UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

SISTEMA DE RECONOCIMIENTO DE ROSTROS PARA SEGURIDAD EN EL USO DE CELULARES CON S.O. ANDROID.

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

CARLOS FREDDY MONTENEGRO ARIAS

DIRECTOR: ING. RODRIGO PROAÑO

Quito, Abril 2015

DECLARACIÓN

Yo **CARLOS FREDDY MONTENEGRO ARIAS**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Carlos Freddy Montenegro Arias

1717572711

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título "**Sistema de reconocimiento de rostros para seguridad en el uso de celulares con s.o. android**", que, para aspirar al título de **Ingeniero en Informática y Ciencias de la Computación** fue desarrollado por **Carlos Freddy Montenegro Arias**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 18 y 25.

_____

Ing. Rodrigo Proaño

DIRECTOR DELTRABAJO

C.I. 170854904-1

DEDICATORIA

A Dios que supo guiarme durante todo el trayecto de mi carrera y apoyarme en los problemas que se presentaban, ayudándome a enfrentar las adversidades sin perder nunca la fe.

A mi madre por sus consejos, amor y comprensión durante todo el tiempo, ya que ayudó a formar todos mis valores, principios y lo necesario para conseguir mis objetivos a lo largo de mi vida.

Para mi familia que fue un pilar importante en el transcurso de mi carrera, donde cada uno contribuyó de la manera más cordial, y el apoyo que me brindaron fue de gran ayuda en mis estudios.

AGRADECIMIENTO

Agradezco a Dios por haberme apoyado y dado fuerza durante esos momentos difíciles a lo largo de toda mi carrera, permitiéndome llevar a cabo el desarrollo mi tesis.

Le doy gracias a mi madre Lic. Mireya Arias por los buenos valores que me ha inculcado, ya que gracias a su esfuerzo pude realizar mis estudios en esta universidad, porque sin su apoyo no lo hubiera logrado.

A todos los que conforman mi familia, por confiar y siempre darme ánimo para seguir adelante, luchando para conseguir mis objetivos.

Quiero agradecer a mi Director de Tesis Ing. Rodrigo Proaño por su importante aporte y participación durante el desarrollo de esta tesis. Agradeciendo su disponibilidad y paciencia mientras realizaba mi tesis, enriqueciendo el trabajo realizado.

A mis profesores de la universidad, ya que gracias a sus enseñanzas y consejos he podido formarme como un profesional con valores.

Gracias a todos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

PÁGINA

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE ANEXOS

RESUMEN

Esta tesis tuvo como finalidad desarrollar una aplicación que brinda seguridad en el uso de dispositivos móviles con sistema operativo (s.o.) Android. La misma que permite bloquear cualquier aplicación instalada y realizar un reconocimiento de rostros usando la cámara frontal del dispositivo para identificar al usuario que puede acceder a dicha aplicación. La seguridad se implementa mediante el uso de técnicas de inteligencia artificial y librerías que permitan realizar el reconocimiento del rostro. Además permite el ingreso de una clave como medida adicional en caso de fallo de la cámara o cualquier variación del ambiente, como falta de iluminación que no permita una identificación visual del usuario. Se realizó un análisis de algunas técnicas que existen actualmente relacionadas con el reconocimiento de rostros, como redes neuronales, eigenfaces, entre otras. También se investigó el uso de las librerías neuroph y opencv. Como resultado final se obtuvo una aplicación capaz de realizar el reconocimiento del rostro para permitir el acceso a las aplicaciones bloqueadas del dispositivo móvil.

ABSTRACT

This thesis aimed to develop an application that provides security in the use of mobile devices with Android operating system. It allows block any installed application and perform a face recognition using the front camera of the device to identify the user who can access using the application. Security is implemented by using artificial intelligence techniques and libraries that allow for face recognition. It also allows the entry of a key as an additional measure in case of failure of the camera or any variation of the environment as lack of lighting which does not allow visual identification of the user. An analysis of some techniques that are currently associated with face recognition, such as neural networks, eigenfaces, among other was performed. The use of Neuroph and opencv libraries are also investigated. As a final result an application capable of performing face recognition to allow access to blocked applications mobile was obtained.

INTRODUCCIÓN

1 INTRODUCCIÓN

El rostro es nuestro principal punto de atención en la sociedad, ya que permite identificarnos y mostrar nuestras emociones. Pese a que la capacidad de inducir el carácter de la apariencia facial es sospechosa, la capacidad humana de reconocer caras es notable. Se puede reconocer miles de caras aprendidas a lo largo de nuestra vida e identificar caras conocidas con una sola mirada, incluso después de años de no haberle visto a esa persona. Esta habilidad es bastante eficiente, aunque haya habido grandes cambios visuales, ya sean condiciones de expresión, envejecimiento, y otros factores tales como bigote, gafas o cambios de estilo de cabello.

A pesar que las personas son buenas en el reconocimiento de rostros, no es del todo claro cómo se codifican o decodifican las caras en el cerebro. Por lo tanto, el reconocimiento facial es una tarea de la visión por computador de muy alto nivel, en el que muchas técnicas pueden estar involucradas, ya que el primer paso de identificación del rostro humano es extraer las características relevantes a partir de imágenes faciales.

Hoy en día el reconocimiento facial o reconocimiento de rostros se ha convertido en un problema importante en muchas aplicaciones tales como sistemas de seguridad, la verificación de identidad en tarjetas de crédito, identificación criminal y últimamente está siendo utilizado en dispositivos móviles, ya que esto permite tener mayor facilidad y control al momento de poner seguridad a los mismos, permitiendo a los usuarios acceder al uso de su celular de manera más rápida y segura, identificando su rostro frente a la cámara del dispositivo móvil, con lo cual podrá acceder a todas sus aplicaciones.

Al desarrollar aplicaciones de reconocimiento de rostro, se debe tomar en cuenta que no están exentos de errores, por eso las aplicaciones que comienzan a tener éxito se basan en librerías previamente entrenadas y

probadas para minimizar este error, además deben tener una interfaz de fácil uso para el usuario.

