Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

Sistema de seguridad basado en reconocimiento facial utilizando una Raspberry Pi

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por los Bachilleres
Karina De Sousa y Carlos Mora
para optar al título de
Licenciado en Computación

Tutores:
Prof. Francisco Sans
Prof. Miguel Astor

Caracas, 21 Octubre de 2016

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por los Bachilleres Karina De Sousa C.I.: 23696989 y Carlos Mora C.I.: 23595856, con el título *Sistema de seguridad basado en reconocimiento facial utilizando una Raspberry Pi*, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 21 de Octubre de 2016, a las 8:00 am, para que sus autores lo defendieran en forma pública, en el *Centro de Computación Gráfica*, lo cual se realizó mediante una exposición oral de su contenido, y luego respondieron satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de diecinueve (19) puntos.

En fe de lo cual se levanta la presente acta, en Caracas a los 21 días del mes de Octubre del año dos mil dieciseis, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Francisco Sans.

Prof. Francisco Sans

Prof. Miguel Astor

Profa. Ana Morales

Prof. Rhadamés Carmona

Agradecimientos

Gracias a los profesores Francisco Sans y Miguel Astor por la constante ayuda y guía brindada para realización de esta investigación.

Gracias a Deyban Perez y a Adelis Nieves por su ayuda en la realización de las pruebas de de rendimiento y usabilidad.

Gracias a Joaquin Molina en la ayuda brindada durante el desarrollo de la aplicación.

Gracias a Marisol, Leonel, Jesús y Miguel, nuestros hermanos. Quienes siempre creyeron en nosotros y estuvieron a nuestro lado sin importar la distancia que nos separa.

Gracias a Diego De Sousa, mi sobrino y ahijado, por ser siempre una luz en mi vida y un motivo para seguir adelante.

Gracias a mi tía Coromoto, por estar ahí en todo momento durante este camino, sin condiciones, sin falta.

Gracias a nuestros padres, por ser la columna que nos levantó y nos mantuvo firmes, no solo en el proceso de desarrollo de la investigación, sino en todo el camino que recorrimos desde que comenzamos a formarnos. Parece que fue ayer cuando pisamos por primera vez las

aulas para emprender el camino hacia el éxito académico. No ha sido un camino fácil, pero si muy gratificante.

Gracias por estar ahí en los momentos difíciles y también en los más gratificantes. Gracias Leonor, António, Aura y Chuchú por todo el esfuerzo y confianza puestos en nosotros, sin dudar ni en un solo momento de nuestra capacidad, perseverancia y pasión por lo que hacemos. Ustedes se merecen esto y mucho más, por eso les dedicamos este trabajo, reflejo de ese camino que hemos recorrido.

Los amamos.

Dedicado a mi *Nona* Zoila, esto también es para ti, un logro que merece ser celebrado desde lo más alto del cielo y lo más profundo de nuestros corazones, donde siempre estarás.

Dedicado a mis abuelos António, José y María que siempre han cuidado de mi desde el cielo y a mi abuela María de Jesús por mantenerme en su memoria y seguir a nuestro lado.

Resumen

Título:

Sistema de seguridad basado en reconocimiento facial utilizando una Raspberry Pi

Autores:

Karina De Sousa, Carlos Mora

Tutores:

Prof. Francisco Sans, Prof. Miguel Astor

El procesamiento digital de imágenes cuenta con técnicas utilizadas en campos como la seguridad. Por otro lado, la capacidad de procesamiento de los *Single Board Computers* ha aumentado, además de que sus precios se han reducido considerablemente en lo últimos años. Esto, sumado a la problemática de inseguridad y falta de recursos existente en la Facultad de Ciencias de la UCV, da base a la búsqueda de soluciones de bajo costo a esta situación. Por esto, el trabajo plantea el proceso de diseño y desarrollo de un sistema de seguridad basado en reconocimiento facial, para ser ejecutado sobre una Raspberry Pi, de bajo costo, el cual se compone de un módulo de reconocimiento facial, una interfaz web y una interfaz de programación de aplicaciones (API). Al culminar el desarrollo, el sistema se sometió a diferentes pruebas para determinar su rendimiento, donde fueron medidas las tasas de cuadros por segundo, tamaño de imágenes, posición de la cámara, condiciones de iluminación, aciertos y fallos, utilizando diferentes tipos de cámaras, resultando en un sistema versátil, confiable, rápido y usable, con un porcentaje de aciertos superior al 85 %, distancia máxima entre los rostros y la cámara de 1.5 metros, y una tasa de entre 1.3 y 3.1 FPS. Finalmente, todos los objetivos propuestos en este trabajo fueron alcanzados de manera satisfactoria.

Palabras clave:

Reconocimiento Facial, *Single Board Computers*, Sistema de Seguridad, Raspberry Pi, Bajo costo

Tabla de contenidos

Lista de figuras

Lista de tablas

Listados de código

Capítulo 1

Introducción

El procesamiento digital de imágenes es un campo de investigación abierto, relacionado con la computación gráfica. El constante progreso de la misma ha sido en conjunto con otras áreas con las cuales está relacionada, como las matemáticas, la minería de datos y el conocimiento cada vez mayor de ciertos órganos del cuerpo humano que intervienen en la percepción y la manipulación de las imágenes. A todo esto se suma la inquietud del hombre por imitar y usar ciertas características del ser humano como apoyo en la solución de problemas. El avance del procesamiento digital de imágenes se ve reflejado en la medicina, la astronomía, geología, entre otros. Los sistemas de seguridad basados en reconocimiento facial tienen sustento gracias a estos avances.

Los SBC (*Single Board Computers* - Computadores de Placa Reducida) son computadores funcionales, contenidos en una tarjeta de una única placa. Normalmente son embebidos como controladores e interfaces de otros dispositivos. Para entender el aumento en la popularidad de los mismos, se debe ir a sus inicios hace 10 años en Ivrea, Italia, donde un grupo de diseñadores querían desarrollar un microcontrolador de bajo presupuesto y fácil uso. La rápida aceptación en el mercado de este proyecto marcó el inicio de lo que actualmente se conoce como SBC [1]. Muchos de los SBC existentes hoy en día poseen tanto poder que han comenzado a superar la capacidades de los PCs y tabletas modernas [1].

Un ejemplo de estos SBC es la Raspberry Pi, un computador de bajo costo desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

El avance del procesamiento digital de imágenes y la existencia de SBC como la Raspberry Pi, permiten el desarrollo de proyectos de investigación que los combinen para crear sistemas de seguridad de bajo prosupuesto basados en reconocimiento facial. Es en esta área, donde se ubica el presente trabajo de investigación.

1.1. Justificación y planteamiento del problema

En los últimos años, la situación de inseguridad se ha agravado en la escuela de computación de la facultad de ciencias de la Universidad Central de Venezuela, lo cual ha llevado

a las autoridades a considerar la implementación de nuevas medidas para evitar que el escenario empeore y siga afectando el funcionamiento normal que debería tener la casa de estudios. Gracias a los avances tecnológicos, se han diseñado diferentes alternativas que intentan solventar esta clase de problemas, haciendo uso del reconocimiento facial y otras áreas relacionadas con la biometría. Las mismas suelen tener un alto costo de implementación, instalación y mantenimiento, tanto en *hardware* como en *software*, lo que supone otro problema debido a la situación económica de la universidad.

Una de las ventajas de la escuela de computación es que cuenta con recursos humanos capacitados para desarrollar sistemas similares a los mencionados anteriormente, lo cual implica reducciones en el costo final de los mismos. Sabiendo esto, queda solo considerar los costos de *hardware* para la puesta en marcha de la solución.

La escuela de computación se compone de diversos centros de investigación y laboratorios, que cuentan con recursos de *hardware* destinados al uso docente y estudiantil, tales como computadores personales, servidores, cámaras IP, cámaras web, SBC, etc. Es por esto que existe la posibilidad de considerarlos como parte de una solución para el problema antes descrito. La Raspberry Pi es una buena opción a la hora de implementar sistemas de seguridad de bajo costo.

El laboratorio ICARO, perteneciente al centro de investigación CICORE de la escuela de computación, cuenta con una Raspberry Pi modelo B y una cámara web, que fueron utilizadas para implementar un sistema de seguridad basado en reconocimiento facial que ayudará a mejorar el control de acceso a este sitio y además podrá ser utilizado en otras locaciones de la escuela que cuenten con las condiciones ambientales adecuadas, en una SBC similar a la mencionada.

1.2. Objetivo general

Desarrollar un sistema de seguridad, para ser utilizado en el laboratorio ICARO de la Escuela de Computación de la Universidad Central de Venezuela, basado en reconocimiento facial utilizando una Raspberry Pi y una cámara web.

1.3. Objetivos específicos

- Implementar el módulo encargado del reconocimiento facial, haciendo uso de los algoritmos proporcionados por OpenCV, para ser ejecutado sobre la Raspberry Pi.
- Desarrollar el módulo de entrenamiento y gestión manual de rostros conocidos para el sistema de seguridad.
- Desarrollar el módulo de visualización de visitantes en tiempo real.
- Desarrollar el módulo de alarmas que informe a los administradores del sistema la existencia de algún intruso.

- Realizar el despliegue del sistema de reconocimiento facial en el laboratorio seleccionado, considerando la instalación y el entrenamiento del mismo.
- Realizar las pruebas necesarias para poder evaluar el rendimiento del sistema.
- Documentar el proceso de instalación, configuración y uso del sistema.

1.4. Alcance del Sistema

El sistema será instalado en el laboratorio ICARO de la Escuela de Computación en la Facultad de Ciencias de la Universidad Central de Venezuela. Estará compuesto por una Raspberry Pi a la cual se conecta una cámara web o IP. Un vez instalado el sistema, se podrá acceder a la aplicación a través de un navegador web. El mismo podrá ser instalado en una Raspberry Pi.

El sistema de seguridad posee un módulo de alarmas, el cual evalúa a las personas que son detectadas en la puerta y determina si son una posible amenaza. De ser así, envía un correo electrónico informando a los administradores de la situación. Dicho sistema no se encarga de abrir la puerta del laboratorio, sino de reconocer a las personas que transiten por la puerta del mismo y de informar sobre posibles amenazas. Tampoco utilizará varias cámaras simultáneamente.

1.5. Distribución del documento

El presente documento describe los procesos de investigación y desarrollo utilizados para cumplir con los objetivos antes descritos. El documento está dividido en 5 capítulos: el presente (Capítulo 1), contiene la justificación y planteamiento del problema, junto con los objetivos (general y específicos) de la investigación. El Capítulo 2, detalla los basamentos teóricos necesarios para comprender el resto del documento y un resumen de trabajos existentes, así como la metodología usada durante el desarrollo del sistema. En el Capítulo 3, se presenta de forma detallada el proceso de diseño y desarrollo del sistema. El Capítulo 4, describe los resultados obtenidos en las pruebas realizadas, y las especifíca. Finalmente, en el Capítulo 5, se muestran las conclusiones del trabajo realizado, describiendo los aportes realizados, junto al planteamiento de posibles trabajos futuros.

Capítulo 2

Marco teórico

Este Capítulo describe exhaustivamente las bases teóricas sobre las cuales se fundamenta la investigación, de modo de tener un entendimiento completo de cada una de las técnicas y herramientas a utilizar. Se describirá el proceso de reconocimiento facial desde una visión general hasta un grado alto de detalle, tomando en cuenta cada una de las etapas del mismo y las técnicas y métodos que se utilzan para llear a cabo cada una de ellas. Luego se describiran las características más relevantes de las herramientas de *hardware* y *software* disponibles en el mercado. Por último se hará un recorrido por el estado del arte haciendo referencia a algunas investigaciones hechas en los últimos años relacionadas con el área en cuestión, y se explicará en detalle la metodología usada en el desarrollo de la presente investigación.

2.1. Reconocimiento facial

Los sistemas de reconocimiento facial involucran varios sub-problemas a resolver, como detección, extracción y clasificación, como se muestra en la Figura 2.1. 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.

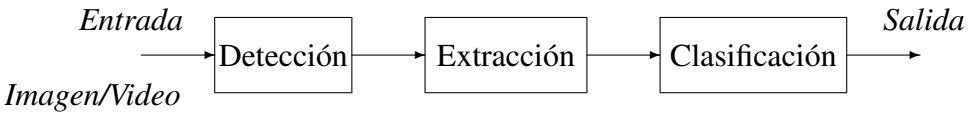

Figura 2.1: Etapas del reconocimiento facial

Todo sistema de reconocimiento facial necesita en principio de un flujo de datos como una imagen o un video para tomarlo como entrada y producir una salida que representa la identificación de una persona o un conjunto de personas que aparecen en dicha entrada.

Entre el proceso de entrada y salida podemos definir etapas que desglosan el reconocimiento facial en si. Para esto definimos tres pasos [2]: detección facial, extracción de características y clasificación.

La detección facial se define como el proceso de extracción de características y procesamiento de las mismas para determinar las posibles regiones donde existe la de alguna cara, todo esto sobre una imagen o secuencia de imágenes. El proceso de extracción de características obtiene detalles faciales importantes de los datos de entrada, como por ejemplo espaciado entre los ojos y arco de las cejas. Finalmente, se lleva a cabo el clasificación en si, que podría ser combinado con el uso de alguna base de datos para identificar a algún individuo. Las fases anteriormente descritas pueden ser mezcladas o manipuladas con la finalidad de mejorar el rendimiento final del sistema.

2.2. Detección facial

La detección comprende el proceso de análisis de características de la imagen con el fin de identificar la ubicación de una cara o un conjunto de caras, lo que puede llegar a ser complejo ya que depende de muchas variables que no son fácilmente controlables, como la posición e intensidad de la luz, posición e inclinación de la cara, expresión facial, oclusión, orientación de la imagen, etc. Por otra parte, el reconocimiento facial es el proceso de análisis de características específicas de un rostro para determinar a quién corresponde.

Se podría decir que la detección facial es un proceso que responde a la pregunta ¿Dónde está la persona?, mientras que el reconocimiento facial puede responder a la pregunta de ¿Quién es esa persona?

El proceso de detección de rostros a través del uso de un computador puede ser llevado a cabo de diferentes maneras, y presenta grandes dificultades [2]. Algunas aplicaciones de reconocimiento facial no requieren realizar el proceso de detección. En algunos casos, las imágenes almacenadas en alguna base de datos pueden estar normalizadas, es decir, tener un formato de imagen de entrada estándar por lo que no hay necesidad de realizar el proceso antes nombrado. 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 de detección varían de acuerdo a su tipo. Pueden involucrar el uso del análisis de las componentes principales [3], autovalores y autovectores [4], máquinas de soporte vectorial [5] o redes neuronales [5]. Por otra parte, las imágenes pueden ser capturadas en un entorno no controlado en relación a condiciones de iluminación y posición de los sujetos, lo que se traduce en grandes retos para el desarrollo de los sistemas de reconocimiento facial [5] [6]. Estos retos se pueden atribuir a varios factores [7]:

- **Variación de la pose**
  El escenario ideal para la detección facial es en el que sólo estuviesen involucradas imágenes de frente al sujeto. Pero esto es poco probable cuando se trabaja bajo un

ambiente no controlado. Por otra parte, el rendimiento de los algoritmos de detección facial decae severamente cuando hay grandes variaciones en la pose del sujeto. Dichas variaciones pueden ocurrir durante un movimiento del sujeto o del ángulo de la cámara.

- **Oclusión facial**
  Causada por la presencia de elementos como barba, lentes o sombreros, es decir que las caras pueden estar parcialmente cubiertas por objetos u otras caras, lo que se traduce en pérdida de características.

- **Expresión facial**
  Las características faciales pueden variar drásticamente debido a diferentes gestos.

- **Condiciones de la imagen**
  Diferentes tipos de cámaras y variaciones en las condiciones ambientales, como tipo de iluminación, posición de la luz o su intensidad pueden afectar la calidad de la imagen de entrada, perjudicando la apariencia del rostro.

2.2.1. Estructura del Problema de Detección Facial

La detección es el problema de forma general, que puede ser subdivido para estudiarlo más a fondo, ya que es un concepto que involucra otros sub-problemas. Se encuentran sistemas que detectan y localizan rostros al mismo tiempo para luego mantener un rastreo del mismo. Dicho proceso se muestra en la Figura 2.2. En este trabajo se utilizó este enfoque para llevar a cabo la detección facial.

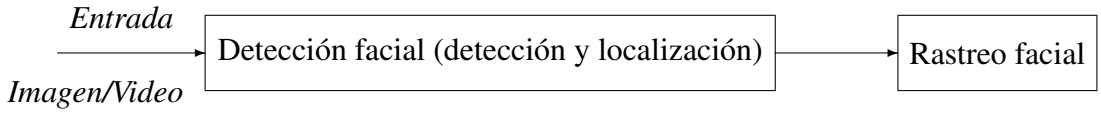

Figura 2.2: Procesos de detección facial

Generalizando el proceso de detección facial, se realizan pasos comunes en muchos sistemas de este tipo. Primero se realiza algún tipo de reducción de dimensionalidad, para obtener un tiempo de respuesta aceptable [7], luego se intenta extraer ciertas regiones faciales relevantes; posteriormente se hacen mediciones sobre los rostros y se extraen características importantes. Por último se analiza toda la información recolectada para decidir si la región evaluada corresponde o no a un rostro.

2.2.2. Enfoques para la detección facial

Existen diversos enfoques en los que se puede agrupar la detección facial. Debido a que no hay una clasificación unificada que sea aceptada mundialmente, diversos autores crean la

suya. Tal es el caso de Marques [7], cuya clasificación ha sido tomada como referencia para el desarrollo de este documento. El mismo presenta dos criterios de clasificación, detección facial dependiendo del escenario y métodos de detección divididos en categorías.

La detección facial dependiendo del escenario involucra entornos controlados, imágenes a color e imágenes en movimiento. Los métodos de detección divididos en categorías incluyen métodos basados en conocimiento, métodos basados en plantillas, métodos basados en apariencia, métodos basados en distribución, redes neuronales, máquinas de soporte vectorial, redes dispersas de Winnow, clasificadores bayesianos ingénuos y modelos ocultos de Markov [7]. Para el desarrollo de esta investigación se utilizaron métodos basados en apariencia.

Los métodos basados en apariencia generan una plantilla a partir de imágenes de entrenamiento. Se realiza análisis estadístico y técnicas de aprendizaje automático para encontrar las características relevantes de los rostros.

El análisis suele ser costoso, por lo que es necesario llevar a cabo algún tipo de reducción de dimensionalidad. Un vector de características derivadas de la imagen es visto como una variable aleatoria X. Un método común es utilizar dicha variable para clasificar a un candidato como un rostro a través de un clasificador Bayesiano. La desventaja es que la dimensionalidad de X puede ser tan grande que la implementación de dicho clasificador se puede volver muy compleja.

2.3. Extracción de características

Los humanos pueden identificar rostros desde muy temprana edad. Parece ser un proceso automatizado y dedicado a esta tarea en nuestros cerebros, aunque es un tema muy debatido. Lo que queda claro es que podemos reconocer a personas que conocemos, incluso si estas llevan lentes o sombreros [7]. No es díficil para nosotros ver fotos antiguas de nuestros familiares y reconocerlos, a pesar de que se vean mucho más jóvenes en las imágenes. Todo este proceso puede parecer trivial para nosotros, pero representan todo un reto para los computadores.

El problema principal al momento de extraer rostros de una o más fotografías es obtener información relevante de éstas, a este proceso se le conoce como extracción de características.

Los pasos para la extracción de las características son, reducción de la dimensionalidad, extracción de características y seleccción de características. Estos pasos podrían solaparse y la reducción de dimensionalidad podría verse como una consecuencia de los algoritmos de extracción y selección de características [7].

Es importante hacer una distinción entre la extracción de características y la selección de características, ya que ambos términos suelen ser usados de manera indiferente. Un algoritmo de extracción obtiene características de los datos, creando así, nuevas características basadas en transformaciones o combinaciones de los datos originales, para seleccionar un subespacio apropiado. Por otro lado, un algoritmo de selección de características elige el mejor subconjunto del conjunto de características de entrada, descartando las no-relevantes. La selección de características se realiza despúes de la extracción [7], tal cual se muestra en la

Figura 2.3. De esta forma se extraen características de las imágenes de rostros, y luego solo un subconjunto optimizado de estas es seleccionado.

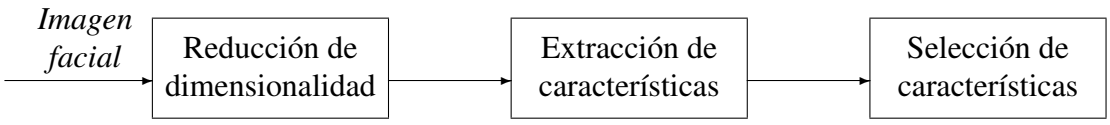

Figura 2.3: Proceso de extracción de características

2.3.1. Métodos para la extracción de rostros

La extracción de características es un proceso clave para el reconomiento de rostros, así como para cualquier tarea de clasificación [2]. Existen muchos algoritmos de extracción de características, la mayoría de ellos son usados en otras áreas distintas al reconocimiento de rostros [7]. Para lograr sus propósitos, los investigadores especializados en computación visual han usado, modificado y adaptado muchos algoritmos y métodos ya existentes, como PCA (*Principal Components Analysis* - Análisis de Componentes Principales). Este método 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, etc [8].

Los métodos para realizar la extracción de características de un rostro, se pueden clasificar en técnicas basadas en geometría, técnicas basadas en plantillas, aproximación basada en apariencia (utilizada para el desarrollo de esta investigación) y métodos basados en el color [2].

Aproximación basada en apariencia

Estas técnicas procesan la imagen como patrones en dos dimensiones, modificando el concepto de característica, ya que aquí no se refieren solo a caracerísticas faciales, sino de cualquier característica extraída de una imagen. Este método ha tenido unos de los mejores resultados en el área de extracción de características faciales, ya que se mantiene con tan solo la información importante de una imagen y deshecha el resto.

Métodos como PCA e ICA (*Independent Component Analysis* - Análisis de Componentes Independientes) son usados para extraer el vector de características [9]. El propósito principal de PCA en estas técnicas es reducir la gran dimensionalidad de las variables observadas a una menor, que contenga variables independientes, sin perder mucha información [9]. Sin embargo, existen casos en los que ICA funciona mejor que PCA, por ejemplo el habla o las imágenes naturales, ya que la mejor forma de describirlas es como una combinación lineal de fuentes con una distribución Gaussiana.

2.3.2. Métodos para la selección de características faciales

Como ya se dijo anteriormente, la selección de características consiste en elegir un subconjunto de características, de las obtenidas en el proceso de extracción. Los algoritmos de selección de características faciales tratan de realizar este proceso, para así obtener el error de clasificación más pequeño posible. La importancia de este error es lo que hace a la selección de características dependiente del método de clasificación [7]. La forma más directa de atacar este problema es examinando cada uno de los posibles subconjuntos y escoger el que se ajuste a un criterio definido previamente. Sin embargo, esto puede ser una tarea inasequible en términos de tiempo computacional. En la Tabla 2.1se muestran los algoritmos de selección propuestos por Jain et al. [10]

Tabla 2.1: Métodos de selección de características [10].

Método	**Definición**	**Comentarios**
Búsqueda exhaustiva	Evalua todos los subconjuntos de características posibles	Método óptimo, pero muy complejo
Ramificación y poda (*branch and bound*)	Usa el algoritmo de *branch and bound*	Puede ser óptimo y posee una complejidad máxima de $O(2^n)$
Mejor característica individual	Evalúa y selecciona las características individualmente	Computacionalmente simple, pero no es probable que consiga el mejor subconjunto de características

La selección de características es un problema NP-complejo, es por esto que los investigadores se esfuerzan en encontrar algoritmos satisfactorios, en vez de óptimos [7]. La idea es crear un algoritmo que seleccione el subconjunto de características más satisfactorio, minimizando la dimensionalidad y complejidad de los datos.

El método basado en apariencia seleccionado para esta investigación es el *Clasificador de Haar*, que involucra tanto el proceso de detección facial como el de selección de características, utilizando un enfoque de ramificación y poda.

Clasificador de Haar

Es un método de deteccion de objetos propuesto por Paul Viola y Michael Jones [11], basado en aprendizaje automático donde una función en cascada es entrenada a partir de un conjunto de imágenes positivas (imágenes de rostros) y negativas (imágenes sin rostros).

El algoritmo necesita extraer características, denominadas *Haar Features*, 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. La Figura 2.4 muestra una característica de Haar.

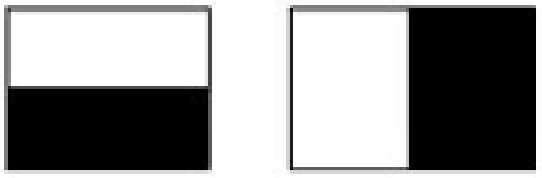

Figura 2.4: Característica de Haar

La extracción de las características utilizando la imagen integral se lleva a cabo de una manera muy rápida, pero genera demasiadas características que no aportan ningún valor a la detección final, por lo que deben ser descartadas de alguna manera.

La idea del método se basa en utilizar una ventana deslizante de $n \times m$ píxeles sobre la imagen original, lo que implica llevar a cabo el proceso repetidas veces. Entonces calcular 180000 características contenidas en una ventana de 24x24 píxeles se vuelve poco práctico [11].

Para lograr esto se utiliza el concepto de clasificadores en cascada, que agrupa las características en diferentes etapas de clasificación y las aplica una por una. Si una ventana falla en la primera etapa, se descarta, sino se pasa a la segunda etapa y así sucesivamente.

Este método de detección facial es uno de los más eficientes gracias a la aplicación de la imagen integral y la clasficiación en cascada [12] y puede detectar rostros humanos a una tasa de por lo menos 5 FPS en un procesador de 2 GHz [11].

