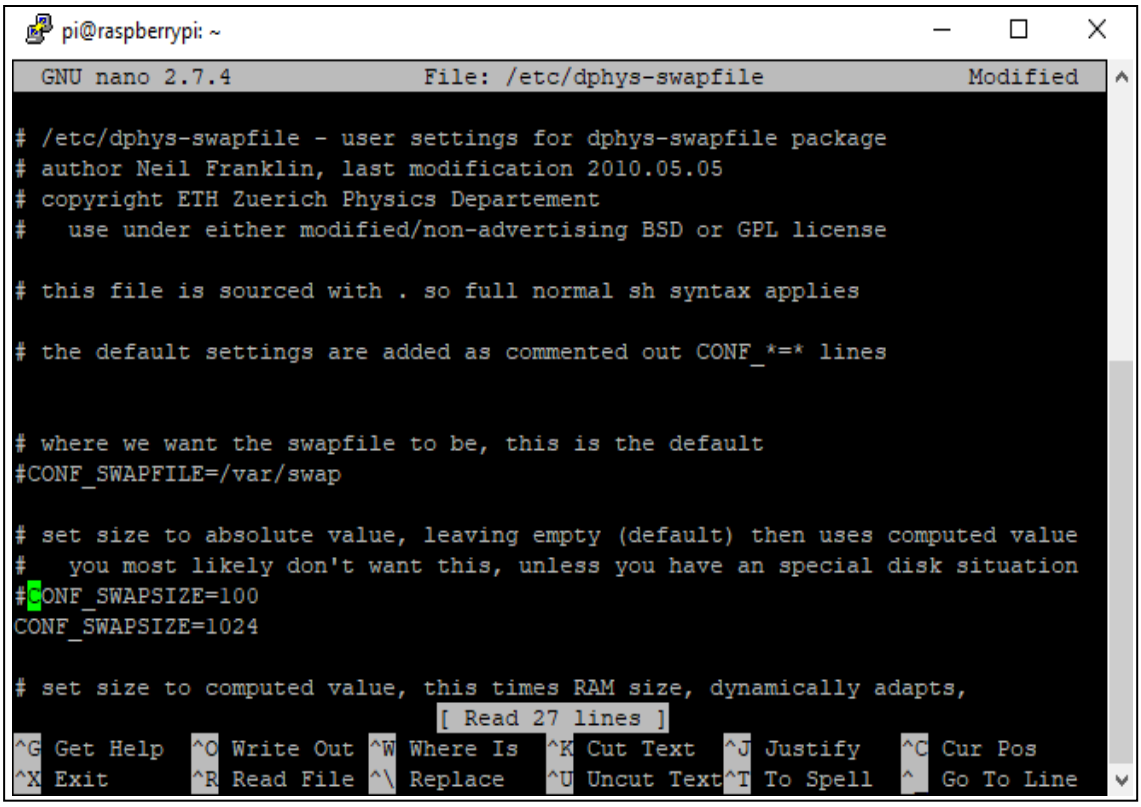
- cd ~/opencv-3.4.0/
- mkdir build
- c make -D CMAKE_BUILD_TYPE=RELEASE \
  -D CMAKE_INSTALL_PREFIX=/usr/local
  DINSTALL_PYTHON_EXAMPLES=ONDOPENCV_EXTRA_MOD
  ULES_PATH=~/opencv_contrib-3.3.0/modules \-D
  BUILD_EXAMPLES=ON \.

Antes de comenzar el proceso de compilación, se debe aumentar el tamaño de espacio de intercambio (*swap)*. Esto permite que *OpenCV* compile con los cuatro núcleos del Raspberry PI sin que la compilación se cuelgue debido a problemas de memoria. Para esto se debe acceder al archivo de la siguiente manera:

- sudo nano /etc/dphys-swapfile

Una vez dentro del archivo se debe configurar la línea *CONF_SWAPSIZE*, por defecto esta viene con un valor de 100, este debe ser cambiado a 1024 mientras se dé la compilación:

Figura 21. Asignación de CONF_SWAPSIZE a 1024



Fuente: Falcón y Mendoza (2018).

Para activar los cambios recién realizados se debe ejecutar los siguientes comandos:

- sudo /etc/init.d/dphys-swapfile stop
- sudo /etc/init.d/dphys-swapfile start

Finalmente se procede a compilar con el siguiente comando:

- make -j4

Una vez que la compilación concluya se procede a instalar *opencv:*

- sudo make install
- sudo ldconfig

Con el proceso de instalación culminado sin errores, *opencv* se deberá encontrar en el directorio */usr/local/lib/python3.5/dist-packages/* en el cual se deberá encontrar el archivo con extensión *.so*, para facilitar su uso este será modificado usando los siguientes comandos:

- cd /usr/local/lib/python3.5/dist-packages/
- sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so

Con los cambios realizados se procederá a crear un link simbólico para concluir con la instalación, esto se realizará con el siguiente comando:

- ln -s /usr/local/lib/python3.5/dist-packages/cv2.so cv2.so

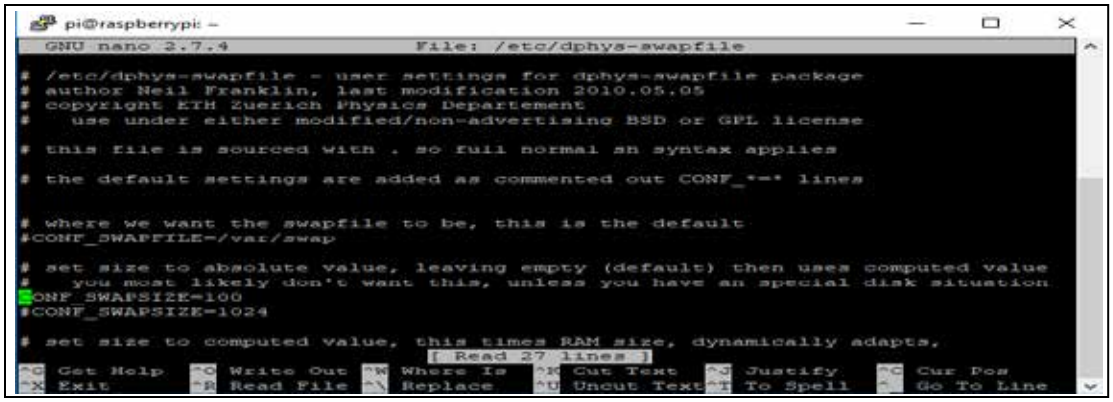
Para el ahorro de almacenamiento se pueden eliminar ambos archivos descargados con los siguientes comandos:

- rm opencv-3.4.0
- rm opencv_contrib-3.4.0

Por último es recomendable volver a cambiar el valor de la variable *CONF_SWAPSIZE* en el archivo de configuración *swap* a su valor original, es decir, 100:

- sudo nano /etc/dphys-swapfile

Figura 22. Asignación del CONF_SWAPSIZE a 100



Fuente: Falcón y Mendoza (2018).

4.2.1.4 Instalación de Mysql

MySql es un sistema de gestión de base de datos relacional y de código abierto, para trabajar la gestión de los datos para el reconocimiento facial y subsecuente al sistema de seguridad se utilizará el gestor de base de datos para la implementación y registrar todo lo necesario para el sistema. Para instalar un servidor mysql se utiliza el siguiente comando:

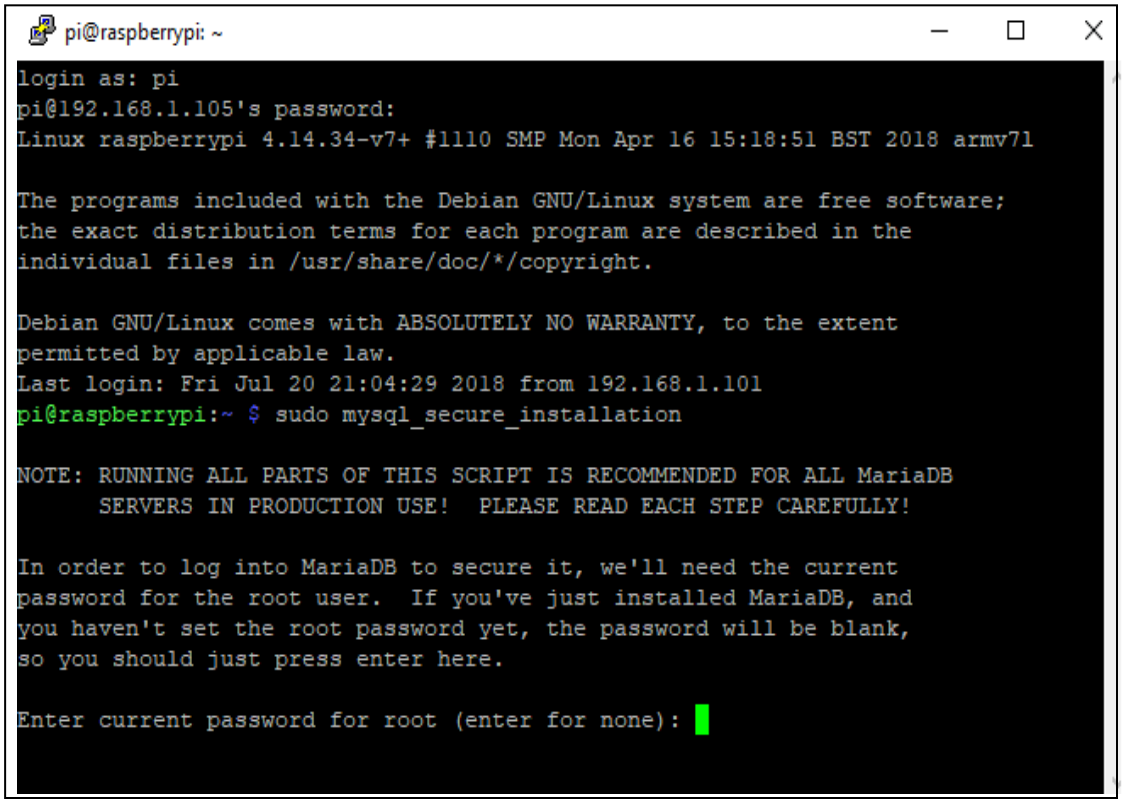
- sudo apt-get install mysql-server

Una vez la instalación concluya se debe configurar la base de datos utilizando el siguiente comando:

- Sudo mysql_secure_installation

Al ejecutar el comando aparecerá un conjunto de configuraciones a realizarse de acuerdo al uso que se dará, la primera configuración será la clave de acceso para el servidor:

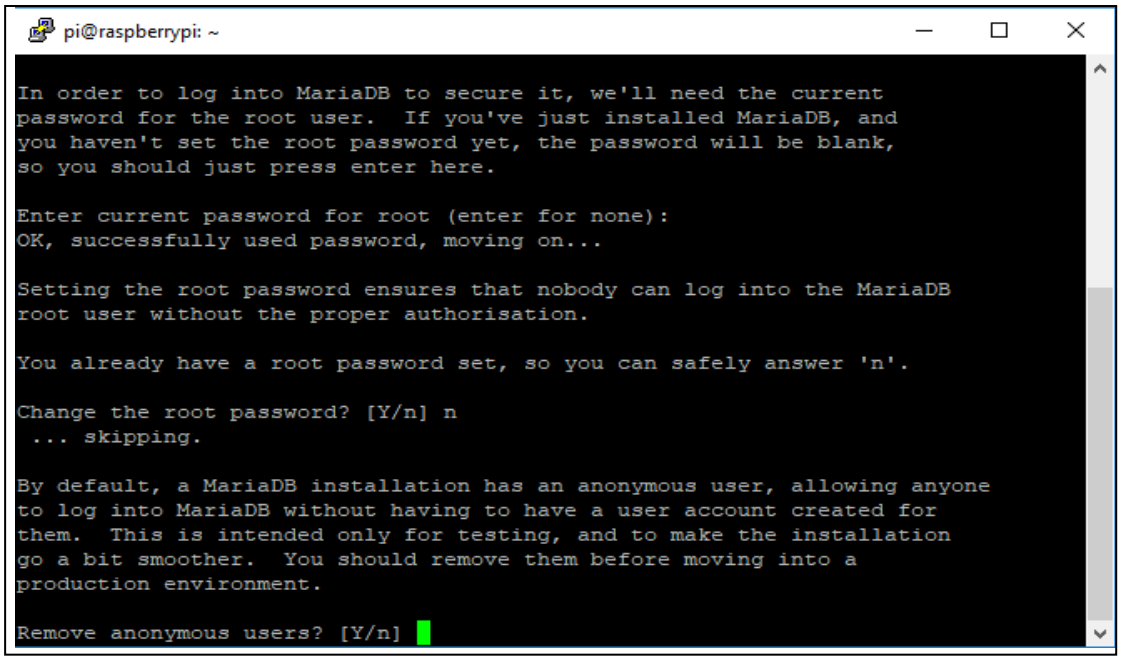
Figura 23. Configuración de la clave del servidor



Fuente: Falcón y Mendoza (2018).

Lo siguiente a configurar son los usuarios anónimos que posee mysql, estos usuarios pueden ingresar al servidor sin la necesidad de tener una cuenta creada, esta configuración será desactivada:

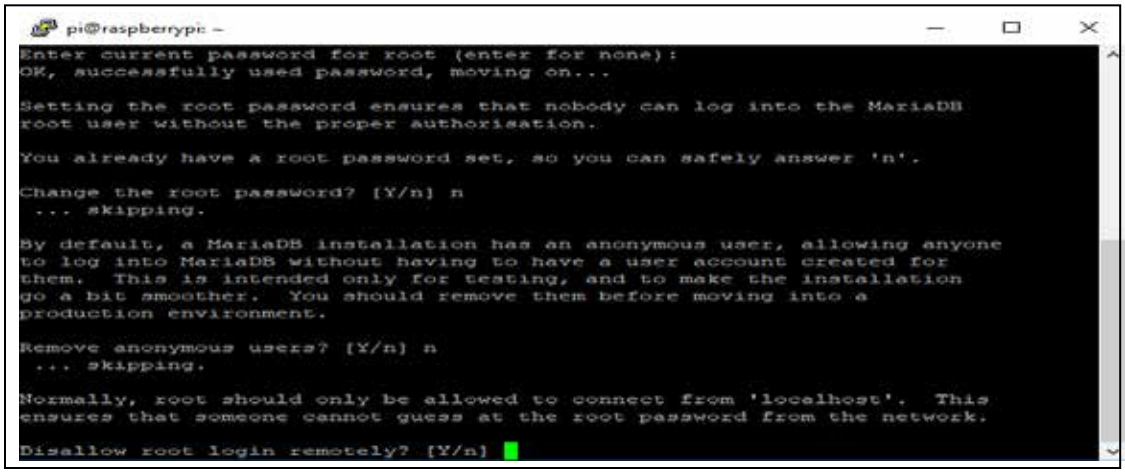
Figura 24. Configuración de usuarios anónimos



Fuente: Falcón y Mendoza (2018).

La próxima configuración se trata de los accesos al servidor como administrador, por defecto el servidor solo puede ser accedido como administrador por la máquina que lo aloja conocida como *localhost.* Para este proyecto se ha decidido que el administrador pueda acceder de manera remota al servidor.