En el mercado existen muchas aplicaciones de reconocimiento facial para Android, como facelock pro (FaceLock, 2013), visidon applock (Visidon Ltd, 2015), entre otras, las mismas que poseen 2 versiones, una de pago y otra gratis. Las versiones de pago cuentan con la opción de bloquear cualquier aplicación instalada en el dispositivo móvil, además tienen una mejor interfaz de usuario y más funcionalidades para el nivel de seguridad, mientras que las versiones gratis no cuentan con todas las funcionalidades que traen las de pago, por ejemplo solo permiten bloquear una aplicación a la vez, y en algunos casos caducan al poco tiempo. Por eso se pretende desarrollar una aplicación que permita bloquear cualquier aplicación instalada en el dispositivo móvil para mantener mayor seguridad en el mismo.

El reconocimiento facial está teniendo múltiples aplicaciones, por ejemplo la aplicación nametag (Facial Network Inc., 2015) que sirve para búsqueda de personas en las redes sociales, esto se logra tomando una foto de cualquier persona desconocida, donde se compara el rostro de la misma con los millones de registros de las redes sociales como Facebook o Twitter, permitiendo identificarla.

Una de las librerías que últimamente está siendo muy utilizado para desarrollar este tipo de sistemas es opencv (Opencv devteam., 2014), debido a que cuenta con funciones y métodos previamente entrenados que facilitan mucho al momento de realizar el reconocimiento facial para dispositivos móviles.

El uso de las redes neuronales también es muy utilizado para aplicaciones de reconocimiento facial debido a su capacidad de aprendizaje, por eso facebook tiene uno de los mejores sistemas de reconocimiento facial del mundo, utilizado en su sistema llamado deepface (Taigman, Yang, Ranzato, & Wolf, 2014).

Un problema hoy en día es que existe un alto riesgo de pérdida o sustracción de los dispositivos móviles, quedando la información que contienen de una manera expuesta que puede incluso ser utilizada con fines maliciosos por personas inescrupulosas. Por este motivo si bien algunos dispositivos móviles utilizan técnicas de seguridad, el propósito de la aplicación es usando técnicas de inteligencia artificial, visión artificial, o alguna otra que se acople al modelo, hacer un reconocimiento del rostro del usuario previamente registrado, dándole acceso al uso de su dispositivo.

También el usuario podrá elegir cualquiera de las aplicaciones instaladas en su celular para bloquear con este sistema de reconocimiento de rostros, como por ejemplo el acceso a los contactos del dispositivo móvil, la mensajería, etc. Además se tendrá la opción de añadir una clave adicional para poder ingresar en caso de que por algún motivo su rostro no pueda ser reconocido y el sistema no le permita el acceso.

El objetivo es desarrollar una aplicación de reconocimiento de rostros para seguridad en el uso de dispositivos móviles con s.o. Android, que pueda utilizarse en cualquier dispositivo móvil con un mínimo de características de hardware y software, para este proyecto se probará en el teléfono Samsung Transform Ultra.

Como objetivos específicos se proponen los siguientes.

1. Investigar modelos de reconocimiento de patrones e inteligencia artificial que ayuden a solucionar el problema de reconocimiento de rostros.

2. Buscar e interactuar con librerías de Android que permitan realizar este tipo de proyectos.

3. Investigar el esquema de seguridad de los dispositivos móviles que usan Android.

4. Implementar como prototipo un sistema de reconocimiento de rostros para seguridad en el uso de dispositivos móviles con s.o. Android.

MARCO TEÓRICO

2 MARCO TEÓRICO

El reconocimiento del rostro humano se ha estudiado durante más de veinte años, y sin embargo desarrollar un modelo computacional de reconocimiento de rostro perfecto todavía es bastante difícil, porque los rostros son complejos.

Por eso se han creado varios métodos para el reconocimiento de rostros. Un método se basa en la extracción de vectores de características, a partir de las partes básicas de un rostro como son: los ojos, la nariz, la boca y barbilla, etc. (Agarwal, Agrawal, Jain, & Kumar, 2010).

Otro método se basa en análisis de componentes principales (Kumar Gupta, Kumar Sahu, 2013*)* el cual ha demostrado que cualquier cara en particular puede ser representada en términos de un sistema de coordenadas denominado como "eigenfaces". También se está estudiando las redes neuronales artificiales, que se utilizan debido a su capacidad de aprender a partir de datos observados. (Agarwal, Agrawal, Jain, & Kumar, 2010)

Un sistema de reconocimiento facial es una aplicación informática para identificar o verificar de forma automática a una persona por medio de su rostro ya sea desde una cámara digital, webcam, una imagen o a partir de una fuente de vídeo. Una de las maneras de hacerlo es mediante la comparación de rasgos faciales seleccionados de la imagen del rostro y una base de datos faciales. Normalmente el reconocimiento de rostros se utiliza en los sistemas de seguridad y se puede comparar con otros datos biométricos, tales como huellas dactilares o los sistemas de reconocimiento del iris del ojo.

Las redes neuronales también son usadas para sistemas de reconocimiento de rostros, ya que se caracterizan por aprender de la experiencia adquirida, pueden procesar datos distorsionados y seguir funcionando a pesar de lesiones (Ñustes, Hurtado, Bedoya & Marin, 2013). Una librería que ayuda mucho al momento de crear estructuras de redes neuronales es neuroph, ya

que cuenta con funciones y arquitecturas de redes previamente probadas. (Sevarac et .al, 2014).

2.1 MODELO DE RECONOCIMIENTO DE ROSTROS GENÉRICO

Un sistema de reconocimiento de rostros genérico puede ser desarrollado con una serie de pasos que se describen a continuación. Primero la localización o detección del rostro que es el proceso de extraer la región de imagen que contiene un rostro. Luego se debe realizar una normalización del rostro con el fin de estandarizar cierta escala y rotación en el plano de la imagen. El siguiente paso de la extracción de características, como son los rasgos faciales relevantes de los datos, debe ser eficiente en términos de tiempo de cálculo y uso de memoria. La reducción de dimensionalidad es otro paso esencial de cualquier reconocimiento de sistema de patrones. Después tenemos que clasificar las imágenes del rostro, y el rendimiento del clasificador dependerá a la cantidad de imágenes de muestra tomadas. Por último el sistema deberá tomar la decisión para comprobar si se encuentra un rostro conocido. En la Figura 1 se muestra la estructura de un sistema genérico de reconocimiento de rostros. (*Behan & Mansoor, 2013*).