2.4. Minería de datos y clasificación de rostros

Para realizar el reconocimiento facial es necesario pasar por varias etapas, como lo muestra la Figura 2.1. En las Secciones 2.2 y 2.3 se habló sobre dos de ellas (detección facial y extracción de rostros) y en la presente Sección se explicará la última fase, conocida como clasificación de rostros. Terminando así, el proceso de reconocimiento, donde el resultado final es la posible identidad del rostro.

Antes de hablar de la clasificación de rostros, es importante tener en cuenta que es, como su nombre lo indica, un problema de clasificación, los cuales pertenecen al área de la minería de datos. Por lo que a continuación se da una breve introducción a la misma y se explican algunos algoritmos que se encargan de solucionar este tipo de problemas.

2.4.1. Conceptos Básicos de minería de datos

La minería de datos estudia la recolección, limpieza, procesamiento, análisis y obtención de información relevante sobre un conjunto de datos. Existe una amplia variación en términos de dominio de un problema, usos, formulación y respresentación de los datos que pueden ser

encontrados en aplicaciones reales. Por lo que, "minería de datos" se usa para abarcar todos estos aspectos del procesamiento de datos [13].

Desde un punto de vista análitico, la minería de datos es compleja y representa un gran reto, debido a la diferencia tan marcada que hay entre problemas y tipo de datos que pueden ser encontrados. Sin embargo, las aplicaciones de minería suelen estar altamente relacionadas con uno de estos cuatro problemas: agrupación, clasificación, asociación o reconocimiento de patrones y detección de *outliers* [13] [1]. Estos problemas son de gran importancia, ya que sirven como una asbtracción y ayudan a conceptualizar y extructurar el área de la minería de datos de forma más eficiente.

2.4.2. Proceso de minería de datos

El proceso de minería de datos es un *pipeline* que contiene varias fases, representadas en la Figura 2.5, las cuales serán explicadas a continuación. Hay que notar que el proceso analítico muestra múltiples bloques que representan la creación de una solución para una aplicación en particular. Esta parte del diseño algoritmico es dependiente de las habilidades del analista y usualmente usa uno o más de los cuatro problemas principales como un bloque de construcción.

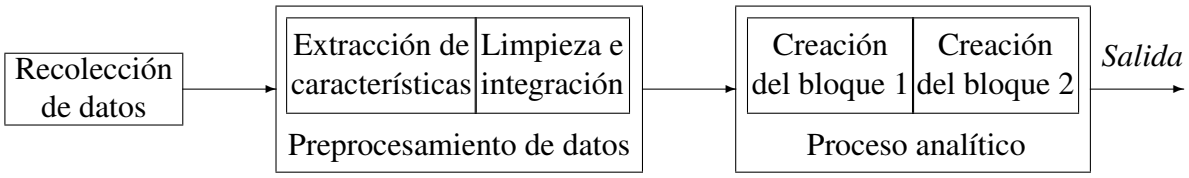

Figura 2.5: Proceso de minería de datos

- **Recolección de datos:** Mientras este escenario es altamente específico a la aplicación en la que se este usando, y normalmente afuera del campo de los analistas de minería de datos, es sumamente importante porque buenas decisiones en esta etapa pueden impactar de manera significante al proceso de minería de datos.

- **Extracción de características y limpieza de los datos:** cuando los datos son recolectados, usualmente no vienen en un formato que sea apropiado para el procesamiento. Es crucial extraer las características relevantes para el proceso de minado.

  El objetivo de la reducción de datos es representar los mismos de forma más compacta. Existen distintos tipos de formas para reducir la dimensionalidad de los datos [13]:

  1. Muestreo de datos.
  2. Reducción de datos con rotación de ejes.

[1] En estadística, un valor atípico o *outlier* es una observación que es numéricamente distante del resto de los datos.

3. Reducción de datos con transformación de tipos.

   PCA es una de las técnicas que han tenido más éxito en el reconocimiento y comprensión de imagénes. 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 reducción de los datos como mencionamos anteriormente y además es usado para predicción, eliminación de redundancia, extracción de características, comprensión de los datos, ect [8].

   El objetivo de PCA es rotar los datos en un sistema de ejes, donde la mayor cantidad de varianza es capturada en una dimensión de menor tamaño. Es importante mencionar que dicho sistema de ejes es afectado por la correlación que existe entre cada uno de los atributos, y que la varianza del set de datos junto con una dirección en particular pueden ser expresadas directamente en términos de su matriz de covarianza.

- **Procesamiento analítico y algoritmos:** la parte final del proceso de minado es diseñar de forma efectiva métodos analíticos de la data procesada.

En general, estos pasos son similares a los descritos en la Figura 2.1, ya que el proceso de reconocimiento facial se inspira en el *pipeline* de minería de datos.

2.4.3. Clasificación en minería de datos

El problema de clasificación puede ser definido de la siguiente forma:

Dada una matriz de entrenamiento D de $n \times d$ (base de datos D) y un valor de etiqueta de clase entre $\{1...k\}$ asociado a cada una de las n filas de D (observaciones en D), determinar la etiqueta de clase de una instancia que no posea clase en la data de prueba. Existen múltiples variaciones de este problema [14].

Los algoritmos de clasificación típicamente contienen dos fases:

- **Etapa de entrenamiento:** en esta fase, se construye un modelo con las instancias de los datos de entrenamiento.

- **Etapa de prueba:** en esta fase, el modelo es usado para asignar una etiqueta a una instancia de los datos de prueba que no posea etiqueta.

La salida de un algoritmo de clasificación puede presentarse ante una instancia de prueba en alguna de estas dos formas:

- **Etiqueta discreta:** en este caso, una etiqueta es retornada para la instancia de prueba.

- **Puntaje númerico:** se asigna un puntaje númerico a cada combinación de etiqueta de clase e instancia de prueba.

Métodos de clasificación

Entre los métodos normalmente utilizados para la clasificación se encuentran:

- **Métodos de selección de características**: La primera etapa de prácticamente todos los algoritmos de clasificación es la selección de características. En la mayoría de los escenarios de minería de datos, una gran variedad de información es recolectada por personas que no manejan el área. Claramente, Las características irrelevantes podrían resultar en que se creen modelos pobres, ya que las mismas no están relacionadas a las etiquetas de clases.

- **Métodos probabilísticos**: Los métodos probabilísticos son los más fundamentales en la clasificación de datos [15]. Estos algoritmos usan inferencia estadística para conseguir la mejor clase para una entrada específica. En adición a esto, este tipo de algoritmos dan como salida la probabilidad posterior correspondiente a la instancia de prueba. Este valor es definido como la probabilidad obtenida despúes de observar la característica específica. Por otro lado, la probabilidad previa es simplemente la fracción de observaciones de entrenamiento que pertenecen a cada clase en particular, sin conocimiento de la instancia de prueba. Despúes de obtener la probabilidad posterior, se usa teoría de toma de decisiones para determinar la membresía de clase para cada nueva instancia. Para calcular este valor se puede usar un clasificador bayesiano ingenuo o un modelo discriminante.

- **Árboles de decisiones**: Los árboles de decisiones crean una partición jerarquica de los datos, que relaciona las hojas con las diferentes clases. Esta división en cada nivel es creada mediante el uso de un criterio. El mismo podría usar una condición (o predicado) sobre un atributo (división univariada) o sobre múltiples de ellos (división multivariada). En general, este enfoque consiste en dividir la data de entrenamiento recursivamente de forma que se maximize la discriminación entre las distintas clases sobre diferentes nodos. Esto ocurre cuando el nivel de distorsión entre las clases de un nodo es maximizado. Se usa una medida como el índice de Gini o entropía para medir dicha distorsión.

- **Métodos basados en reglas**: Los métodos basados en reglas están relacionados con los árboles de decisión, con la excepción de que ellos no crean una jerarquía de partición estricta de los datos de entrenamiento. Por el contrario, la sobreposición es válida para crear una mayor robustez para el modelo. Cualquier camino en un árbol de decisión puede ser interpretado como una regla, que asigna una instancia de prueba a una etiqueta en particular.

  Los clasificadores basados en reglas pueden ser vistos como un modelo más general que los árboles de decisión, ya que estos producen un conjunto de reglas que no se sobreponen, mientras que este no es el caso para los basados en reglas.

- **Métodos basados en instancias**: en aprendizaje basado en instancias, la primera fase al construir el modelo de entrenamiento usualmente es dejado de lado. La instancia de

prueba se relaciona directamente a la de entrenamiento para crear un modelo de clasificación. Esto es conocido como aprendizaje ingenuo, porque estos métodos esperan por el conocimiento de la instancia de prueba antes de crear un modelo local optimizado, el cual es específico para la obervación. La ventaja de esto es que pueden ser personalizados para la instancia de prueba particular, y pueden evitar pérdida de información asociada a la falta de datos en el modelo de entrenamiento.

- **Métodos SVM**: los métodos SVM[2] usan condiciones lineales para separar las clases entre ellas tan bien como se pueda. Los clasificadores SVM pueden considerarse un árbol de decisiones de un solo nivel, con una sola condición de separación multivariante. Como la efectividad del enfoque depende únicamente en esta condición, es crítico definirla cuidadosamente. Las máquinas de soporte vectorial son generalmente definidas para problemas de clasificación binaria [15].

- **Redes neuronales**: las redes neuronales intentan simular sistemas biológicos, que corresponden con el cerebro humano. En éste, las neuronas están conectadas entre ellas a través de puntos, conocidos como sinapsis. En estos sistemas, el aprendizaje es realizado mediante cambios en la fuerza de las conexiones sinápticos en respuesta a impulsos.

  La analogía biológica se mantiene en las redes neuronales artificiales. La unidad básica de computación en ellas es una neurona. Las mismas pueden ser ordenadas en distintos tipos de arquitecturas a través de conexiones entre ellas. La más básica es la llamada perceptron, que contiene un conjunto de nodos de entrada y un único nodo de salida.

2.4.4. Clasificación de rostros

El proceso que sigue a la detección de rostros y selección de características, de acuerdo a la Figura 2.1, es la clasificación, otra de las fases necesarias para llevar a cabo el reconocimiento facial. La idea es agrupar de alguna manera un conjunto de rostros detectados anteriormente, para clasificarlos como un individuo en particular.

Los algoritmos de clasificación hacen uso de métodos de aprendizaje, bien sean supervisados, no supervisados o semisupervisados. El uso de métodos no supervisados es el más complicado, ya que no hay ejemplos definidos que puedan servir como base para el entrenamiento. También es posible encontrar escenarios donde existan ejemplos identificados como individuos, que son utilizados por los métodos supervisados. Ahora bien, puede darse el caso en que el conjunto de datos identificados sea pequeño, y además conseguir nuevas muestras con identificación puede ser muy difícil. En estos casos se utilizan los algoritmos de aprendizaje semisupervisado.

[2]Las SVM (*Support Vector Machines* - Máquinas de Soporte Vectorial) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik [16].

2.4.5. Clasificadores

Jain, Duin y Mao [17] exponen tres enfoques para la construcción de un clasificador, los cuales se presentan a continuación:

- **Similaridad**
  Es el clasificador más sencillo e intuitivo. El objetivo principal es agrupar a los patrones en conjuntos, según el grado de similitud que tengan. La idea es establecer una métrica para definir el grado de similitud de las muestras para así agruparlas en clases comunes. Por ejemplo, si utilizamos a la distancia euclideana como métrica y representamos a una clase como el vector promedio de todos los patrones que pertenecen a dicha clase, se puede utilizar la clasificación de los vecinos más cercanos con estos parámetros. El rendimiento de este tipo de clasficación es generalmente bueno, aunque requiere de altos recursos computacionales cuando la cantidad de muestras es grande. Vecinos más cercanos no es la única técnica. Por ejemplo se pueden utilizar técnicas basadas en plantillas para comparar los rostros no identificados con los identificados y agrupar.

- **Probabilidad**
  Se utilizan modelos probabilisticos para clasificar rostros. Por ejemplo, las reglas de decisión Bayesiana, que pueden ser usadas para crear clasificadores óptimos que utilicen el error Bayesiano para evaluar características. Un enfoque es definir la regla de decisión del máximo a posteriori [18].

- **Árboles de decisión**
  La idea principal detrás de los árboles de decisión es minimizar el error entre el patrón de candidato y los patrones de prueba. Un método muy utilizado es el del discriminante lineal de Fisher [4], que está relacionado con el análisis de las componentes principales. Este método modela la diferencia entre las clases de datos y es utilizado para minimizar el error cuadrático medio o el error absoluto.

  Los árboles de decisión son un tipo específico de clasificador, estos se entrenan a través de una selección iterativa de características individuales que se representan en cada nodo del árbol. Durante la clasificación se evalúan sólo las características necesarias.

Uno de los métodos basados en similaridad utiliza el análisis discriminante de la imagen para determinar los grupos que contienen rostros y los que no. Los patrones son proyectados sobre un espacio de menor dimensión al original y se lleva a cabo dicho análisis. El método basado en similaridad seleccionado para esta investigación es el *Fisherfaces*, debido a que entrega la tasa de errores más baja entre los métodos más comúnes [4].

Fisherfaces

El algoritmo utiliza el discriminante lineal de Fisher [19] [4], que lleva a cabo una reducción de dimensionalidad específica por clase. Para encontrar la combinación de características que separa las clases de mejor manera, el discriminante de Fisher maximiza la relación

o clases externas e internas, en vez de maximizar la dispersión general. La idea es sencilla: las clases similares deberían agruparse de manera firme, mientras que las clases diferentes estarán tan lejos como se pueda en la representación de menor dimensión.

Fisherfaces agrupa a los rostros de entrenamiento en clases etiquetadas con un identificador para cada una, y para detectar la identidad de un rostro, se calcula la distancia entre la imagen y cada una de las clases entrenadas. Dicha distancia es utilizada como una medida para determinar el grado de similitud con alguna de las clases.

2.5. Herramientas de *Software* y *Hardware*

Un SBC es un computador completo construído sobre una única tabla de circuitos, con microprocesadores, memoria, entrada/salida y otros componentes, que sigue la arquitectura de Von Neumann, y se puede adquirir a un precio bastante bajo. Por su naturaleza pueden ejecutar cualquier tipo de *software* cuyos requerimientos de *hardware* sean satisfechos por las prestaciones de la placa, como diferentes bibliotecas de visión por computador que ejecutan algoritmos de reconocimiento de imágenes y minería de datos. En esta Sección se hará un recorrido por diferentes herramientas de *hardware* y *software* que pueden ser utilizadas para desarrollar sistemas de seguridad.

2.5.1. Herramientas de *hardware*

Generalmente las prestaciones de un SBC son menores que las otorgadas por un computador personal del momento u otros computadores de mayores prestaciones.

Por ejemplo, las capacidades de procesamiento, memoria y entrada/salida suelen ser mucho más reducidas que las de un PC. Sin embargo el precio final de estas placas, el consumo de energía y su tamaño son factores que influyen en la elección de este tipo de sistemas para ejecutar tareas con un solo objetivo, como por ejemplo, un sistema de reconocimiento facial que se ejecute todo el tiempo.

Los sistemas de reconocimiento facial necesitan detectar la presencia de al menos un individuo en la escena, debido a que no es necesario utilizar recursos de computo mientras no haya que ejecutar algoritmos de detección, en caso de que no haya nadie presente. Sin embargo, deben permanecer activos en todo momento para así poder detectar cuando alguien se acerque a la cámara, por lo que deben estar ejecutando tareas de detección de objetos sencillas, para así llevar a cabo la detección y reconocimiento facial en el momento en que se necesite. Entonces ¿Por qué no utilizar simplemente un computador personal o de otro tipo para que sólo se ejecute el *software*? La respuesta se traduce en ahorro.

Para esta investigación se seleccionaron cuatro SBC presentes en el mercado actual, que se pueden encontrar en cualquier inventario de tienda electrónica de gran popularidad. Una vez analizadas las cuatro SBC, se pudo establecer una comparación y finalmente decidir cual de ellas es la más adecuada para destinar a la ejecución de un *software* de reconocimiento facial en esta investigación. La Tabla 2.2 generada a partir de la investigación, compara

exhaustivamente las especificaciones técnicas, precio y características importantes como documentación, para mostrar de manera más objetiva las diferencias entre todas ellas.

La CPU de cada una de las placas tiene características similares, pero la de la Raspberry Pi es la que mejor relación entre velocidad de reloj y cantidad de núcleos brinda. El conjunto de instrucciones es de 64 bits tanto en la Raspberry Pi como en la Minnowboard, mientras que en las demás placas es de 32 bits. La memoria RAM, otros puertos, entrada de energía y sistemas operativos son factores medianamente estándar que no representan mayor peso al momento de realizar una elección. No es el caso de los puertos USB, WiFi y documentación, donde la Raspberry Pi lleva la delantera sobre cada una de las demás opciones.

Tabla 2.2: Comparación de placas SBC

	Raspberry Pi 3 B	**Banana Pi 2**	**BeagleBone Black**	**MinnowBoard MAX v2**
CPU	ARM Cortex-A53 quad core 1.2 GHz	ARM Cortex-A7 dual core 1 GHz	ARM Cortex-A8 quad core 1 GHz	Intel Atom E3825 dual core 1.33 GHz
Conjunto de instrucciones	64 bits	32 bits	32 bits	64 bits
Memoria RAM	1GB SDRAM	1 GB DDR3	512 MB DDR3	1 GB DDR3
Almacenamiento externo	Micro SD	SD	N/A	Micro SD
Puertos USB	4	2	1	2
Otros puertos	Gigabit Ethernet, GPIO, HDMI, Audio 3.5 mm	HDMI, Audio 3.5 mm, Gigabit Ethernet, GPIO, IR	Gigabit Ethernet, HDMI, GPIO	HDMI, SATA2, Gigabit Ethernet
Entrada de energía	5 V			
Precio	US$ 35	US$ 35	US$ 55	US$ 99
Wifi	Incluído	A través de USB, dispositivo externo		
GPU	VideoCorde IV 256 MB	ARM Mali400MP2 128 MB	PowerVR SGX530	Intel HD Graphics
Sistemas operativos	Raspbian, Debian, Fedora, Linux ARM, FreeBSD, otros	Raspbian (Banana), Android, Debian, Ubuntu, OpenSUSE, otros.	Angstrom, Debian, Ubuntu, Android	Debian, Windows 8.1, Yocto, Android 4.4
Documentación y comunidad	Extensa	Reducida	Reducida	Moderada

Todas las placas escogidas gozan de características y prestaciones comunes, y pueden ser utilizadas para diferentes fines, uno de ellos la ejecución de algún *software* de reconocimiento facial. Debido a que todas ellas pueden ejecutar sistemas operativos basados en Linux, es posible desarrollar *software* de un tipo en específico y recompilarlo en cualquiera de ellas. La decisión depende de las necesidades específicas del proyecto a desarrollar.

En el desarrollo de ésta investigación se ha escogido la Raspberry Pi como *hardware* para soportar el *software* de reconocimiento facial. Esta elección se ha hecho en base a diversos factores: si bien no es la placa con mayor capacidad de procesamiento entre las comparadas, sus 1.2 GHz pueden ser capaces de entregar buenos tiempos de respuesta en la ejecución de los algoritmos; su memoria RAM es un factor estándar en todas las placas escogidas además su precio es el más competitivo entre todas las placas, poniendose al mismo nivel que la Banana Pi. Uno de los más influyentes factores en la decisión es la cantidad de documentación y la gran comunidad que soporta a la placa, ya que al disponer de más información el proceso de desarrollo se agiliza. Otro factor importante es el soporte de bibliotecas de visión por computador, descritas posteriormente, que pueden ser utilizadas en cualquiera de las placas mientras que estén soportadas por el sistema operativo instalado.

2.5.2. Raspberry Pi

La Raspberry Pi es un computador de una sola placa y del tamaño de una tajeta de crédito, desarrollada en el Reino Unido por la fundación Raspberry Pi [20] con la intención de estimular la enseñanza de las ciencias de la computación en las escuelas.

La placa se conecta a un monitor HDMI o un televisor RCA y utiliza un teclado y ratón estándar. Es un dispositivo que le permite a personas de todas las edades explorar la computación y aprender lenguajes de programación como Scratch y Python. Es capaz de realizar casi cualquier tarea realizada por un computador común, desde reproducir video de alta definición hasta ejecutar *software* de ofimática y videojuegos.

Uno de los detalles más interesantes de la Raspberry Pi es la capacidad de interactuar con el mundo exterior a través de un conjunto de sensores y actuadores, conectados por medio de sus pines de entrada/salida de propósito general (GPIO) programables. Ha sido utilizada repetidas veces en el mundo de la creación de proyectos digitales, robótica, Internet-of-Things, controladores embebidos, sistemas de seguridad, etc.

Hardware

La versión más reciente del dispositivo con precio superior a los US$30 es la Raspberry Pi 3 Modelo B, que cuenta con:

- Microchip Broadcom BCM2837.
- CPU ARM Cortex A53 de 1.2GHz.
- GPU Broadcom VideoCare IV @ 250 MHz, con soporte para OpenGL ES 2.0.
- Memoria RAM de 1GB (Compartida con la GPU).

Utiliza memorias SD para encendido y almacenamiento a largo plazo. El dispositivo es alimentado a través de un conector USB de 5V. La cantidad de miliampérios que requiere para funcionar depende de los dispoitivos conectados a él. Generalmente el modelo B+ utiliza al menos 700 mA, dependiendo de los dispositivos conectados, mientras que el modelo A utiliza 500 mA sin periféricos conectados.

La Raspberry Pi está equipada con cuatro puertos USB 2.0. Estos están conectados a un chip IC3 LAN que es otro dispositivo USB conectado directamente a la placa. Estos puertos permiten la conexión con teclados, ratones, cámaras web y otros dispositivos. En general, cualquier dispositivo soportado por linux con *drivers* para procesadores ARM puede ser utilizado en una Raspberry Pi.

La especificación de USB define tres velocidades para dispositivos: lentos, medianos y rápidos. La mayoría de teclados y ratones son lentos, dispositivos de audio vía USB de velocidad normal y los de video son de velocidad rápida. Se definen los tres niveles debido a que la conexión de dispotivos a diferentes grupos puede reducir el rendimiento en la transmisión de datos. Los puertos USB tienen un diseño para cargar 100mA cada uno, suficiente intensidad para alimentar a un dispositivo lento, como teclados y ratones. Otros, como adaptadores de WiFi, discos duros USB, memorias flash y cualquier otro de alto consumo deben ser alimentados a través de un concentrador externo.

Se provee un puerto HDMI, salida de video y un banco especial para conexión del módulo de cámara, diseñado exclusivamente para ser utilizado con las SBC y conectado a través de un puerto Hirose DF30FC-24DP-0.4V, no disponible en las PC convencionales. Soporta resoluciones de video desde 640 x 480 VGA hasta 1920 x 1080. También tiene soporte para salida de audio y video RCA NTSC y PAL hacia televisores de tecnología CRT.

Software

Existe un sistema operativo particular optimizado para esta placa: Raspbian, lanzado en Julio de 2012. Debido a que Raspbian está basado en Debian, teóricamente el sistema de reconocimiento facial desarrollado podría ser ejecutado sobre cualquier distribución de Linux basada en el mimo sistema operativo. La imagen del firmware es conocida como *binary blob* cuyos drivers son propietarios, accesibles a través de bibliotecas. El *software* de aplicación utiliza llamadas a bibliotecas cargadas en tiempo de ejecución que se comunican con los drivers del kernel de Linux.

Las aplicaciones de vídeo utilizan OpenMAX, las aplicaciones 3D y 2D utilizan OpenGL ES 2.0 y OpenVG respectivamente.

En Febrero de 2012, la fundación Raspberry Pi lanzó su primera prueba de concepto para cargar una imagen de sistema operativo, la cual estaba basada en Debian 6.0 (Squeeze). Luego, el 8 de Marzo se lanzó el Raspberry Pi Fedora Remix que es la distribución de Linux diseñada para la placa y fue desarrollada por el Senaca College en Canadá.

Acoplamiento de dispositivos

Teóricamente cualquier dispositivo soportado puede ser acoplado a la Raspberry Pi, siempre y cuando los *drivers* pertinentes funcionen de manera correcta. Generalmente los dispositivos son conectados a través de un puerto USB y reconocidos por el sistema operativo en ejecución, lo que permite conectar cámaras web de una manera muy sencilla. Si se trabaja con una interfaz gráfica, el sistema mostrará un ícono que representa el dispositivo conectado. Sin embargo no todos los *drivers* están desarrollados para todos los dispositivos, por lo que la fundación Raspberry Pi pone a disposición de los usuarios una sección de preguntas y ayuda para reportar este tipo de casos.

2.5.3. Cámaras digitales

Una cámara digital es un dispositivo generalmente portátil que contiene una superficie fotosensible que graba imágenes a través de un lente [21], para posteriormente ser visualizadas y/o procesadas en un computador. Las cámaras digitales han ganado terreno en el mercado a través de los años. En 1997, se ofrecieron las primeras cámaras fotográficas para consumidores de un megapíxel.

Con la finalidad de definir el alcance de la investigación y determinar qué tipo de cámaras digitales pueden ser utilizadas para servir como entrada al sistema antes mencionado, se lleva a cabo una descripción de cada uno de los tipos de cámaras más populares de la actualidad.

Cámaras réflex

Poseen un visor réflex, gracias al cual se puede ver directamente a través del objetivo y no una recreación digital en una pantalla. Los objetivos son intercambiables, llegando a tener un surtido muy variado de focales, calidades y precios. Entre sus características más importantes está el empleo de sistemas de control para la automatización de la mayoría de los mecanismos, tanto de dispositivos de obturación, sincronización con flashes (tanto internos como externos), y en general la mayoría de funciones de la cámara, aunque se siguen comportando en la mayoría de aspectos (enfoque, disparo, estabilización) como dispositivos electromecánicos al igual que sus predecesoras.