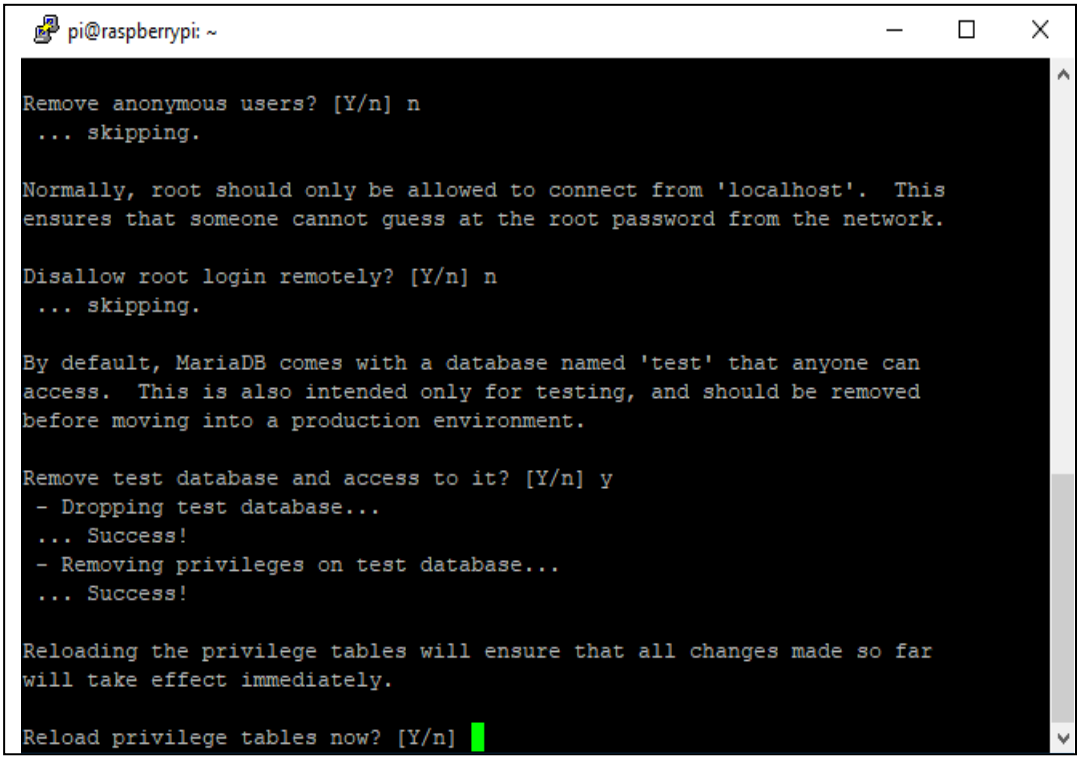
Figura 25. Configuración de acceso remoto



Fuente: Falcón y Mendoza (2018).

Por defecto el servidor provee una base de datos de prueba llamada *test* a la cual todos pueden acceder, dado que se trabajara con una base de datos definida esta base de datos será removida, además la tabla de privilegios será actualizada para que los cambios se hagan efectivos y así culminar con la configuración del servidor.

Figura 26. Eliminando la base de datos *test* y aplicación de cambios



Fuente: Falcón y Mendoza (2018).

4.2.1.5 Instalación de Python-Mysql

Una vez configurado el servidor es necesario poder acceder a este a través del lenguaje de programación *Python*, para esto se procederá a instalar el paquete *python-mysqldb* para realizar la interconexión entre ambas:

- Sudo apt-get install python-mysqldb

4.2.1.6 Instalación de Phpmyadmin

Phpmyadmin es una interfaz amigable que permite visualizar el servidor mysql en el cual se pueda interactuar con mucha más facilidad con las distintas bases de datos que pudieran estar alojadas en este. Durante la instalación se pedirá la clave de acceso al servidor introducida en la configuración de *mysql*, para iniciar la configuración se utiliza el siguiente comando:

- sudo apt-get install phpmyadmin

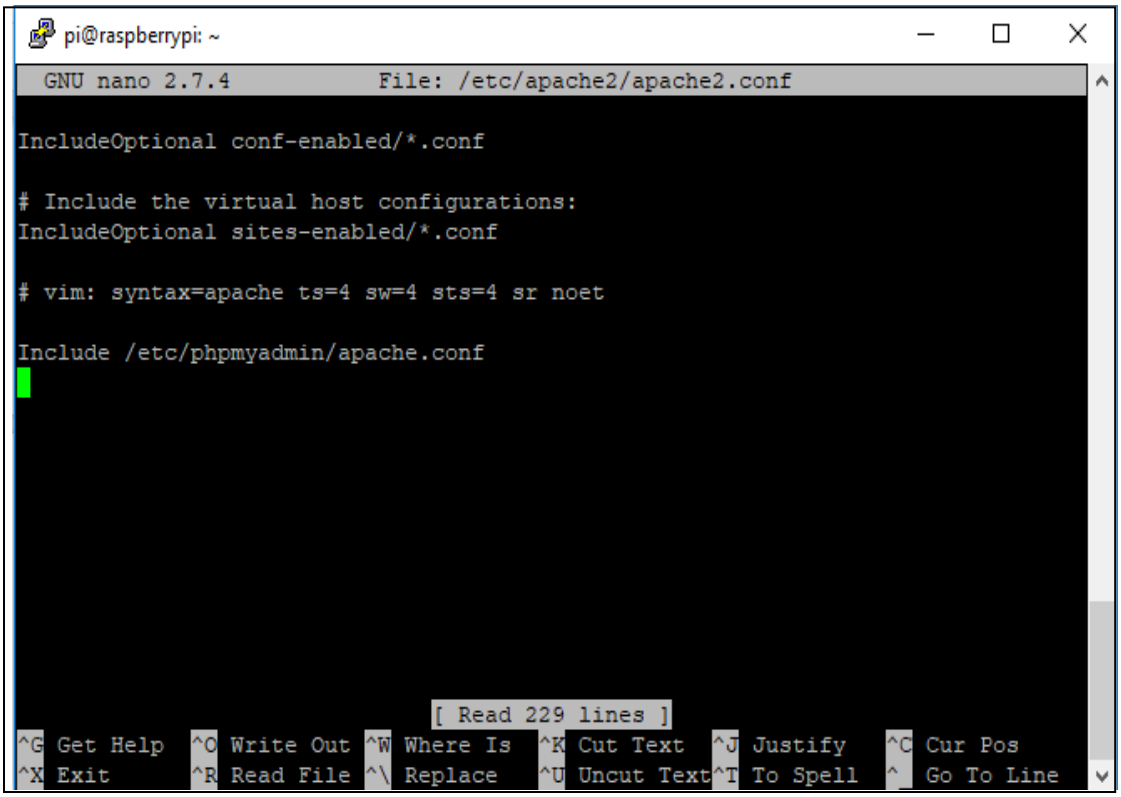
Una vez instalado se procede a configurar *apache* para ser utilizado junto a *phpmyadmin* para esto se debe configurar el siguiente archivo:

- sudo nano /etc/apache2/apache2.conf

Una vez dentro del archivo se debe ir al fin de página y colocar la siguiente línea:

- Include /etc/phpmyadmin/apache.conf

Figura 27. Configuración de *apache* con *phpmyadmin*



Fuente: Falcón y Mendoza (2018).

Una vez modificado el archivo se procede a reiniciar el servidor con el siguiente comando:

- sudo /etc/init.d/apache2 restart

4.2.1.7 Instalación de Flask

Flask es un *microframework web* desarrollado en *python*, utilizando esta librería se realizará una interfaz de usuario para poder manejar las distintas funcionalidad y el *streaming* de las distintas cámaras que puedan estar conectadas a la red para poder ser accedidas a través del servidor principal utilizando la dirección *IP* correspondiente a la cámara. Para instalar el *microframework* se hará uso del gestionador de paquetes *Pip*, para esto se debe ejecutarse el siguiente comando:

- sudo pip3 install flask

Ya con el *microframework* instalado se procederá a instalar dos componentes del mismo *flask* para manejar formularios y la conexión con la base de datos de la siguiente manera:

- sudo pip3 install flask-wtf # Componente para manejar formularios
- sudo pip3 install flask-mysqldb # Componente para manejar la conexión a la base de datos

4.2.1.8 Instalación de Paramiko

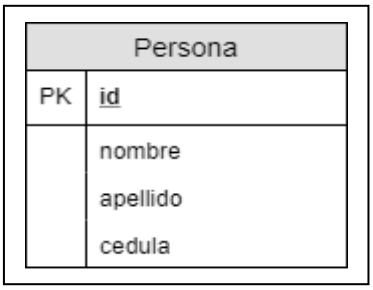
Paramiko es una implementación de *Python* del protocolo *SSHv2* el cual provee de funcionalidades tanto de cliente como de servidor, adicionalmente este trabaja con una librería de criptografía llamada *pyCrypto* por lo cual también será necesario instalar para su correcto funcionamiento, toda la instalación se puede realizar a través de *Pip* ejecutando los siguientes comandos:

- Sudo pip3 install pyCrypto
- Sudo pip3 install paramiko==2.4.1

4.2.2 Diseño e Implementación de la Base de Datos

Tomando en cuenta los requerimientos mencionados anteriormente la base de datos mínima necesaria para que el sistema de reconocimiento facial realice su función será diseñada como una base de datos relacional tomando la premisa de que el sistema de reconocimiento facial necesita de información básica para poder reconocer al individuo por ende se definió una entidad Persona en donde se incluirán todas las entidades aquellas que tengan acceso a las instalaciones.

Figura 28. Modelo ER para la base de datos mínima del reconocimiento facial

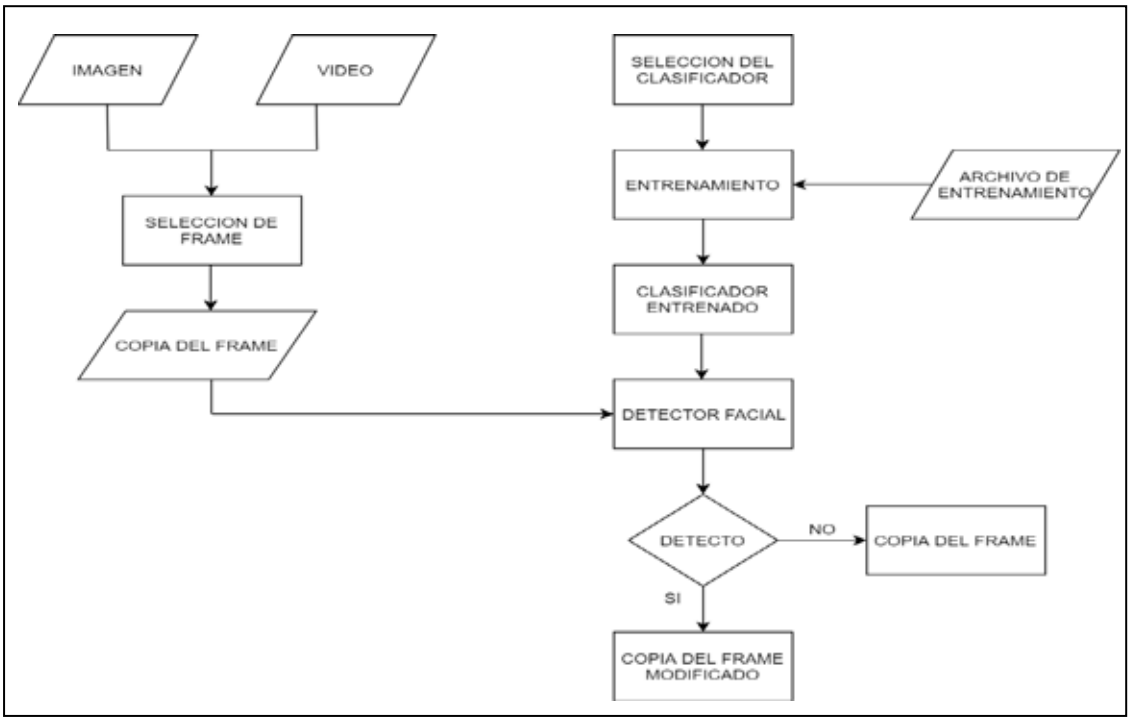


Fuente: Falcón y Mendoza (2018).

4.2.3 Diseño e Implementación de la Detección Facial

Las primeras versiones realizadas para la detección fueron realizadas a través de *scripts* de python para el *haarcascade* de rostros frontales, por lo cual todo el sistema se enfocará en estas características para realizar la detección facial. El código de esta implementación puede observarse en el anexo A.

Figura 29. Diagrama de funcionamiento de la detección facial



Fuente: Falcón y Mendoza (2018).

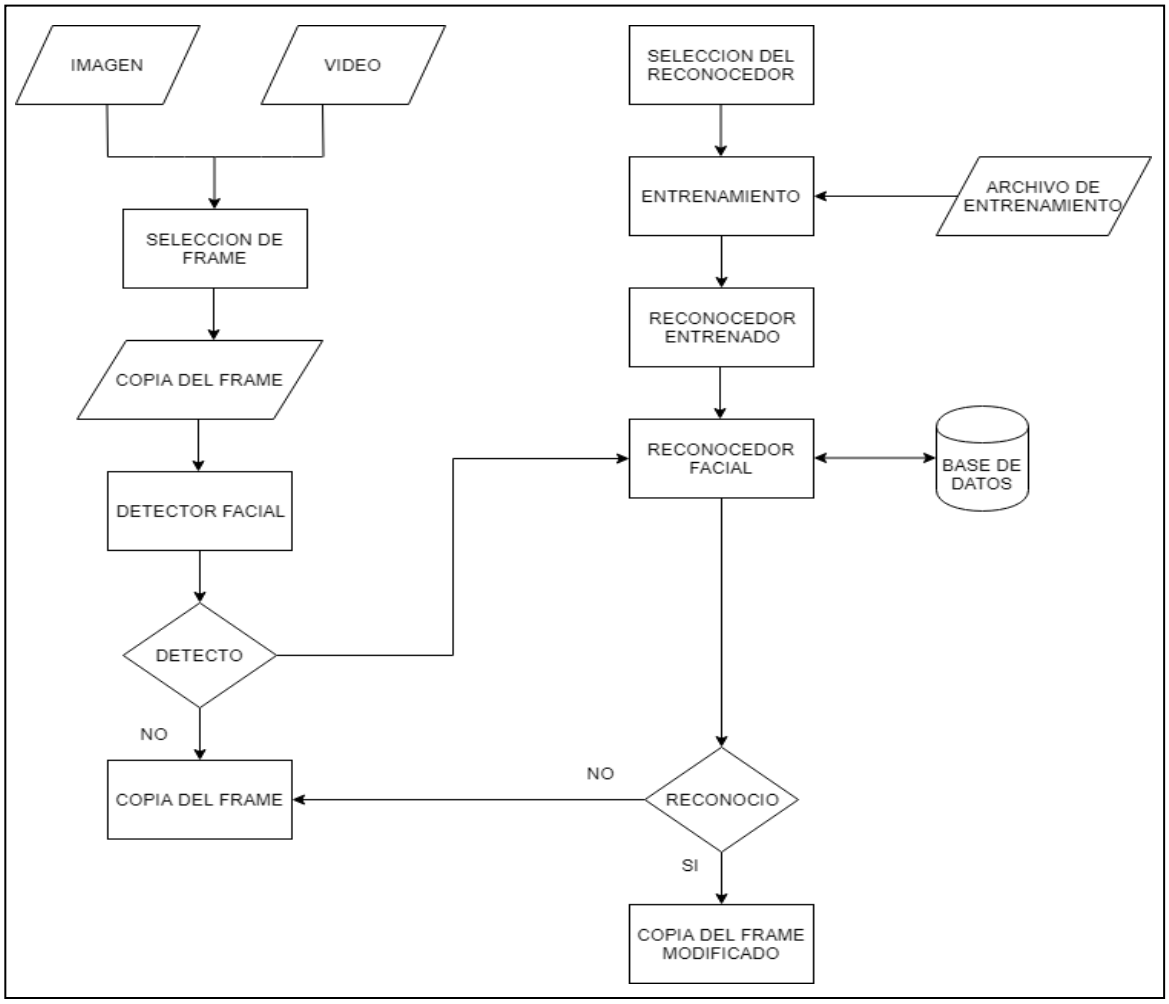
Una vez el detector esté entrenado para la extracción de características faciales la detección facial consiste en pasarle distintos *frames* ya sea proveniente de una imagen o cámara. Una vez el detector detecte algún *frame* entrante este lo analiza en busca de características faciales otorgando una copia del *frame* en donde se resaltan todos los rostros que hayan coincido con el análisis, en caso de no haber detectado ningún rostro se retorna una copia del *frame* sin ninguna alteración.

4.2.4 Diseño e Implementación del Reconocimiento Facial

Una vez implementado el detector facial es momento de asociar un rostro con una entidad Persona en particular, para esto se seleccionará un reconocedor. Opencv-python provee un conjunto de reconocedores listo por usarse a los cuales se les puede entrenar mediante la entrada de archivos, debido a la funcionalidad que el sistema debe otorgar se utilizara el LBPH que posee una respuesta rápida eficiente para el sistema.