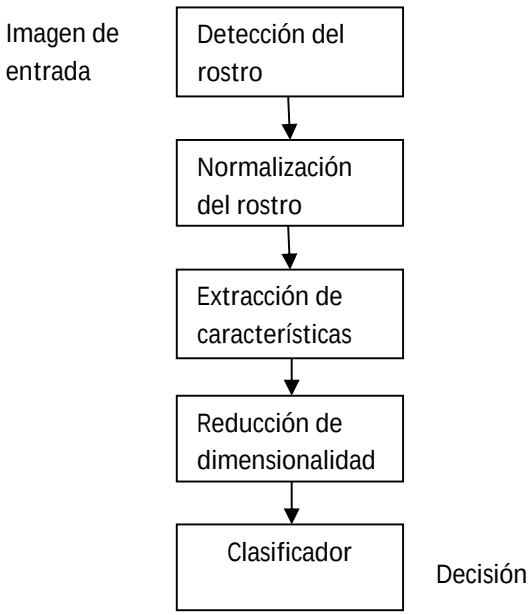

Figura 1. Sistema genérico de reconocimiento de rostros

A continuación se describe algunos de los métodos más utilizados para el reconocimiento facial.

2.2 EIGENFACES (ANÁLISIS DE COMPONENTES PRINCIPALES)

Para el proceso de los clasificadores de rostros en eigenfaces se necesita calcular los valores propios y vectores propios de la formación de la matriz de covarianza, el conjunto de eigen vectores definen un nuevo espacio en donde las imágenes son representadas. Dada la colección de vectores propios, se construye un conjunto de vectores de características para cada imagen.(Alvarado, Pedrycz, Reformat, &Kwak, 2006)

Los primeros trabajos de reconocimiento facial se remontan por lo menos hasta la década de 1950 pero el reconocimiento facial realmente se inició en la década de 1970 por el autor Kelly y Kanade.(Zhao, Chellappa, Phillips, & Rosenfeld, 2000).

La idea de usar eigenfaces fue dada por una técnica desarrollada por Sirovich y Kirby entre 1987 y 1990 (Turk & Pentland, 1991) para representar eficazmente las imágenes de rostros utilizando análisis de componentes principales (PCA). El mismo que a partir de un conjunto de imágenes de rostros originales, calculó un mejor sistema de coordenadas para la compresión de imágenes, donde cada coordenada en realidad es una imagen que ellos llamaron una eigenpicture. Afirmaron que, al menos en principio, cualquier colección de imágenes del rostro puede ser aproximadamente reconstruida mediante el almacenamiento de una pequeña colección de pesos para cada rostro y un pequeño conjunto de imágenes estándar. (Turk & Pentland, 1991)

Por otra parte si un gran conjunto de imágenes del rostro puede ser reconstruido por sumas ponderadas de una pequeña colección de rasgos característicos también llamados eigenpictures, entonces una forma eficiente

de aprender y reconocer rostros podría ser la construcción de rasgos característicos por medio de la experiencia a través del tiempo, y reconocer rostros concretos mediante la comparación de las características necesarias para reconstruir con los pesos asociados a los rostros de las personas conocidas. Por lo tanto, cada rostro de una persona se caracteriza por un pequeño conjunto de características o pesos necesarios para poder describirlo y reconstruirlo.

2.2.1 PASOS PARA EL RECONOCIMIENTO DE ROSTROS

Este reconocimiento de rostro implica los siguientes pasos:

1. Primero se debe obtener una serie de imágenes del rostro.

2. Luego se calculan los eigenfaces del conjunto de entrenamiento, de las cuales solo se tomarán los valores propios más altos. Las imágenes obtenidas definirán el espacio del rostro. Cuando se toma nuevas muestras de rostros, los eigenfaces deben ser actualizados o recalculados.

3. Después se calcula la distribución de acuerdo al espacio de peso para cada individuo conocido, mediante la proyección de sus imágenes del rostro en el espacio de la cara.

Una vez que el sistema se haya iniciado, se utilizan los siguientes pasos para reconocer nuevas imágenes del rostro:

Primero se debe calcular un conjunto de pesos sobre la base de la imagen de entrada y los eigenfaces mediante la representación de la imagen de entrada en cada uno de los mismos.

Luego determinar si la imagen es un rostro conocido o desconocido por medio de la comprobación, para ver si la imagen se encuentra suficientemente cerca del espacio de cara.

Si se trata de un rostro, clasificar el patrón de peso, ya sea como una persona conocida o como desconocida.

Actualizar los eigenfaces o patrones de peso (opcional).

Si se observa el mismo rostro de manera repetitiva, calcular su peso patrón característico e incorporar a los rostros más conocidos (opcional). (Rabbani & Chellappan, 2007).

2.2.2 CALCULO DE EIGENFACES

De acuerdo a Turk y Pentland, si tomamos una imagen de cara I(x,y) de 2 dimensiones nxn de 8 bits, puede ser vista como un vector de N^2 de tal manera que una imagen de tamaño de 10 por 10 se convierte en un vector de dimensión 100. Las imágenes de rostros son consideradas como un sub espacio dimensional parcialmente bajo de todo el espacio de la imagen.

La idea primordial del PCA es encontrar los vectores que mejor cuenten para la distribución de imágenes de rostros dentro de todo el espacio de la imagen y los vectores definen el sub espacio de imágenes de la cara, lo que llamamos espacio de la cara o facespace.

En la Figura 2 se muestra la imagen original y sus primeros 5 eigenfaces.

Figura 2. Rostro con sus eigenfaces

(Heseltine T., Pears N., Austin J., 2002)

2.3 REDES NEURONALES ARTIFICIALES

En los últimos años, las redes neuronales artificiales (RNA) han tomado un notable interés en el campo de reconocimiento de rostros. Una red neuronal artificial es un sistema adaptativo que cambia su estructura basada en la

información externa o interna que fluye a través de la red durante la fase de aprendizaje. Kohonen fue el primero que utilizó redes auto asociativas de memoria para almacenar y recuperar imágenes del rostro. (Mayank, Nikunj, Kumar & Himanshu, 2010).