Cámaras compactas

Suelen ser cámaras de fácil manejo, baratas y destinadas a un público que se inicia en el mundo de la fotografía. Suelen tener un zoom óptico entre 3x y 5x por lo que son más indicadas para fotografía de paisaje, arquitectónica o cualquiera que no exija focales muy largas.

Normalmente su funcionalidad está limitada en comparación con las réflex, ya que su objetivo no es desmontable, aunque suelen ser más ligeras y fáciles de transportar, lo que las hace ideales para llevarlas de viaje.

Cámaras ultrazoom

Son un poco más avanzadas que las compactas, y su principal diferencia es el zoom óptico que puede llegar hasta los 18x. Siguen siendo cámaras con el objetivo fijo. Son más versátiles que las compactas ya que el zoom óptico permite otro tipo de fotografías donde se utilizan focales más largas. Algunos defectos de las cámaras de gama media de este tipo es la lentitud de enfoque, o dificultad para enfocar en líneas generales, cuando intentamos realizar una captura con muchos aumentos. Por otro lado, la trepidación será muy acusada en muchas tomas.

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 [22] 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.

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 transmitida en la red.

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.

2.5.4. Herramientas de *software*

Al igual que en la Sección 2.5.1, se hizo una comparación exhaustiva de algunas de las herramientas de *software* más populares para llevar a cabo el reconocimiento facial, en la Tabla 2.3 generada a partir de la investigación, se muestra una comparación de las carecterísticas más relevantes de las mismas.

Se puede notar que los sistemas operativos que soportan cada una de las bibliotecas son un factor estándar, lo que no sucede con los lenguajes de programación, algoritmos de reconocimiento facial y documentación. OpenCV puede ser utilizado en una mayor variedad de de lenguajes de programación que las otras bibliotecas, provee tres algoritmos de reco-

nocimiento facial implementados al igual que EmguCV y posee extensa documentación, a diferencia de GNU Octave y PyFaces.

Para la realización de esta investigación fue seleccionada la biblioteca OpenCV por ser la que posee más opciones de algoritmos de reconocimiento facial implementados, lo cual brinda más flexibilidad a la hora de buscar la solución al problema de reconocimiento. Esto, sumado a la amplia documentación que posee y el hecho de ser multiplataforma contribuyó a la decisión de seleccionarla. Igualmente, OpenCV es utilizada mundialmente en una gran cantidad de desarrollos, incluyendo aquellos que son ejecutados sobre una Raspberry Pi, lo que supone una ventaja en la investigación una vez elegida la herramienta de *hardware*.

Tabla 2.3: Comparación de herramientas de *software*

	Sistemas operativos	**Lenguajes de programación**	**Algoritmos de Reconocimiento Facial**	**Documentación**
OpenCV	Linux, Windows, MAC OS X, Android y iOS	C++, Java, etc. Existen implementaciones de interfaces para Python, Ruby, Matlab y otros	*eigenfaces*, *fisherfaces* y LBP	Posee una extensa documentación con ejemplos y explicaciones detalladas, así como una gran comunidad
OpenCV-Python	Linux, Windows y MAC OS X	Python, pero por ser un *wrapper* de OpenCV ejecuta código en C/C++ por lo que es más lento que OpenCV	*eigenfaces*, *fisherfaces* y LBP	Posee una extensa documentación
EmguCV	Windows, Linux, Mac OS X, iOS, Android y Windows Phone	.Net, pero por ser un *wrapper* de OpenCV ejecuta código en C/C++ por lo que es más lento que OpenCV	*eigenfaces*, *fisherfaces* y LBP	Posee una extensa documentación
GNU Octave	Unix y Windows	GNU Octave	*eigenfaces*	Posee poca documentación
PyFaces	Windows, Linux, Mac OS-X, Unix y Solaris	Python	*eigenfaces*	Posee poca documentación

Una vez seleccionada OpenCV, también se decidió el lenguaje de programación a usar,

tomando en cuenta la amplia variedad de lenguajes que son compatibles con la biblioteca y el nivel de experiencia que se posee con cada uno de ellos. Debido a la naturaleza de la solución implementada, basada en web, se decidió usar Python, que aunque en su base es un *wrapper* de la biblioteca en C++, entrega un rendimiento bastante bueno y se comporta muy bien para las necesidades presentes.

OpenCV es una biblioteca de código libre originalmente desarrollada por Intel en 1999 por Gary Bradsky, especializada en la visión por computador. Está escrita en C y C++ con parte en lenguaje ensamblador y funciona en los sistemas operativos Linux, Windows, MAC OS X, Android y iOS. Existen implementaciones de interfaces para Python, Ruby, Matlab y otros lenguajes. Su primera versión fue publicada en el año 2000 y actualmente su versión estable es la 3.1.0. Interfaces para operaciones de alta velocidad en la GPU, basadas en CUDA y OpenCL se encuentran en desarrollo.

OpenCV fue diseñada para ser eficiente computacionalmente, enfocada en aplicaciones de tiempo real [23] y toma ventaja de sistemas con procesadores multinúcleo, además hace uso de características especiales de los procesadores como las extensiones multimedia MMX de Intel o las extensiones NEON de los procesadores ARMv7. Uno de los objetivos de OpenCV es proveer una infraestructura para visión por computador fácil de usar, que ayude a las personas a construir aplicaciones sofisticadas de esta área rápidamente [23].

La biblioteca contiene más de 500 funciones que cubren muchas áreas de la visión por computador [23], incluyendo imágenes médicas, seguridad, interfaces de usuario, calibración de cámaras y robótica. Además como estas áreas y la de aprendizaje automático (*machine learning*) van de la mano, OpenCV provee una biblioteca de propósito general llamada *Machine Learning Library* (MLL). Esta biblioteca se enfoca en el reconocimiento estadístico de patrones y la ramificación.

Para resolver el problema de reconocimiento facial, OpenCV posee tres algoritmos distintos ya implementados, *eigenfaces*, *fisherfaces* y LBP.

Sumado a OpenCV se utilizaron diferentes herramientas de *software* de código abierto disponibles en la actualidad, para el desarrollo del sistema de reconocimiento facial. En esta Sección se describirá cada una de ellas a groso modo, para dar una idea general de su funcionamiento.

Crossbar.io

Crossbar.io es un enrutador de código abierto para el protocolo WAMP (*Web Application Messaging Protocol* - Protocolo de Comunicación de Aplicaciones Web) diseñado para permitir la creación de componentes en aplicaciones distribuidas [24]. Estos componentes pueden ser páginas web, aplicaciones de *back-end*, clientes móviles, microservicios o dispositivos de IoT (*Internet of Things* - El Internet de las Cosas). El objetivo de Crossbar.io es permitir a cada componente, conectarse con los demás como iguales.

WAMP es un subprotocolo de *WebSockets* [25] que ofrece patrones de comunicación como rRPC (*routed Remote Procedure Call* - Llamadas enrutadas a Procedimientos Remotos) y PubSub (*Publish/Subscribe* - Publicación/Suscripción) [26]. WAMP acepta conexiones desde clientes y enruta llamadas y eventos entre ellos.

Los clientes se conectan a Crossbar.io y se utilizan *WebSockets* como medio de transporte, lo que significa que los componentes pueden ejecutarse en cualquier lugar donde existan conexiones HTTP. Una vez que la conexión se establece, la misma es bidireccional.

Para rRPC, un componente registra un procedimiento con Crossbar.io, mientras que otro componente puede llamarlo para obtener el resultado a través del enrutamiento de Crossbar.io.

En PubSub, un componente publica tópicos (información) y otro se suscribe a los mismos, el protocolo se encarga de entregar los mensajes correctos a los suscriptores apropiados. Esto permite la distribución eficiente de información a través de los componentes de la aplicación.

Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma, asíncrono, de código abierto para desarrollar *software* del lado del servidor. Está basado en el *ECMAScript* [27] y posee una arquitectura orientada a eventos.

Existe un hilo de ejecución principal, enlazado con un ciclo de eventos, que escucha a llamadas desde otros hilos, implementados como un *pool* en C++. Las tareas diferidas son encapsuladas, entrando y saliendo del entorno de ejecución por medio de *callbacks*. Las operaciones de entrada/salida generan flujos de datos con eventos, de modo que se mantenga el paradigma en caso de que se realicen alguna de estas operaciones [28]. La concurrencia es manejada por el sistema y las dependencias se incluyen a través de su manejador de paquetes, npm (*Node Package Manager* - Manejador de Paquetes de Node) [28].

Mongo DB

MongoDB es un sistema de base de datos de código abierto, no relacional y orientado a documentos. Su unidad principal de información es el documento, y utiliza el formato JSON para almacenar la información.

Este sistema fue diseñado para ser utilizado en aplicaciones web e infraestructura de Internet [29]. El modelo de datos y las estrategias de persistencia están construidas para alto tráfico de lectura y escritura, y la habilidad de escalar fácilmente, por lo que puede entregar un rendimiento bastante alto.

AngularJS

AngularJS es un framework de JavaScript de código abierto, mantenido por Google, construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de *software*, mientras que la programación imperativa es para expresar la lógica de negocio [30]. Este framework adapta y amplía el HTML tradicional para servir contenido dinámico a través de lo que se denomina *data binding* o enlance de los datos (*two-way binding*), que permite la sincronización automática de modelos y vistas [30]. Como resultado, AngularJS pone menos énfasis en la manipulación del DOM.

Gracias a sus características AngularJS logra disociar la manipulación del DOM de la lógica de la aplicación. Esto mejora la capacidad de prueba del código [30]. Así como también se separa el lado del cliente del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados lados de la aplicación (cliente y servidor).

AngularJS sigue el patrón MVC (*Model View Controller* - Modelo Vista Controlador). Con el uso de la inyección de dependencias maneja servicios que tradicionalmente se realizarían en el lado del servidor en controladores dependientes de la vista. En consecuencia, gran parte de la carga en el *back-end* se reduce, lo que conlleva a aplicaciones web mucho más ligeras [30].

REST

REST (*Representational State Transfer* - Transferencia de Estado Representacional) se refiere a un estilo de arquitectura de la web. Jakl [31] define REST como un conjunto coordinado de restricciones de arquitectura que intenta minimizar la latencia y cantidad de comunicación en la red mientras maximiza la independencia y escalabilidad de los componentes implementados. REST permite la reutilización de interacciones, sustitución dinámica de componentes y procesamiento de acciones por intermediarios, de modo de cumplir con las necesidades de los sistemas distribuidos de hipermedia.

Express.js

Express.js es un *framework* robusto, rápido, flexible y muy simple. Ofrece un mecanismo que combina una dirección, un método de solicitud HTTP y uno o más controladores, con el fin de realizar peticiones de conexión con bases de datos y definir rutas para el direccionamiento dentro de la aplicación [29].

2.6. Estado del arte

En los últimos años se han desarrollado soluciones de vigilancia basadas en reconocimiento facial que involucran el uso de la Raspberry Pi como uno de sus componentes principales. En esta Sección se hará un recorrido por algunas investigaciones hechas desde 2010 hasta la fecha, que tratan acerca de este tipo de soluciones.

2.6.1. Sistemas de seguridad con reconocimiento facial utilizando *Single Board Computers*

Kanzanriya et al. [32] utilizaron una Raspberry Pi como microprocesador para construír un sistema de monitoreo a través de vídeo en tiempo real utilizando Internet. Los datos del mismo son capturados a través del módulo de cámara de la Raspberry Pi, comprimidos a un formato MPEG y transmitidos utilizando cualquier conexión disponible en la placa. Utilizan

Linux Embebido como sistema operativo, y sobre él se ejecuta todo el flujo del proceso, como lo es: inicialización de la cámara, captura, compresión, transmisión y presentación del vídeo.

Rajni et al. [33] propusieron una red sensorial multimedia para detección de intrusos y actividades ilegales en fronteras. Debido al bajo consumo de energía eligieron una Raspberry Pi como placa principal, el módulo de cámara de la misma y utilizaron también un receptor pasivo de infrarrojo adjuntado directamente a la misma. La red monitorea el entorno desde varias localizaciones, donde en cada una se encuentra un nodo constituído por una Raspberry con cámara y sensor infrarrojo. Un transceptor [3] es utilizado para transmitir la información generada por los sensores y cámaras hacia una estación central que reúne toda la data.

Amaya et al. [34] desarrollaron un prototipo de sistema de seguridad vehicular que utiliza reconocimiento facial. Dicho sistema captura el rostro del conductor a través de una cámara situada dentro del automóvil de manera de determinar si el individuo está autorizado y así permitir o no el encendido del mismo. Como herramienta de *software* utilizaron el textitMatlab Image Processing Toolbox y como herramientas de *hardware* se propuso la utilización de una cámara digital con visión nocturna, un SBC UDOO y una pantalla táctil desarrollada por UDOO. Sin embargo, el desarrollo final se llevó a cabo utilizando un computador portátil Hp Pavilion 14 con cámara web *Truevision* integrada, y una placa Arduino para mostrar dos señales de salida a través de leds (rojo y verde), indicando si el individuo fue reconocido o no.

Senthilkumar et al. [35] trabajaron con la Raspberry Pi para desarrollar un sistema de reconocimiento facial embebido que trabajase sobre la placa en dos fases: la primera, adquisición de los datos a través de una cámara conectada a la Raspberry y la segunda, ejecución del *software* de reconocimiento facial desarrollado utilizando el algoritmo de *eigenfaces*. El objetivo principal del proyecto fue que todo el sistema trabajase únicamente sobre la Raspberry Pi sin necesidad de componentes adicionales para llevar a cabo el cómputo más demandante.

Singh et al. [36] utilizaron cámaras IP conectadas a una Raspberry Pi para desarrollar un sistema de vigilancia capaz de detectar rostros en la escena y transmitir el vídeo para ser visualizado en tiempo real en cualquier navegador a través de Internet. Hicieron uso de OpenCV como herramienta de *software* para llevar a cabo el proceso de detección facial, donde fue utilizado el clasificador de Haar. Los autores aseguran que los resultados entregados por el sistema podrían mejorar utilizando fuentes de luz montadas sobre la cámara IP.

Sistemas de control de acceso para puertas han sido desarrollados utilizando técnicas de reconocimiento facial, como es el caso de Sahani et al. [37] quienes desarrollaron una solución web constituída por dos componentes, un WCU (*Wireless Control Unit* - Unidad de Control Inalámbrica) y una WIU (*Wireless Information Unit* - Unidad de Información Inalámbrica), enlazadas a través de una red con la tecnología ZigBee. La WIU cuenta con un módulo de servicio general de paquetes vía radio (*General Packet Radio Service*) para transmitir la información hacia la red pública, utiliza una Raspberry Pi modelo B+ como unidad de procesamiento principal, una cámara digital para obtener la información de vídeo y se encuentra conectado a la web haciendo uso del protocolo TCP/IP. Por otra parte, la WCU constituye el sistema que permite la apertura automática de puertas a través de cerraduras

[3] Un transceptor es un dispositivo que funciona como transmisor y receptor de información.

electromagnéticas. Al momento de llegar algún visitante, la Raspberry Pi envía un mensaje de texto y correo electrónico a un destinatario específico, con el nombre y fotografía del sujeto (cuando aplique) para que el propietario pueda tomar acciones. Éste puede visualizar en tiempo real la transmisión del vídeo a través de una página web. El módulo de reconocimiento facial fue desarrollado utilizando PCA y *eigenfaces*. La web para visualización de vídeo fue desarrollada utilizando HTML, JavaScript, PHP y SQLite.

En los trabajos consultados, si bien existe *software* de reconocimimento facial, siendo ejecutado sobre un *SBC*, no se provee ninguna interfaz para administrarlo, no existe ninguna alarma que alerte a algún administrador sobre actividades sospechosas, ni se definen pruebas estadísticas que demuestren cuál es el rendimiento del sistema. El *software* desarrollado en este trabajo especial de grado, puede ser accedido desde una interfaz común en la actualidad, como lo es un navegador web, para visualizar la escena en tiempo real y administrar cada uno de los datos de la aplicación de una manera sencilla, sin la necesidad de que el encargado de realizar este trabajo cuente con experiencia en programación. Posee un módulo de alarma que envía una alerta vía correo electrónico en caso de detectar rostros desconocidos. Además, el trabajo provee resultados basados en pruebas estadísticas que demuestran el buen funcionamiento del mismo. Igualmente, el *software* desarrollado en esta investigación es más flexible que los tomados como referencia, debido a que permite realizar conexión con distintos tipos de cámaras, como lo son web e IP.

2.7. Marco metodológico

El desarrollo se llevó a cabo utilizando una metodología *Ad-hoc*, debido al número de miembros del equipo de desarrollo y la naturaleza del trabajo. Esta metodología utilizó además, algunos conceptos y prácticas pertenecientes a XP (*Extreme Programming*, Programación Extrema) como base para definir la forma de trabajo. A continuación se describe cada uno de estos conceptos en conjunto con una definición de XP, haciendo énfasis en los elementos utilizados.

2.7.1. *Extreme Programming*

XP (*Extreme Programming*, Programación Extrema) es una metodología ágil propuesta por Kent Beck [38], con la finalidad en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

2.7.2. Ciclo del proceso de XP

El ciclo del proceso puede ser observado en la Figura 2.6, donde se define una entrega que debe ser planificada, dividida en iteraciones que producen entregas parciales hasta que no

existan más iteraciones. Una vez que se lleva a cabo una entrega parcial, se realizan pruebas de aceptación para definir si hay errores, de modo de mantener la calidad del producto.

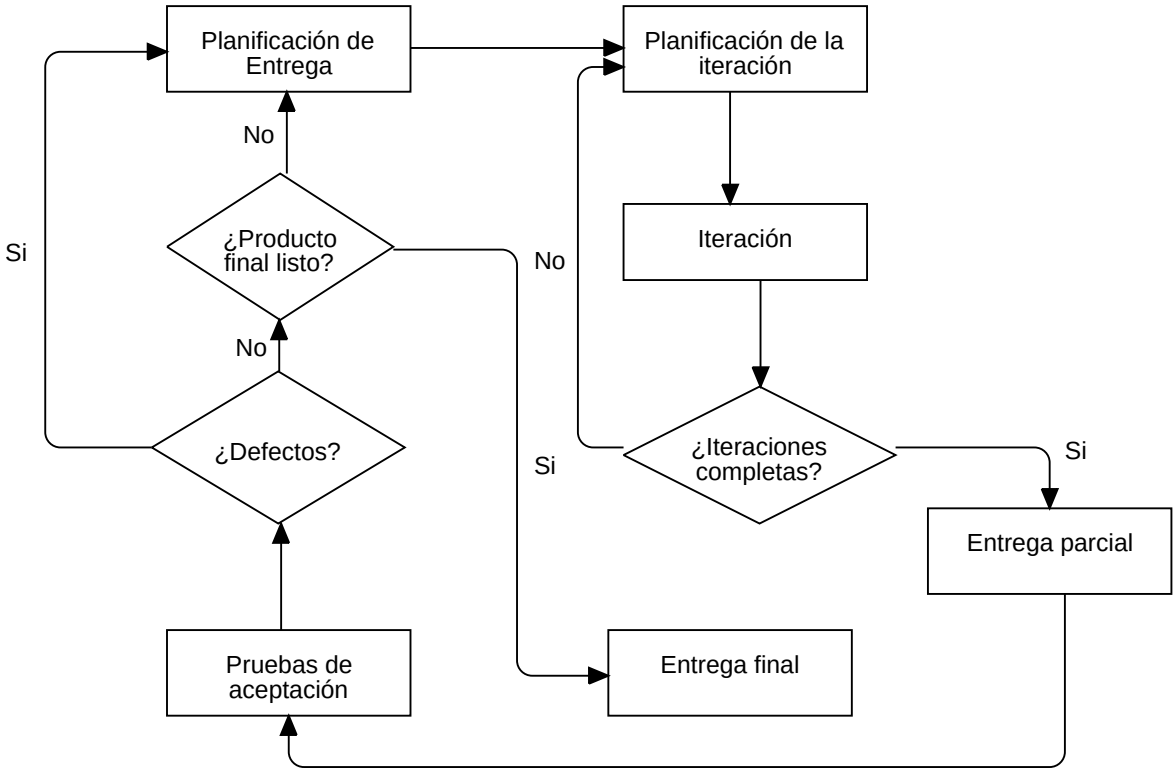

Figura 2.6: Ciclo del proceso de XP

XP es iterativo e incremental, y está manejado por ciclos definidos por tiempo. La metodología define 6 niveles de actividades: Ciclos de vida del producto, lanzamientos, iteraciones, tareas, desarrollo y retroalimentación.

1. **Ciclos de vida del producto**

   También llamado fase de exploración, involucra el conjunto de características, definición y planificación. El cliente requiere características de alto valor, las cuales se definen como historias de usuario.

2. **Lanzamientos**

   También llamada fase de compromiso, incluye la reunión del equipo para verificar el avance del proyecto, añadir o eliminar nuevos requerimientos, presentar nuevas historias y discutirlas. Los desarrolladores determinan la aproximación técnica para solucionar el problema y su riesgo, de modo de proveer estimaciones. El cliente le da prioridad a las historias y define en conjunto con los desarrolladores, qué módulo debe ser desarrollado primero.

3. **Iteraciones**

Incluye las iteraciones sobre el sistema antes de ser entregado, involucrando las reuniones del equipo con el cliente para definir el alcance de cada una de las iteraciones. Los desarrolladores determinan los riesgos y maneras de cumplir con la iteración, comienzan el desarrollo y despliegan el sistema hasta el punto en el que se ha desarrollado.

4. **Tareas**

   Dividen las iteraciones en elementos más pequeños, definiendo episodios de desarrollo para implementar las historias. Los desarrolladores aseguran que las historias para la iteración están completas con pruebas de aceptación.

5. **Desarrollo**

   Involucra el desarrollo de cada historia, que puede ser una actividad continua y dinámica. Para esto se usa un esquema de programación en pares, quienes verifican el entendimiento de la historia, determinan el enfoque apropiado para el desarrollo, aseguran que las pruebas se lleven a cabo de manera correcta, integran el código al sistema base y revisan el progreso frecuentemente.

6. **Retroalimentación**

   La última fase del ciclo, en la que los pares se comunican entre ellos para recibir una evaluación informal acerca de su trabajo. El cliente puede estar involucrado de una manera contínua.

2.7.3. Roles de XP

Existen diferentes roles en la metodología, que pueden variar según quien la defina, aunque la definición original de Beck [38] incluye sólo seis roles:

- **Programador**

  El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

- **Cliente**

  El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

- **Encargado de pruebas**

  El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

- **Encargado de seguimiento**

  El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

- **Entrenador**

  Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- **Consultor**

  Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

- **Gestor**

  Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

2.7.4. Metodología *Ad-hoc*

La metodología utilizada para desarrollar la investigación estuvo basada en iteraciones que sirvieron para que la solución se construyera de manera incremental. Como puede observarse en la Tabla 2.4, la primera iteración estuvo destinada a diseñar de manera conceptual la arquitectura del sistema, como se describe en la sección 3.1. Luego se llevaron a cabo las siguientes iteraciones para desarrollar cada uno de los módulos previamente definidos, que internamente se dividieron en tareas manejadas a través del método Kanban [39], el cual utiliza tableros que mantienen las tareas en diferentes etapas, con el fin de mantener la consistencia y organización de las mismas, y así evitar la pérdida de tiempo. Se utilizó la herramienta Trello para la definición y uso de estos tableros. En el proyecto, fueron definidos tres tableros que cumplieron una función específica: ***TODO***, para alojar todas las tareas definidas por desarrollar, ***In Progress***, referente a las funcionalidades en proceso de desarrollo, y ***DONE***, para los requerimientos desarrollados y probados. La Figura 2.7 muestra una captura de pantalla de la aplicación con el progreso.

Tabla 2.4: Iteraciones realizadas en el proceso de desarrollo

Iteración	Descripción
1	Definición de la arquitectura de referencia, módulos candidatos e iteraciones siguientes.
2	Desarrollo del módulo de reconocimiento facial.
3	Desarrollo de la interfaz de programación de aplicaciones para establecer conexión con la base de datos.
4	Desarrollo del módulo de visualización y entrenamiento utilizando una interfaz web.

Los roles de programador y cliente fueron los únicos presentes en el desarrollo de la solución, debido a que el equipo no tenía la cantidad suficiente de integrantes para ocupar cada uno de los puestos.

También se utilizó el esquema de programación en pares (*Pair Programming*) [40], para el desarrollo de código fuente. Cada uno de los programadores desarrolló módulos distintos, pero mientras uno de los miembros escribía, el otro daba sugerencias y analizaba el código de la forma como lo define el esquema.