Esta versión del reconocimiento facial se realizó utilizando el algoritmo *LBPH* y un archivo de entrenamiento junto al detector *haarcascade* utilizado anteriormente separando cada proceso en un *script* diferente utilizando una base de datos sencilla realizada bajo el esquema *sqlite*, por lo que primero se posee un *script* para la creación de la base de datos, luego uno que utilice el reconocedor para agregar nuevas personas al sistema para luego crear el archivo de entrenamiento a utilizar por el reconocedor y por último el reconocedor. El código de implementación puede observarse en el anexo B.

Figura 30. Diagrama de funcionamiento del reconocimiento facial



Fuente: Falcón y Mendoza (2018).

Una vez que el reconocedor está entrenado es momento de indagar en la base de datos para reconocer efectivamente a alguien, para lograr esto se trabajar con el *frame* utilizado en la detección, una vez que el detector haya encontrado una o más caras en el *frame* el reconocedor compara cada cara detectada del *frame* con el conjunto de imágenes de las distintas entidades Personas y tomará la que posea el mayor nivel de aceptación de las características faciales de ambos lados para luego buscar la información asociada a la entidad Persona acierto en la base de datos.

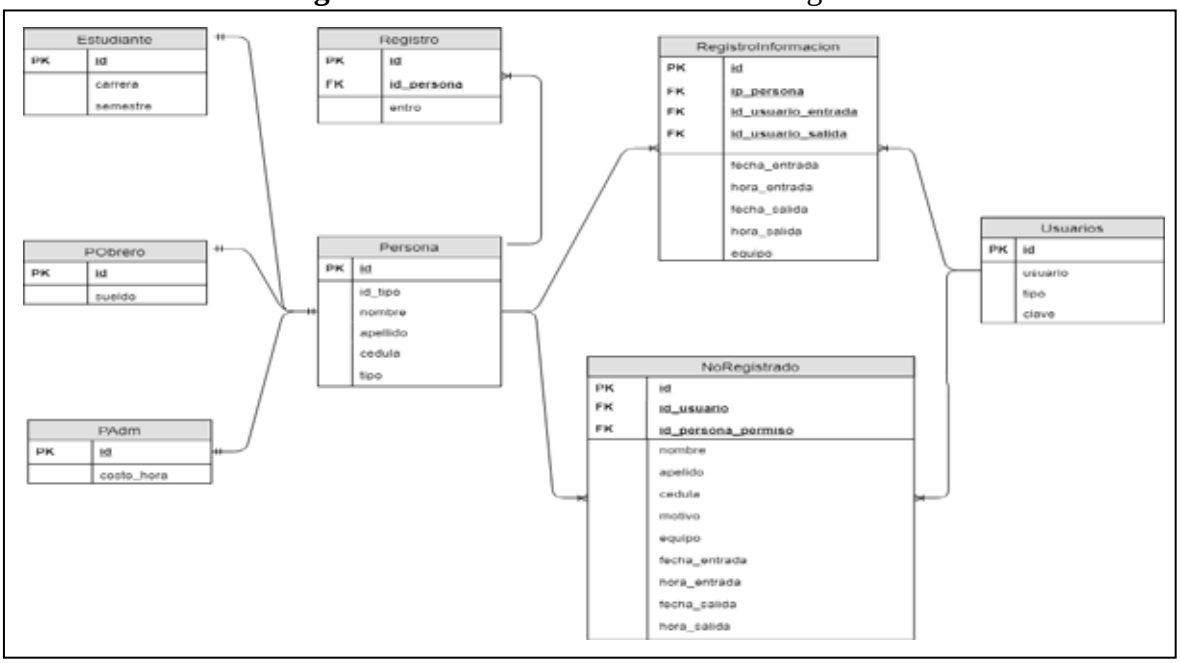
4.3 Fase III: Desarrollar la Estructura del Sistema de Seguridad con la Ayuda de un Raspberry Pi y Técnicas Biométricas

Una vez realizado el sistema de reconocimiento facial con la Ayuda de un Raspberry Pi y Técnicas Biométricas, es momento de integrarlo a un sistema de seguridad, para esto se adaptara la base de datos del reconocimiento facial a las condiciones del sistema de seguridad según los requerimiento compilados en la fase I, se definirá cómo se llevará a cabo la comunicación entre ambos sistemas, se diseñara e implementara un entorno amigable para usar el sistema.

4.3.1 Adaptación de la Base de Datos

Para el sistema de seguridad la base de datos a utilizar será una adaptación de la base de datos del reconocimiento facial, para esto se asoció la entidad persona a su respectivo tipo (estudiante, personal administrativo, obrero), se definieron los registros necesarios que se deben tomar en cuenta tanto de las personas con acceso y sin acceso para llevar un control de las personas que se mueven dentro y fuera de las instalaciones.

Figura 31. Modelo ER del sistema de seguridad



Fuente: Falcón y Mendoza (2018).

4.3.1.1 Descripción de las Entidades del Modelo

La base de datos planteado posee un conjunto de entidades y condiciones que se debe tomar en cuenta al momento de implementar, para esto se describirán cada una para un mejor entendimiento:

- **Persona:** Es toda aquella persona que posee algún tipo de autorización para entrar a las instalaciones, se relaciones con **PObrero, PAdm** y **Estudiante,** la entidad posee una variable llamada **tipo** la cual nos indicará con qué otra entidad se relaciona:
  a. **tipo = estudiante (0) -> Estudiante**
  b. **tipo = padm(1) -> PAdm**
  c. **tipo = pobrero(2) -> PObrero**
- **Estudiante:** Contiene la información adicional sobre los estudios que actualmente está cursando una entidad **Persona** del **tipo estudiante (0).**
- **PAdm:** Contiene toda la información administrativa de una entidad **Persona** del **tipo padm (1).**
- **PObrero:** Contiene toda la información administrativa de una entidad **Persona** del **tipo pobrero (2).**
- **Usuarios:** Es toda aquella entidad que posea algún tipo de autorización dentro del sistema, requiere tener un nombre de usuario, así como de una contraseña, los tipos de usuarios pueden ser:

  *** Tipo = adm (0) -> Administrador del sistema.** Puede crear cualquier otro tipo de cuenta.

  *** Tipo = gaceta (1) -> Módulos de vigilancia.** Su función consta en mantener el registro de acceso a las instalaciones. Puede indicar si la **Persona** posee un equipo electrónico al momento de ingresar. Puede agregar una entidad **NoRegistrado** en caso de que la persona no tenga acceso a las instalaciones. No puede crear otros usuarios.

*** Tipo = registro (2) -> Módulo de registro con autorización.** Su función consta en introducir nuevas **Personas** a la base de datos. No puede crear otros usuarios.

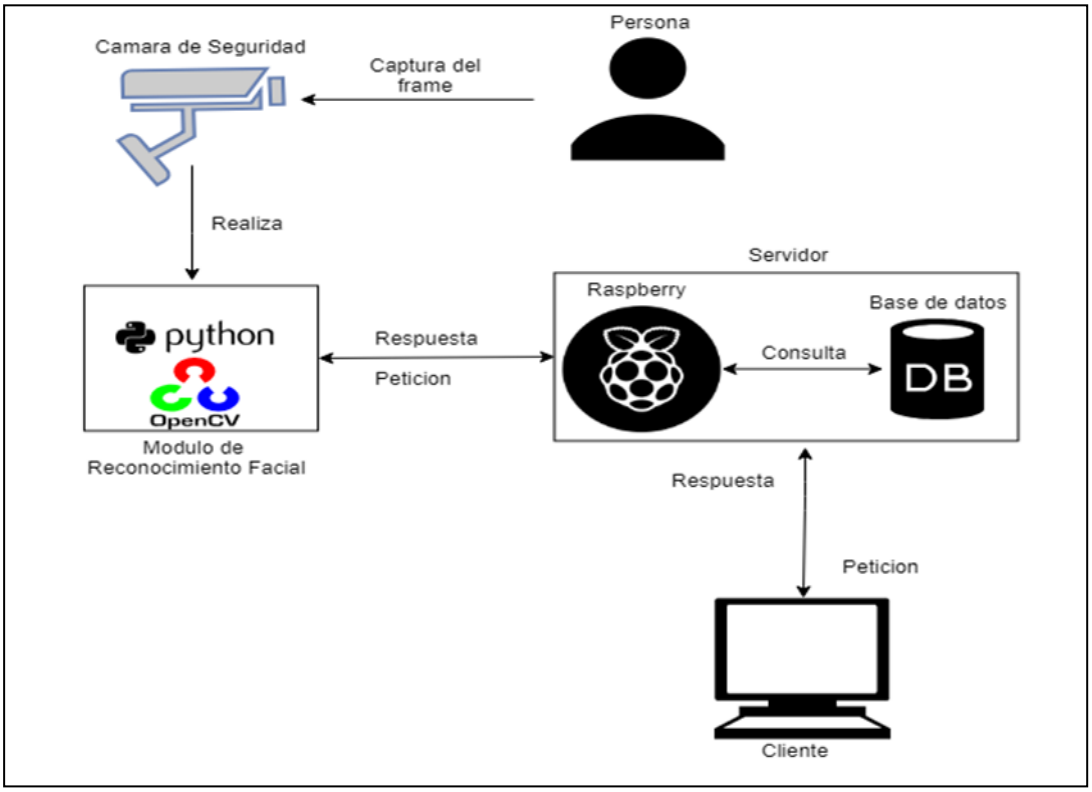
- **RegistroInformacion:** Almacena la información del acceso y salida de las instalaciones de una entidad Persona a través de uno de los módulos de vigilancia.

- **NoRegistrado:** Entidad de entidades que no posean autorización para entrar a las instalaciones, está almacenará el nombre, apellido, cedula, motivo de la visita y si posee algún equipo electrónico o de otro tipo al momento de ingresar a través de uno de los módulos de vigilancia.
- **Registro:** Es una entidad especial de sincronización para la entrada y salida de una entidad **Persona** para evitar accesos repetidos de una misma persona múltiples veces sin ninguna salida. Posee dos variables de tipo *bool* entro y salio para llevar a cabo esta sincronización.

4.3.2 Comunicación Entre Sistemas

La comunicación entre los sistemas es posible gracias a que esta se comunica a través de una red *LAN* privada, exclusiva y centralizada siendo el servidor central el *Raspberry Pi* en el cual se almacena la base de datos y los clientes son todas las cámaras que se encuentren dentro de la red. Los clientes realizan consultas al servidor central para obtener un reconocimiento efectivo y poder registrarlos para llevar un seguimiento.En la figura 32 se observa como a traves del diagrama como es la comunicación entre los sistemas.

Figura 32. Diagrama de Comunicación Entre Sistemas

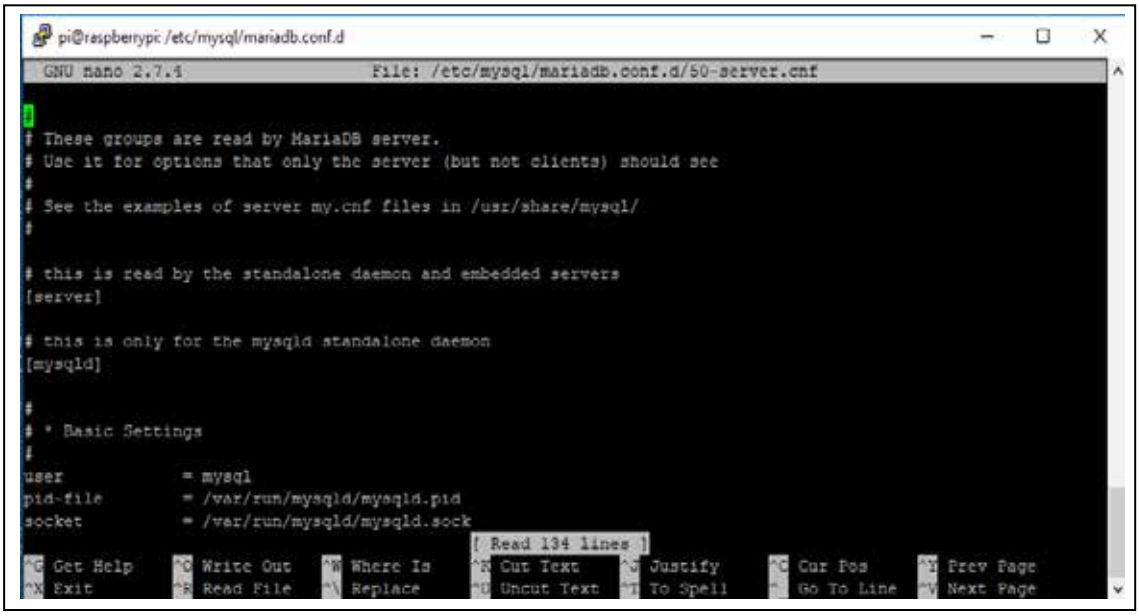


Fuente: Falcón y Mendoza (2018).

Debido a que el sistema se intercomunica a través de una *LAN* el acceso hacia el servidor, debe poder acceder desde diversos puntos, es decir, debe poseer acceso remoto desde otras direcciones dentro de la misma red, para esto se debe configurar el archivo 50-server.cnf con el siguiente comando:

- sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf

Figura 33. Archivo de configuración 50-server.cnf

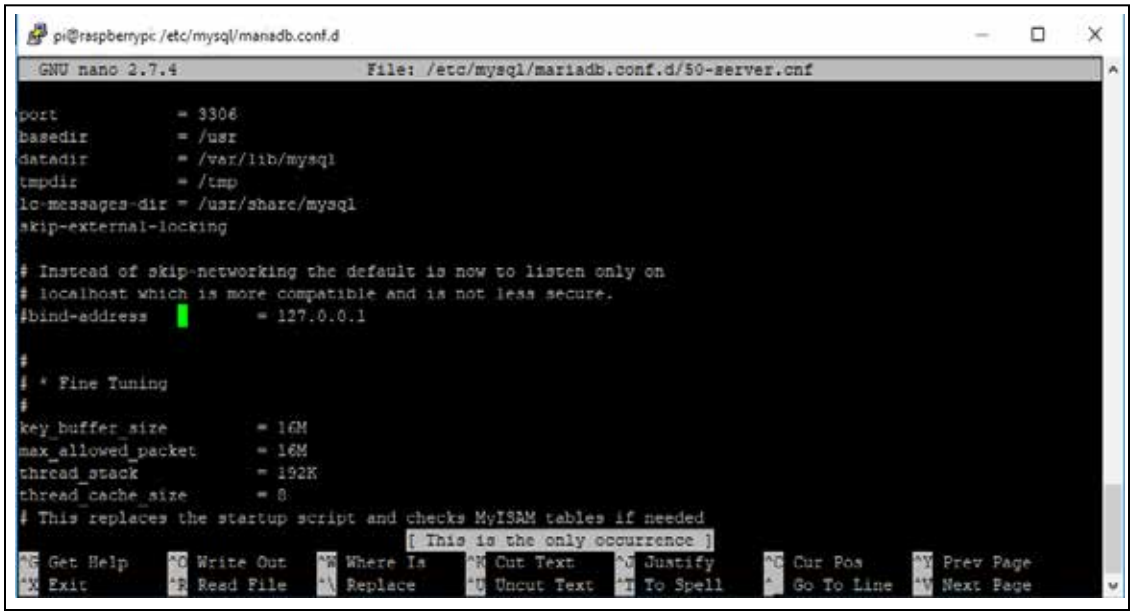


Fuente: Falcón y Mendoza (2018).

Una vez ubicado el archivo se debe comentar la línea *bind-addres = 127.0.0.1*, esa línea de configuración permite acceso a la base de datos solamente a traves de la direccion *127.0.0.1* la cual es la propia máquina, al comentar la linea se está permitiendo a cualquier *IPV4* pueda acceder a la base de datos. Para encontrar con mucha más facilidad la línea requerida puede realizarse lo siguiente:

- Activar la búsqueda de cadenas dentro de *nano* con *ctrl + w*, una vez se escribe *"bind"* y este nos llevara a la línea que coincida con la búsqueda, una vez ahí se procede a comentar y guardar los cambios.

Figura 34. Configuración para la conexión remota con el servidor.



Fuente: Falcón y Mendoza (2018).