La redes neuronales artificiales (RNA) son estructuras de procesamiento de información sin memoria compartida, donde cada uno de sus elementos opera solo cuando toda la información de entrada está disponible, parecido a estructura de flujo de datos. Las RNA principalmente se han convertido en el punto de atención porque está relacionada con la búsqueda de soluciones satisfactorias para solucionar problemas para los que existe un conocimiento escaso o nulo sobre el proceso. (Rabunal, Dorado & Pazos, 2009).

2.3.1 USO DEL ALGORITMO GENÉTICO Y REDES NEURONALES DE RETRO PROPAGACIÓN PARA EL RECONOCIMIENTO DE ROSTROS.

Esquema del sistema

El diseño e implementación del sistema de reconocimiento de rostros o Face Recognition System (FRS) se divide en dos partes. Una parte es el procesamiento de imágenes y otra son las técnicas de reconocimiento. La primera parte se encarga del procesamiento de imágenes que consiste en la adquisición de imágenes a través del barrido del rostro, mejora y recorte de imagen, filtrado, detección perimetral y extracción de características. La segunda parte consiste en la inteligencia artificial que está compuesto por el algoritmo genético y la red neuronal de retro propagación.

En la primera parte del sistema de reconocimiento de rostros se utiliza varias técnicas de procesamiento de imágenes. Primero se obtiene la imagen del rostro el cual se puede obtener mediante una cámara digital, cámara web o el uso de escáner, luego se realiza un recorte de la imagen mediante el algoritmo de detección con un punto de inicio y punto final. Después se detectan los bordes utilizando filtro de paso alto, filtro de mediana o algún método de detección de bordes. Para finalizar se extraen las características

y se incorporan después en el algoritmo genético y red neuronal de retro propagación.

Para la segunda parte se usan dos técnicas uno se basa en el Algoritmo genético y otro se basa en la red neuronal de retro propagación. En la primera parte, las características extraídas se guardan en la memoria y el uso de algoritmo genético, en la segunda parte las características extraídas se introducen en la entrada de la red neural multicapa y la red está capacitada para crear una base de conocimientos, para que luego se realice el reconocimiento del rostro.(Anam et al., 2009).

Pasos para el reconocimiento del rostro

A continuación en la Figura 3, se muestra un diagrama general del sistema de reconocimiento de rostros usando redes neuronales de retro propagación.

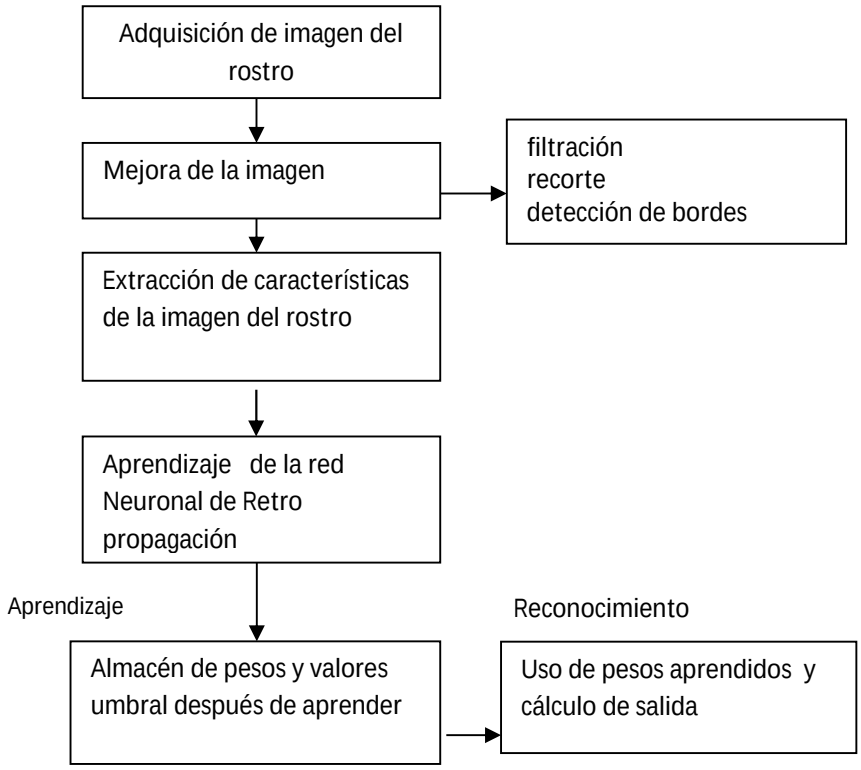

Figura 3. Diagrama general del sistema de reconocimiento de rostros

1) Adquisición de las imágenes del rostro.

Para obtener las imágenes del rostro, se puede utilizar una cámara o cualquier dispositivo que pueda capturar una foto, en este caso se ha utilizado un scanner. Después del escaneado, la imagen se puede guardar en varios formatos tales como mapa de bits. JPEG, GIF, etc.

2) Filtrado y recorte

La imagen de rostro que ingresa al sistema puede contener ruido y datos basura que deben ser eliminados. Después de filtrar, la imagen se recorta para obtener los datos necesarios que se requieren para la eliminación del fondo innecesario que rodea la imagen.

3) Detección de bordes

Existen varios métodos para la detección de bordes de una imagen, el procedimiento para la determinación de los bordes de una imagen puede realizarse mediante el uso de máscaras. Distintos tipos de máscaras se pueden aplicar como Sobel, Prewitt, Kirsch (Esqueda & Palafox, 2005) que son máscaras rápidas para obtener el borde de una imagen de la cara. A continuación en la Figura 4 se muestra la imagen del rostro aplicada la máscara.

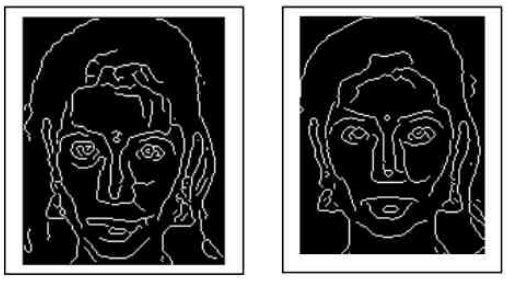

Figura 4. Aplicación de mascara para detección de bordes.

(Anam et al., 2009)

4) Escala de imagen

Hay varias técnicas para ajustar la escala de la imagen. Esto dependerá del tamaño con el cual queremos trabajar en el reconocimiento de rostros.