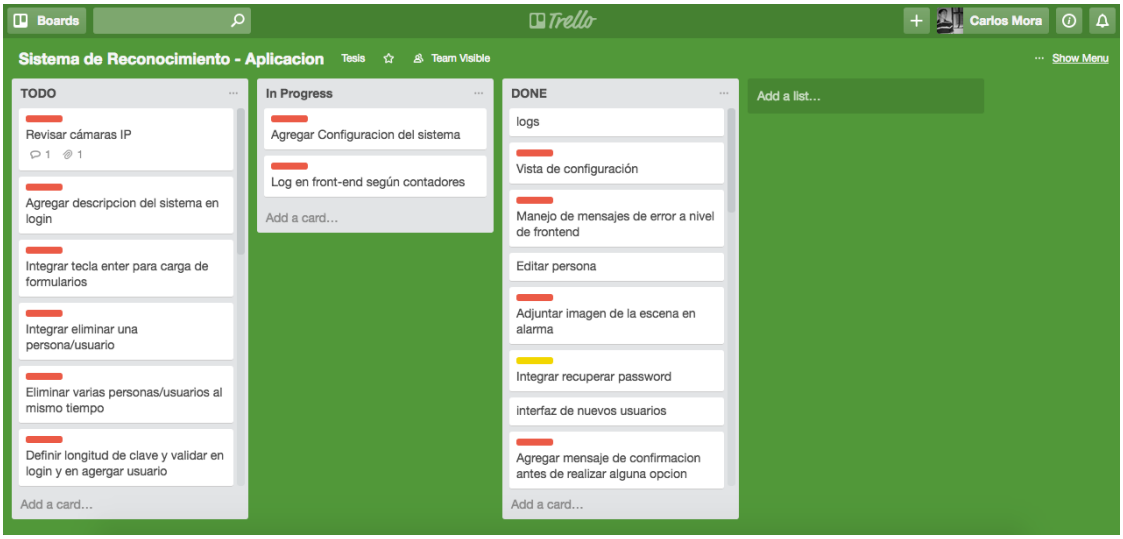

Figura 2.7: Tablero en Trello para organización de tareas

Capítulo 3

Implementación

En este Capítulo, se presenta la explicación del diseño de la arquitectura del sistema y el flujo de la aplicación final, haciendo uso del diagrama de casos de uso para ilustrar, de forma clara el alcance de la misma. Finalmente, se expone de manera detallada, los pasos que se siguieron para la implementación del sistema.

3.1. Diseño arquitectónico

Para el siguiente trabajo se propuso e implementó una arquitectura para un sistema de seguridad basada en una Raspberry Pi a nivel de *hardware* y en MEAN *Stack* para los componentes del *software*. La Figura 3.1 ilustra como se relacionan los distintos elementos de la aplicación.

Los componentes de *hardware* del sistema son una cámara web que captura la imagen de la escena y se encuentra conectada a una Raspberry Pi, en donde se ejecutan los elementos de *software* de la aplicación. Como se ve en la Figura 3.1, también es necesario un dispositivo que haga la función de cliente, el cual puede ser cualquier dispositivo que posea un navegador web.

MEAN es el acrónimo que referencia arquitecturas desarrolladas con MongoDB, Express, AngularJS y Node.js [41]. Para la aplicación en cuestión se usa un esquema basado en MEAN *Stack*, al cual se le agregan un módulo de reconocimiento facial desarrollado sobre OpenCV-Python y el uso de Crossbar.io para el manejo de websockets. Todos estos elementos conforman el *software* del sistema de seguridad. Como se muestra en la Figura 3.1 la Raspberry Pi envía la escena capturada por la cámara al módulo de reconocimiento facial, en donde la misma es procesada y los resultados se envían a través de un websocket al cliente.

El módulo de *front-end* fue realizado utilizando AngularJS y se despliega en el cliente, quien se comunica con el API realizando peticiones (GET, POST, UPDATE, DELETE), ya que no existe una conexión directa entre el cliente y la base de datos. Dicho API se hizo usando Node.js y Express, y funciona como un intermediario entre el cliente y la base de datos. La Figura 3.1 muestra de forma gráfica como se conectan estos tres componentes.

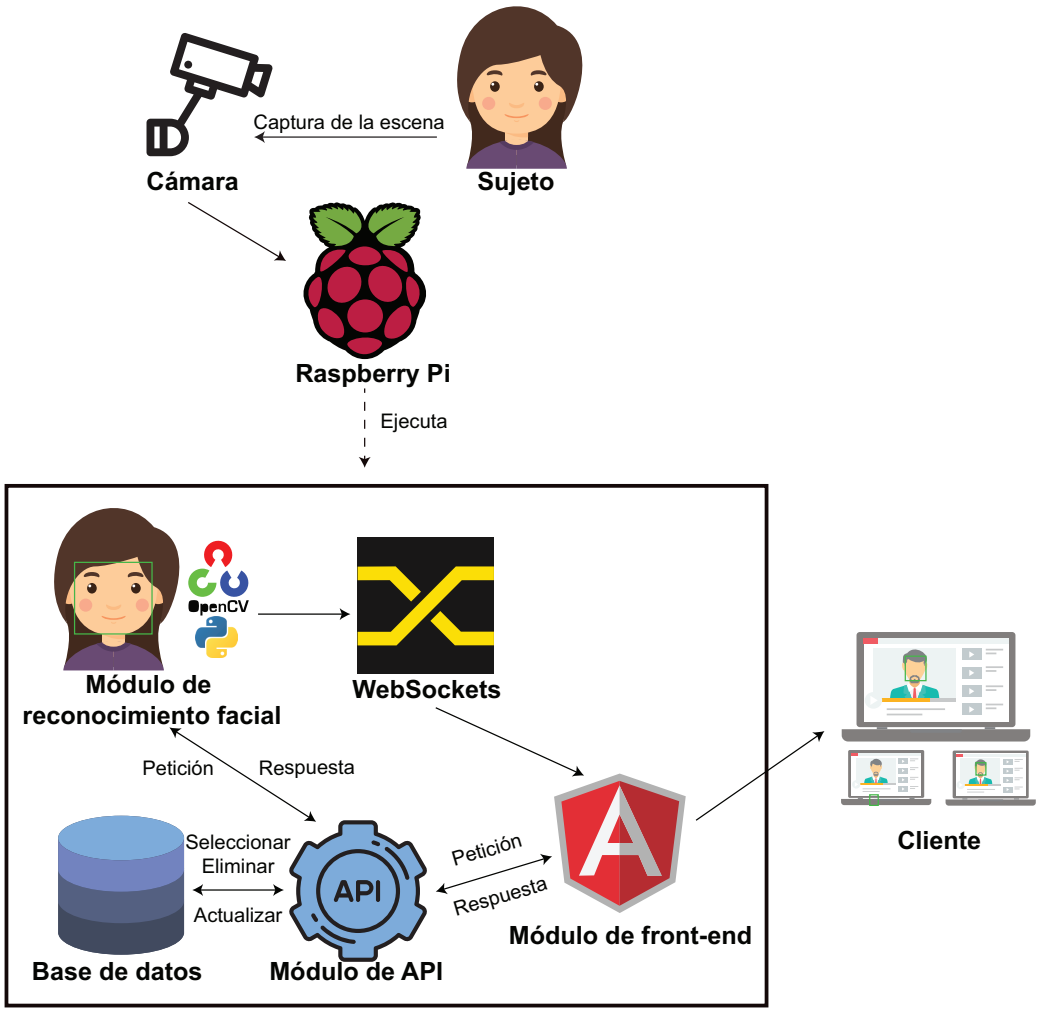

Figura 3.1: Diseño general de la arquitectura del sistema

3.2. Diagrama de casos de uso

La Figura 3.2 muestra el diagrama de casos de uso de la aplicación desarrollada, el cual es de nivel 1 y cuenta con dos actores: Administrador y Usuario.

El administrador cumple el rol de superusuario, con la posibilidad de utilizar todas las funciones del sistema, como lo son el inicio de sesión, gestión de personas registradas en el sistema para ser reconocidas (*people*), gestión de usuarios registrados, configuración del sistema y la visualización de la escena y los *logs*.

Para el caso de las sesiones, gestión de personas y gestión de usuarios es posible llevar a cabo tareas de inserción, eliminación, modificación y listado de los datos, mientras que las configuraciones sólo pueden ser modificadas y listadas.

El usuario general cuenta con un rol restringido, sin la posibilidad de llevar a cabo ciertas tareas en el sistema. Sólo tiene la posibilidad de iniciar sesión, modificar sus datos, ver la

escena y los *logs*.

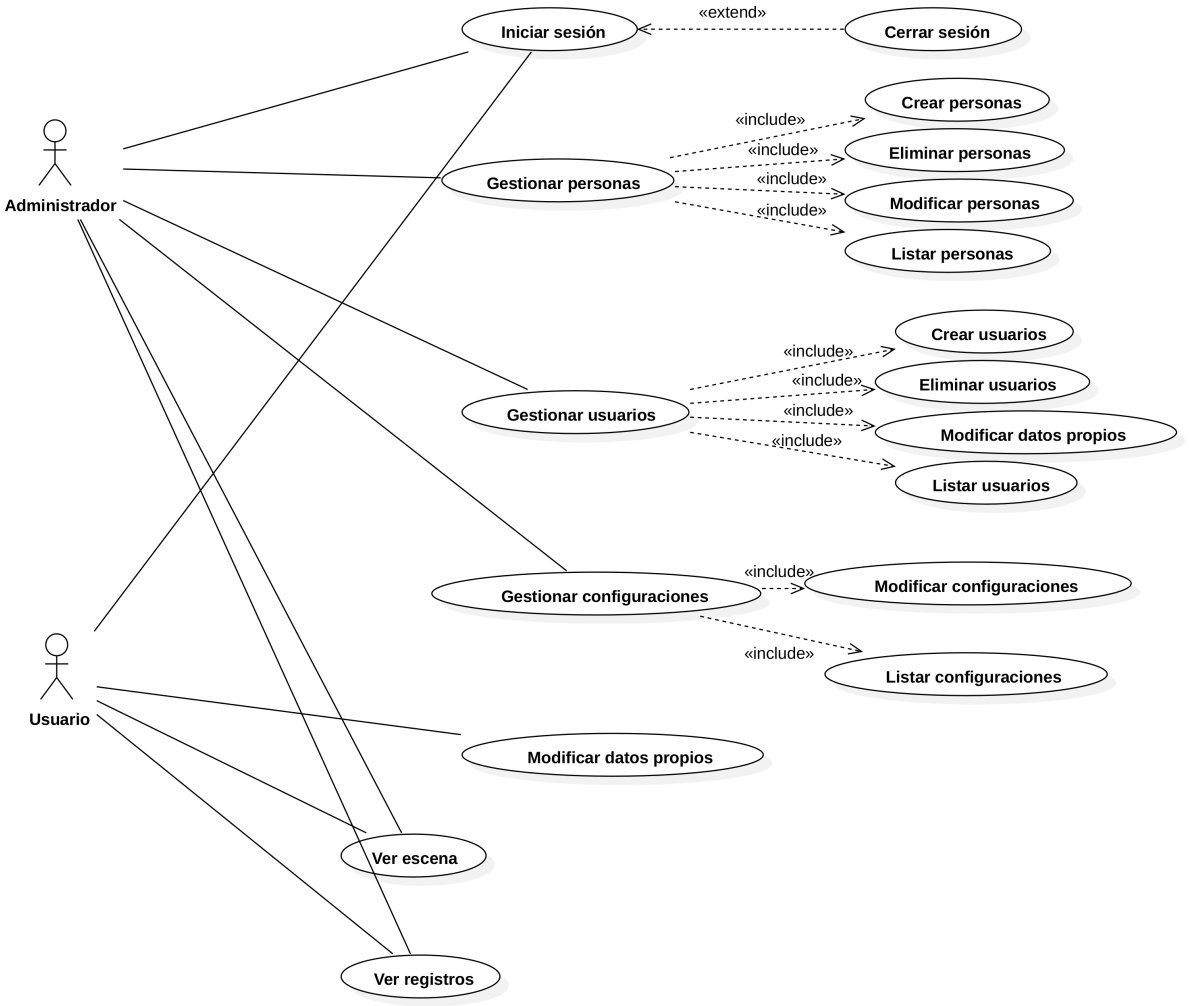

Figura 3.2: Diagrama de casos de uso del sistema

3.3. Diagrama de objetos de dominio

Para ilustrar los objetos que actúan en la aplicación y sus relaciones a muy alto nivel se diseñó el diagrama en cuestión. La Figura 3.3 lo muestra, y describe la cardinalidad de cada uno de sus objetos.

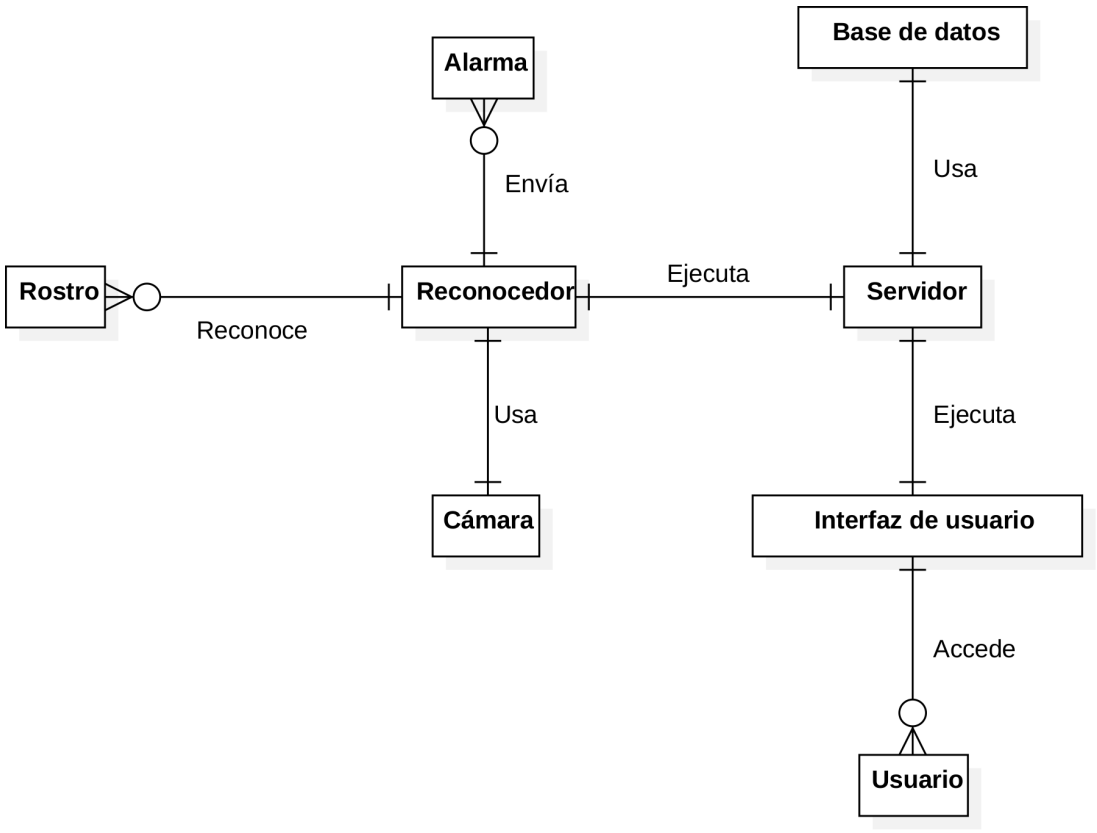

Figura 3.3: Diagrama de objetos de dominio del sistema

Cada uno de estos objetos tienen una función específica: El **reconocedor** debe ser el encargado de reconocer los **rostros** obtenidos a través de la escena, captada por la **cámara** y enviar las respectivas **alarmas** en caso de ser necesario. Este reconocedor en conjunto con la **interfaz de usuario** deben ser ejecutados en un **servidor**, que utiliza una **base de datos**. Finalmente los **usuarios** deben tener la capacidad de utilizar todas las funcionalidades a través de la **interfaz de usuario**.

3.4. Diagrama de componentes

La Figura 3.4 muestra el diagrama de componentes de la aplicación, en el que se muestra de qué manera está dividido cada módulo de la aplicación, así como las dependencias existentes entre cada módulo utilizado.

Cada uno de los componentes siguen la arquitectura definida en la Sección 3.1, agrupándose en los diferentes módulos, los cuales se conectan entre si a través del protocolo HTTP/1.1. El objetivo del diagrama de componentes es especificar con mayor detalle la estructura de la solución.

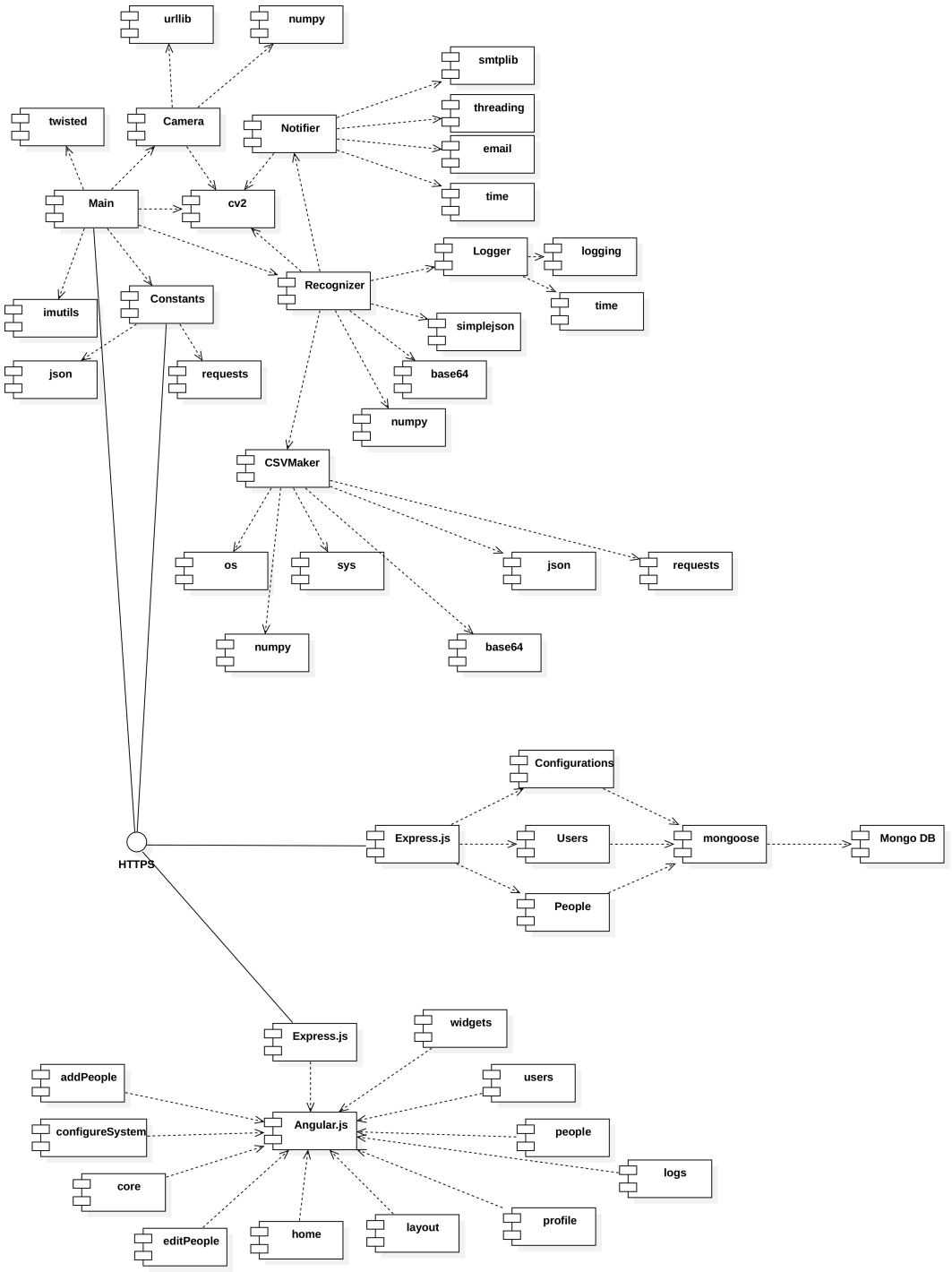

Figura 3.4: Diagrama de componentes del sistema

El módulo ***recognizer*** es el primero de los descritos en 3.1, y está compuesto por varios submódulos desarrollados en el lenguaje de programación Python. ***Main*** es el encargado de definir el *loop* principal de la aplicación, y utiliza ***Camera*** para obtener la información de la

escena, ***Recognizer*** para ejecutar cada uno de los pasos del proceso de reconocimiento facial; ***cv2*** como paquete que encapsula toda la funcionalidad de *OpenCV*; ***Constants*** para obtener todas las configuraciones; junto a otros módulos de desarrolladores externos.

El módulo ***Notifier*** implementa toda la funcionalidad de las notificaciones. ***CSVMaker*** se encarga de crear la estructura de datos necesaria para entrenar al módulo ***Recognizer***.

El módulo **API** se compone de diferentes submódulos creados sobre **Express.js**, como lo son ***Configurations***, ***Users*** y ***People*** para exponer todas las rutas por las cuales es posible acceder a la interfaz. El módulo **Mongoose** para encapsular todas las funcionalidades de la base de datos **MongoDB**.

Finalmente el módulo ***front-end*** utiliza **Express.js** como servidor, sobre el cual se ejecuta **AngularJS**, que a su vez define diversos módulos para acceder a las vistas de la aplicación. Cada uno de estos módulos cuenta con un controlador y una vista tal como lo define la arquitectura de AngularJS.

3.5. Flujo de aplicación

Para el uso del sistema de seguridad se creó una aplicación web, a la cual solo tienen acceso los usuarios registrados en la base de datos. La misma comienza con el módulo de *login* donde se solicitan las credenciales y se presenta una opción para recuperar la contraseña, la cual envía un correo al usuario con una nueva. Una vez que se han ingresado y validado la contraseña y el correo electrónico, como se muestra en la Figura 3.5, se inicia la sesión del usuario y se despliega la página inicial de la aplicación.

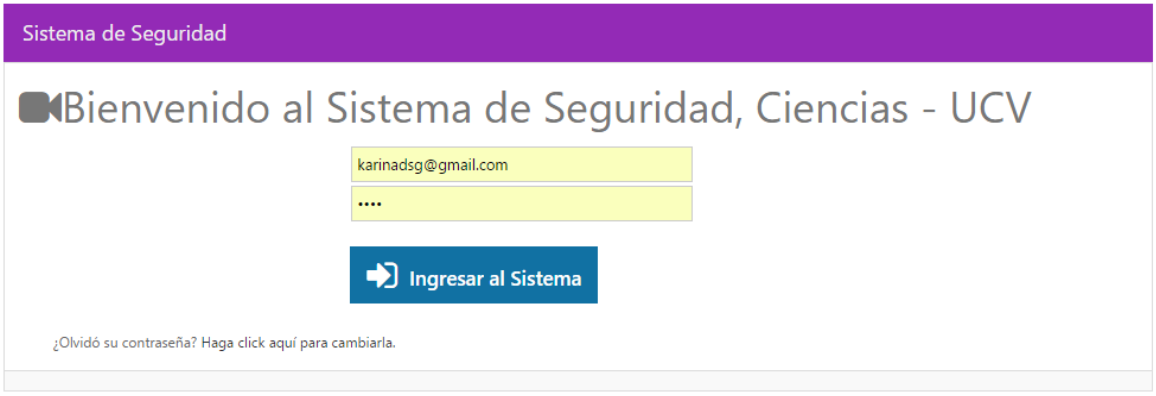

Figura 3.5: Módulo de *login*

La Figura 3.6 presenta la página de inicio de la aplicación, en donde se puede observar la transmisión en tiempo real de la cámara y el listado de personas que han sido reconocidas por el sistema con la hora de la observación. En la imagen de la cámara, se presenta el nombre con el cual ha sido identificada la persona. En la Figura 3.6 se observa un menú superior y lateral, desde los cuales es posible navegar a los distintos módulos de la aplicación o salir de la misma. Este menú lateral varía de acuerdo a los permisos que posea el usuario, esto se

explica en la Sección 3.2. La imágenes de la presente Sección muestran todas las opciones disponibles para un usuario de tipo administrador.

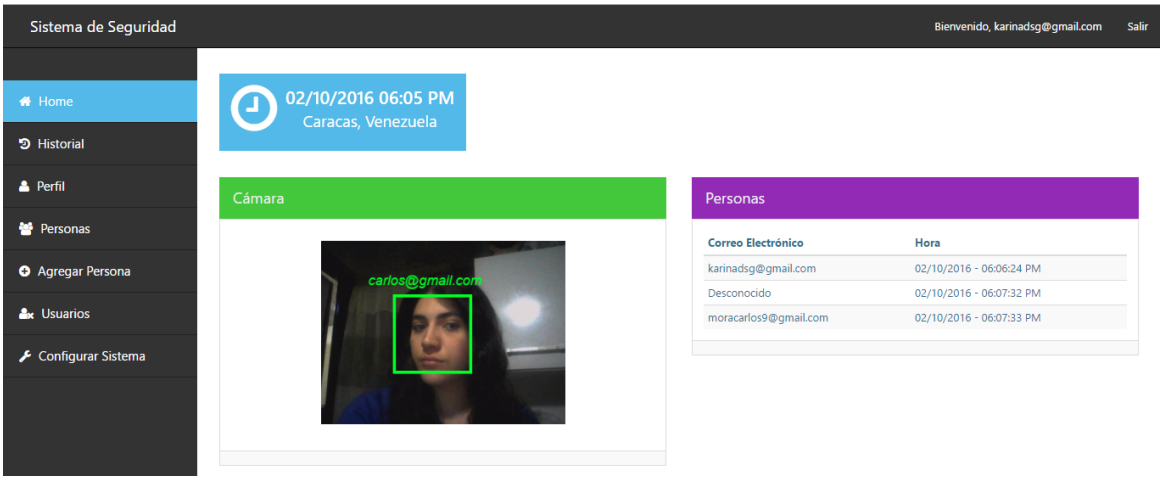

Figura 3.6: Página de inicio de la aplicación

El listado de personas que se muestran en el *home* de la aplicación se puede ver en detalle en el módulo de *logs* de la Figura 3.7. Aquí se muestran todas las observaciones del sistema en tiempo real y la confianza de cada una. En el Capítulo 4 se explica de forma detallada cómo funcionan los *logs* y qué representa la confianza de una observación.