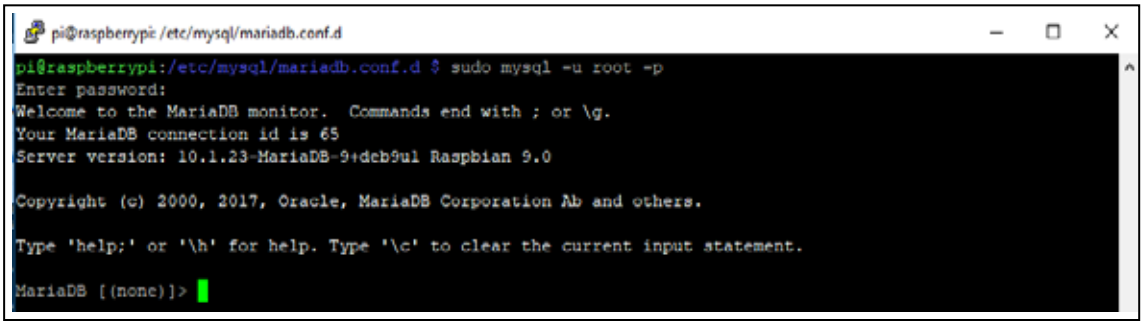
Una vez guardado el archivo es recomendable reiniciar los servicios del servidor con los siguientes comandos:

- sudo service apache2 restart
- sudo service mysql restart

Las bases de datos a pesar de tener habilitado el acceso remoto estos no poseen ningún privilegio con las bases de datos almacenadas en este, para esto se debe otorgar privilegios a los distintos usuarios que accedan al sistema, para otorgar privilegios a los usuarios se debe acceder a la consola de *mysql* a través del comando el cual a ser ejecutado pedirá la clave de acceso la cual fue configurado cuando se instaló *mysql:*

- sudo mysql -u root –p

Figura 35. Accediendo a la consola *mysql.*

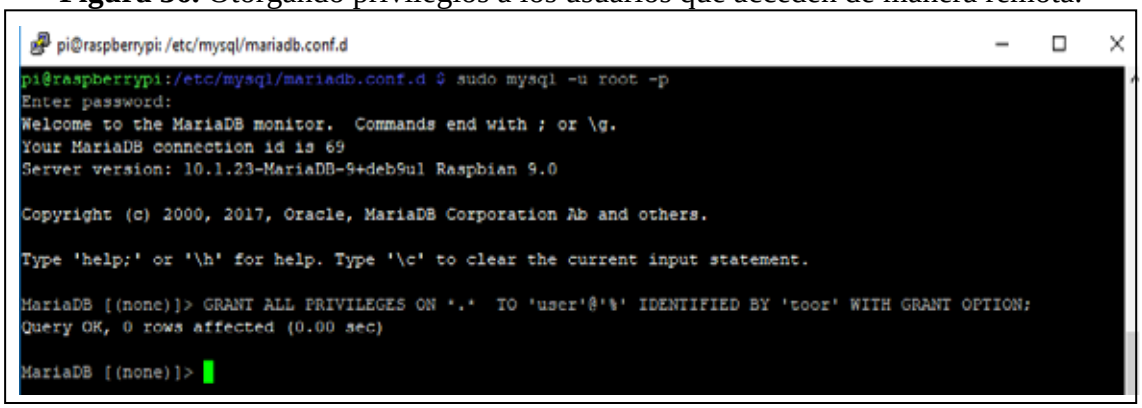


Fuente: Falcón y Mendoza (2018).

Una vez dentro de la consola se deberá otorgar los privilegios utilizando el siguiente comando:

- GRANT ALL PRIVILEGES ON *.* TO 'user'@'%' IDENTIFIED BY CLAVE 'WITH GRANT OPTION;

Figura 36. Otorgando privilegios a los usuarios que acceden de manera remota.



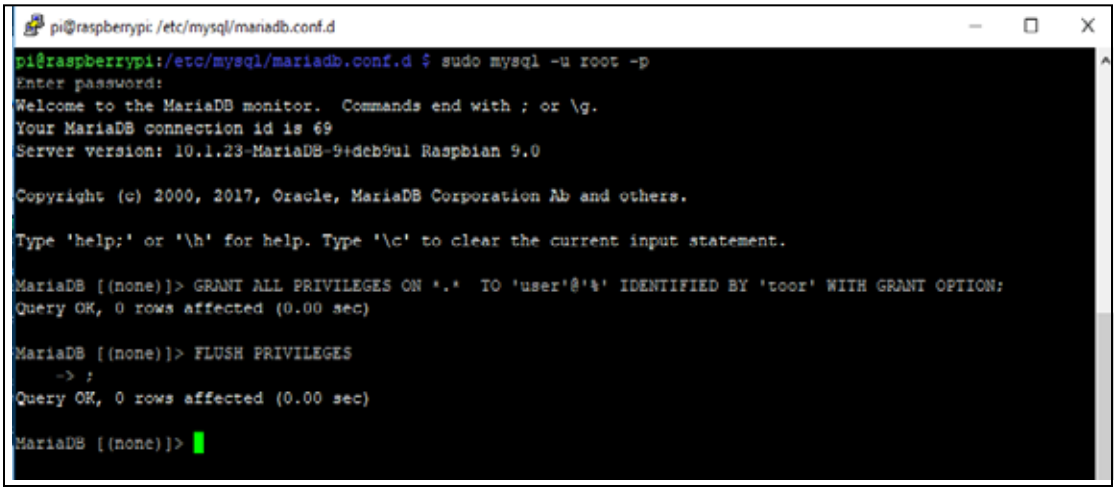
Fuente: Falcón y Mendoza (2018).

Este comando otorgó todos los privilegios en todas las bases de datos al usuario "*user*" conectado desde el comodín "%" (el comodín hace referencia a cualquier *IP*

dentro de la red *LAN)* identificado con la clave “CLAVE”. Una vez el comando se haya ejecutado es momento de aplicar los cambios con el siguiente comando:

- FLUSH PRIVILEGES;

Figura 37. Aplicando los cambios de los privilegios.

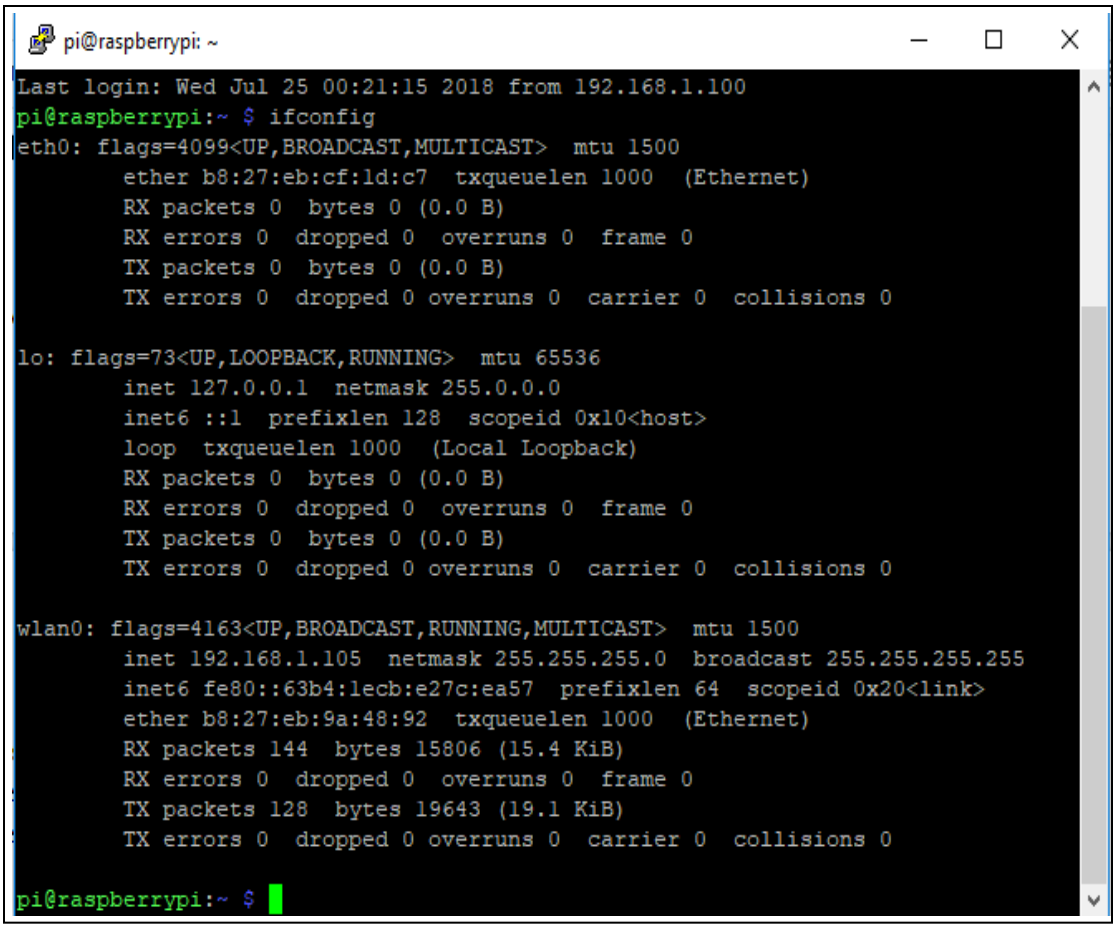


Fuente: Falcón y Mendoza (2018).

Ya los distintos usuarios que acceden al servidor a través de una conexión remota tienen todos los privilegios, la conexión con el servidor se realiza a través de la dirección *IP* del dispositivo, por esto, la dirección del dispositivo debe mantenerse estática para evitar problemas de acceso cada vez que el dispositivo se conecte a la red. Para realizar esto lo primero que se debe realizar es observar las interfaces de red disponibles con el siguiente comando:

- Ifconfig

Figura 38. Interfaces de red

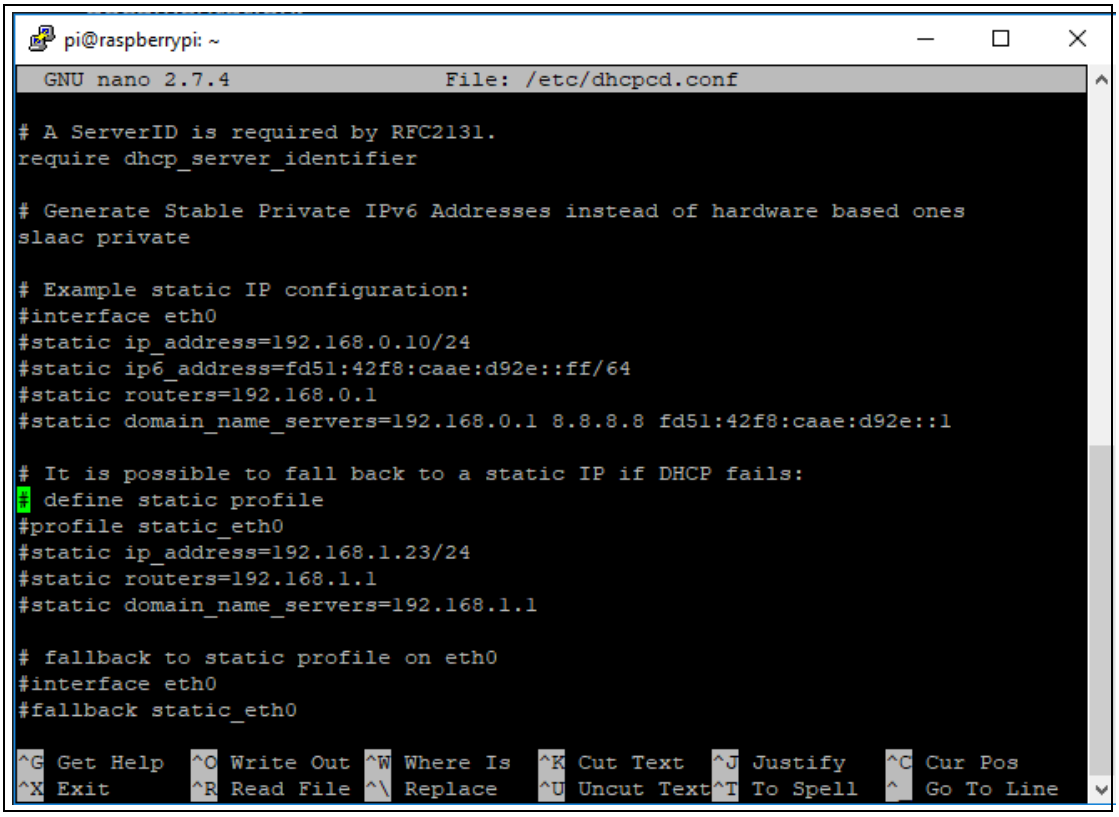


Fuente: Falcón y Mendoza (2018).

Ubicando la interfaz *wlan0* con una conexión establecida se procede a realizar la configuración del archivo *dhcpcd.conf* (Figura38) utilizando la dirección *IP* que nos muestre la interfaz en caso de querer utilizar la misma dirección, para acceder al archivo se utiliza el siguiente comando:

- sudo nano /etc/dhcpcd.conf

Figura 39. Archivo de configuración *dhcpcd.conf*.



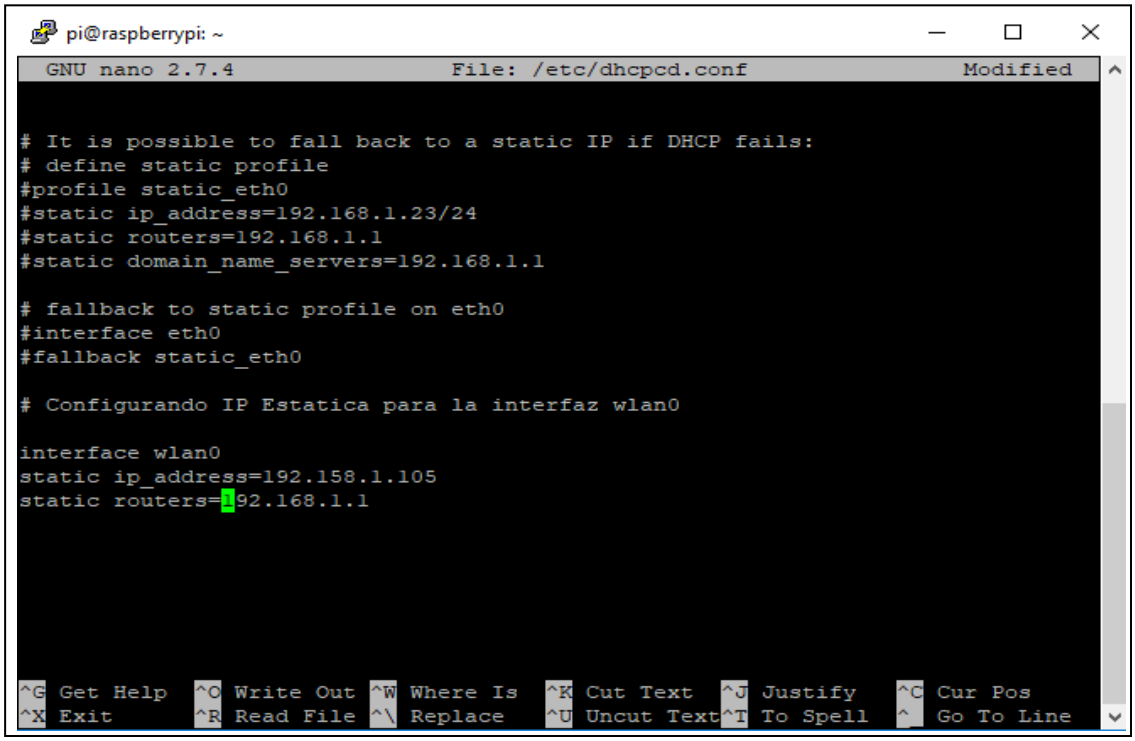
Fuente: Falcón y Mendoza (2018).

Para definir la *IP* estática dentro en la interfaz *wlan0* se debe configurar el archivo de la siguiente manera:

- Interface INTERFAZ
- Static ip_address=IP
- Static routers=ROUTER

Donde INTERFAZ es la interfaz de red a la cual se le asignará la *IP* estática, IP es la dirección que tiene el dispositivo dentro de la INTERFAZ y ROUTER es la dirección del router, configurando el archivo obtenemos lo que se muestra en la figura 39.