5) Características de extracción

Para extraer los rasgos de un rostro la imagen se convierte a un número binario. Después desde el centro de gravedad, el rostro se convierte a nivel de gris y las características son recogidas. Como se muestra en la Figura 5.

Figura 5. Conversión de imagen a un número binario

(Anam et al., 2009)

6) Por último el reconocimiento

Las características extraídas de las imágenes del rostro han sido alimentadas al algoritmo genético y la Red neuronal de retro propagación para el reconocimiento.(Anam et al., 2009)

2.4 KOHONEN

2.4.1 RECONOCIMIENTO DE ROSTROS HUMANOS UTILIZANDO EL MAPA AUTO ORGANIZADO (SOM) DE KOHONEN.

Otra de las técnicas de inteligencia artificial para el reconocimiento de rostros es el mapa auto organizado (Self Organized Map SOM), cuyo objetivo principal del sistema de reconocimiento facial es obtener un modelo que sea fácil de aprender es decir, la minimización del tiempo de aprendizaje, reaccionar bien con diferentes expresiones faciales con entrada ruidosa y optimizar el reconocimiento como sea posible.

Esta técnica se ha utilizado con éxito para el reconocimiento facial. SOM es más efectiva que las técnicas PCA e ICA para base de datos dada y el clasificador utilizado.

En el sistema de reconocimiento facial, tenemos una base de datos de imágenes almacenados en el sistema. Cada vez que obtenemos una nueva imagen, se compara con la base de conocimiento de imágenes que ya está almacenada en el sistema. Las Redes Neuronales hacen uso de la nueva imagen del rostro y las imágenes de rostros almacenados para determinar si hay una coincidencia.

Kohonen describe una red asociativa con un simple algoritmo de aprendizaje que puede reconocer o clasificar imágenes del rostro y recuperar una imagen del rostro de una entrada incompleta o versión ruidosa a la red. (Turk y Pentland,1991)

2.4.2 RECONOCIMIENTO FACIAL CON SOM

Cuando se usa el mapa de auto-organización SOM como método para extracción de características en aplicaciones de reconocimiento de rostros ofrece buenas perspectivas, ya que el aprendizaje es sin supervisión, y no hay datos de imagen pre-clasificados ya que no son necesarios. A continuación se mencionan las principales etapas para llevar a cabo el reconocimiento de rostros.

Selección y Toma de Muestras

En una base de datos puede haber un sinnúmero de relaciones de valor, y tener millones de registros involucrados y miles de variables, la minería de datos inicial normalmente está restringida a muestras computacionalmente definidas de un almacén de datos completo.

El muestreo es la elección de los puntos que queremos que se represente una imagen dada. Dada una imagen analógica, un muestreo representa una asignación de la imagen a partir de un espacio continuo de puntos a un conjunto discreto. Dada una imagen digital, el muestreo representa una asignación de un conjunto discreto de puntos a otro conjunto más pequeño.

Dimensionalidad y Reducción

En la fase de reducción del proceso de reconocimiento de la cara, los conjuntos de datos se reducen al mínimo posible a través del muestreo o las estadísticas de resumen. Por ejemplo, las tablas de datos pueden estar reemplazadas por las estadísticas descriptivas como media y desviación estándar.

Extracción de Características

El proceso de extracción de características se refiere a los patrones de extracción de los datos mediante el uso de técnicas tales como la clasificación, la regresión, análisis de enlaces, la segmentación, o detección de la desviación.

Clasificador

La clasificación consiste en mapeo de datos en una de varias clases predefinidas o descubierto.

2.4.3 EL MAPA AUTO-ORGANIZADO (SOM)

El mapa de auto-organización (SOM) es un proceso de aprendizaje no supervisado, que aprende la distribución de un conjunto de patrones sin ningún tipo de información de clase. Tiene la propiedad de la topología de la preservación. Hay una competencia entre las neuronas que se activan o prenden. El resultado es que sólo una neurona que gane el concurso se dispara y se llama el ganador. SOM pueden ser unidimensional, bidimensional o multidimensional, pero los más frecuentes son los mapas unidimensionales o bidimensionales.

El número de conexiones de entrada depende del número de atributos para ser utilizados en la clasificación. La neurona con pesos cercanos al vector de datos de entrada se declara el ganador durante el entrenamiento. Entonces los pesos de todas las neuronas en la vecindad de la neurona ganadora se ajustan por una cantidad inversamente proporcional a la distancia.

Los algoritmos de aprendizaje no supervisado o mapa auto organizado son métodos muy populares para el descubrimiento de patrones significativos o características en los datos de entrada. Algoritmos como: Análisis componentes Principales (PCA), mapa auto-organizado (SOM), y Análisis de Componentes Independientes (ICA) son ampliamente utilizados para el reconocimiento de rostros. (Ghorpade, Ghorpade, Mantri, &Ghorpade, 2010).

2.5 PERCEPTRÓN MULTICAPA

El perceptrón multicapa es un tipo de red neuronal, con una variación del perceptrón simple debido a sus limitaciones, el cual puede aprender a partir de un conjunto de ejemplos y aproximar relaciones no lineales, esto lo ha hecho muy adecuado para abordar problemas de la vida real como son: reconocimientos de objetos, rostros, patrones, etc.

Aunque este tipo de red es una de las más conocidas y utilizadas, no implica que sea la mejor y que los resultados obtenidos sean los mejores, ya que posee limitaciones cuando se trata de problemas que intervengan muchas variables**.**

2.5.1 ARQUITECTURA DEL PERCEPTRÓN MULTICAPA

Está formada por tres capas básicas, las mismas que tienen agrupadas un conjunto de neuronas, las capas que la conforman son: capa de entrada, capa oculta y capa de salida.

La capa de entrada es la que se encarga de recibir los datos que se van a ingresar a la red, la capa oculta puede ser una o más, y es la que realiza el procesamiento no lineal de los datos recibidos, y la capa de salida proporciona la respuesta de la red para cada conjunto de datos ingresado.

En la Figura 6 se muestra la arquitectura del perceptrón multicapa.