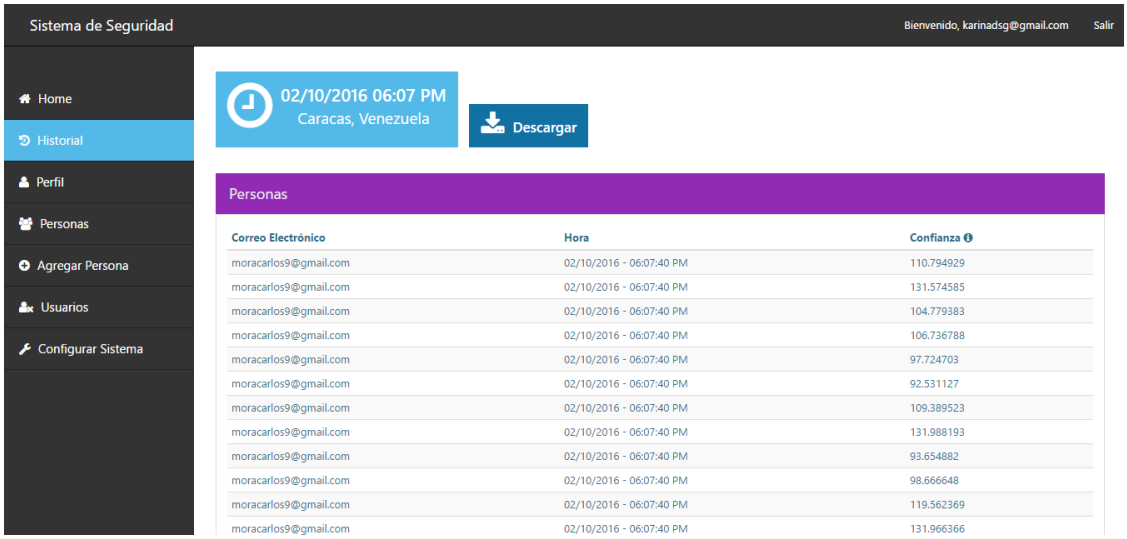

Figura 3.7: Módulo de *logs* en tiempo real

3.5.1. Administración de usuarios

La aplicación permite la administración de usuarios, como se muestra en la Figura 3.8, en donde se listan los usuarios registrados en el sistema, identificando con un escudo a los

administradores. Desde este módulo es posible eliminar, agregar y ver el detalle de los usuarios existentes. También se brinda la opción de búsqueda, ya sea por nombre o por correo electrónico.

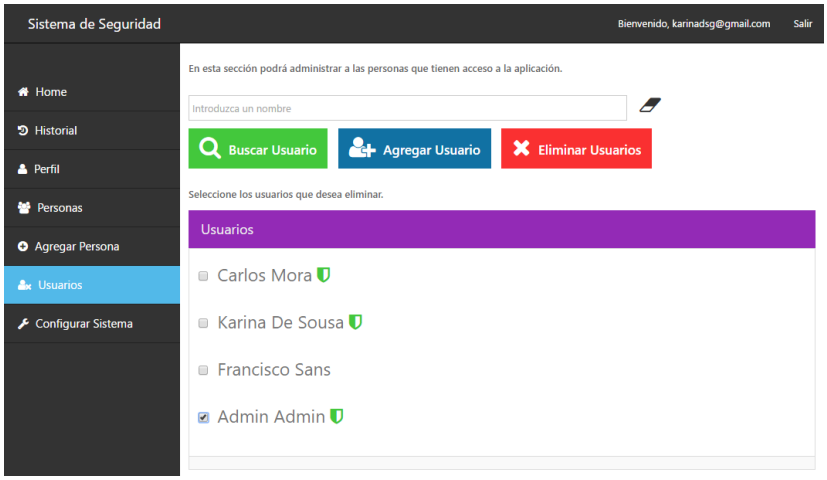

Figura 3.8: Listado de usuarios

La opción de agregar nuevos usuarios se muestra en la Figura 3.9, en la cual se solicita el correo electrónico, contraseña, nombre, apellido y escuela del nuevo usuario. También presenta la opción de dar permisos de administrador al mismo. Al agregar un nuevo usuario, el mismo podrá acceder al sistema usando sus credenciales (correo eléctronico y contraseña).

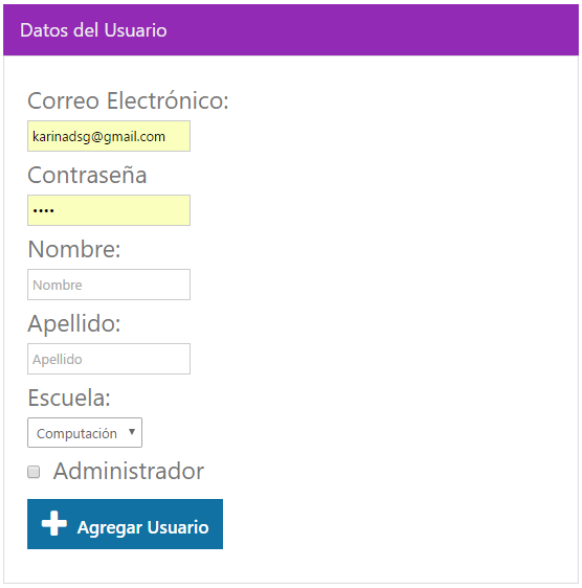

Figura 3.9: Campos requeridos para agregar usuarios al sistema

La Figura 3.10 muestra el detalle de un usuario registrado en el sistema. Los campos mostrados son los mismos que fueron ingresado al momento de agregar un usuario.

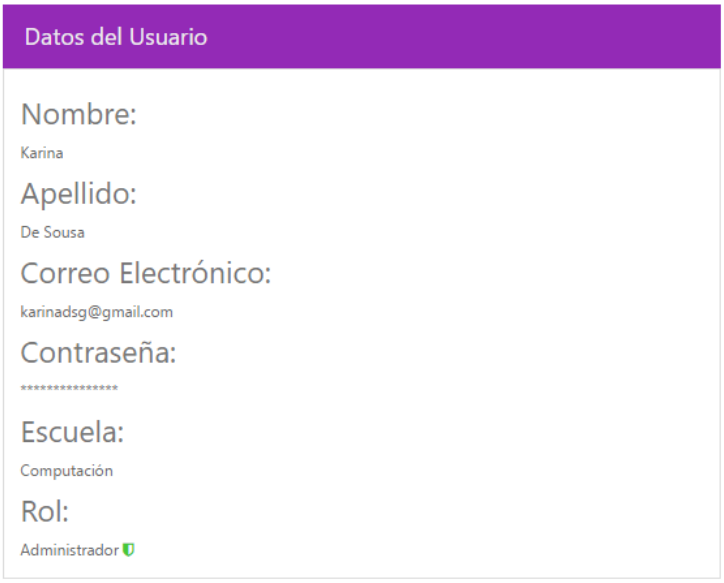

Figura 3.10: Detalle de un usuario registrado en el sistema

La Figura 3.11 muestra el módulo del perfil de usuario, donde se pueden consultar los datos del mismo y editarlos, como se muestra en la Figura 3.12. Estos campos son los mismos ingresados al momento de agregar un usuario.

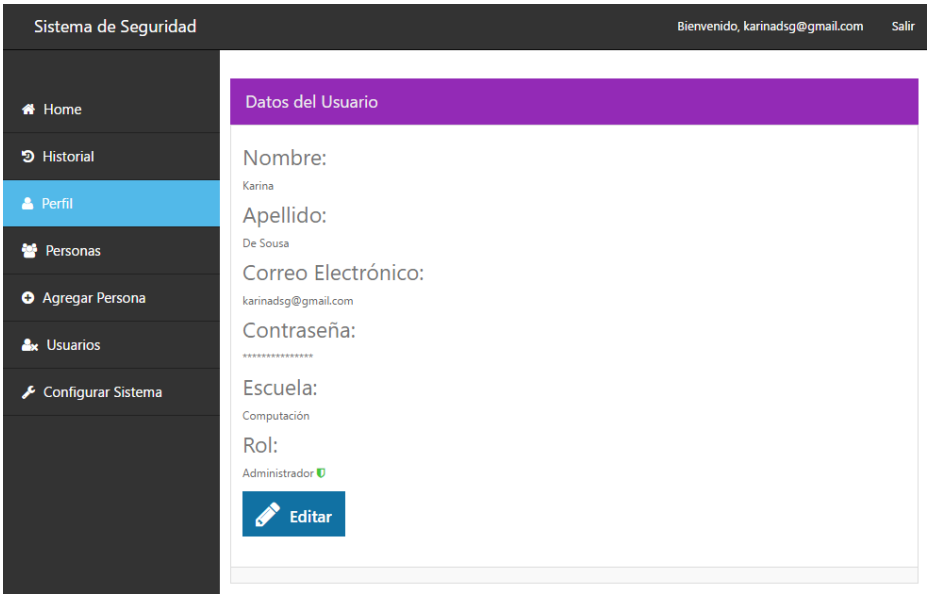

Figura 3.11: Perfil de usuario

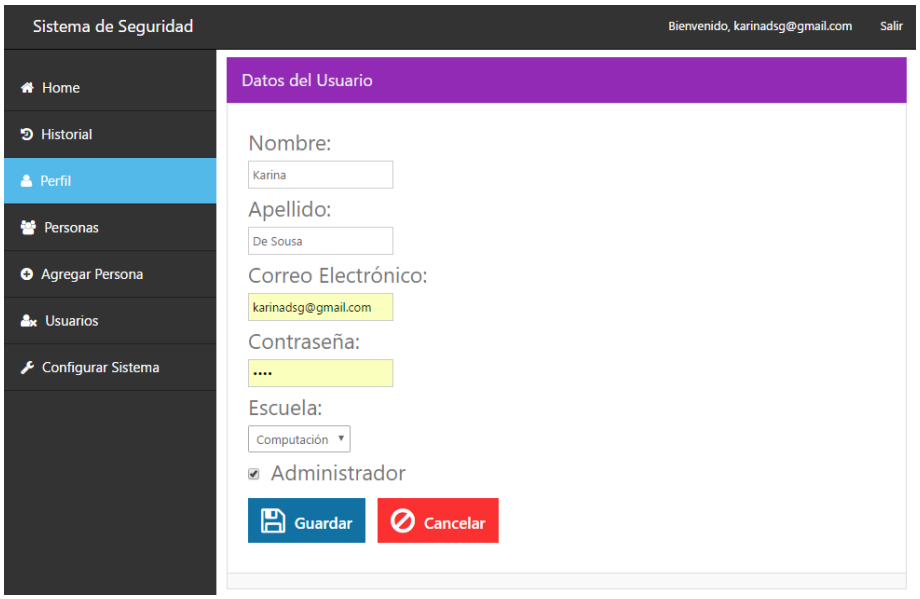

Figura 3.12: Edición del perfil de usuario

3.5.2. Administración de personas

El módulo de personas despliega el listado de personas registradas en el sistema, como lo muestra la Figura 3.13. Es posible ver, editar, agregar y eliminar personas. También se brinda la opción de búsqueda, ya sea por nombre o por correo electrónico.

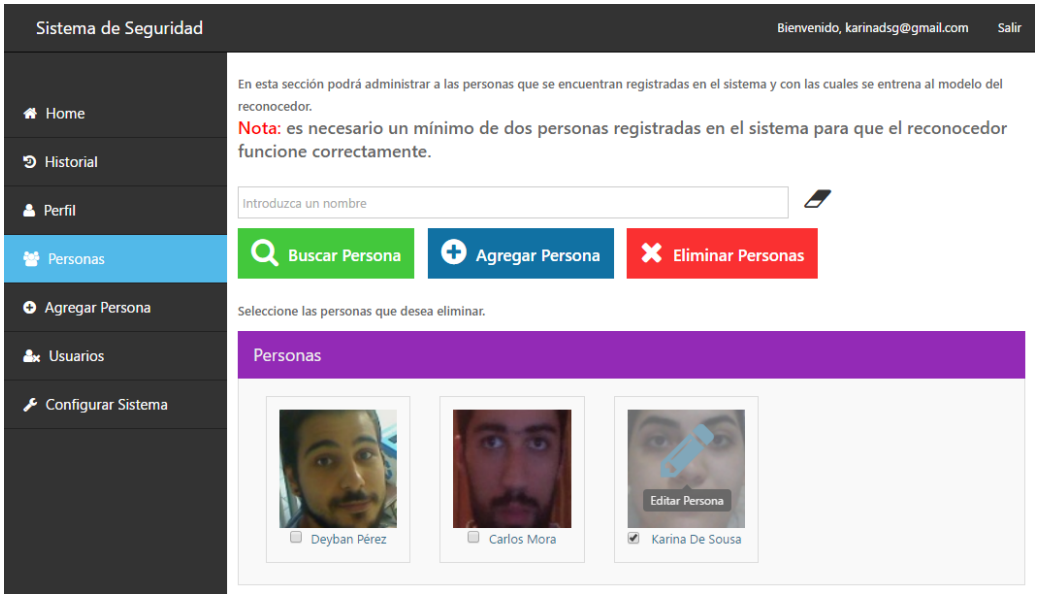

Figura 3.13: Módulo de personas

Para editar personas se muestra una interfaz similar a la de agregar personas. En la Figura

3.14 se observa que en esta vista se pueden modificar los datos de la persona y agregar imágenes para reentrenar al reconocedor.

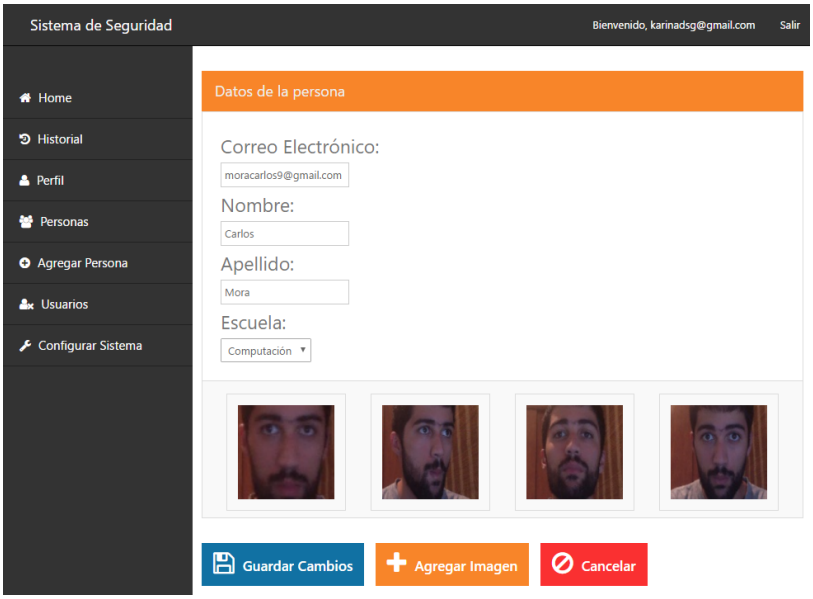

Figura 3.14: Edición de persona

Para agregar imágenes al sistema se implementó un módulo de cámara en donde se puede ver la transmisión en tiempo real, tomar fotos y cortarlas para seleccionar solo el rostro de la persona. Este módulo se muestra en la Figura 3.15 y es usado tanto para editar como para agregar personas.

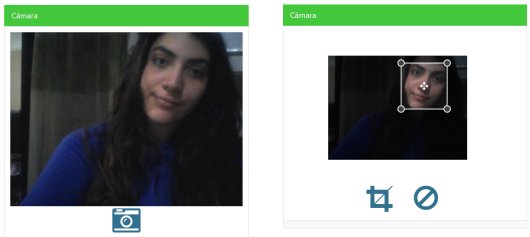

Figura 3.15: Transmisión de la cámara en tiempo real y corte final de la imagen

La última opción de administración de personas es el módulo de agregar personas, mostrado en la Figura 3.16, que como se mencionó anteriormente, presenta una vista similar a la de la Figura 3.14 y también hace uso del módulo de cámara.

Figura 3.16: Agregar persona

3.5.3. Configuración del sistema

Para realizar el reconocimiento facial y el envío de alarmas, existen ciertos parámetros que son modificables desde el módulo de configuración del sistema, mostrado en la Figura 3.17.

Este módulo se divide en configuraciones de envío y otras configuraciones del sistema. Ambas secciones se muestran en la Figura 3.17. La primera presenta un formulario, en el cual se puede editar el asunto y contenido del correo electrónico enviado cuando existe una alarma de intrusos en el sistema. Además se puede modificar la lista de destinatarios que recibirán dicho correo.

La segunda sección da la opción de editar el límite superior de confianza y el porcentaje de presencia que debe poseer el sujeto para enviar la alarma. Los detalles del significado de la confianza se explican en el Capítulo 4. Esta sección también permite activar y desactivar la alarma y configurar la hora de inicio y fin de la misma. Finalmente, permite indicar el tipo de cámara y en caso de ser una cámara IP se debe especificar la URL en la cual se almacenan las imágenes.

3.6. Detalles de implementación

En esta Sección, se da una descripción específica de cada uno de los módulos desarrollados, como fueron definidos en la Sección 3.1. El enfoque modular seguido se basó en la idea de proveer la mayor independencia posible para que estos módulos, pudiesen ser ejecutados bien sea en el mismo SBC como en distintas piezas de *hardware*, si fuese necesario. Cada

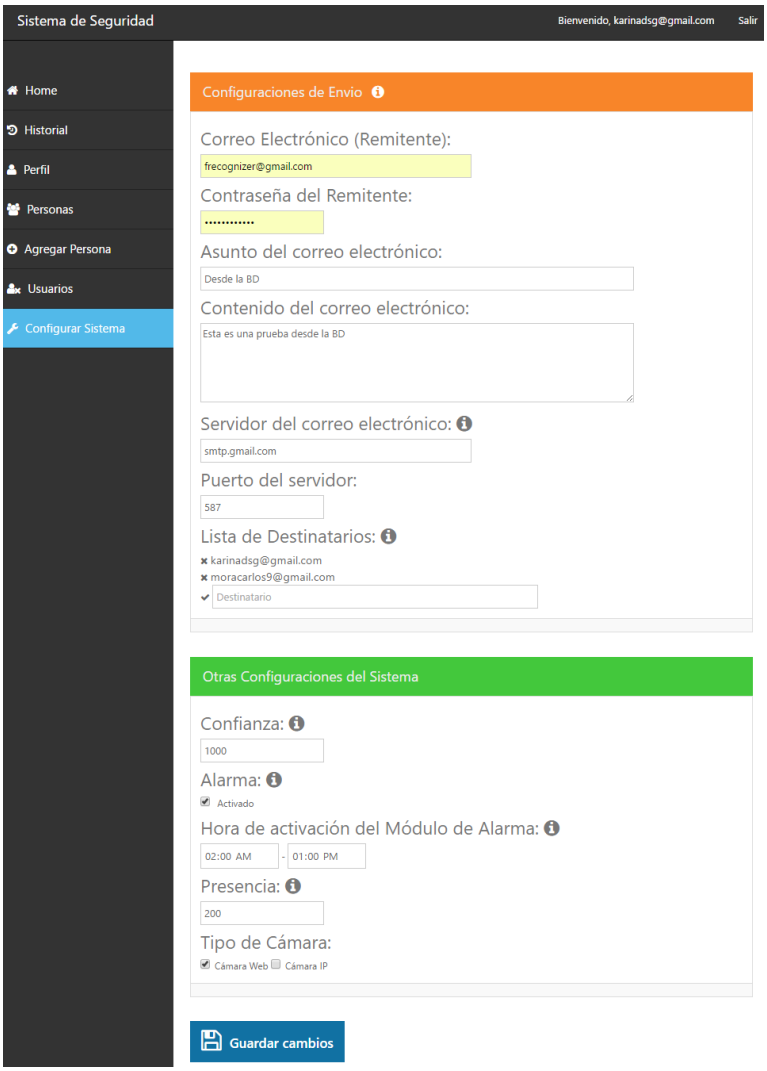

Figura 3.17: Módulo de configuración del sistema

uno de ellos utiliza un conjunto de herramientas de *software* de código abierto, y necesitan ser configurados propiamente para garantizar su interoperabilidad.

3.6.1. Módulo de reconocimiento facial

El primer módulo desarrollado fue el de reconocimiento facial, que lleva a cabo las tareas de detección, extracción de características y clasificación facial. Éste ejecuta un conjunto de rutinas encapsuladas en varias clases de OpenCV para poder llevar a cabo su función, así mismo, este módulo utliza Crossbar.io (definido en la Sección 2.5.4) para realizar todo el proceso de transmisión de imágenes a los clientes, siguiendo el patrón PubSub [26]), además este módulo implementa el submódulo de alarmas y de registro de acceso (*logs*).

Para el desarrollo de este módulo se utilizó el lenguaje de programación Python, con un

paradigma de programación orientada a objetos. La clase ***Main*** se encarga de instanciar al ***Logger***, ***Recognizer***, ***Camera*** y ***Constants***, calcular los cuadros por segundo y definir el bucle principal del programa.

La cámara se comunica con la clase cv2.*VideoCapture*, que se encarga de realizar la conexión con la cámara conectada al sistema y obtener cada uno de los cuadros. Debido a que *VideoCapture* no está en la capacidad de conectarse con el módulo de cámara de la Raspberry Pi, *Camera* se comunica con el módulo *picamera* de Python para llevar a cabo esta tarea. Finalmente está en la capacidad de distinguir si existe o no una conexión con el módulo de cámara y utilizarlo en caso de que exista.

La clase ***Recognizer*** lleva a cabo tres tareas principales: detección facial, clasificación y codificación. La detección se lleva a cabo utilizando un método basado en apariencia [7] llamado Clasificador de Haar, que se encuentra disponible pre-entrenado en OpenCV 2.4. No surgió la necesidad de utilizar una base de datos de rostros para llevar a cabo el entrenamiento en esta investigación dado que el clasificador en cuestión se ajusta a las necesidades de este trabajo. La clasificación es realizada utilizando el algoritmo de Fisherfaces [4] para agrupar los rostros en clases y predecir, con cierta confianza, la presencia de un individuo en la escena. Para que el reconocedor pueda ser entrenado, es necesario que existan al menos dos individuos registrados en la base de datos. Finalmente, la codificación se realiza con la finalidad de transmitir (a través la clase *Main*) los cuadros con las predicciones en un objeto JSON. Dicha codificación se hace en formato Base64 [42]. La estructura de este objeto se muestra en el Código 3.1.

```
{
  "name": String, //Email del usuario reconocido
  "log": String,  //Define si se debe mostrar o no en el Front-end
  "confidence": Float,  //Confianza de la predicción
  "x":Float, //Coordenada X
  "y":Float, //Coordenada Y
  "w": Float, //Tamaño en X
  "h": Float //Tamaño en Y
}
```

Listado 3.1: Estructura del objeto JSON de predicción del sistema

La clase ***Logger*** maneja todos los procesos referentes a los registros de acceso en el sistema, mientras que la clase ***Constants*** se conecta al módulo API para obtener todos los parámetros necesarios para enviar correos, imágenes de entrenamiento para el reconocedor y umbrales para confianza y alarma.

El módulo de reconocimiento facial basado en Crossbar.io es el encargado de transmitir los datos hacia los clientes. Para esto utiliza enlaces TCP, el protocolo HTTPS y WebSockets seguros. Por esto, se asigna una dirección IP exclusivamente para él. Todas las transmisiones se realizan de manera segura a través de SSL (*Secure Sockets Layer* - Capa de Puertos Seguros) con el protocolo HTTPS. Se utiliza además un mecanismo de autenticación llama-

do CRA (*Challenge-Response Authentication* - Autenticación por Desafío-Respuesta) [24], en donde se define un conjunto de usuarios con roles específicos que permiten llevar a cabo las tareas de publicación, registro, suscripción y llamada a procedimientos remotos en Crossbar.io.

Submódulo de alarma

La clase ***Notifier*** implementa la funcionalidad de envío de correos electrónicos que sirven como alarma cuando algún individuo no reconocido ingresa al sistema. Se crea un hilo de ejecución para llevar a cabo éste proceso de manera asíncrona. El correo enviado incluye una fotografía adjunta con el estado de la escena en el momento en el que el sistema detecta al intruso, como se muestra en la Figura 3.18. El desarrollo de las notificaciones se hizo según los criterios establecidos en las Ecuaciones 3.1 y 3.2, para que el sistema no enviase un correo electrónico por cada cuadro en el que un desconocido estuviese presente, para esto un método registra las apariciones de cada individuo, y al momento de que se presente un número alto de apariciones, se envía la alarma. El porcentaje de cada aparición es calculado a través de la Ecuación 3.1. Este porcentaje es re-calculado en cada iteración, y es decrementado según lo define la Ecuación 3.2.

Figura 3.18: Correo electrónico enviado como alarma

$$Porcentaje'(Porcentaje, FPS) = \begin{cases} MIN\left(Porcentaje+1/0{,}060(FPS), 100\right) & FPS \neq 0 \\ MIN\left(Porcentaje+1/0{,}060(1), 100\right) & CC \end{cases} \tag{3.1}$$

$$Porcentaje'(Porcentaje, FPS) = \begin{cases} MAX\left(0, Porcentaje - \left({}^{1}/_{0{,}2(FPS)}\right)\right) & FPS \neq 0 \\ MAX\left(0, Porcentaje - \left({}^{1}/_{0{,}2(1)}\right)\right) & CC \end{cases} \tag{3.2}$$

La finalidad del decremento es disminuir el nivel de presencia de cada individuo por cada cuadro, de modo tal que si este es captado por la cámara y se aleja por cierto tiempo, el sistema pueda ser capaz de reenviar la notificación.