Figura 40. Archivo configurado para la *IP* estática del *Raspberry*



Fuente: Falcón y Mendoza (2018).

4.3.3 Implementación del sistema.

Para llevar a cabo una implementación ordenada para un mejor control de errores y modificaciones sencillas se construyó una librería llamada *pycam* la cual es la encargada de todos el proceso del reconocimiento facial como de la creación de la base de datos a través de una clase denominada PyCam, adicionalmente toda la comunicación por vía *SSH* se realiza a través de una clase denominada *SSHConnection* dentro de la misma librería. Por último se desarrolló un sistema *web* en conjunto con la librería *flask* para generar la *GUI.*

4.3.3.1 Definición de clase PyCam.

Esta clase se desarrolló para llevar todo el control de los procesos relacionados a al reconocimiento facial, para lograr esto la clase se define con los siguientes atributos y funciones (véase Anexo C para observar el código de la clase).

Atributos:

- Usuario : Nombre del usuario asociado a la camara
- Video : Entrada de video a utilizar, puede ser una cámara web o una cámara IP
- Out : Nombre del video y formato en el que se guardará todo el registro de las cámaras
- Database: Encargada de la configuración con la base de datos
- Entrada: Indica si la cámara sirve de entrada (1) o salida (0)
- Face_recognizer: Reconocedor a utilizar
- _tipo_persona : Diccionario de asociación numérica con respecto al tipo de persona
- _tipo_usuario : Diccionario de asociación numérica con respecto al tipo de usuario
- _initial_sql : Query inicial para crear la base de datos del sistema
- Face_cascade : Clasificador a utilizar
- Fourcc : Formato del video
- File_name : Nombre del archivo de video
- Host : Dirección a utilizar para la conexión *SSH*
- Username : Nombre de usuario para la conexión *SSH*
- Pw: Clave del usuario para la conexión *SSH*

Funciones:

- __init__(self,train,usuario,entrada) : Constructor de la clase, los parámetros son *self* el cual hace referencia al propio objeto, *train* hace referencia a si el objeto ya posee un archivo de entrenamiento o no, usuario es el nombre del usuario al cual está asociado la cámara y entrada indica si la cámara es utilizada en la entrada o la salida
- __del__ (self): Destructor libera el uso de la cámara, cierra la conexion con la base de datos y guarda el video en una direccion específica, sus parámetros son *self* el cual hace referencia al propio objeto.
- getFrame(self) : Función que analiza los *frames* de video y hace las búsquedas necesarias para el reconocimiento. Retorna el *frame* procesado en formato *bytes* y en formato original
- create_database(host,username,password,dbname,sql): Funcion que permite crear una nueva base de datos, los parámetros son *host* el cual indica dónde se encuentra la base de datos alojada, *username* el usuario de acceso para poder conectarse a la base de datos, *password* clave de acceso para el usuario, *dbname* nombre de la base de datos a utilizarse, *sql* indica el query inicial de la base de datos.
- connect_database(self,host,username,password,dbname): Funcion para conectarse a la base de datos, los parámetros son *self* el cual hace referencia al propio objeto, *host* el cual indica dónde se encuentra la base de datos alojada, *username* el usuario de acceso para poder conectarse a la base de datos, *password* clave de acceso para el usuario, *dbname* nombre de la base de datos a utilizarse
- close_database(self): Función que cierra la conexión con la base de datos, los parámetros son *self* el cual hace referencia al propio objeto
- record_face(self,nombre, apellido, cedula,tipo,carrera,semestre,sueldo,costo_hora): Función que permite

registrar nuevas personas con cierto acceso, los parámetros son *self* el cual hace referencia al propio objeto, nombre, apellido, cedula hacen referencia a los datos de la persona a registrar, tipo hace referencia a si la persona es estudiante, personal administrativo u obrero, carrera y semestre son parámetros opcionales utilizados sólo si la persona es estudiante, sueldo es un parámetro adicional solo si la persona es personal obrero y costo_hora es un parámetro adicional solo si la persona es personal administrativo

- generate_seed_users (self): Función que genera un conjunto de usuarios predeterminados, su parámetro es *self* el cual hace referencia al propio objeto. Esta función se utiliza para crear un usuario tipo administrador, dos tipos gacetas, uno tipo registro y una persona ficticia la cual será utilizada para la relación de las personas no registradas
- detect_face(img): Función que detecta rostros, su parámetro es *img* el cual hace referencia al *frame* que se desea analizar, retorna el *frame* modificado con la detección facial
- train_data(): Función que genera el archivo de entrenamiento

4.3.3.2 Definición de la clase SSHConnection.

Esta clase se desarrolló para llevar todo el control de los procesos relacionados a las transferencias de archivos *SSH* entre los sistemas, para lograr esto la clase se define con los siguientes atributos y funciones (véase Anexo D para observar el código de la clase).

Atributos:

- Sftp : Objeto de la conexión *SFTP* para la transferencia de archivos
- Sftp_open: Indica si la conexión ha sido abierta o no
- Transport: Objeto que indica las preferencias de seguridad de un transporte *SSH*

Funciones:

- __init__(self, host, username, password, port): Constructor de la clase, los parámetros son *self* el cual hace referencia al propio objeto, *host* indica la dirección de la máquina remota, *username* y *password* hacen referencia a los datos de acceso usuario y clave para la máquina remota, *port* indica el puerto en el cual se trabajara
- _openSFTPConnection(self): Funcion que abre la conexion *SFTP* para la transferencia de archivos
- get(self, remote_path, local_path): Función para transferir archivos de la máquina remota a la máquina local, los parámetros son *self* que hace referencia al propio objeto, *remote_path* el cual indica la ruta del archivo a transferir en la máquina remota y *local_path* indica la ruta en donde se copiara el archivo en la máquina local
- put(self, local_path, remote_path): Función para transferir archivos de la máquina local a la máquina remota, los parámetros son *self* que hace referencia al propio objeto, *local_path* el cual indica la ruta del archivo a transferir en la máquina local y *remote_path* indica la ruta en donde se copiara el archivo en la máquina remota
- create_folder(self,remote_path): Funcion que crea directorios utilizando la conexión *SFTP* donde los parámetros son *self* el cual hace referencia al propio objeto y *remote_path* es la ruta del nuevo directorio a crear
- close(self): Función que cierra todas las conexiones

4.3.3.3 Desarrollo de la Interfaz Web

Usando *flask* y sus conjuntos de componentes se ha desarrollado una GUI para que los usuarios pueda acceder al sistema y comprobar su funcionalidad, el sistema de seguridad como se recalcó en el diseño de la base de datos consta de tres tipos de usuarios que son, administrador, gaceta y registro cada una con su funcionalidad

particular cada usuario posee su propio alcance dentro del sistema por lo cual se definieron módulos para cada tipo de la siguiente manera:

1. Administrador: El administrador es el único usuario que puede crear otros usuarios, puede ver los registros de seguridad, así como de las personas que se encuentran dentro del sistema y posee acceso de poder registrar personas al sistema. Adicionalmente este usuario es el encargado de generar el archivo de entrenamiento que se utilizan para el reconocimiento facial.
2. Gaceta: La gaceta consta de un conjunto de funcionalidades limitadas, cada usuario de gaceta posee una relación con las cámaras ubicadas en los puntos críticos mencionados con anterioridad las cuales son las encargadas de monitorear el registro de acceso a las instalaciones. A su vez este tipo de usuario puede registrar personas que no se encuentren registradas dentro del sistema con la opción de asociarla con una persona que sí posea acceso.
3. Registro: El usuario registro es el encargado de registrar nuevas personas en el sistema, para esto consta de un módulo donde puede registrar ya sea a un estudiante, personal administrativo u obrero.

Para observar la GUI diseñada se recomienda visitar el Anexo E.

4.4 Fase IV: Realizar las pruebas para evaluar el Rendimiento.

Las pruebas realizadas durante el proyecto se fueron realizando a medida que se iban desarrollando a medida que este tomaba forma, desde un principio no se trataron las pruebas de una manera aislada a la implementación. Debido a esto el proyecto empezó arrojar resultados desde el comienzo, los resultados de las pruebas durante todo el proceso comenzaron con resultados de menor grado que sirvieron como guía para ir adaptando el proyecto hasta culminar con el sistema completo. Las pruebas fueron realizadas dentro de las instalaciones de la Universidad José Antonio Páez.

Estas se basaron en dos aspectos importantes los cuales son necesarios para obtener la mejor confiabilidad de los resultados, el primer conjunto de pruebas consistió en el registro de personas nuevas al sistema y la cantidad de imágenes necesarias para obtener la mejor precisión al momento del reconocimiento, el segundo conjunto consistió en observar la eficiencia del reconocimiento en el entorno de los puntos críticos. Cabe destacar que ambos conjuntos se realizaron bajo distintas condiciones de luz no controladas.

4.4.1 Pruebas de Registro

El sistema de registro consiste en otorgar los datos necesarios de la persona a la cual se le concederá el acceso y obtener los positivos necesarios de cada individuo para actuar en el reconocimiento facial. Estas pruebas se llevaron a cabo dentro del salón de técnicos en el quinto piso del edificio de odontología de la Universidad José Antonio Páez, la prueba consistió en registrar una cantidad de usuarios obteniendo una cierta cantidad de positivos y probar el reconocer en el mismo sitio.

El reconocedor al necesitar una cierta cantidad de positivos de cada individuo se probaron distintas cantidades y se probaron con el reconocedor para conocer la exactitud que esta tiene, al registrar a la persona se le cree una carpeta personal en la cual se encontraron los positivos de esta, estas carpetas tienen el formato **Px** donde **x** es el ***id*** que tiene la persona en la base de datos.

Figura 41. Registro de cuatro personas en el sistema.

Fuente: Falcón y Mendoza (2018).

Figura 42. Carpeta de positivos de cada persona en el sistema.

Fuente: Falcón y Mendoza (2018).

Como se mencionó anteriormente el registro guarda una cantidad **n** de positivos, esta cantidad **n** se varió para comprobar la efectividad que implica tener pocos a muchos positivos para los entrenamientos del reconocedor y el tiempo que le toma al registro capturar los **n** positivos del individuo, así como el tiempo. A continuación, se muestra en la Tabla 1 las distintas cantidades de positivos. Dado que el tiempo tomado al momento de captar los positivos de cada persona, al momento del registro varió estos se concentraron en un rango de tiempo de acuerdo a la cantidad n de muestras obteniendo así un valor mínimo y un valor máximo de tiempo de duración.

Tabla 1. Distintas cantidades de positivos

N	**Duración**	**Tamaño**	**Efectividad**
30	5 - 10 segundos	100 KB	E1
100	30 - 40 segundos	1.10 MB	E2
200	1 minuto y medio	2.27 MB	E3
500	3 - 5 minutos	4.50 MB	E4

Fuente: Falcón y Mendoza (2018).

La efectividad para cada caso se ve afectada por las condiciones de luz del entorno, la inclinación y la diferencia de cómo se tomaron los positivos, de esta manera cada efectividad de **n** se definió de la siguiente manera:

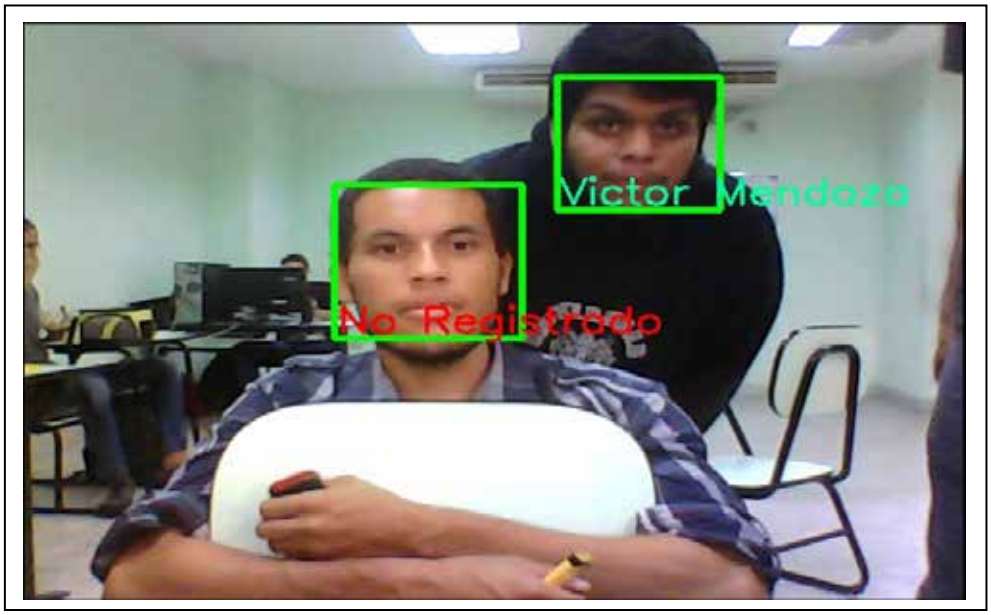
- **E1:** La efectividad E1 para el caso **n = 30** resulto ser útil siempre y cuando no se encuentren muchos usuarios en el mismo frame ya que puede que más de una persona coincida con alguien distinto debido a los pocos positivos que se tiene de cada individuo.
- **E2:** La efectividad E2 para el caso **n = 100** fueron mejores que el caso anterior obteniendo precisiones más acertadas, pero con un margen de error considerable de acuerdo a las expresiones de cada positivo.
- **E3:** La efectividad E3 para el caso **n = 200** resulto ser bastante eficiente logrando reconocer efectivamente a distintas personas en el mismo frame siempre y cuando las condiciones sean las más ideales posibles.
- **E4:** La efectividad E4 para el caso **n = 500** resultó ser eficiente ya que se tiene una cantidad alta de positivos de cada individuo, la deficiencia de esta recae en el hecho de que requiere un tiempo considerable al momento del registro lo cual no es lo ideal para el funcionamiento del sistema.

A continuación, se muestra en las Figuras 42,43 y 44 las pruebas del reconocedor dentro del mismo entorno utilizado los datos ingresados en el sistema:

Figura 43. Reconocimiento fallido debido a las condiciones de luz e inclinación. Dos registrados, ninguno detectado.

Fuente: Pruebas en Universidad José Antonio Páez

Figura 44. Reconocimiento con una aceptación del 50% debido a las condiciones de iluminación.



Fuente: Falcón y Mendoza (2018).

Figura 45. Reconocimiento parcialmente exitoso debido a las condiciones de movimiento, iluminación e inclinación.

Fuente: Falcón y Mendoza (2018).

4.4.2 Pruebas de Reconocimiento en Puntos Críticos

En esta sección de pruebas se verificó tanto el posicionamiento de las cámaras, como esta se ve afectada por la iluminación natural, así como también verificar que el sistema de seguridad lleve el registro de las personas que entren y salgan de las instalaciones indicando fecha, hora y el punto crítico donde se dio el registro. También se puso a prueba la función de la detección facial en un entorno donde el paso de personas es constante.

En la primera prueba realizada se puso a prueba la cantidad de personas capaces de ser reconocidos por el sistema en un ambiente normal. La prueba duró aproximadamente un minuto en el cual se observaron que entraron cinco personas de las cuales dos fueron detectadas como usuarios no registrados debido a que no se encuentran en el sistema.

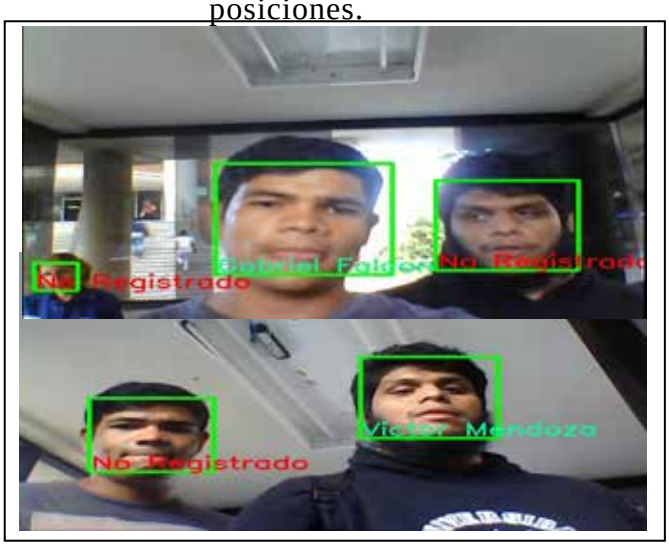
Cabe mencionar que las detecciones son más eficientes cuando la cámara se encuentra de frente a la persona, esta prueba se realizó sin tomar el caso ideal. Estas detecciones se pueden observar en la figura 45.