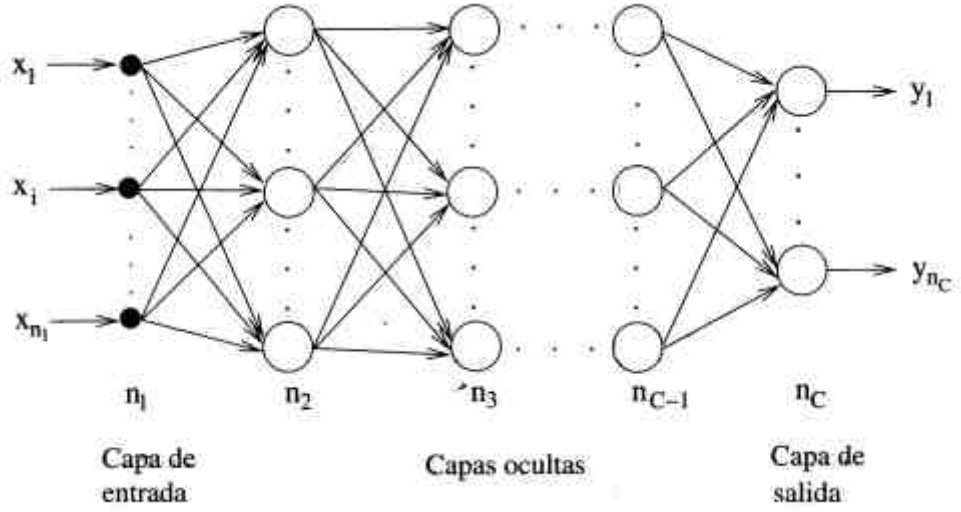

Figura 6. Arquitectura del perceptrón multicapa

(Isasi P. & Galvan M., 2004)

La activación de las neuronas de la capa de entrada está dado por la siguiente formula

$a_i^1 = X_i p \qquad i = 1,2, \ldots \ldots, n1$

Donde x= x_1, x_2,….x_{n1} que representan los valores de entrada a la red

La activación de las neuronas de la capa oculta procesa la información recibida aplicando la función de activación, a la suma de activación por sus pesos correspondientes.

La activación de las neuronas de la capa de salida, está dada por la suma de los productos de entrada por sus pesos.

Las funciones de activación más utilizadas en el perceptrón multicapa son la sigmoide y tangente, las cuales trabajan con un rango entre [0,1] o [-1,1] y están dadas por las siguientes expresiones. (Isasi P. & Galvan M., 2004)

Función Sigmoidal

$f1(x) = \frac{1}{1+e^{-x}}$

Función tangente hiperbólica

$$f(x2) = \frac{1-e^{-x}}{1+e^{-x}}$$

2.6 RECONOCIMIENTO DE ROSTROS UTILIZANDO OPENCV

2.6.1 INTRODUCCIÓN

OpenCv es una librería de visión por computador, fue creado por Intel en 1999 (Opencv devteam., 2014)**.** En el año 2008 Willow Garage se hizo cargo y OpenCv ahora cuenta con una interfaz de programación en C, C ++, Python y Android, se utiliza en proyectos académicos y productos comerciales.

A partir de la versión 2.4.2 viene con una clase FaceRecognizer para el reconocimiento de rostros, misma que cuenta con 3 algoritmos disponibles para el reconocimiento como: eigenfaces, fisherfaces, histogramas de patrones binarios (LBPH).

2.6.2 HISTOGRAMAS DE PATRONES BINARIOS

La idea de esta técnica es no tratar toda la imagen como un arreglo sino tomar las características locales de la misma. Esto se logrará haciendo una comparación de cada pixel con sus vecinos, donde se toma un pixel como centro y el umbral de sus vecinos en contra, si la intensidad del pixel vecino es igual o mayor al pixel central toma el valor de 1 caso contrario 0. Con 8 píxeles circundantes que va a terminar con 2^8 combinaciones posibles, llamados patrones binarios locales. En la Figura 7 se muestra la forma en que se trabaja con los pixeles. (Opencv devteam., 2014)

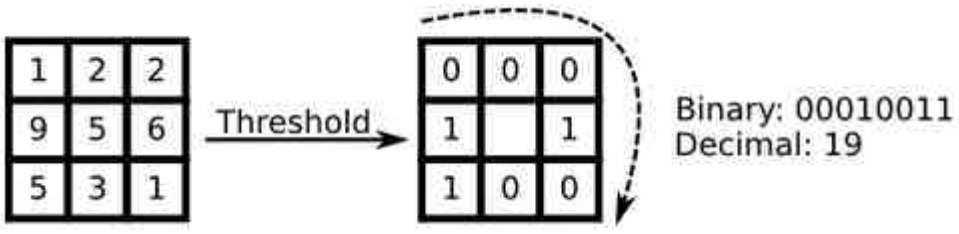

Figura 7. Pixeles algoritmo lbph

(Opencv devteam., 2014)

En la Figura 8 se puede apreciar las fotos del rostro después de aplicarle el histograma de patrones binarios.

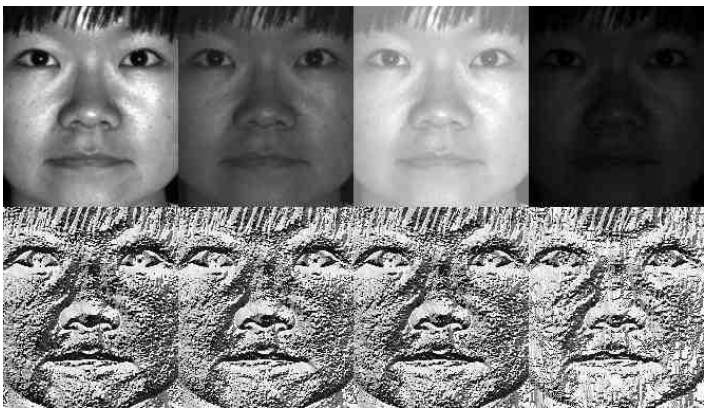

Figura 8. Imagen aplicada lbph

(Opencv devteam., 2014)

Con las técnicas y métodos descritos anteriormente se ha hecho un breve repaso de los conceptos que más se usan para los sistemas de reconocimientos de rostros, a continuación pasaremos a dar una breve introducción, arquitectura y otras características del sistema Android.