La elección de estas funciones se hizo de manera empírica. De esta manera, independientemente de sobre qué *hardware* se ejecute el reconocedor, el tiempo para enviar la alerta será aproximadamente el mismo.

3.6.2. Módulo API

Con la finalidad de exponer un servicio que encapsule las funciones de conexión con la base de datos para que estas no se hagan de forma directa, se desarrolló el módulo denominado API. La implementación está hecha sobre Node.js utilizando el *framework* Express.js.

El módulo define tres modelos de datos: *users*, *people* y *configurations*. El modelo *users* almacena información acerca de los usuarios que cuentan con credenciales para acceder al sistema y administrarlo a través del módulo *front-end*, mientras que *people* mantiene la información de los individuos registrados en el sistema para ser reconocidos por el módulo de reconocimiento facial. Finalmente *configurations* almacena la información de configuración básica del sistema, más específicamente los parámetros de configuración del módulo de reconocimiento facial y de alarmas.

Para cada uno de los modelos se define la funcionalidad haciendo uso los métodos HTTP utilizados en servicios REST, de modo de implementar el CRUD (*Create, Read, Update, Delete* - Crear, Leer, Actualizar y Borrar) en su totalidad, para poder manipular la información.

La API se ejecuta sobre el protocolo HTTPS, con certificados auto-firmados y utiliza un mecanismo de autorización básico, basado en token, generado a partir del correo electrónico y contraseña del usuario, codificado en formato Base64. Dicho mecanismo de autenticación utiliza la cabecera HTTP Authorization, que se construye de la siguiente forma:

***Authorization**: Basic <correo electrónico>:<contraseña>*

Con *<correo electrónico>:<contraseña>* codificados en Base64. Así por ejemplo, con correo electrónico “email@email.com” y contraseña “1234”, se obtendría:

***Authorization**: Basic ZW1haWxAZW1haWwuY29tOjEyMzQ=*

3.6.3. Módulo de *front-end*

Una vez desarrollados los módulos de reconocimiento facial y del API, era necesario una interfaz a través de la cual los usuarios pudiesen acceder al sistema y manejarlo de forma

más sencilla. Es por esto que se desarrolló un módulo de *front-end*. La implementación está hecha con los lenguajes HTML5 y CSS3, utilizando el *framework* AngularJS, que permite el desarrollo de aplicaciones web de una forma rápida y sencilla.

Se siguió la estructura recomendada por la documentación oficial de AngularJS, en donde cada módulo se separa por rutas. A su vez cada ruta posee su propio controlador y vista asociada.

La conexión con el API se realizó usando la directiva *$http*, que brinda los métodos básicos para hacer solicitudes (*request*) a un API *RESTFul*. Todas las llamadas al API fueron centralizadas en un *factory*. De esta forma se pueden reutilizar a través de los distintos módulos de aplicación.

Como se mencionó en la Sección 3.5, se implementó un módulo de cámara para tomar y recortar las fotos de las personas registradas en el sistema. Para esto se creó un *modal* que transmite en tiempo real y posee un controlador asociado en donde se procesan las imágenes y se retornan al módulo que invocó a la cámara.

Capítulo 4

Pruebas y Resultados

El proceso de pruebas comenzó desde el momento en que el desarrollo fue tomando forma, es decir, no se trazó una línea clara entre la implementación y las pruebas, que dividiese a ambas fases de manera excluyente. Por esto se comenzaron a obtener resultados, de menor grado, pero de igual importancia que sirvieron como guía en el proceso, razón por la cual se toman en cuenta y se explican en la presente Sección.

Una vez culminada la implementación del sistema de reconocimiento facial, se llevaron a cabo un conjunto de pruebas que sirvieron como base para medir el rendimiento general del mismo en diferentes ámbitos. La idea básica fue determinar qué tan bien se comportaba la solución implementada, no solamente en términos de velocidad, sino también en confiabilidad y usabilidad. Éstas pruebas fueron el pilar fundamental para detectar errores cometidos en la fase de desarrollo y captar nuevas necesidades que no se pudieron determinar a priori.

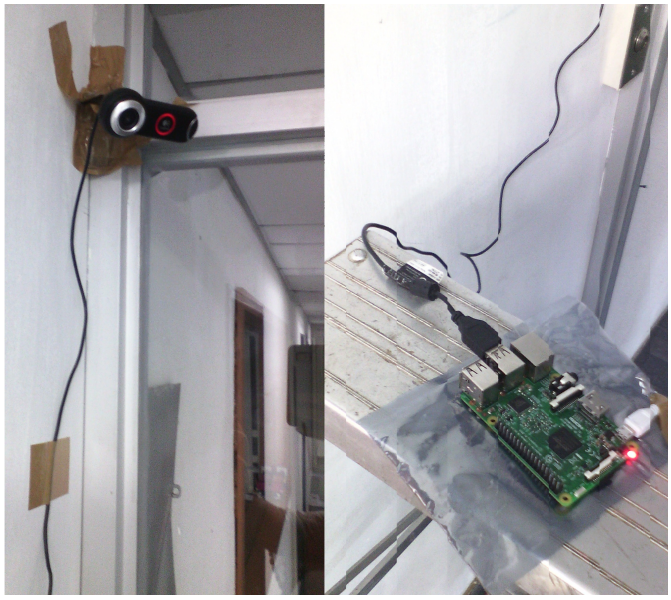

Figura 4.1: Instalación del sistema de seguridad para la realización de las pruebas

Estas pruebas se realizaron en el laboratorio ICARO de la escuela de computación de la Facultad de Ciencias de la UCV. La Figura 4.1 muestra la instalación realizada para las pruebas, que consistía en una cámara Logitech *QuickCam* Pro 9000, que se encontraba conectada a una Raspberry Pi, en donde se ejecutaba la aplicación. Para la fase de pruebas se utilizaron cámaras web, el módulo de cámara de la Raspberry y cámaras IP.

Dichas pruebas se dividieron en tres categorías: pruebas empíricas, pruebas estadísticas y pruebas de usabilidad.

4.1. Definición de escenario de pruebas

Se extienden desde el inicio del proceso de desarrollo del sistema de seguridad, hasta la puesta en marcha del mismo en el laboratorio seleccionado. Se definieron 5 escenarios para las pruebas, descritos a continuación.

4.1.1. Transmisión de los datos

Luego de determinar bajo qué protocolo se realizaría la transmisión de los datos, se produjo la interrogante de ¿es posible transmitir los datos desde la Raspberry Pi a través de la red de modo de mantener los FPS entregados por la cámara?, ya que se debía determinar el límite de datos podría enviar y así acotar la transmisión con respecto a algún valor. Para determinar esto, se tomó en cuenta un caso aproximado, en condiciones normales. La Raspberry Pi, con una cámara web USB conectada o en su defecto, el módulo de Cámara, entrega un promedio de 20 FPS, a una resolución de 1280px $\times$ 960px sin llevar a cabo ningún tipo de procesamiento sobre los cuadros, por lo que dicho número representa la cota superior para la tasa de lectura del sensor. Basándose en el número máximo de cuadros entregados, se llevaron a cabo transmisiones desde la Raspberry hacia la red, utilizando el protocolo WAMP y no se observó diferencia alguna entre la visualización en el *loopback* (interfaz *lo*) y en los clientes conectados a la red. Cabe destacar que esta prueba fue realizada utilizando una interfaz LAN Gigabit Ethernet 802.3ab en la Raspberry Pi, los clientes y el concentrador de red. Además, se obtuvieron los mismos resultados utilizando una red WLAN 802.11b/g con esta especificación en cada uno de los nodos. Se puede concluír que el sistema puede llevar a cabo la transmisión de 20 cuadros por segundo a una resolución de 1280px $\times$ 960px a través de una interfaz LAN, o WLAN con un nivel de uso del medio compartido relativamente bajo para este caso. Como dicha resolución representa una cota superior para la cantidad de datos por segundo a transmitir y al momento de ejecutar los algoritmos de reconocimiento facial la tasa de cuadros disminuye, los datos pueden ser transmitidos a través de la red LAN o WLAN.

4.1.2. Tamaño de las imágenes de entrada y distancia entre el sujeto y la cámara

Una vez que culminó el desarrollo del módulo de reconocimiento facial se observó un decremento importante en la tasa de cuadros por segundo entregados por la Raspberry Pi, el

cual era directamente proporcional al tamaño de las imágenes de entrada, por lo que se procedió a redimensionarlas de forma tal que este no se convirtiera en crítico para la aplicación final, y se pudiese apreciar la escena con fluidez. La disminución de dicho tamaño afecta de forma proporcional a la distancia máxima a la cual puede estar el sujeto con respecto a la cámara. Como lo indica la Tabla 4.1, en total se probaron tres resoluciones (320px $\times$ 240px, 640px $\times$ 480px y 1280px $\times$ 960px), con lo cual se obervó que la resolución de la imagen es inversamente proporcional a la cantidad de FPS entregados luego del procesamiento, al igual que la distancia máxima a la cual el sujeto puede ser reconocido. Con un tamaño de imagen de 320px $\times$ 240px el sistema detectará y reconocerá rostros a aproximadamente 2 metros de distancia, como máximo; con un promedio de 3.314847 FPS. Para las otras dos resoluciones los FPS fueron menores por ser inversamente proporcionales, 1.545634 FPS y 0.695849 FPS para 640px $\times$ 480px y 1280px $\times$ 960px respectivamente.

Tabla 4.1: Tasas de FPS y distancia máxima entre el sujeto y la cámara, en función de la resolución de la imagen de entrada

	FPS con reconocimiento	**Distancia máxima aproximada**
1280x960	0.695849	7.5 metros
640x480	1.545634	4 metros
320x240	3.314847	2 metros

Otra de las consecuencias que se desprenden del redimensionamiento de la imagen de entrada es la distancia máxima a la cual se pueden ubicar los individuos en la escena para que el proceso de detección facial se lleve a cabo, debido a que a menor tamaño de imagen, más cerca deben encontrarse los sujetos. Entonces, empíricamente se buscó un tamaño de imagen que maximizara la tasa de FPS y la distancia máxima a la cual se puede situar el sujeto.

4.1.3. Posición de la cámara y condiciones de iluminación

Desglosando las condiciones de iluminación, el sistema se comportará bien con solo mantener un balance de blancos constante, sin sobreexposiciones ni subexposiciones muy grandes. La cantidad de luz debe ser suficiente como para distinguir todos los rasgos faciales a una distancia máxima dependiente del tamaño de la imagen, para nuestro caso 320px $\times$ 240px. Técnicamente, la cámara puede encontrarse a cualquier altura siempre y cuando no existan puntos de luz muy brillantes en la escena (como por ejemplo, un bombillo de luz blanca) y exista una visualización completa de los rostros.

Uno de los casos borde referentes a las condiciones de iluminación se presentó en el momento en que se intentó hacer detección facial con una variación considerablemente grande en las condiciones de iluminación de la escena, más específicamente, con la cámara apuntando hacia diferentes luces blancas. Esto se debe a que al existir puntos con brillos muy altos, el sensor regula la cantidad de luz total que percibe y genera subexposición en las zonas más oscuras, por lo que la detección facial no se lleva a cabo con éxito. Se obtuvo el mismo

resultado con todas las cámaras probadas. Esto, dependerá también de la calidad del sensor que posea la cámara que se use en un momento dado y de qué tan bien se haga la regulación automática de blancos en la misma. Una solución para este problema, es utilizar fuentes de luz adicionales que sirvan para mantener al sujeto mejor iluminado y así intentar equilibrar la cantidad de luz que generan los puntos de iluminación provenientes de los bombillos con el resto de la escena, de modo de no generar subexposición. Las Figuras 4.2 y 4.3 muestran la problematica existente cuando la cámara apunta en dirección a una fuente de luz y cuando no hay sobreexposición, respectivamente.

Figura 4.2: Detección facial fallida debido a las condiciones de iluminación

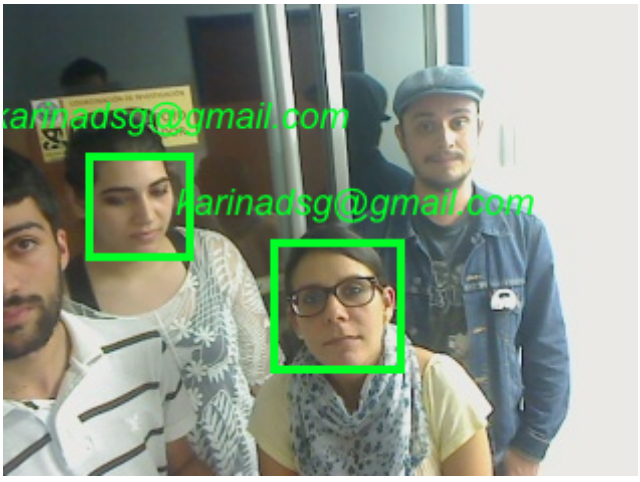

Figura 4.3: Detección y reconocimiento facial exitosos debido a las condiciones de iluminación

4.1.4. Posición e inclinación de rostros

La inclinación es otro factor importante que genera condiciones para el correcto funcionamiento del sistema, más específicamente para llevar a cabo de forma satisfactoria el proceso de detección facial. Los rostros deben apuntar a la cámara y mostrar tres características principales: boca, dos cejas y naríz. Los mismos pueden rotar sobre su propio eje (*yaw*) hasta que todas las características puedan observarse. Por lo tanto, no es posible hacer detección facial con un rostro de lado. También, el sistema requiere que los rostros no presenten una rotación *roll* superior a los 10 grados aproximadamente, tomando como punto inicial una alineación con el eje Y. Estas restricciones se definen con el fin de mejorar el rendimiento del proceso de detección facial.

4.1.5. Entrenamiento dependiente de las condiciones de iluminación

También se obtuvo un resultado empírico al momento de llevar a cabo el reconocimiento facial sobre condiciones de iluminación diferentes a las existentes en la base de datos de rostros. Al cambiar de luz, el sistema no era capaz de reconocer de forma acertada a los sujetos con los cuales había sido entrenado. Seguido de esto se hizo un reentrenamiento en el nuevo entorno y se observó que el reconocimiento se realizaba de forma correcta. Esto lleva a la conclusión de que es necesario llevar a cabo los entrenamientos cada vez que las condiciones de iluminación del entorno cambien.

4.2. Pruebas estadísticas

Con el fin de obtener indicadores que dieran información certera referente a que tan bien funciona el sistema de reconocimiento facial, se llevaron a cabo pruebas que acumularon datos para posteriormente ser analizados. Se diseñaron 4 pruebas distintas, con la intención de obtener información de falsos y verdaderos positivos [1], confianza promedio [2] y número de cuadros por segundo (FPS); para así tener una visión aproximada del sistema en diferentes condiciones.

4.2.1. Condiciones ideales

La idea de esta prueba fue obtener información referente a la confiabilidad del sistema en buenas condiciones de iluminación, posición de la cámara, oclusión facial y entrenamiento adecuado. La cámara fue posicionada de forma tal que no apuntase directamente a una fuente de luz artificial, con los objetivos bien iluminados, a una distancia de la cámara comprendida

[1] En estadísticas se considera un verdadero positivo al descubrimiento acertado de algún elemento. Por ejemplo, cuando se reconoce correctamente un rostro. Mientras que identificar de forma errada un rostro se consideraría un falso positivo.

[2] La confianza de una observación es la distancia entre las clases entrenadas y la clase candidata (rostro) y hace referencia a la veracidad del resultado obtenido. Para el contexto de la aplicación mientras menor sea este valor se considera que es mas acertado.

entre 0.3 y 1.5 metros, sin oclusión facial y con un entrenamiento previo bajo las mismas condiciones de iluminación.

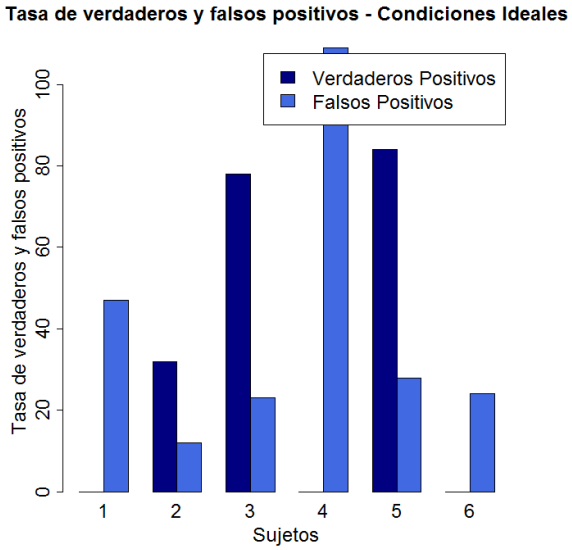

Figura 4.4: Resultados de la tasa de verdaderos y falsos positivos por sujeto, en las pruebas realizadas en condiciones ideales

El muestreo se realizó en un total de 30 segundos, donde el sujeto podía moverse libremente a través de la escena mientras estuviese dentro de los límites de distancia antes mencionados, de manera de simular la espera tras la puerta que estaba siendo vigilada por el sistema de seguridad. En este tiempo, se obtuvieron un total de 437 observaciones (una observación es el registro de un rostro en el *log* [3]). A partir de esto se obtuvo una tasa de verdaderos positivos, de 0.4439359. Esto fue calculado sin tomar en cuenta la confianza de las observaciones. Además, se calculó el promedio de confianza para las observaciones acertadas (verdadero positivo) y las erradas (falso postivo), obteniendo las cifras 264.7759 y 382.9676, respectivamente. La Tabla 4.2 muestra los resultados de las pruebas realizadas por cada sujeto. En la misma se considera un verdadero positivo aquella observación donde la predicción es igual a la identidad real del sujeto, y no se toma en cuenta la confianza obtenida. Es por esto que solo hay aciertos para aquellas personas que se encontraban registradas en el sistema.

Con respecto a la cantidad de verdaderos y falsos positivos obtenidos se puede observar en la Figura 4.4 que hay una gran cantidad de falsos positivos y que no existen verdaderos positivos para algunos sujetos, ya que estos no se encuentran registrados en el sistema.

Es posible ver en la Tabla 4.2, que la confianza promedio de las observaciones consideradas como aciertos, de sujetos registrados en el sistema, se encuentra por debajo de 300. Este campo no aplica para personas que no se encuentren registradas, ya que estas nunca producirán un verdadero positivo. Además, se puede observar que para todos los sujetos la confianza promedio de fallos siempre esta por encima de 300.

[3] Un *log* se refiere al registro detallado y secuencial de la actividad del sistema.

Tabla 4.2: Resumen de las pruebas bajo condiciones ideales

	Verdaderos positivos	**Falsos positivos**	**¿Registrado?**	**Confianza promedio de aciertos**	**Confianza promedio de fallos**
Sujeto 1	0	47	No	N/A	377.2362
Sujeto 2	32	12	Sí	207.7038	398.9698
Sujeto 3	78	23	Sí	273.3610	355.7645
Sujeto 4	0	109	No	N/A	369.0793
Sujeto 5	84	28	Sí	278.5456	433.3546
Sujeto 6	0	24	No	N/A	416.5512

La Figura 4.5 presenta la cantidad de observaciones que son consideradas como aciertos según varíe la confianza. Así se ve que 300 es el punto a partir del cual se comienza a aceptar una cantidad razonable de observaciones.

Analizando estos resultados se llega a la conclusión de que bajo condiciones ideales un umbral de confianza para aceptar una observación podría ser $[0, 300]$. Entonces, todas las entradas del *log* que posean una confianza por encima de 300 se pueden considerar como falsos positivos. Usando dicho intervalo se obtiene un error [4] de 0.13 y aumentando el intervalo a $[0, 350]$ el error es de 0.09. La Figura 4.6 muestra como aumentan los verdaderos positivos y como disminuyen los falsos positivos en comparación a la Figura 4.4. Para este caso se considera un verdadero positivo a aquellas observaciones donde la predicción es igual al sujeto real y la confianza se encuentra dentro del rango mencionado anteriormente.

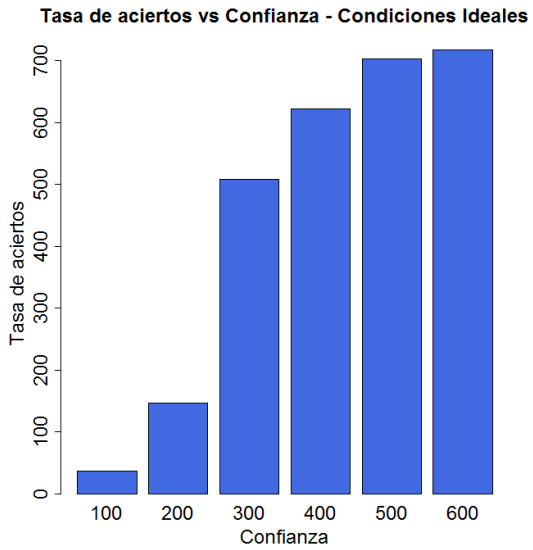

Figura 4.5: Cambio de la tasa de aciertos de acuerdo a la confianza, en las pruebas realizadas en condiciones ideales

[4]El error vine dado por la división de la sumatoria de los falsos positivos entre la suma de la población

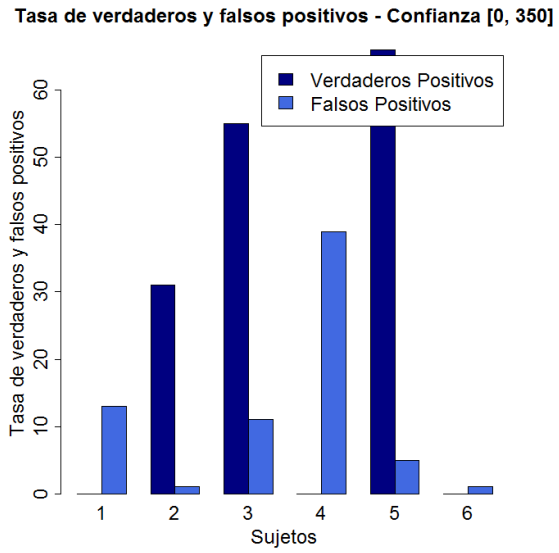

Figura 4.6: Resultados de la tasa de verdaderos y falsos positivos por sujeto, en las pruebas realizadas en condiciones ideales, tomando como verdaderos positivos las observaciones con confianza menor o igual a 350

4.2.2. Distancia corta

Con la finalidad de medir las diferencias de aciertos cuando la distancia varía, se realizó una prueba en la que los rostros se encontraran a 50 centímetros de la cámara aproximadamente, de modo que todos los rasgos faciales se obtuvieran con buena definición.

Tabla 4.3: Resumen de las pruebas con distancia entre la cámara y los rostros de 50 cm

	Verdaderos positivos	**Falsos positivos**	**Confianza promedio de aciertos**	**Confianza promedio de fallos**
Sujeto 1	111	96	300.0357	357.4301
Sujeto 2	74	46	353.3571	351.2490
Sujeto 3	129	0	317.6251	N/A

Al igual que en las pruebas anteriores, la duración de ésta fue de 30 segundos, en los cuales se registraron 456 observaciones. Se comenzó el análisis considerando como verdaderos positivos aquellas entradas del *log* donde el sujeto y la predicción eran iguales, con lo cual se obtuvo una precisión [5] de 0.6885965, confianza promedio para aciertos de 319.8282 y confianza promedio para fallos de 355.4278. Esto significa que el sistema se comporta bastante bien cuando el sujeto se encuentra más cerca de la cámara. Sin embargo, la confianza promedio requerida para aciertos es similar al caso anterior en condiciones ideales.

[5]La precisión viene dada por la división de la suma de los verdaderos positivos y los verdaderos negativos entre la suma de la población

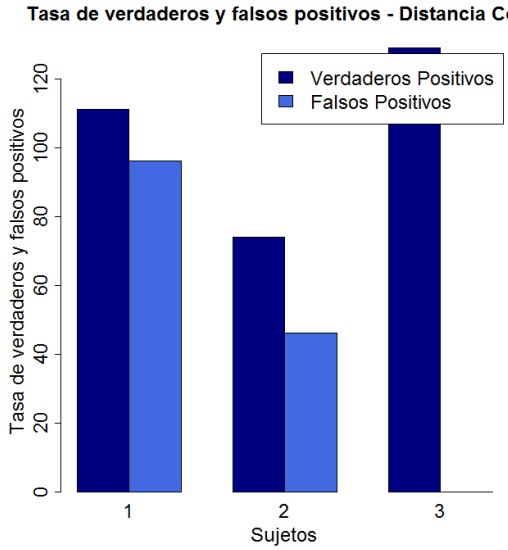

Figura 4.7: Resultados de la tasa de verdaderos y falsos positivos por sujeto, en las pruebas realizadas a distancia corta

La Tabla 4.3 muestra los resultados de las pruebas realizadas. Para esta prueba se usaron 3 sujetos que se encontraban registrados en el sistema. Se puede obervar que para el tercer sujeto el reconocimiento se realizó perfectamente, por lo que no aplica la columna de confianza promedio de fallos. En cambio, para los dos primeros sujetos la cantidad de verdaderos y falsos positivos es similar, lo cual se aprecia mejor en la Figura 4.7.