Figura 46. Detecciones faciales en un punto crítico.

Fuente: Falcón y Mendoza (2018).

La prueba subsiguiente se realizó dentro del módulo de vigilancia como se muestra en la figura 46, para conocer la efectividad del sistema para registrar la entrada aislando lo más posible la luz natural del entorno, adicionalmente la cámara se encontró en una posición lo más ideal posible para verificar efectividad del reconocedor. Adicionalmente se cambió la posición de la cámara con una cierta inclinación para observar qué efecto tendría.

Figura 47. Reconocimiento efectivo aislando la luz natural y usando diferentes posiciones.



Fuente Falcón y Mendoza (2018).

La siguiente prueba se realizó a contra de la luz natural que posee el entorno colocando la cámara en una posición más ideal como es en el caso de que esta se encuentre lo más de frente a la persona. También se realizó la prueba en donde la cámara se encuentra dentro del módulo de vigilancia apuntando hacia uno de los puntos críticos, con estas ambas pruebas se pudo observar la reacción del sistema ante los cambios de posicionamiento con respecto a la luz.

Figura 48. Comparación de cómo la luz natural afecta al sistema.

Fuente: Falcón y Mendoza (2018).

Por último, cabe mencionar que en todas las pruebas dentro de los puntos críticos las cámaras se comportaron tanto para registrar la entrada o la salida de las personas a las instalaciones. La cámara encargada del registro a la entrada se le apodó gaceta1 mientras que la cámara encargada del registro a la salida se le apodo gaceta2. Gracias a esto se pudo registrar los momentos en que los individuos con acceso se le registraron.

En la figura 48 se puede observar que un registro quedó sin salida indicando que la persona o no ha salido o no fue detectada en las pruebas. Cuando las persona una vez ingresado se le asigna como salida determinada la misma por la que entró, esto se debe se rea;ozp para evitar conflictos con la base de datos, cuando la persona una vez salga de las instalaciones se le actualiza la salida como se puede observar en el segundo registro.

Figura 49. Registro de entrada y salida a las instalaciones de los individuos.

nombre	usuario_entrada	usuario_salida	fecha_entrada	hora_entrada ▲ 1	fecha_salida	hora_salida ▲ 2
Gabriel	gaceta1	gaceta1	2018-07-26	15:56:06	*NULL*	*NULL*
Victor	gaceta1	gaceta2	2018-07-26	16:02:37	2018-07-26	16:10:50

Fuente: Falcón y Mendoza (2018).

CAPITULO V
CONLUSIONES Y RECOMENDACIONES

5.1 Conclusiones.

- Los requerimientos funcionales y no funcionales se realizaron, a través, de la recolección de datos obteniéndose que los puntos críticos más importantes son las vigilancias peatonales y vehiculares, el control de registro y la posibilidad de agregar personas al sistema para que estos puedan ser reconocidos a la hora de ingresar a la universidad.

- Los pilares del sistema fueron la detección y reconocimiento facial, los cuales se desarrollaron utilizando los algoritmos Haar Cascade y LBPH, siendo estos los que tienen mayor rendimiento, debido a los ajustes en variaciones de iluminación, dando como resultados a los usuarios un buen funcionamiento en condiciones ideales logrando detectar y reconocer los diferentes rostros sin importar las oclusiones que se presenten en este, a comparación de los resultados obtenidos en condiciones no ideales el cual afecta al momento de detectar y reconocer los rostros.

- El uso de cámaras IP requiere de la modificación de los algoritmos utilizados y mejores equipos de hardware, puesto que los utilizados no cumplen con el rendimiento esperado.

- La interfaz web integro la detección y reconocimiento facial en una página web amigable con el usuario de tal forma, que se puedan observar los registros de entrada y salida de los usuarios, así como el control de los invitados que ingresen a las instalaciones.

- El proyecto presentado no logro solventar el problema de la inseguridad en la Universidad José Antonio Páez debido a que existe una gran población a cubrir, lo cual conlleva, a realizar una gran cantidad de procesamiento por parte del sistema, generando demoras al momento de registrar la entrada y

salida de cada una de las personas haciendo esta opción no viable para la disminución de la inseguridad en las instalaciones.

- El reconocimiento facial, como sistema de seguridad no cumplió con los objetivos propuestos, debido a que el procesamiento necesario no se dispone con los recursos tecnológicos, económicos y algoritmos necesarios en el momento en que se realizó el presente trabajo.

- El archivo de entrenamiento que se utilizó para el reconocimiento suelen ser archivos muy pesados cuando se trabaja con un gran número de persona, la transferencia del archivo de entrenamiento a otras máquinas puede conllevar retrasos a la red.

5.2. Recomendaciones.

- Impartir dentro del pensum materias que traten el tema de Machine Learning y su usabilidad en conjunto con otros dispositivos ya sea en el área de la domótica u otras, esto debido a que las tecnologías actuales se han enfocado mucho en el avance de esta tecnología y sus aplicaciones en las distintas áreas.

- Adicionar dentro del pensum materias que traen el tema de las SBC como el raspberry y cómo estas pueden utilizar para diferentes proyectos ya que los dispositivos SBC han ganado popularidad en los últimos años y se han empezado a utilizar como centro de control en diversas áreas de la tecnología.

- Usar el sistema de reconocimiento para tratar temas como el pago de cuotas, asistencia de profesores y accesos a laboratorios.

- Desarrollar un detector que funcione para la detección de rostros de distintos ángulos y expresiones faciales para así conseguir una mayor precisión al momento de la detección.

- Para contrarrestar el paso del tiempo en las personas se puede adaptar al reconocedor para que este actualice los positivos de las personas a medida que entra para siempre tener positivos actuales de las personas.

- Implementar un sistema de seguimiento facial utilizando redes neuronales a través de las distintas cámaras ubicadas en las instalaciones para una mayor seguridad.

- Utilizar una red segura y privada limitando su acceso solo a los dispositivos que se encuentren dentro del sistema para evitar posibles ataques y congestionamiento de red.

- Emplear técnicas de Machine Learning para contrarrestar los efectos de la luz al momento de la detección y reconocimiento.

- Se da advertencia a la universidad José Antonio Páez que si alguna empresa de seguridad llega a ofrecer un sistema similar al presentado en esta investigación tomar en cuenta que la universidad no posee las condiciones de hardware y software necesarias para llevar a cabo una implementación de gran nivel.

BIBLIOGRAFIA

.

Amaya A. (2015). **Sistema alternativo de seguridad vehicular basado en reconocimiento facial.** Ambato: Ecuador.

Arias, F. (1999). **El proyecto de investigación: Introducción a la metodología científica.** 3ra Edición. Caracas: Editorial Episteme.

Arias, F. (2006). **El proyecto de investigación: Introducción a la metodología científica.** 5ta Edición. Caracas: Editorial Episteme.

Arias, M. (2008). **Definición de lenguaje de programación**. Recuperado en: http://catedraprogramacion.forosactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos

Booch G. (1996). **Análisis de Diseño Orientado a Objetos con Aplicaciones**.2ª Edición. S.A. Editorial Alhambra Mexicana.

Buendia, L.; Colas, P. y Hernández, F. (1998). **Métodos de Investigación en Psicopedagogía.** Madrid: McGraw-Hill.

Bradski G. y Kaehler A. (2008). **Learning OpenCV: Computer vision with the OpenCV library**. O'Reilly Media, Inc.

Cajas M. y Viri P. (2017). **Diseño e implementación de un sistema de seguridad vehicular mediante reconocimiento facial a través de visión artificial**. Universidad Politécnica Salesiana. Cuenca-Ecuador.

Dubs de Moya, R. (2002). El **Proyecto Factible: una modalidad de investigación.** Caracas, Venezuela.

Espinoza D. y Jorguera P. (2015) **Reconocimiento facial**. Universidad católica de Valparaíso. Chile.

González M. e Infante. M. (2017). **Construcción de un prototipo de robot social basado en un miniordenador raspberry pi 3 y arduino. Universidad José Antonio Páez**. Venezuela.

Gonzalez, R. (2015). **Phyton para** 1era Edición. España: Autoedición

Hurtado, J. (2000). **Metodología de Investigación Holística (3ª. Ed.)**. Caracas: Editorial Sypal.

Jain, A.; Ross, A. y Nandakumar, K. (2011). Introduction to Biometrics. EEUU: Editorial Springer.Mijares, H; García, L. (2007). **Normas para la Elaboración y Presentación de los Anteproyectos, Proyectos y Trabajos de Grado**. Carabobo, Venezuela.

Leblanc, S. (2018). **¿Qué es la técnica de iluminación rebrand y como se utiliza?** Recuperado en: https://mott.pe/noticias/que-es-la-tecnica-de-iluminacion-rembrandt-y-como-se-utiliza/

Léniz, J. (2014). **25 trucos para salir mejor en las fotos**. Recuperado en: https://www.cutypaste.com/estilo-de-vida/fotografia/25-trucos-para-salir-mejor-en-las-fotos/

López, M. (2015). **Un nuevo algoritmo para reconocer rostro**. Recuperado en: https://www.unocero.com/ciencia/un-nuevo-algoritmo-para-reconocer-rostros/

Logan S. (2016). **Python Programming Course: Learn the Crash Course to Learning the Basics of Python.** United States: Createspace Independent Publishing Platform.

Palella, S. y Martins, F. (2010). **Metodología de la investigación cuantitativa.** Caracas: Editorial Fedupel.

Pressman R. (2006). **Ingeniería del Software: Un Enfoque Práctico.** 7ma Edición. Editorial McGraw-Hill.

Sabino, C. (1996). **Introducción a la Metodología de Investigación**. Caracas: Editorial: Panapo.

Sierra Bravo R. (1984). **Técnicas de investigación Social Teoría y ejercicios**. Madrid: Editorial Paraninfo.

Trespalacios, J.; Vázquez, R y Bello L. (2.005). **Investigación de Mercados**. España: Editorial International Thomson Editores.

Tamayo, M. (1998). **El proceso de la investigación científica.** 3ra edición. México: Editorial Limusa.

ANEXOS

GLOSARIO

D

Dhcpcd: Es una implementación del cliente DHCP que obtiene la host información (dirección IP, rutas, etc.) desde un servidor DHCP y configura la red interfaz de la máquina en la que se está ejecutando.

F

Flask: Es un microframework para Python, que proporciona un núcleo funcional y conciso para su aplicación.

Framework: Es una estructura estratificada que indica qué tipo de programas pueden o deben construirse y cómo se interrelacionarán.

H

Haar Cascade: Es básicamente un clasificador que se utiliza para detectar el objeto para el que se ha entrenado, desde la fuente.

Highgui: Es una aplicación a gran escala para que las funcionalidades de los marcos de interfaz trabajen rápidamente y se puedan visualizar los resultados.

I

IP: Es una dirección única que dispositivos informáticos, como ordenadores personales, tabletas y teléfonos inteligentes utilizan para identificarse y comunicarse con otros dispositivos en la red IP.

L

LBPH: Es un operador de textura simple pero muy eficiente que etiqueta los píxeles de una imagen mediante el umbral de la vecindad de cada píxel y considera el resultado como un número binario.

Localhost: Es el nombre predeterminado que describe la dirección de la computadora local también conocida como la dirección de bucle invertido.

M

Microframework: Es un framework que intenta proporcionar solo los componentes que son absolutamente necesarios, para que un desarrollador cree una aplicación o puede enfocarse en proporcionar la funcionalidad de un área en particular.

Modelo ER: Es un modelo de datos que consiste en un conjunto de objetos básicos llamados Entidades y sus respectivas relaciones entre sí, sirve para diseñar el esquema de la base de datos antes de desarrollarla.

MySQL: El un sistema de gestión de bases de datos SQL de código abierto.

P

Pip: Es un sistema de administración de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

Phyton: Es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica.

Q

Query: Es una solicitud de datos o información de una tabla de base de datos o combinación de tablas. Estos datos pueden generarse como resultados devueltos por Structured Query Language (SQL) o como gráficos, gráficos o resultados complejos.

Raspberry pi: Es una computadora de bajo costo con tamaño de tarjeta de crédito que se conecta a un monitor de computadora o TV y utiliza un teclado y mouse estándar.

R

Raspbian: Es el conjunto de programas básicos y utilidades que hacen funcionar su Raspberry Pi.

Red LAN: Es una red que conecta los ordenadores en un área relativamente pequeña y predeterminada

S

SBC: Es una computadora completa construida en una sola placa de circuito, con microprocesador, memoria, entrada / salida (E / S) y otras características requeridas de una computadora funcional.

SFTP: Es una versión segura del Protocolo de transferencia de archivos (FTP), que facilita el acceso a los datos y la transferencia de datos a través de un flujo de datos Secure Shell (SSH). Es parte del protocolo SSH.

SSH: Es un protocolo de red que proporciona a los administradores una forma segura de acceder a una computadora remota.

Streaming: Es una técnica para transferir datos para que pueda procesarse como un flujo constante y continuo.

ANEXO A

Código de implementación del detector Facial.

```python
#Se importa la libreria de opencv
import cv2

'''
Funcion que convierte una imagen en BGR a RGB
arg -> img <- Imagen a convertir
function convertToRGB(img)
return img convertida a RGB
'''

def convertToRGB(img):
    return cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

'''
Funcion que convierte una imagen en BGR a Gray
arg -> img <- Imagen a convertir
function convertToGrey(img)
return img convertida a Gray
'''
def convertToGray(img):
    return cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
'''
Funcion que detecta rostros en una imagen determinada
f_cascade -> clasificador a utilizar
colored_img -> Imagen con la que se trabajara
scaleFactor -> Factor de escala para la deteccion, por defecto tendra el valor de 1.1
retorna -> Copia de la imagen con las detecciones realizadas
'''

def detect_faces(f_cascade, colored_img, scaleFactor =1.1):
    # Se realiza una copia de la imagen
    img_copy = colored_img.copy()
    # Se convierte a formato Gray la copia de la imagen
    gray_img = convertToGray(img)
    # Utilizando el clasificador se procede a detectar rostros en la imagen
    faces = f_cascade.detectMultiScale(gray_img, scaleFactor=scaleFactor, minNeighbors=5)
    # Iteracion entre todas los rostros que se consiguieron en la imagen
    for(x,y,w,h) in faces:
        # Se dibuja un rectangulo en todos los rostros que hayan sido detectados
        cv2.rectangle(img_copy,(x,y),(x+w,y+h),(0,255,0),2)
    # Retorna la copia de la imagen original modificada
    return img_copy

# Imagen a la cual se detectara rostros
img = cv2.imread('new_test.png')
# Clasificador con el cual se trabajara, en nuestro caso se utilizara el haar cascade para rostros frontales
haar_face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
#Se realiza la deteccion de rostros y se guarda la copia de la imagen en la variable faces_detected_imgs
faces_detected_imgs = detect_faces(haar_face_cascade,img)
#Se muestra la imagen modificada por pantalla
cv2.imshow('Test',faces_detected_imgs)
```

Fuente: Falcón y Mendoza (2018).

ANEXO B.

Código de implementación del reconocedor facial

A-1. Creación de la base de datos

```
# Se importa la libreria sqlite3 para python
import sqlite3

# Se realiza la conexion a una base de datos llamada database
conn = sqlite3.connect('database.db')

# Se hace uso del cursor para poder manipular la base de datos
c = conn.cursor()

# Se genera un query inicial para la base de datos
sql = """
DROP TABLE IF EXISTS persona;
CREATE TABLE persona(
                    id integer unique primary key autoincrement,
                    nombre varchar(50),
                    apellido varchar(50),
                    cedula varchar(50)
);
"""

# Haciendo uso del cursor se ejecuta el query para crear la base de datos
c.executescript(sql)
# Se guardan los cambios realizados en la base de datos
conn.commit()
# Se cierra la conexion con la base de datos
conn.close()
```

Fuente: Falcón y Mendoza (2018).