2.7 ANDROID

2.7.1 INTRODUCCIÓN

Android es un sistema operativo diseñado para dispositivos móviles inteligentes y es la primera plataforma integral completa de código abierto. Está basado en el kernel de Linux que interactúa con el hardware y software del dispositivo.

Android fue diseñado pensando en los desarrolladores y los controles de seguridad fueron creados para reducir la carga del diseño de aplicaciones.

Además todos los recursos del dispositivo con S.O. Android, como las funciones de la cámara, los datos GPS, Bluetooth, funciones de telefonía, conexiones de red, etc., se acceden a través del sistema operativo.

2.7.2 ARQUITECTURA DE CAPAS ANDROID

Está dividida en 4 capas como son la capa de pila de Android que se coloca en la parte superior de las capas Android, las mismas que son apoyados por debajo de 3 capas que incluyen al framework de aplicaciones, Android runtime, y el kernel Linux.

El Kernel Linux se utiliza como una abstracción entre el hardware y la pila de software de android. El Componente básico de ejecución en Android es la Máquina virtual dalvik, diseñado específicamente para la plataforma Android y optimizada para dispositivos móviles. Cada proceso se ejecuta en su propia instancia separada de dalvik vm.

Las aplicaciones de Android están escritas en Java, pero la ejecución de código de bytes de Java no está soportado directamente por Android por eso el sistema Android utiliza una herramienta que convierte código dx java dalvik en código de bytes comprensible.

Una aplicación de Android puede tener cuatro componentes que son actividad, servicio, receptor de radio difusión, y el proveedor de contenido. La actividad es una pantalla visual a través de las cuales el usuario interactúa con una aplicación por ejemplo, etiquetas en el menú de la parte superior hacia arriba o varias imágenes que aparecen en el formulario de caja en la pantalla del móvil, etc.

Una aplicación puede consistir en una o más de las actividades en función de su arquitectura y el diseño. Los proveedores de contenidos son normalmente Bases de datos basadas en sqlite que apoyan el intercambio y

el acceso de datos entre aplicaciones. El Desarrollador de Android también podría construir proveedor de contenido utilizando el almacenamiento de archivos Android o XML. Cada aplicación Android tiene su propio archivo AndroidManifest.xml, donde junto con un poco de otra aplicación de la información declara sus componentes.

Además de la declaración de los componentes, AndroidManifest.xml contiene también alguna meta información relacionados con la seguridad de la aplicación de políticas Android. Algunos de los recursos del sistema Android se accede a través de componentes, mientras que algunos se accede directamente a través de su API (tales como GPS, red, micrófono, etc.)(Banuri et al., 2011).

Android busca ser el sistema operativo más seguro y útil para las plataformas el mismo que pretende proteger los datos del usuario, proteger los recursos del sistema, proporcionar aislamiento de aplicaciones.

Para lograr esto Android proporciona las siguientes características de seguridad clave:

Seguridad robusta a nivel de sistema operativo a través del kernel Linux, Sandbox de aplicación obligatoria para todas las aplicaciones, comunicación segura entre procesos, firma de aplicaciones, permisos de aplicaciones definidas por el usuario. En la Figura 9 se muestra la arquitectura de Android.

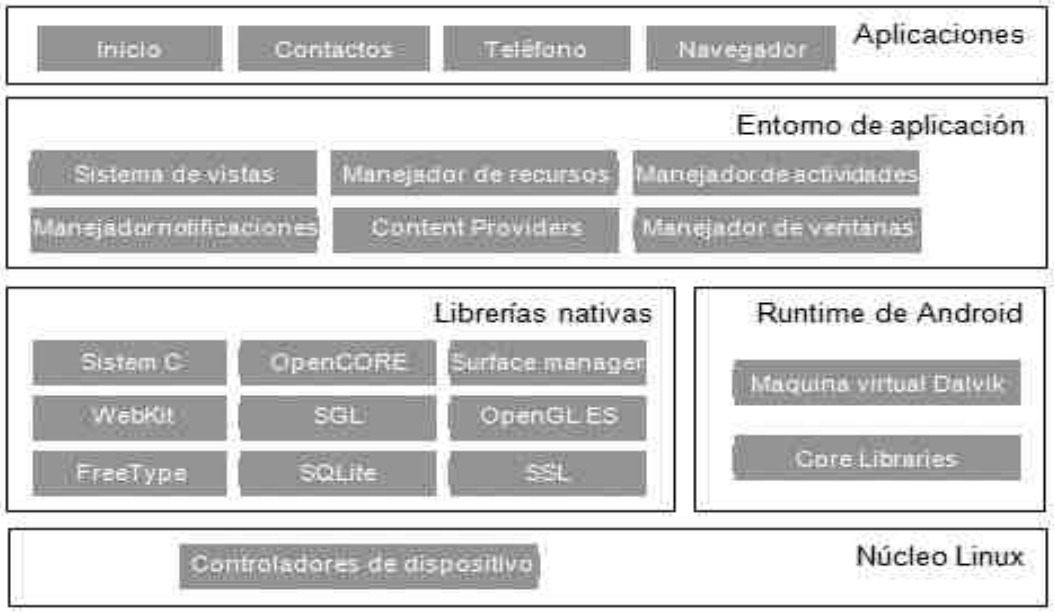

Figura 9. Arquitectura de Android

(Gironés, 2012)

El núcleo Linux

Esta capa de Android está formado por el sistema operativo Linux versión y proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, y el soporte de drivers para dispositivos. Es la única que es dependiente del hardware.

Runtime de Android

Se basa en la máquina virtual utilizado en Java. Algunas características de la máquina virtual dalvik que facilitan esta optimización de recursos son la ejecución de ficheros dalvik ejecutables (.dex), formato optimizado para ahorrar memoria. Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual dalvik.

Librerías nativas

Android trae consigo un conjunto de librerías en C/C++. Están compiladas en código nativo del procesador. Algunas de estas librerías son utilizadas en proyectos de código abierto como por ejemplo:

Media Framework, Surface Manager, WebKit, Librerías 3D, FreeType, SQLite, etc.