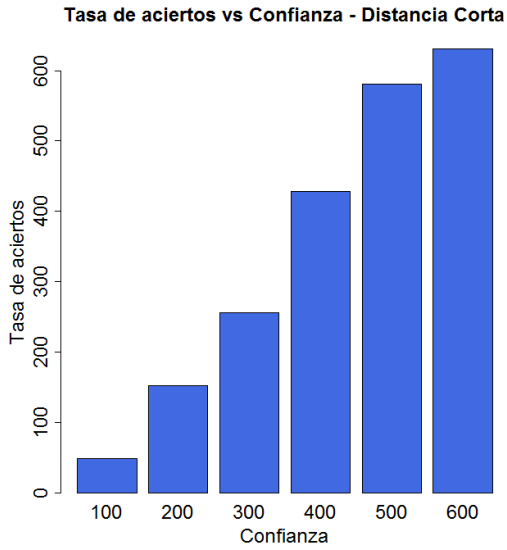

Figura 4.8: Cambio de la tasa de aciertos de acuerdo a la confianza, en las pruebas realizadas a distancia corta

Al observar la Figura 4.8 se ve que el umbral de aceptación de la prueba anterior puede ser una buena elección. Tomando como verdaderos positivos las observaciones en donde la confianza era menor o igual a 350 y el sujeto real era igual a la predicción del sistema, se obtiene un error de 0.3, esto se ve representado en la Figura 4.9, donde a pesar de tener un error mayor al de las pruebas anteriores, al compararla con la Figura 4.7 la cantidad de falsos positivos disminuye considerablemente.

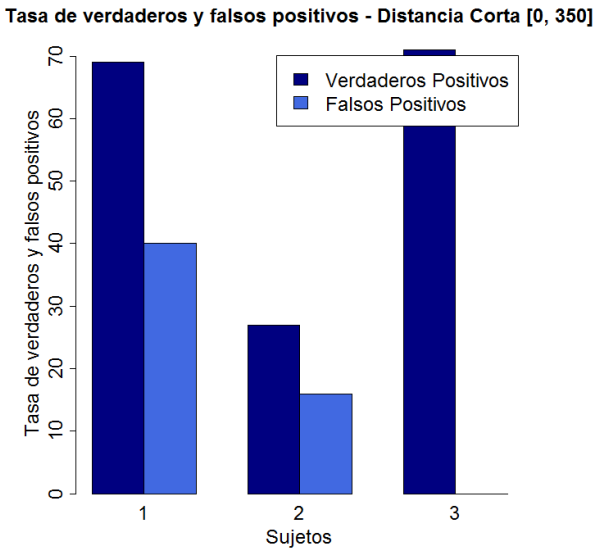

Figura 4.9: Resultados de la tasa de verdaderos y falsos positivos por sujeto, en las pruebas realizada a distancia corta, tomando como verdaderos positivos las observaciones con confianza menor o igual a 350

4.2.3. Oclusión facial

Para estas pruebas se variaron los elementos de oclusión facial presentes en los rostros de los sujetos para analizar cómo se comporta el sistema si hay objetos que cambian la apariencia de los mismos. Se utilizaron gorras y lentes de sol para recrear las oclusiones más comunes de los rostros.

Tabla 4.4: Resumen de las pruebas con oclusión facial

	Verdaderos positivos	**Falsos positivos**	**¿Registrado?**	**Confianza promedio de aciertos**	**Confianza promedio de fallos**
Sujeto 1	0	50	No	N/A	230.0541
Sujeto 2	11	9	Sí	293.7792	366.9340
Sujeto 3	47	5	Sí	386.7485	378.4288
Sujeto 4	0	52	No	N/A	438.2221

En los 30 segundos se obtuvieron 174 muestras, un número mucho menor al de las pruebas anteriores, lo que supone que la etapa de detección facial se hace con menor precisión y en muchos casos no es posible que el sistema detecte la presencia de un rostro. Además, la precisión fue de 0.33333 y las confianzas promedio para aciertos y fallos fueron 369.1164 y 340.3863 respectivamente. Para este caso particular, el valor de la confianza para los aciertos es mayor a la de los fallos. A partir de esta información es posible deducir que cuando existe oclusión facial, el sistema se comporta bastante mal, lo que teóricamente es correcto ya que esto representa uno de los mayores retos en sistemas de este tipo.

La Tabla 4.4 muestra los resultados de las pruebas realizadas, en las cuales se usaron 4 usuarios, dos de ellos registrados en el sistema y dos que no lo estaban. Al igual que en las pruebas en condiciones ideales, los campos de confianza promedio de aciertos no aplican para aquellos usuarios que no se encuentran registrados en la base de datos por no poseer verdaderos positivos.

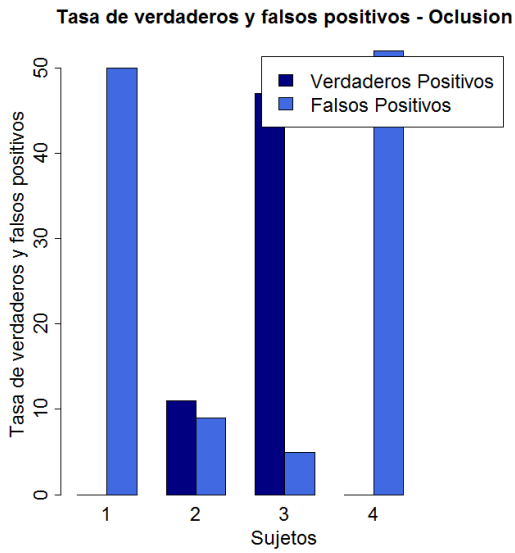

Figura 4.10: Resultados de la tasa de verdaderos y falsos positivos por sujeto, en las pruebas realizadas con oclusión facial

La notable diferencia entre la cantidad de observaciones del usuario 2 y 3 se debe a que el rostro del primero no pudo ser detectado en todos los cuadros por el sistema, a la hora de detectar su rostro. En cambio, con el usuario 3 existe una mayor cantidad de observaciones, donde la mayoría fueron verdaderos positivos. Lo interesante de este usuario es la confianza promedio para aciertos, porque la misma se encuentra por encima de la confianza promedio de fallos.

La Figura 4.10 muestra los verdaderos y falsos positivos de las pruebas de oclusión por usuario. Aquí se observa que los usuarios 1 y 4 son los que no se encuentran registrados en el sistema. Como se dijo anteriormente el usuario 2 posee pocas observaciones en el *log*, mientras que el usuario 3 presenta una gran cantidad de aciertos.

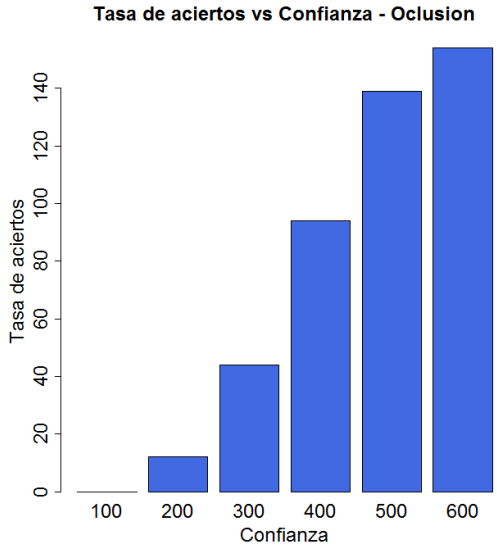

Figura 4.11: Cambio de la tasa de aciertos de acuerdo a la confianza, en las pruebas realizadas con oclusión facial

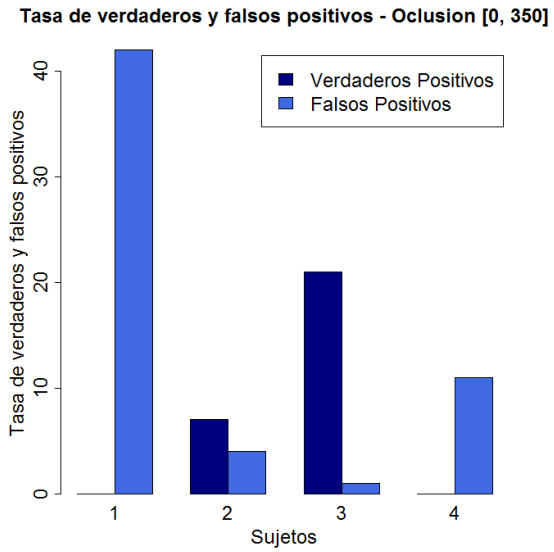

Figura 4.12: Resultados de la tasa de verdaderos y falsos positivos por sujeto, en las pruebas realizadas con oclusión facial, tomando como verdaderos positivos las observaciones con confianza menor o igual a 350

Al igual que en las pruebas anteriores, se toma la Figura 4.11 como guía para determinar el umbral adecuado de aceptación. En este caso se observa que de nuevo 350 puede ser un buen límite superior de este intervalo y con el mismo se obtiene un error de 0.17, lo cual es

bastante bajo considerando que este tipo de pruebas son las más díficiles para los sistemas de seguridad de este tipo, por no tener una visión completa de los rostros a reconocer.

La Figura 4.12 muestra como quedan los verdaderos y falsos positivos al usar el umbral mencionado anteriormente. Claramente disminuyen los falsos positivos tanto del usuario 1, como del 2; pero al mismo tiempo también disminuyen los verdaderos positivos.

4.2.4. Número de rostros en la escena

Con la finalidad de determinar los efectos que se producen sobre el rendimiento del sistema, causados por la existencia de más de un rostro presente en la escena, se llevó a cabo un muestreo de 30 segundos con 1 y 4 rostros presentes por separado, llevando el registro de aparición en cada individuo y la tasa de FPS de ese momento, para luego ser comparados. Los resultados se pueden apreciar en la Figura 4.13. Si se analiza el comportamiento de la Figura, se visualiza constancia para 1 rostro, y un leve decrecimiento en los FPS cuando existen 4. Los círculos representan la tasa de cuadros por segundo en función de las observaciones, ya que las mismas se tomaron de manera progresiva.

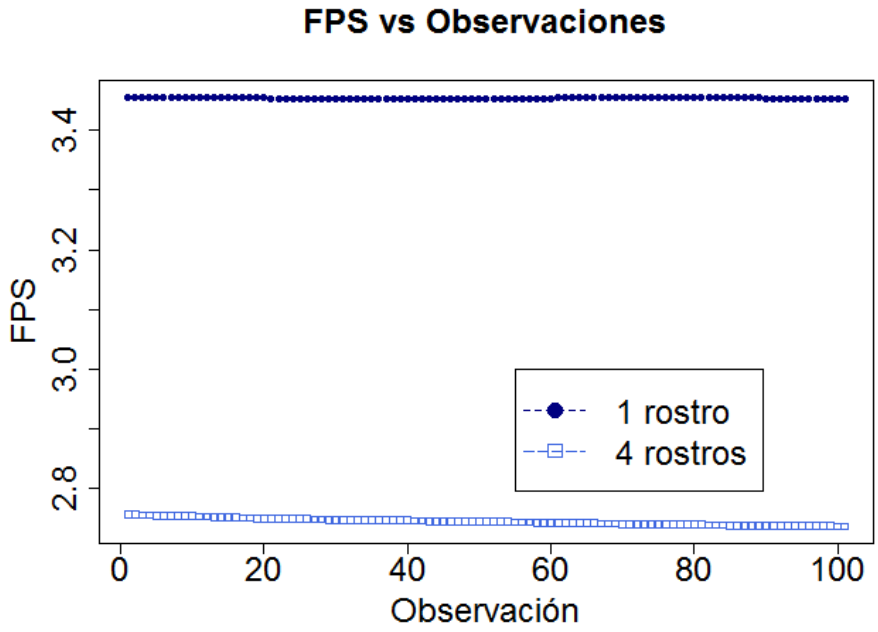

Figura 4.13: Comparación de tasa de cuadros por segundo para un rostro y cuatro rostros presentes en la escena.

Para esta prueba, el sistema estuvo ejecutándose por unos segundos antes de comenzar a tomar las observaciones, esto para descartar posibles cambios en los FPS derivados de la planificación del sistema operativo o memoria. El comportamiento es predecible, debido a que el sistema mantiene una tasa casi constante, pero con una diferencia entre los dos casos.

Es posible notar que el rendimiento se ve afectado para el caso de cuatro personas en la escena, con una pérdida de 0.7 FPS aproximadamente, lo que no supone una diferencia de mayor impacto en un ambiente de producción. Además, como premisa del estudio, se hizo una suposición de que el uso promedio de las instalaciones del laboratorio no implicará tener una cantidad muy grande de personas al mismo tiempo en la escena esperando en la puerta. Finalmente es posible concluír que el número de rostros siendo reconocidos en un momento en particular afecta la tasa de FPS entregada por el reconocedor.

4.3. Pruebas con cámara IP

La última prueba de rendimiento realizada se realizó usando una cámara IP DLink DCS-5300, bajo condiciones ideales. En la misma se tomaron observaciones por 30 segundos y se tomó en cuenta la tasa de FPS. Con esta cámara no se realizaron las pruebas anteriores (condiciones ideales, distancia corta, oclusión facial y número de rostros en la escena) porque éstas son independientes a la cámara que se use, siempre y cuando se use una resolución similar. En cambio, el rendimiento del sistema si es dependiente del tipo de cámara. Los resultados de esto se ven en la Figura 4.14, donde se tomaron 100 observaciones y se ve como la tasa no es constante pero siempre se encuentra por debajo de 1.5 FPS. Para esta prueba el promedio de FPS fue de 1.32325, mientras que en la prueba presentada en la Sección 4.2.4 el promedio fue de 3.169988. La Figura 4.15 muestra la comparación entre los resultados de un solo rostro usados en la Figura 4.13 y los obtenidos en esta prueba. Aquí se ve como el rendimiento disminuye considerablemente cuando se usa una cámara IP.

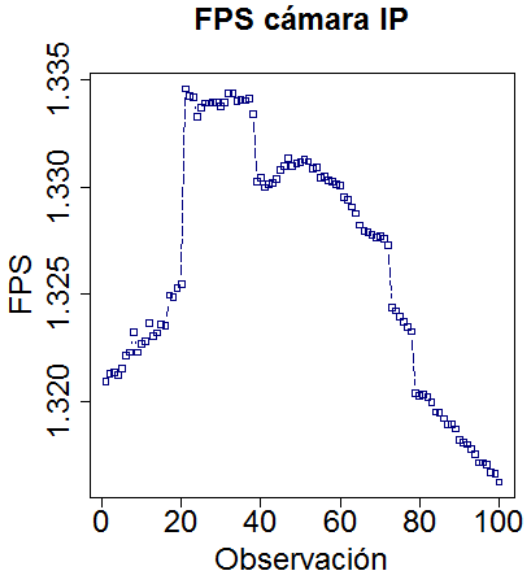

Figura 4.14: Comparación de tasa de cuadros por segundo para un rostro usando una cámara IP.

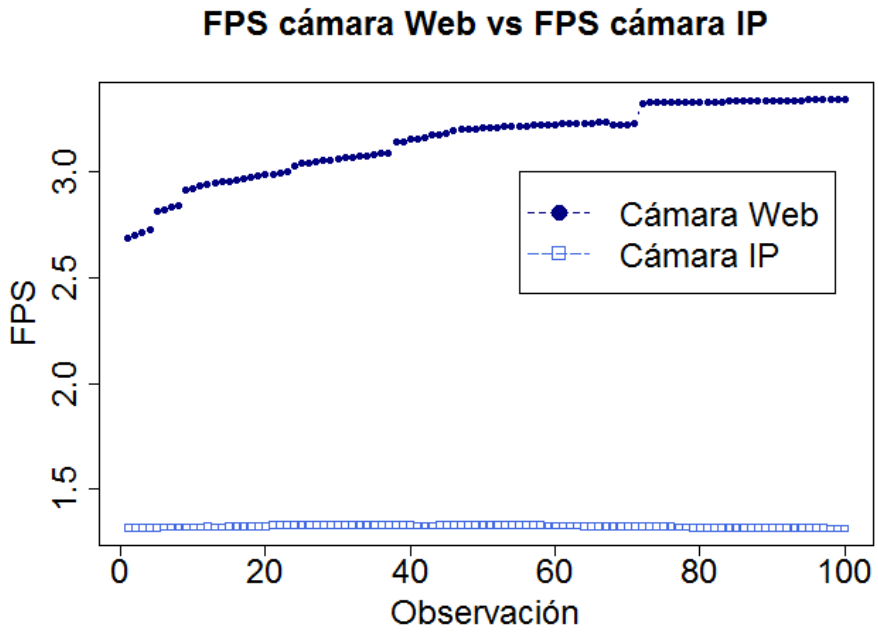

Figura 4.15: Comparación de tasa de cuadros por segundo para un rostro usando una cámara web y una cámara IP.

4.4. Pruebas de usabilidad

Para esta prueba se llevó a cabo una evaluación heurística definida por Nielsen [43], con la finalidad de encontrar problemas en la interfaz de usuario, para poder ser atendidos y corregidos en el proceso de desarrollo de la investigación. La muestra tomada fue de 5 individuos, debido a que con este número es posible para obtener un resultado válido [43]. Los encuestados respondieron a un total de 13 preguntas de selección simple, con 5 opciones para cada una. Dichas opciones permitieron que se respondiera de manera concisa, para evitar ambigüedades y estuvieron basadas en la escala de Likert [44]. Las opciones de respuesta para todos los casos fueron:

1. Totalmente de acuerdo.
2. De acuerdo.
3. Ni de acuerdo ni en desacuerdo.
4. En desacuerdo.
5. Totalmente en desacuerdo.

Las pruebas se dividieron en dos etapas. En la primera se recolectaron las respuestas y a partir de esto se mejoró la aplicación, para luego pasar a la segunda etapa en donde se verificó que los resultados mejoraran con respecto a los obtenidos previamente. Para ambas etapas se requirió que cada uno de los encuestados hicieran un recorrido por el sistema y utilizaran las funcionalidades en su totalidad, para fundar la opinión que posteriormente plasmarían en su respuesta. A continuación se discuten los resultados de cada uno de los ítems del cuestionario tanto en la primera como en la segunda encuesta.

1. **El sistema posee visibilidad del estado en el que se encuentra**

   Como se puede observar en la Tabla 4.5 y en la Figura 4.16 en términos generales, los encuestados estuvieron a gusto con la visibilidad del estado del sistema, lo que supone que el mismo permite que se pueda determinar fácilmente en qué vista de la aplicación se encuentra. Es posible afirmar que hay un buen nivel de satisfacción para este caso particular. La segunda etapa obtuvo un porcentaje mayor de respuestas positivas que la primera, como se puede observar en la Tabla 4.6 y en la Figura 4.17, lo que tiene sentido ya que se llevaron a cabo mejoras para indicarle al usuario en qué parte de la aplicación se encontraba.

2. **Se utiliza un lenguaje natural y simple**

   Este resultado fue un poco más disperso que el anterior en la primera encuesta, como se puede apreciar en la Tabla 4.5 y en la Figura 4.16, donde 2 de los encuestados estuvieron totalmente de acuerdo en que el lenguaje utilizado era natural y simple, otros 2 indicaron que estaban de acuerdo y el encuestado restante no estuvo ni de acuerdo ni en desacuerdo. Dado que se detectó que el lenguaje utilizado podría ser un poco más natural, se simplificaron los términos en general, antes de llevar a cabo la segunda encuesta. Como se observa en la Tabla 4.6 y en la Figura 4.17, se obtuvo una mejora total en la satisfacción de los usuarios, ya que los 5 encuestados estuvieron totalmente de acuerdo con la afirmación planteada.

3. **Es posible deshacer y rehacer las acciones de forma sencilla**

   En una interfaz de usuario es de gran importancia que las acciones puedan rehacerse y deshacerse. Para este caso, en principio 4 de los encuestados opinaron que la funcionalidad se provee, mientras que uno de ellos presenta una opinión central, sin satisfacción ni insatisfacción, como se aprecia en la Tabla 4.5 y en la Figura 4.16. Esto permite inferir que algunas de las acciones no se ajustaron a este requerimiento. Para la segunda encuesta, el índice de satisfacción incrementó, ya que como lo muestra la Tabla 4.6 y la Figura 4.17, todos los usuarios tuvieron una respuesta positiva respecto a este punto.

4. **Se pueden realizar las tareas de manera sencilla**

   Como se puede apreciar, en la Tabla 4.5 y la Figura 4.16, 3 de los encuestados estuvieron totalmente de acuerdo en que las tareas son muy sencillas de realizar, mientras que los 2 restantes estuvieron de acuerdo. El sistema, en términos generales permite llevar esto a cabo de una manera simple. Sin embargo, con la finalidad de mejorar aún

más la simplicidad de la aplicación, se incorporaron funcionalidades de manipulación de usuarios y personas en lotes. Luego de esta mejora, 4 de los encuestados estuvo totalmente de acuerdo con la sencillez de las tareas (Tabla 4.6 y Figura 4.17).

5. **El proceso para tomar fotos al momento de agregar o editar a una persona, es sencillo**

   En la primera etapa, 4 de los encuestados opinaron que el proceso de tomar fotografías en el sistema era sencillo, mientras que uno de ellos no tuvo una opinión polarizada. Esto puede ser observado en la Tabla 4.5 y en la Figura 4.16. Despúes de es esta primera encuesta se realizaron cambios generales en la aplicación y en la segunda encuesta se obtuvo un 100 % de satisfacción total con el proceso, como lo muestra la Tabla 4.6 y la Figura 4.17.

6. **Las acciones del sistema son consistentes**

   Los usuarios deben poder aprender una secuencia de acciones en una parte del sistema y aplicarla en otras para obtener resultados similares. La Tabla 4.5 y la Figura 4.16 muestran los resultados de la primera encuesta, donde la mayoría de los encuestados estuvo de acuerdo con esto, mientras que otros estuvieron en desacuerdo o no tuvieron una opinión al respecto. Esto pudo significar que las acciones del sistema no fueron completamente consistentes. Por esa razón se consultó con los encuestados, quienes indicaron que existían maneras diferentes de eliminación de datos en distintas vistas, por lo que se procedió a unificarlas, y como lo muestra la Tabla 4.6 y la Figura 4.17, el porcentaje de satisfacción aumentó hasta el punto en que todos los encuestados tuvieron una respuesta positiva en la segunda encuesta.

7. **El diseño de la aplicación es consistente**

   En términos generales, todos los encuestados estuvieron de acuerdo con esta interrogante, lo que significa que a lo largo de la aplicación todos los elementos de diseño son aplicados de la misma manera. Esto puede evidenciarse en la Tabla 4.5 y en la Figura 4.16. Sin embargo, los resultados fueron distintos en la segunda encuesta, mostrados en la Tabla 4.6 y en la Figura 4.17, donde 1 miembro de la muestra indicó no estar ni de acuerdo ni en desacuerdo con la interrogante. Esto pudo producirse debido a que alguno de los cambios aplicados pudo generar una diferencia de opinión en la segunda encuesta.

8. **La aplicación posee una buena distribución**

   Todos los elementos se encuentran distribuidos de manera lógica, adecuada y consistente, según la Tabla 4.5 y la Figura 4.16 lo evidencian, donde los encuestados coinciden con esto. En la segunda encuesta se obtuvo un resultado similar, con un nivel de satisfacción mayor (Tabla 4.6 y Figura 4.17).

9. **El diseño de la aplicación previene la ocurrencia de errores**

Los usuarios deberían sentirse seguros de las acciones que realizan, para prevenir errores humanos, y la aplicación debe ayudar a respaldar esta seguridad. En ambas encuestas, hubo satisfacción general respecto a esto, como se puede observar en las Tablas 4.5, 4.6 y Figuras 4.16, 4.17.

10. **Las instrucciones y opciones para el uso del sistema están a la vista**

    Todos los botones, secciones y demás deben estar a la vista para que el usuario pueda ser consciente acerca de su presencia. Los encuestados se sintieron a gusto con estos elementos en la aplicación, además de que los resultados obtenidos fueron exactamente iguales en ambas encuestas (Tablas 4.5, 4.6 y Figuras 4.16, 4.17).

11. **El sistema se adapta para usos frecuentes**

    La Tabla 4.5 y Figura 4.16 muestra cómo los usuarios en su mayoría, sintieron que podían ingresar a la aplicación de manera frecuente y navegar a gusto, mientras que un porcentaje menor no tuvo una opinión parcial al respecto. Luego de los cambios realizados, el índice de satisfacción se incrementó, y ningún usuario opinó de manera neutral, como lo muestra la Tabla 4.6 y la Figura 4.17.

12. **Los diálogos de la aplicación poseen información irrelevante**

    La ayuda provista en el sistema no debería ser irrelevante, ya que la idea es ayudar al usuario, no confundirlo. Los encuestados opinaron que el sistema no posee información de este tipo. La Tabla 4.5 y la Figura 4.16 lo evidencian. En la segunda encuesta, con resultados especificados en la Tabla 4.6 y en la Figura 4.17, más usuarios manifestaron que la información de la aplicación era relevante.

13. **El sistema posee la documentación necesaria para realizar las acciones**

    El sistema debe ayudar al usuario a despejar dudas que puedan surgir a través de herramientas como *tooltips*. En la primera iteración con resultados mostrados en la Tabla 4.5 y en la Figura 4.16, 4 encuentados opinaron que el sistema lo provee, mientras que uno de ellos no opinó lo mismo, y se sintió en desacuerdo con esto. Se procedió a indagar en la causa de este hecho y los usuarios manifestaron su poco entendimiento de la información situada en las cajas de ayuda, por lo que se procedió a modificarlas para que fuesen más explicativas. En la segunda iteración, el porcentaje de satisfacción respecto a la documentación se elevó, ya que como se puede apreciar en la Tabla 4.6 y en la Figura 4.17, todos los usuarios tuvieron una respuesta positiva.

Los cambios más importante realizados en la aplicación despúes de la primera etapa de la prueba de usabilidad fueron:

- Algunas de las palabras usadas en la aplicación se encontraban en inglés, las mismas fueron traducidas. Esto hizo que el lenguaje se hiciera más natural y simple para los usuarios.

- La opción de cancelar fue integrada para todas las acciones que se pueden realizar en la aplicación.
- La eliminación de usuarios y personas fue unificada para que las acciones del sstema fueras consistentes.
- Se integró la eliminación de personas y usuarios por lote.
- Los mensajes de ayuda de la aplicación fueron reescritos para que fueran más sencillos y facilitaran la interacción de los usuarios con el sistema.