A-2. Registro de personas

```
# Se importan todas las librerias necesarias
# opencv -> Manejo y procesamiento de imagenes
# os -> Libreria de manipulacion sistema
import cv2
import os

# Se realiza la conexion con la base de datos
conn = sqlite3.connect('database.db')

# Se verifica si exis
if not os.path.exists('./dataset'):
    os.makedirs('./dataset')

# # Se hace uso del cursor para poder manipular la base de datos
c = conn.cursor()

# Se seleccion el clasificador de rostros frontales haarcascade
face_Cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Se realiza la video captura de la camara web
cap = cv2.VideoCapture(0)

# Se piden los datos de ingreso de la persona
uname = input("Ingrese su nombre")
uname2 = input("Ingrese su apellido")
ucedula = input("Ingrese cedula")

# Se ejecuta un query en la base de datos para agregar una nueva persona
c.execute('INSERT INTO persona (nombre,apellido,cedula) VALUES (?);',(uname,uname2,ucedula))

# Se toma el id de la persona recien agregada para crear la carpeta utilizara para el entrenamiento
uid = c.lastrowid
```

Fuente: Falcón y Mendoza (2018).

A-3. Registro de persona

```
# Contador de los positivos que se tomaran
sampleNum = 0

# Ciclo que obtendra 20 positivos de la persona que se esta agregando al sistema
while True:
    '''
        Se lee un frame del video captura
        ret -> Respuesta de la lectura True si fue satisfactoria de lo contrario False
        img -> Imagen con la que se trabajara
    '''
    ret,img = cap.read()
    # Si la lectura fue satisfactoria se procede a guardar los positivos
    if ret:
        # Se transforma la imagen original a formato Gray
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        # Utilizando el clasificador se detectan rostros
        faces = face_Cascade.detectMultiScale(gray,1.3,5)
        # Para todos los rostros detectados
        for (x,y,w,h) in faces:
            # Se incrementa el numero de muestra positivo
            sampleNum = sampleNum + 1
            # Se guarda el positivo en una carpeta con el formato dataset/User.(numero id del query).(numero de la muetra).jpg
            cv2.imwrite("dataset/User."+str(uid)+"."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            cv2.waitKey(100)
        cv2.imshow('img',img)
    cv2.waitKey(1)
    if sampleNum > 20:
        # Una vez se hayan tomado todos los positivos concluye la recoleccion de positivos
        break
# Se libera la camara web
cap.release()
# Se ejectuan los cambios realizados
conn.commit()
# Se cierra la conexion con la base de datos
conn.close()
cv2.destroyAllWindows()
```

Fuente: Falcón y Mendoza (2018).

A-4. Creación de archivo de entrenamiento.

```
'''
    Se realizan las importaciones necesarias
    os -> Libreria para la manipulacion del sistema
    cv2 -> Libreria para el manejo y procesamiento de imagenes
    numpy -> Libreria para el computo cientifica
    PIL -> Libreria de imagenes para python
'''
import os
import cv2
import numpy as np
from PIL import Image

# Se crea el reconocedor utilizando el algoritmo LBPH
recognizer = cv2.face.LBPHFaceRecognizer_create()
# Nombre del directoria donde se encuentran los positivos con los cuales se entrenara
path = 'dataset'

# Se verifica si no existe una carpeta para el archivo de entrenamiento, en caso de no existir se crea
if not os.path.exists('./recognizer'):
    os.makedirs('./recognizer')
```

Fuente: Falcón y Mendoza (2018).

A-5. Creación de archivo de entrenamiento.

```
'''
    Funcion que crea la asociacion de los positivos con el id de la persona
    path -> ruta donde se encuentran los positivos de entrenamiento
    Retorna -> Un array tipo numpy de IDs y una lista de los positivos
'''
def getImagesWithId(path):
    # Se listan todas las rutas de las imagenes a utilizar
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    '''
        Se crean dos listas con las cuales se trabajaran
        faces -> Lista que almacenara los rostros
        IDs -> Lista que almacenara los id de las personas
    '''
    faces = []
    IDs = []
    # Se recorre cada ruta en imagesPahs
    for imagePath in imagePaths:
        # Se toma la imagen de la ruta selecciona para ser convertida en formato Gray
        faceImg = Image.open(imagePath).convert('L')
        # Se crea un array con la libreria numpy para relacionar las imagenes
        faceNp = np.array(faceImg,'uint8')
        # Utilizando la ruta se selecciona el ID correspondiente a los positivos
        ID = int(os.path.split(imagePath)[-1].split('.')[1])
        # Se almacena los valores en la lista faces
        faces.append(faceNp)
        # Se almacena los valores en la lista IDs
        IDs.append(ID)
        cv2.imshow("training",faceNp)
        cv2.waitKey(10)
    return np.array(IDs),faces

# Se relacionan todos los positivos con su respectivo ID
Ids,faces = getImagesWithId(path)
# Se entrena el reconocedor con los ID y positivos obtenidos anteriormente
recognizer.train(faces,Ids)
# Se guarda el archivo de entrenamiento para futuros usos
recognizer.save('recognizer/trainingData.yml')
cv2.destroyAllWindows()
```

Fuente: Falcón y Mendoza (2018).

A-6. Reconocedor usando el archivo de entrenamiento

```
'''
    Se realizan las importaciones necesarias
    os -> Libreria para la manipulacion del sistema
    cv2 -> Libreria para el manejo y procesamiento de imagenes
    numpy -> Libreria para el computo cientifica
    sqlite3 -> Libreria para manipular bases de datos tipo sqlite
'''
import cv2
import numpy as np
import sqlite3
import os

# Se realiza la conexion con la base de datos y se crea el cursor de la misma
conn = sqlite3.connect('database.db')
c = conn.cursor()

# Nombre del archivo de entrenamiento a utilizar
fname = "recognizer/trainingData.yml"

# Verifica si el archivo de entrenamiento existe si no manda una advertencia
if not os.path.isfile(fname):
    print("Entrene la data primero")
    exit(0)

# Se selecciona el clasificador de rostros frontales haarcascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Se inicia la video captura de la camara web
cap = cv2.VideoCapture(0)
# Se crea el reconocedor LBPH
recognizer = cv2.face.LBPHFaceRecognizer_create()
# Se entrena el reconocendor con el archivo de entrenamiento
recognizer.read(fname)
```

Fuente: Falcón y Mendoza (2018).

A-7. Reconocedor usando el archivo de entrenamiento

```
# Se inicia el reconocimiento facial en vivo
while True:
    '''
        Se lee un frame del video captura
        ret -> Respuesta de la lectura True si fue satisfactoria de lo contrario False
        img -> Imagen con la que se trabajara
    '''
    ret,img = cap.read()
    # Se transforma la imagen original a formato Gray
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    # Utilizando el clasificador se detectan rostros
    faces = face_cascade.detectMultiScale(gray,1.3,5)
    # Para todos los rostros detectados
    for (x,y,w,h) in faces:
        # Se crea un rectangulo en el rostro de la persona que fue detectada
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
        '''
            Se realiza la prediccion del rostro detectado utilizando el reconocedor
            ids -> Id de las personas que fueron reconocidas
            conf -> Confianza de la prediccion (entre menor sea el numero mas eficiente es)
        '''
        ids,conf = recognizer.predict(gray[y:y+h,x:x+w])
        # Se ejecuta el query en la base de datos para selecciona el nombre de las personas que fueron reconocidas
        c.execute("select nombre from persona where id = (?);",(ids,))
        # Se toman todas las respuestas dadas por el query
        result = c.fetchall()
        # Se selecciona el nombre de la persona que fue reconocida
        name = result[0][0]
        # Se verifica si la confiabilidad es menor a 50
        if conf < 50:
            # En caso de ser se le coloca el nombre de la persona en la imagen
            cv2.putText(img,name,(x+2,y+h-5),cv2.FONT_HERSHEY_SIMPLEX,1,(155,255,0),2)
        else:
            # En caos de que no se coloca que la persona no ha sido reconocida
            cv2.putText(img,'No Registrado',(x+2,y+h-5),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
    cv2.imshow('Reconocimiento Facial',img)
    k =cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Fuente: Falcón y Mendoza (2018).

ANEXO C

Código Clase PyCam

A-1

```
'''
Se importan las librerias necesarias
    cv2 -> Libreria para el manejo y procesamiento de imagenes
    os -> Libreria para la manipulacion del sistema
    MySQLdb -> Libreria para el manejo de la base de datos Mysql
    numpy -> Libreria para el computo cientifico
    datetime -> Libreria para el manejo de formatos de horas y fechas
    werkzeug.security -> Libreria paa la generacion de claves encriptadas (viene integrado con Flask)
    pycam.shh -> Libreria de la conexion SSH
'''

import cv2
import os
import MySQLdb
import numpy as np
import datetime
from werkzeug.security import generate_password_hash, check_password_hash
from pycam.ssh import SSHConnection

class PyCam(object):

    _tipo_persona = {0 : 'estudiante', 1 : 'padm' , 2 : 'pobrero'}

    _tipo_usuario = {0 : 'adm', 1 : 'gaceta', 2 : 'registro'}

    _initial_sql = [
    """create table IF NOT EXISTS persona(
        id integer AUTO_INCREMENT PRIMARY KEY,
        id_tipo integer not null,
        nombre varchar(50),
        apellido varchar(50),
        cedula varchar(10),
        tipo enum('estudiante','padm','pobrero')
    );""",


    """create table IF NOT EXISTS estudiante(
        id integer AUTO_INCREMENT PRIMARY KEY,
        carrera varchar(50),
        semestre varchar(50)
    );""",
    """create table IF NOT EXISTS pobrero(
        id integer AUTO_INCREMENT PRIMARY KEY,
        sueldo decimal(14,2)
    );""",
    """create table IF NOT EXISTS padm(
        id integer AUTO_INCREMENT PRIMARY KEY,
        costo_hora decimal(14,2)
    );""",
    """create table IF NOT EXISTS usuarios(
        id integer AUTO_INCREMENT PRIMARY KEY,
        usuario varchar(20),
        clave varchar(120),
        tipo enum('adm','gaceta','registro')
    );""",
    """create table IF NOT EXISTS registro(
        id integer AUTO_INCREMENT PRIMARY KEY,
        id_persona integer not null,
        entro bool,
        FOREIGN KEY (id_persona) REFERENCES persona(id)
    );""",
    """create table IF NOT EXISTS registroinformacion(
        id integer AUTO_INCREMENT PRIMARY KEY,
        id_persona integer not null,
        id_usuario_entrada integer not null,
        id_usuario_salida integer not null,
        fecha_entrada date,
        hora_entrada time,
        fecha_salida date,
        hora_salida time,
        equipo varchar(150),
        FOREIGN KEY (id_persona) REFERENCES persona(id),
        FOREIGN KEY (id_usuario_entrada) REFERENCES usuarios(id),
        FOREIGN KEY (id_usuario_salida) REFERENCES usuarios(id)
    );""",
```

Fuente: Falcón y Mendoza (2018).

A-2

```
"""create table IF NOT EXISTS noregistrado(
    id integer AUTO_INCREMENT PRIMARY KEY,
    id_usuario integer not null,
    id_persona integer not null,
    nombre varchar(50),
    apellido varchar(50),
    cedula varchar(10),
    motivo varchar(100),
    equipo varchar(150),
    fecha_entrada date,
    hora_entrada time,
    fecha_salida date,
    hora_salida time,
    FOREIGN KEY (id_persona) REFERENCES persona(id),
    FOREIGN KEY (id_usuario) REFERENCES usuarios(id)
);"""]

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

fourcc = cv2.VideoWriter_fourcc(*'XVID')
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
database = None

file_name = str(datetime.datetime.now()).replace("-","").replace(" ","").replace(":","").replace(".","") +".avi"

# SSH connection data

host = #direccion ip
username = #usuario de la maquina remota
pw = #clave de la maquina remota
 # Activar camara IP

 '''

 self.video = cv2.VideoCapture('rtsp://IP/1')

 self.video = cv2.VideoCapture('rtsp://username:password@IP/1')

 '''

 #Usar multiples camaras por el mismo BUS

 '''

 cap0 = cv2.VideoCapture(0)
 cap0.set(3,160)
 cap0.set(4,120)
 cap1 = cv2.VideoCapture(1)
 cap1.set(3,160)
 cap1.set(4,120)
 ret0, frame0 = cap0.read()
 assert ret0 #succeds
 ret1, frame1 = cap1.read()
 assert ret1 #succeds

 '''
```

Fuente: Falcón y Mendoza (2018).

A-3

```
def __init__(self,train = True, usuario = "gaceta1", entrada = 1):
    self.usuario = usuario
    self.video = cv2.VideoCapture(0)
    self.out = cv2.VideoWriter(self.file_name,self.fourcc,20.0,(640,480))
    # Configuracion para localhost
    self.database = MySQLdb.connect(host="localhost",port=3306,user="root",passwd="",db="initial")
    # Configuracion para el Raspberry
    #self.database = MySQLdb.connect(host="192.168.1.143",port=3306,user="user",passwd="toor",db="initial")
    self.entrada = entrada
    if train == True:
        self.face_recognizer.read("trainer/trainer.yml")

def __del__(self):
    self.video.release()
    self.out.release()
    self.database.close()
    # Cuando se requiere que el video se almacene en el servidor
    '''
    ssh = SSHConnection(self.host,self.username,self.pw)
    ssh.put("C:\\Users\\victo\\Desktop\\TESIS\\PyCam\\V1.0\\"+self.file_name,"/home/pi/PyCam/videos/"+self.file_name)
    ssh.close()
    '''
def get_frame(self):
    # Se accede al curso para trabajar con la base de datos
    cursor = self.database.cursor()
    # Se lee los frames del video
    success, image = self.video.read()
    # Si se lee un frame satisfactorio
    if success:
        # Se convierte el frame en formato Gray
        gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        # Se detecta rostros en el frame
        faces = self.face_cascade.detectMultiScale(gray,1.2,10)
        # Se analizan todos los rostros encontrados
        for (x,y,w,h) in faces:
            cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),1)
            # Se realiza la prediccion de los rostros utilizando el reconocedor
            ids,conf = self.face_recognizer.predict(gray[y:y+h,x:x+w])
            # Se realiza el query para buscar el nombre y apellido de todas las predicciones
            cursor.execute("SELECT id,nombre,apellido FROM persona WHERE id = '%s';",(ids,))
            # se trabajan con los valores resultantes del query
            result = cursor.fetchall()
            id_persona = result[0][0]
            nombre = result[0][1]
            apellido = result[0][2]
            # Se establece el parametro de confiabilida
            if conf < 55:
                entro, salio = 0,0
                # Se realiza un query para obtener la informacion de si la persona ha entrado o no
                sql_responce = cursor.execute("SELECT id,id_persona,entro FROM registro WHERE id_persona = '%s' ORDER BY id  DESC;", (id_persona, ))
                # Se trabaja con el primer dato resultante del query
                registro = cursor.fetchone()
                # Se genera la fecha y hora actual
                now = datetime.datetime.now()
                # Se separa la fecha y la hora en dos variables distintas
                date = now.strftime("%Y-%m-%d")
                time = now.strftime("%H:%M:%S")
                # Se realiza un query para la extraccion de los datos del usuario
                cursor.execute("SELECT id FROM usuarios WHERE usuario = %s;", (self.usuario,))
                id_usuario = cursor.fetchone()[0]
                # Si la camara actua como entrada
```

Fuente: Falcón y Mendoza (2018).

A-4

```
        if self.entrada == 1:
            # Si el query para obtener la informacion no da resultados se crean los nuevos registros
            if sql_response == 0:
                cursor.execute("""INSERT INTO registroinformacion (id_persona,id_usuario_entrada,id_usuario_salida,fecha_entrada,hora_entrada)
                    ALUES (%s,%s,%s,%s,%s);""",(id_persona,id_usuario,id_usuario,date,time))
                cursor.execute("INSERT INTO registro (id_persona,entro) VALUES (%s, %s);",(id_persona,True))
            # Si el qury para obtener la informacion  da resultados se verifica el ultimo ingreso que tuvo la persona
            elif sql_response == 1:
                id_registro , entro = registro[0], registro[2]
                # Query para obtener la informacion del ultimo registro de la persona
                cursor.execute("SELECT id,fecha_salida,hora_salida FROM registroinformacion WHERE id_persona ORDER BY id DESC")
                registro_informacion = cursor.fetchone()
                id_registroinformacion = registro_informacion[0]
                # Se verifica si la persona no haya entrado anteriormente sin una posterior salida, en caso de no ser asi se crean nuevos registros
                if registro_informacion[1] != None and registro_informacion[2] != None and entro == 0:
                    cursor.execute("""INSERT INTO registroinformacion (id_persona,id_usuario_entrada,id_usuario_salida,fecha_entrada,hora_entrada)
                        VALUES (%s,%s,%s,%s,%s);""",(id_persona,id_usuario,id_usuario,date,time))
                    cursor.execute("UPDATE registro SET entro = %s WHERE id =  %s and id_persona = %s;",(True,id_registro,id_persona))
            self.database.commit()
        #Si la camara actua como salida
        elif self.entrada == 0:
            # Si el qury para obtener la informacion  da resultados se actualiza el ultimo registro
            if sql_response == 1:
                id_registro , entro  = registro[0] , registro [2]
                # Si la persona actualmente ya ingreso al sistema se actualizan los registros con la salida
                if entro == 1:
                    # Se realiza un query para obtener el ultimo registro de la persona
                    cursor.execute("SELECT * FROM registroinformacion WHERE id_persona = %s ORDER BY id DESC",(id_persona,))
                    registro_informacion = cursor.fetchone();
                    id_registroinformacion = registro_informacion[0]
                    # Se realiza un query para actualizar la informacion del registro y del registro de control
                    cursor.execute("UPDATE registroinformacion SET id_usuario_salida = %s, fecha_salida = %s, hora_salida = %s WHERE id = %s;",
                                   (id_usuario,date,time,id_registroinformacion))
                    cursor.execute("UPDATE registro SET entro = %s WHERE id =  %s and id_persona = %s;",(False,id_registro,id_persona))
                    self.database.commit()
        else:
            # En caso de que la camara no sea ni de entrada ni salida no se realiza ninguna accion
            pass
        # Se modifica la imagen colocandole el nombre y apellido de la prediccion a la respectiva persona
        cv2.putText(image,nombre+ " " + apellido,(x+2,y+h-5),cv2.FONT_HERSHEY_SIMPLEX,1,(155,255,0),2)
            else:
                # En caso de no haber una prediccion se indica que la persona es No registrada
                cv2.putText(image,'No Registrado',(x+2,y+h-5),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
          ret,jpeg = cv2.imencode('.jpg',image)
          self.out.write(image)
          return jpeg.tobytes(), image

    def create_database(host = "192.168.1.143",username ="user", password = "toor", dbname = 'initial', sql = _initial_sql):
        db = MySQLdb.connect(host=host,port=3306,user=username,passwd=password)
        cursor = db.cursor()
        cursor.execute("create database IF NOT EXISTS "+ dbname +" CHARACTER SET utf8 COLLATE utf8_general_ci;")
        db.select_db(dbname)
        for query in sql:
            cursor.execute(query)
            db.commit()
        db.close()

    def connect_database(self,host,username,password,dbname):
        self.database = MySQLdb.connect(host,username,password,dbname)

    def close_database(self):
        self.database.close()
```

Fuente: Falcón y Mendoza (2018).

A-6

```
def record_face(self,nombre, apellido, cedula,tipo,carrera="",semestre="",sueldo=0.00,costo_hora=0.00):
    #ssh = SSHConnection(self.host,self.username,self.pw)

    # Cuando se necesita trabajar por ssh

    # ssh.create_folder("/home/pi/PyCam/fotos")

    # Cuando se necesita trabajar de manera local

    # Se verifica si el directoria de los positivos de entrenamiento existe, si no existe se crea

    if not os.path.exists('./fotos'):
        os.makedirs('./fotos')

    cursor = self.database.cursor()

    if tipo==0:
        cursor.execute('INSERT INTO estudiante (carrera,semestre) VALUES (%s,%s);',(carrera,semestre))
    elif tipo==1:
        cursor.execute('INSERT INTO padm (costo_hora) VALUES (%s);',(costo_hora,))
    elif tipo==2:
        cursor.execute('INSERT INTO pobrero (sueldo) VALUES (%s);',(sueldo,))

    uid = cursor.lastrowid

    cursor.execute('INSERT INTO persona (id_tipo,nombre,apellido,cedula,tipo) VALUES (%s,%s,%s,%s,%s);',
                   (uid,nombre,apellido,cedula,self._tipo_persona[tipo]))

    uid = cursor.lastrowid

    sampleNum = 0
    sampleNum = 0
    # Cuando se necesita trabajar por ssh

    # ssh.create_folder("/home/pi/PyCam/fotos/P"+str(uid))

    # Cuando se necesita trabajar de manera local

    if not os.path.exists('.fotros/P'+str(uid)):
        os.makedirs('./fotos/P'+str(uid))

    while sampleNum<200:
        success, image = self.video.read()

        if success:
            gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
            faces = self.face_cascade.detectMultiScale(gray,1.2,10)
            for (x,y,w,h) in faces:
                sampleNum = sampleNum + 1
                # Cuando se trabaja de manera local

                cv2.imwrite("fotos/P"+str(uid)+"/face."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])

                # Cuando se requiere trabajar por ssh
                '''
                cv2.imwrite("face."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
                ssh.put("C:\\Users\\victo\\Desktop\\TESIS\\PyCam\\V1.0\\face."+str(sampleNum)+".jpg","/home/pi/PyCam/fotos/P"+str(uid)+
                        "/face."+str(sampleNum)+".jpg")
                os.remove("face."+str(sampleNum)+".jpg")
                '''
                cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
            cv2.imshow("img",image)
        cv2.waitKey(1)
    self.database.commit()
```

Fuente: Falcón y Mendoza (2018
).

A-6

```
def generate_seed_users(self):
    cursor = self.database.cursor()
    # Se realizan todos los query para la creacion de usuarios perdeterminados y persona ficticia
    cursor.execute('INSERT INTO usuarios (usuario,tipo,clave) VALUES ("administrador","adm",%s);',(generate_password_hash("1234"),))
    cursor.execute('INSERT INTO usuarios (usuario,tipo,clave) VALUES ("vigilancia1","gaceta",%s);',(generate_password_hash("1234"),))
    cursor.execute('INSERT INTO usuarios (usuario,tipo,clave) VALUES ("vigilancia2","gaceta",%s);',(generate_password_hash("1234"),))
    cursor.execute('INSERT INTO usuarios (usuario,tipo,clave) VALUES ("registro","registro",%s);',(generate_password_hash("1234"),))
    cursor.execute('INSERT INTO estudiante (carrera,semestre) VALUES ("","");')
    cursor.execute('INSERT INTO padm (costo_hora) VALUES (0);')
    cursor.execute('INSERT INTO pobrero (sueldo) VALUES (0);')
    cursor.execute('INSERT INTO persona (id_tipo,nombre,apellido,cedula,tipo) VALUES (1,"","","","estudiante");')
    self.database.commit()

def detect_face(img):
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(gray,1.2,10)
    if len(faces)==0:
        return None,None
    (x,y,w,h) = faces[0]
```

```
def train_data():
    faces , labels = [] , []

    # se listan todos los directorios de la ruta de las imagenes

    dirs = os.listdir("./fotos")

    # Se iteran a traves de todas las rutas
    for dir_name in dirs:
        # Si el nombre del directorio no comienza ocn P no se realiza nada
        if not dir_name.startswith("P"):
            continue
        # Se modifica el nombre del directorio para extraer la ID de la persona
        label = int(dir_name.replace("P", ""))
        # Se crea la ruta del directorio de la persona
        subject_dir_path = "./fotos/"+dir_name
        # Se listan todos los directorios de la ruta del directorio de la persona
        subject_images_names = os.listdir(subject_dir_path)
        # Se iteran todas las rutas de las imagenes
        for image_name in subject_images_names:
            # Se hace caso omiso de todos los archivos ocultos dentro del directorio
            if image_name.startswith("."):
                continue
            # Se genera la ruta de la imagen
            image_path = subject_dir_path+"/"+image_name
            # Se lee la imagen
            image = cv2.imread(image_path)
            # Se detectan los rostros de la imagen
            face, rect = PyCam.detect_face(image)
            # En caso de encontrar las agrega a las listas de rostros e ids
            if face is not None:
                faces.append(face)
                labels.append(label)
    # Se crea el reconocedor
    face_recognizer = cv2.face.LBPHFaceRecognizer_create()
    # Se entrena el reconocedor con las listas de rostros e ids
    face_recognizer.train(faces,np.array(labels))
```

Fuente: Falcón y Mendoza (2018).

A-7

```
# Si no existe la carpeta para el archivo de entrenamiento se crea
if not os.path.exists('./trainer'):
    os.makedirs('./trainer')
# Se guarda el archivo de entrenamiento del reconocedor
face_recognizer.save("trainer/trainer.yml")

# Cuando se trabaja por SSH
'''
host = # direccion del host local
username = # usuario de la maquina local
pw = # clave del usuario de la maquina local

ssh = SSHConnection(host,username,pw)
ssh.put("/home/pi/PyCam/trainer/trainer.yml","C:\\Users\\victo\\Desktop\\TESIS\\PyCam\\V1.0\\trainer\\trainer.yml")
print("Reconocedor entrenado....")
'''
```

Fuente: Falcón y Mendoza (2018).

ANEXO D

Codificación de implementación Clase SSHConnection

```python
import paramiko

class SSHConnection(object):

    def __init__(self, host, username, password, port=22):
        # Se incializan las conexiones
        self.sftp = None
        self.sftp_open = False
        # Se abre el transporte utilizando el host y puerto dados
        self.transport = paramiko.Transport((host, port))
        # se realiza la conexion a traves del transporte
        self.transport.connect(username=username, password=password)

    def _openSFTPConnection(self):
        # Se abre la conexion si esta no se encuentra abierta
        if not self.sftp_open:
            self.sftp = paramiko.SFTPClient.from_transport(self.transport)
            self.sftp_open = True

    def get(self, remote_path, local_path=None):
        # Se abre la conexion SFTP para transferencia de archivos
        self._openSFTPConnection()
        # Se transfiere el archivo de la maquina remota a la maquina local
        self.sftp.get(remote_path, local_path)

    def put(self, local_path, remote_path=None):
        # Se abre la conexion SFTP para transferencia de archivos
        self._openSFTPConnection()
        # Se transfiere el archivo de la maquina local a la maquina remota
        self.sftp.put(local_path, remote_path)
    def create_folder(self,remote_path):
        """
        Crea un directoria en la maquina remota
        """
        self._openSFTPConnection()
        try:
            self.sftp.chdir(remote_path)
        except IOError:
            self.sftp.mkdir(remote_path)
            self.sftp.chdir(remote_path)

    def close(self):
        # Se cierra la conexion SFTP y SSH
        if self.sftp_open:
            self.sftp.close()
            self.sftp_open = False
        self.transport.close()
```

Fuente: Falcón y Mendoza (2018).

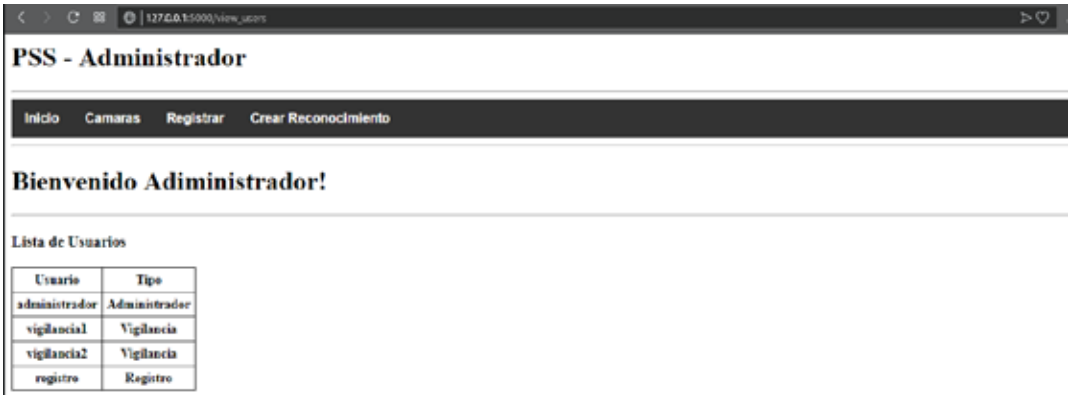
ANEXO E

Registros en la página web

A-1. Login.

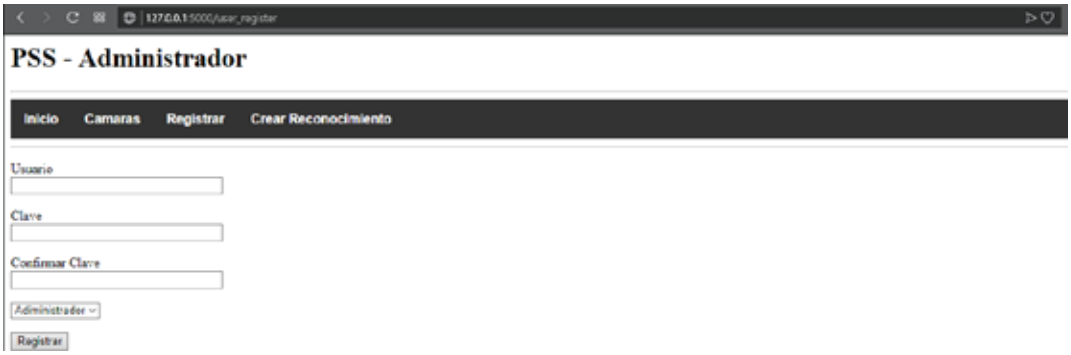
Fuente: Falcón y Mendoza (2018).

A-2. Modulo administrador.



Fuente: Falcón y Mendoza (2018).

A-3. Modulo administrador registro de usuarios.



Fuente: Falcón y Mendoza (2018).

A-4. Modulo vigilancia.

Fuente: Falcón y Mendoza (2018).

A-5. Modulo vigilancia registro de equipos.

Fuente: Falcón y Mendoza (2018).

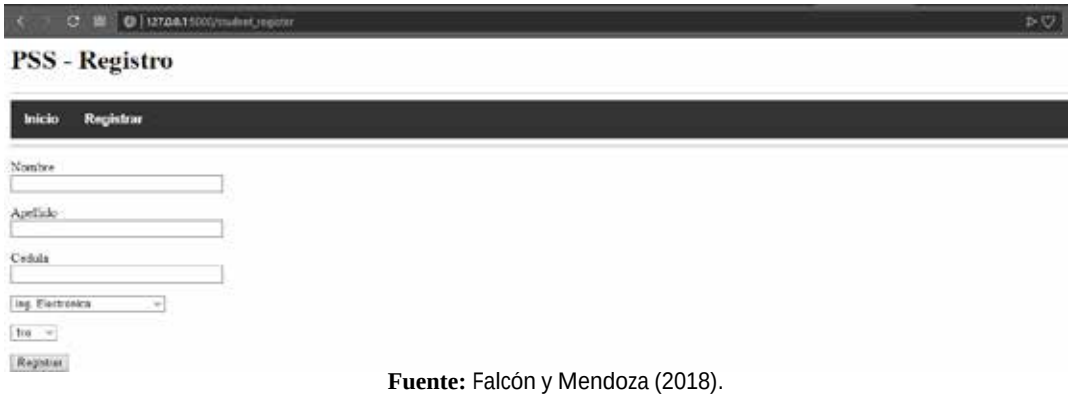
A-6. Modulo vigilancia registro de invitados.

Fuente: Falcón y Mendoza (2018).

A-7. Módulo de registro.

Fuente: Falcón y Mendoza (2018).

A-7. Registro de estudiantes.



Fuente: Falcón y Mendoza (2018).

A-8. Registro de personal administrativo

Fuente: Falcón y Mendoza (2018).

A-9. Registro de personal obrero

Fuente: Falcón y Mendoza (2018).