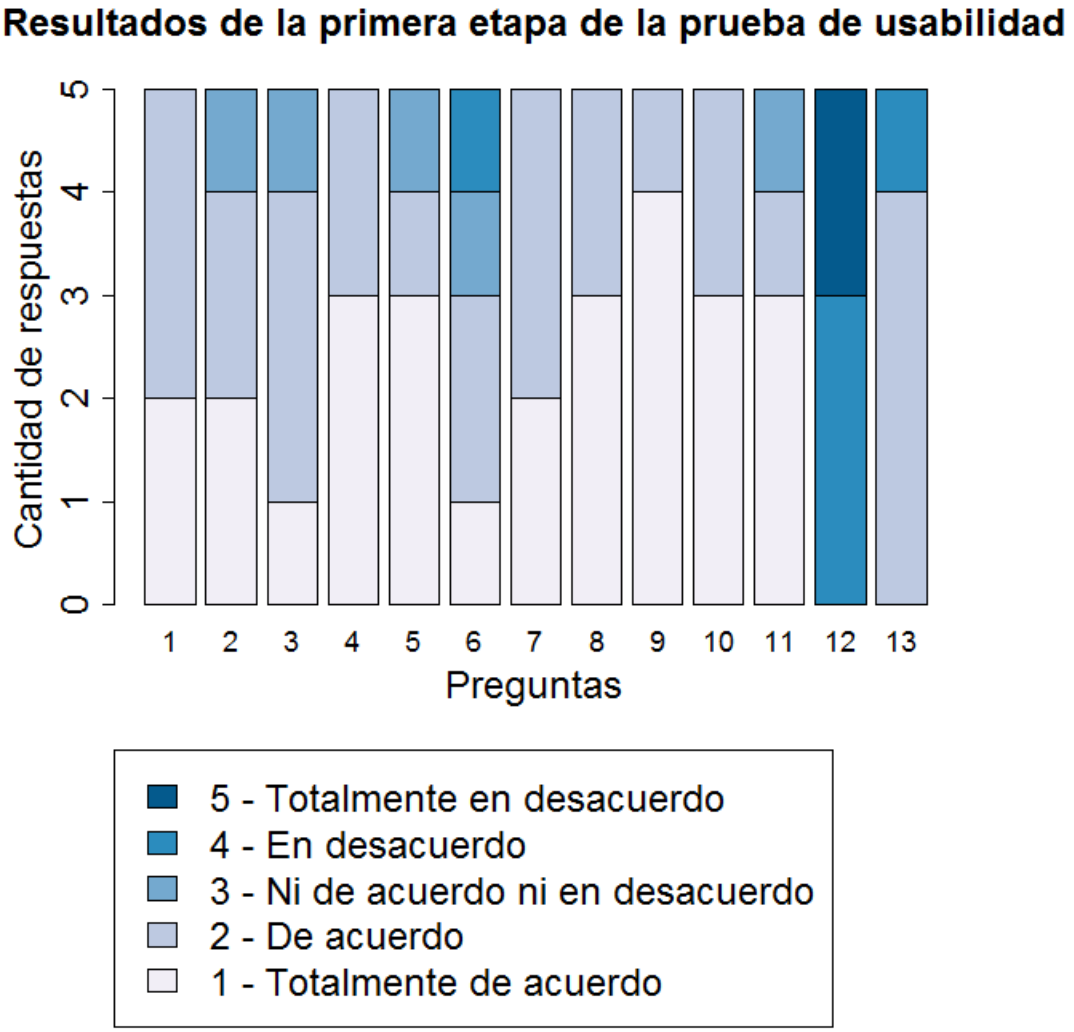

Figura 4.16: Gráfico de resultados referente a la primera etapa de la prueba de usabilidad.

Tabla 4.5: Resultados de la primera etapa de la prueba de usabilidad

	Evaluador 1	**Evaluador 2**	**Evaluador 3**	**Evaluador 4**	**Evaluador 5**
Preguntas	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-Visibilidad del estado de la aplicación	x	x	x	x	x
2-Lenguaje del sistema	x	x	x	x	x
3-Deshacer y rehacer las acciones de la aplicación	x	x	x	x	x
4-Realizar las tareas de manera sencilla	x	x	x	x	x
5-Proceso para tomar fotos	x	x	x	x	x
6-Acciones consistentes	x	x	x	x	x
7-Diseño consistente	x	x	x	x	x
8-Distribución	x	x	x	x	x
9-El diseño previene errores	x	x	x	x	x
10-Instrucciones y opciones a la vista	x	x	x	x	x
11-El sistema se adapta para usos frecuentes	x	x	x	x	x
12-Diálogos poseen información irrelevante	x	x	x	x	x
13- Documentación del sistema	x	x	x	x	x

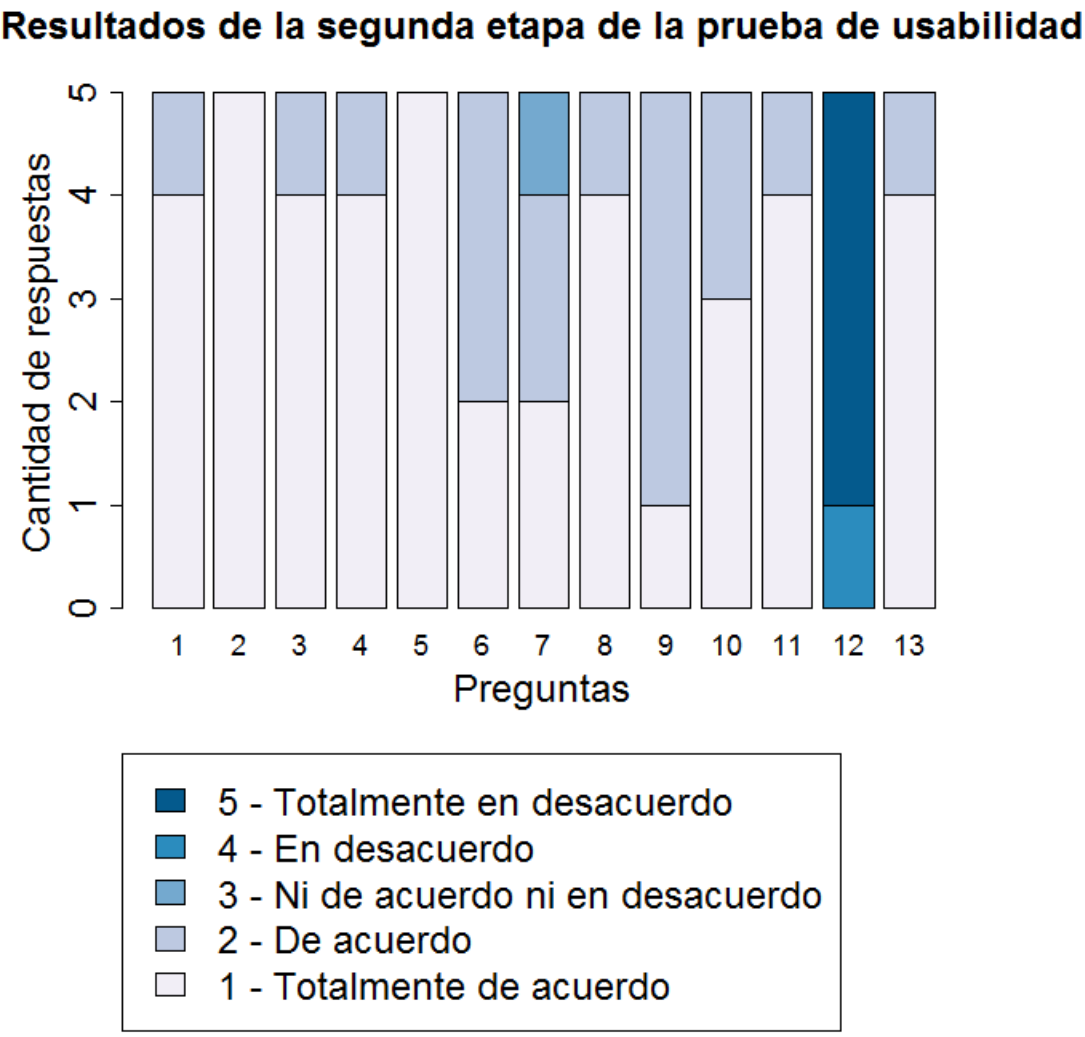

Figura 4.17: Gráfico de resultados referente a la segunda etapa de la prueba de usabilidad.

Los resultados obtenidos antes y después de la aplicación de los cambios mencionados, pueden ser vistos de forma gráfica en las Figuras 4.16 y 4.17. Ambas representan un resumen de los resultados mostrados en las Tablas 4.5 y 4.6 respectivamente. En estas Figuras se puede apreciar como la opinión con respecto a la aplicación mejoró en los 13 aspectos consultados. La pregunta 12 presenta una respuesta negativa, porque en esta se consultaba si los usuatios consideraban que los mensajes del sistema poseían información irrelevante, a lo cual respondieron con totalmente en desacuerdo o en desacuerdo. En el caso particular de dicha pregunta, esas respuestas son consideradas positivas.

Tabla 4.6: Resultados de la segunda etapa de la prueba de usabilidad

	Evaluador 1	**Evaluador 2**	**Evaluador 3**	**Evaluador 4**	**Evaluador 5**
Preguntas	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-Visibilidad del estado de la aplicación	x	x	x	x	x
2-Lenguaje del sistema	x	x	x	x	x
3-Deshacer y rehacer las acciones de la aplicación	x	x	x	x	x
4-Realizar las tareas de manera sencilla	x	x	x	x	x
5-Proceso para tomar fotos	x	x	x	x	x
6-Acciones consistentes	x	x	x	x	x
7-Diseño consistente	x	x	x	x	x
8-Distribución	x	x	x	x	x
9-El diseño previene errores	x	x	x	x	x
10-Instrucciones y opciones a la vista	x	x	x	x	x
11-El sistema se adapta para usos frecuentes	x	x	x	x	x
12-Diálogos poseen información irrelevante	x	x	x	x	x
13- Documentación del sistema	x	x	x	x	x

Capítulo 5

Conclusiones

En el trabajo de investigación se diseñó y desarrolló un sistema de seguridad basado en reconocimiento facial usando una Raspberry Pi, el cual posee una aplicación web, desde la cual es posible administrar a las personas registradas en el sistema y monitorear en vivo las imágenes capturadas por la cámara. Para esto se realizó un estudio previo acerca de las herramientas de *hardware* y *software* disponibles. También se estudiaron los algoritmos y técnicas de reconocimiento facial soportados por las bibliotecas de procesamiento digital de imágenes consultadas previamente. El sistema fue desarrollado para ser ejecutado sobre el sistema operativo Raspbian, el cual está basado en Debian, por lo que teóricamente podría ser ejecutado sobre otros sistemas operativos de este tipo.

Adicional a esto, se diseñaron varias pruebas a las cuales se sometió el sistema, para así determinar su rendimiento. En dichas pruebas se usaron cámaras web e IP. Así, se pudo determinar las limitaciones del sistema y cómo varia el rendimiento de acuerdo al tipo de cámara usado para capturar las imágenes.

Finalmente, se realizó una prueba de usabilidad en donde los usuarios evaluaron la aplicación web del sistema. Esto ayudó a determinar qué aspectos del mismo debían cambiar para ajustarse mejor a las necesidades de los usuarios finales.

En base a lo dicho anteriormente se puede concluir que los objetivos específicos propuestos al comienzo de la investigación y el objetivo general del presente trabajo fueron cumplidos, obteniendo así un sistema de seguridad basado en reconocimiento facial usando una Raspberry Pi.

5.1. Contribuciones

Las contribuciones de este trabajo de investigación son las siguientes:

- Se diseñó y desarrolló un sistema de seguridad con reconocimiento facial de bajo costo y de fácil configuración, el cual posee una aplicación web desde la cual cualquier usuario registrado puede acceder y realizar tareas de forma sencilla.
- Se desarrolló un módulo de alarmas que informa a los usuarios de cualquier actividad

sospechosa detectada por el sistema. Esta alarma es configurable desde la aplicación web del sistema.

- Se creó una aplicación web, la cual es del agrado de los usuarios. La misma hace que el acceso y administración del sistema de seguridad sea sencillo y rápido.

5.2. Limitaciones

Las limitaciones del trabajo son:

- La principal limitación de este trabajo se derivó del tipo de transmisión que brindaba la cámara IP, debido a que fue necesario obtener cada cuadro a través del protocolo HTTP, con codificación JPEG, ya que la cámara no provee un protocolo estándar para codificación de video. Por este motivo el rendimiento se vio afectado.
- La cantidad de cámaras disponibles para probar fue muy limitad, por lo que no se pudieron llevar a cabo estudios más específicos del sistema.

5.3. Trabajos futuros

Se proponen los siguientes trabajos a futuro:

- Realizar pruebas con modelos de cámaras IP más recientes y de esta forma determinar si el rendimiento del sistema es mejor que al usar una cámara web.
- Extender el alcance del sistema para conectarlo con la puerta del laboratorio ICARO de la escuela de computación en la UCV, y de esta forma poder abrir la misma desde la aplicación web.
- Modificar el sistema para usar más de una cámara y verificar que esto no afecte el rendimiento.
- Realizar una auditoría de seguridad informática al sistema para asegurar que éste sea robusto ante ataques y no filtre información a usuarios no registrados en el sistema.
- Ejecutar pruebas sobre sistemas operativos como Windows, MacOS y diferentes distribuciones de Linux de modo de verificar el correcto funcionamiento del sistema.
- Evaluar el comportamiento de sistema cuando los individuos tienen rasgos similares, como familiares, gemelos, etc.

Bibliografía

[1] C. Ortmeyer, "Then and now a brief history of single board computers," *Premier Farnell*, p. 11, Dic. 2014.

[2] W. Zhao, R. Chellappa, P. J. Phillips, y A. Rosenfeld, "Face recognition: A literature survey," *ACM computing surveys (CSUR)*, vol. 35, núm. 4, pp. 399–458, 2003.

[3] L. J. Abdi, Hervé y Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, núm. 4, pp. 433–459, 2010.

[4] P. N. Belhumeur, J. P. Hespanha, y D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, núm. 7, pp. 711–720, July 1997.

[5] M.-H. Yang, D. J. Kriegman, y N. Ahuja, "Detecting faces in images: A survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, núm. 1, pp. 34–58, January 2002.

[6] W. Zhao, R. Chellappa, P. J. Phillips, y A. Rosenfeld, "Face recognition: A literature survey," *ACM computing surveys (CSUR)*, vol. 35, núm. 4, pp. 399–458, December 2003.

[7] I. Marques y M. Grana, "Face recognition algorithms," *Proyecto Fin de Carrera*, p. 66, 2010.

[8] M. Manchanda, "Real time implementation of face recognition system," *Int. Journal of Engineering Research and Applications*, vol. 4, p. 3, October 2014.

[9] U. Bakshi y R. Singhal, "A survey on face detection methods and feature extraction techniques of face recognition," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 3, núm. 3, 2014.

[10] A. K. Jain, R. P. Duin, y J. Mao, "Statistical pattern recognition: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, núm. 1, pp. 4–37, 2000.

[11] P. Viola y M. Jones, "Rapid object detection using a boosted cascade of simple features," en *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–511.

[12] P. I. Wilson y J. Fernandez, "Facial feature detection using haar classifiers," *Journal of Computing Sciences in Colleges*, vol. 21, núm. 4, pp. 127–133, 2006.

[13] C. C. Aggarwal, *Data Mining: The Textbook*. Springer, 2015.

[14] B. Schölkopf y A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[15] C. C. Aggarwal, *Data classification: algorithms and applications*. CRC Press, 2014.

[16] N. Cristianini y J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[17] V. Bevilacqua, L. Cariello, G. Carro, D. Daleno, y G. Mastronardi, "A face recognition system based on pseudo 2d hmm applied to neural network coefficients," *Soft Computing*, vol. 12, núm. 7, pp. 615–621, 2008.

[18] C. Liu y H. Wechsler, "A unified bayesian framework for face recognition," en *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 1. IEEE, 1998, pp. 151–155.

[19] M. Kirby y L. Sirovich, "Application of the karhunen-loeve procedure for the characterization of human faces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, núm. 1, pp. 103–108, 1990.

[20] R. P. Foundation. (2016, Feb.) Raspberry pi - teach, learn, and make with raspberry pi. [En línea]. Disponible en: https://www.raspberrypi.org/

[21] T. F. Dictionary. (2016) Camera. [En línea]. Disponible en: http://www.thefreedictionary.com/camera

[22] N. Memon, "Photo forensics–there is more to a picture than meets the eye," en *International Workshop on Digital Watermarking*. Springer, 2011, pp. 2–2.

[23] G. Bradski y A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. .º'Reilly Media, Inc.", 2008.

[24] Crossbar.io. (2016, Sep.) Crossbar.io - networking for apps. [En línea]. Disponible en: https://crossbar.io/

[25] I. A. N. Authority. (2016, Sep.) Iana - websocket protocol registries. [En línea]. Disponible en: http://www.iana.org/assignments/websocket/websocket.xhtml

[26] B. W. Gregor Hohpe, *Enterprise Integration Patterns: Designing, Building and Deploying Messaging Solutions*. Addison-Wesley, 2004.

[27] E. International. (2016, Sep.) Standard ecma. [En línea]. Disponible en: http://ecma-international.org/

[28] S. Pasquali, *Mastering Node.js*. PACKT, 2013.

[29] K. Banker, *MongoDB in Action*. Manning, 2011.

[30] Google. (2010-2016) Angularjs. [En línea]. Disponible en: https://angularjs.org/

[31] M. Jakl, "Representational state transfer," 2005.

[32] S. Kanzariya y V. Vora, "Real time video monitoring system using raspberry pi," *ETCEE–2015*, p. 19, 2015.

[33] R. Shakya y K. Prabha, "Design and implementation of raspinode for surveillance and border intrusion detection," *International Journal of Innovative Research in Computer and Communication Engineering*.

[34] A. B. Amaya Arcos, "Sistema alternativo de seguridad vehicular basado en reconocimiento facial," 2015.

[35] G. Senthilkumar, K. Gopalakrishnan, y V. S. Kumar, "Embedded image capturing system using raspberry pi system," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 2014.

[36] S. Sneha, A. Pradnya, y B. Yogesh, "Ip camera video surveillance using raspberry pi," *International Journal of Advanced Research in Computer and Communication Engineering*.

[37] M. Sahani, C. Nanda, A. K. Sahu, y B. Pattnaik, "Web-based online embedded door access control and home security system based on face recognition," en *Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on*. IEEE, 2015, pp. 1–6.

[38] K. Beck y M. Fowler, *Planning extreme programming*. Addison-Wesley Professional, 2001.

[39] G. J. Stellman Andrew, *Learning Agile: Understanding Scrum, XP, Lean, and Kanban 1st Edition*. O'Reilly Media, 2013.

[40] K. M. Lui y K. C. Chan, "Pair programming productivity: Novice–novice vs. expert–expert," *International Journal of Human-computer studies*, vol. 64, núm. 9, pp. 915–925, 2006.

[41] P. Pham, "Hybrid mobile application with ionic and mean stack," 2016.

[42] S. Josefsson, "The base16, base32, and base64 data encodings," 2006.

[43] J. Nielsen y T. K. Landauer, "A mathematical model of the finding of usability problems," en *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. ACM, 1993, pp. 206–213.

[44] R. Johns, “Likert items and scales,” *Survey Question Bank: Methods Fact Sheet*, vol. 1, 2010.

Anexos

Anexo A

Guía de instalación

1. Para la instalación del sistema son necesarias las siguientes dependencias.

   - OpenCV 2.4 con soporte para Python
   - Node.js 4.4.4
   - Python-pip 8.1.1
   - Crossbar.io 0.13.2
   - Imutils 0.3.6
   - Bower 1.7.9
   - Gulp 3.9.1

2. Una vez instaladas todas las dependencias se debe clonar el proyecto del repositorio donde se encuentra almacenado. Seguidamente se deben instalar las dependencias del módulo de *front-end* y del API usando npm. Los comandos para realizar este paso se muestran en el Código A.1.

```
$ git clone https://bitbucket.org/tesis-ucv/facerecognition.git
$ cd admin
$ npm install
$ cd frontend
$ npm install
```

Listado A.1: Instalación del sistema

3. Para ejecutar el proyecto, se necesitan tres terminales, una para cada módulo de la aplicación (reconocedor, API, *front-end*). Esto se muestra en los Códigos A.2, A.3, A.4.

```
$ cd admin
$ npm start
```

Listado A.2: Ejecución del API

```
$ cd recognizer
$ crossbar start
```

Listado A.3: Ejecución del módulo de reconocimiento

```
$ cd frontend
$ gulp serve-dev
```

Listado A.4: Ejecución del módulo de *front-end*

4. Para importar la base de datos es necesario utilizar el comando *mongoimport* utilizando los archivos descritos en A.5 y A.6.

```
{
  "_id":{
  "$oid":"5803c6e71fd8cbb22beb23b5"
  },
  "role":"admin",
  "school":"Computacion",
  "lastName":"Admin",
  "firstName":"Admin",
  "password":"$2a$08$d9Lxcu/f1Gcfg8DstHuJHOPRKTm/B.
     mbiYMCM3BMb8hU4OUe6EA2e",
  "email":"admin@admin.com",
  "updated_at":{
  "$date":"2016-10-16T18:28:55.671Z"
  },
"__v":0
}
```

Listado A.5: Semilla de usuario administrador

```
{
  "_id":{
  "$oid":"57e6cab2ca2ec0fd0e58faad"
  },
  "thresholds":{
    "confidence":1000,
    "alarm":200
  },
  "recipients":[
  ],
  "sender":{
    "email":"frecognizer@gmail.com",
    "password":"carloskarina",
    "subject":"Alarma",
    "text":"Se ha detectado la presencia de un intruso",
    "host":"smtp.gmail.com",
    "port":587,
```

```
    "beginTime":200,
    "endTime":1300,
    "active":false
  },
  "camera":{
    "url":"http://190.169.74.220/cgi-bin/video.jpg",
    "cameraType":1
  },
  "__v":0
}
```

Listado A.6: Semilla de configuraciones

La clave por defecto para el administrador es 1234. Al momento de importar las configuraciones y usuarios es necesario transportar A.5 y A.6 a dos archivos y ejecutar los comandos mostrados en A.7.

```
$ mongoimport -h <host> -d facerecognition -c users -p <clave> --
    file <archivo con usuario administrador>.js

$ mongoimport -h <host> -d facerecognition -c configurations -p <
    clave> --file <archivo con configuraciones>.js
```

Listado A.7: Importación de la base de datos

Anexo B

Guía de configuración

Cada uno de los módulos de la aplicación necesitan ser configurados para poder interoperar. Las configuraciones se listan a continuación:

1. **Módulo API:** es necesario definir la dirección IP y puerto para el API, lo que puede ser encontrado en el archivo **admin/bin/www**, como se muestra en B.1.

```
var port = normalizePort(process.env.PORT || "<Puerto>");
app.set("host", <IP>);
```

Listado B.1: Porción de código de admin/bin/www

Igualmente, la conexión con la base de datos debe ser especificada en **admin/app.js**, en la línea mostrada en B.2.

```
mongoose.connect("mongodb://<usuario>:<clave>@<host>:<puerto>/<
    nombre de la base de datos>", function(err){})
```

Listado B.2: Conexión con la base de datos

2. **Módulo de frontend:**

   para el módulo de frontend se deben definir las direcciones IP a través de las cuales se realiza la comunicación con los demás módulos y clientes. Las configuraciones respectivas son:

   En el archivo **frontend/src/client/app/core/constants.js**, como se observa en el Código B.3, SERVICE_URL define la dirección IP y el puerto sobre los cuales se ejecutará el módulo, WEBSOCKET_IP define la dirección IP y puerto utilizado por el módulo Recognizer para hacer la transmisión del video a través de websockets, WEBSOCKET_USER y WEBSOCKET_PASSWORD definen el usuario y contraseña utilizados para autenticarse en el websocket.

```
.constant("SERVICE_URL", <IP>:<puerto>)
.constant("WEBSOCKET_IP", <IP>:<puerto>)
```

```
    .constant("WEBSOCKET_USER", <usuario>)
    .constant("WEBSOCKET_PASSWORD", <clave>)
```

Listado B.3: Configuración de módulo de frontend

3. **Módulo recognizer:**

   la configuración básica de Crossbar.io se encuentra en el archivo **recognizer/.crossbar/config.json**, donde se debe definir la dirección IP y puerto por el cual se hará la transmisión del vídeo a través de websockets, además se deben configurar los datos de nombre de usuario de contraseña para autenticarse. Específicamente esto se debe modificar en B.4.

```
"transports": [
    {
    "type":"websocket",
    "endpoint":{
      "type":"tcp",
      "interface":"<IP>",
      "port":<Puerto>,
      "tls":{
        "key": "server.key",
        "certificate": "server.crt"
      }
    },
  "url":"wss://<IP>:<Puerto>/ws",
  "auth":{
    "wampcra": {
      "type": "static",
      "users": {
        "<usuario>": {
          "secret": "clave",
          "role": "frontend"
        }
      }
    }
  }
  }
]
```

Listado B.4: Configuración de Crossbar.io

Igualmente, es necesario acceder al archivo **recognizer/src/constants.py** para definir los datos de la conexión con el API B.5

```
self.admin_email = '<Usuario administrador (email)>'
self.admin_password = '<Password>'
self.admin_url = '<IP del API>:<Puerto del API>'
```

Listado B.5: Conexión con API en el módulo recognizer