Entorno de aplicación

Esta capa nos brinda una plataforma de desarrollo libre para aplicaciones con grandes prestaciones e innovaciones como sensores, localización, servicios, etc. Esta capa es creada para facilitar la reutilización de componentes. Una de las ventajas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación Java. Los servicios más importantes son:

- **Vistas**: que es la parte visual de los componentes.
- **Administrador de recursos**: Brinda acceso a los recursos que no son código.
- **Administrador de actividades**: permite controlar el ciclo de vida de las aplicaciones.
- **Administrador de notificaciones**: el mismo que permite a las aplicaciones presentar alertas personalizadas en la barra de estado.
- **Proveedores de contenido**: este mecanismo sencillo para acceder a datos de otras aplicaciones, por ejemplo los contactos.

Aplicaciones

Esta capa está representada por el grupo de aplicaciones instaladas en una máquina Android. Todas las aplicaciones corren en la máquina virtual dalvik para garantizar la seguridad del sistema.

La mayoría de aplicaciones Android están escritas en Java. Las mismas que pueden ser desarrolladas utilizando el Android sdk. Para los desarrolladores que prefieran utilizar C/C++ tienen la opción de utilizar el Android ndk (Native Development Kit). (Gironés ,2012).

2.7.3 COMPONENTES DE APLICACIÓN ANDROID

Existen algunos componentes clave en el desarrollo de aplicaciones Android, a continuación se mencionan las más importantes:

Vista (View). Son los elementos que conforman la interfaz del usuario en las aplicaciones como por ejemplo un botón, una entrada de texto, etc. Estos objetos son descendientes de la clase View.

Layout. Es un conjunto de vistas asociadas de una determinada forma. Disponemos de distintos tipos de layout para organizar las vistas de forma vertical, horizontal o alguna manera personalizada. Al igual que las vistas también son descendientes de la clase View.

Actividad (Activity). Son cada una de las pantallas que conforman la aplicación, su función principal es crear la interfaz de usuario. Las actividades trabajan de forma independiente entre sí, pero trabajan para un objetivo común.

Servicio Service). Es un proceso que se ejecuta por detrás de la aplicación sin necesidad de la interacción del usuario, al igual que en Windows.

Existen dos tipos de servicios en Android, los servicios locales son utilizados por aplicaciones del mismo terminal, y servicios remotos que pueden ser utilizados desde otros terminales.

Intención (Intent). Las intenciones representan la voluntad de realizar cualquier acción. Se utilizan cuando se lanza una actividad, servicio, anuncio de tipo broadcast, comunicarnos con un servicio, intercambiar información entre componentes, etc.

Receptor de anuncios (Broadcast Receiver). Recibe y responde ante tipo de anuncios broadcast, como batería baja o llamada entrante que son propios del sistema. No tienen interfaz de usuario pero pueden iniciar una actividad para atender algún anuncio.

Proveedores de contenido (Content Provider). Se utilizan para que las aplicaciones puedan compartir datos sin comprometer la seguridad del sistema de ficheros.

2.7.4 ELEMENTOS DE UN PROYECTO ANDROID

Contiene un descriptor de la aplicación (AndroidManifest.xml), código fuente y algunos recursos, donde cada elemento se almacena en carpetas determinadas. A continuación se detalla la información que contiene cada carpeta.

Src: Contiene el código fuente de la aplicación.

Gen: Contiene el código generado automáticamente por el sdk.

Assets: Pueden contener una serie de ficheros que podrán ser utilizados por la aplicación.

Res: Contiene los recursos usados por la aplicación. También contiene subcarpetas como:

Drawable. Contiene ficheros de imágenes.

Layout. Contiene ficheros xml con las vistas de la aplicación.

Menu. Contiene ficheros del menú.

Values. Contiene ficheros xml para indicar valores de strings, color, estilos.

AndroidManifest.xml En este fichero se indican las actividades, servicios, proveedores de contenido, también se definen los permisos que requiera la aplicación.

Default.properties. Se genera automáticamente por el sdk, no hay que modificarlo, y se utiliza para comprobar la versión del api.

2.7.5 ALMACENAMIENTO DE DATOS EN ANDROID

Almacenar datos de forma permanente es muy importante en la mayoría de aplicaciones, las formas más usadas son el uso de ficheros, base de datos o a través de la red. Al igual que las aplicaciones de escritorio o web, en

Android tenemos diferentes alternativas de almacenar información, a continuación se detallan las más importantes.

Preferencias. Es un mecanismo liviano que permite almacenar datos en forma de pares clave-valor, usualmente se utiliza para almacenar parámetros de configuración. Se puede almacenar variables de tipo booleano, real, entero, string.

Las preferencias son almacenadas en ficheros xml, hay dos alternativas para acceder a las preferencias.

getSharedPreferences( ). Se utiliza cuando se manejan varios ficheros de preferencia o se requiere acceder al mismo fichero desde varias actividades, ya que permite definir el nombre del fichero.

getPreferences( ). No se tiene que indicar ningún nombre de fichero, y se usa cuando solo se necesita un fichero de preferencia.

Estos dos métodos necesitan de permisos que se quieran dar al fichero de preferencia. Los posibles valores son: mode_private, mode_world_readable o mode_world_writeable, de acuerdo al tipo de acceso que necesitemos como lectura, lectura y escritura.

Ficheros. Se pueden almacenar tanto en la memoria interna como en la tarjeta sd del dispositivo y pueden ser usados por la aplicación posteriormente.

Sistema de almacenamiento Interno. De manera predeterminada los ficheros solo son accesibles por la aplicación que los creo, no pueden ser leídos ni si quiera por el usuario del teléfono. Cada aplicación dispone de una carpeta creada en el directorio data/data/nombre_del_paquete y la ventaja es que cuando se desinstala la aplicación los ficheros creados se eliminan.

Sistema de almacenamiento externo. La mayoría de dispositivos disponen de memoria adicional conocida como almacenamiento externo. Por lo

general es una memoria extraíble como tarjeta sd, muy útil para el almacenamiento de imágenes y música.

Desde la versión 1.6 es necesario declarar el permiso `write_external_storage en el AndroidManifest.xml` para poder almacenar ficheros.

Xml. Tipo de archivo que se rige en un estándar. Utilizados ampliamente en internet y algunos archivos generados por el sdk android. Se dispone de las librerías Sax(*Java's Simple API for XML*) y Dom(*DocumentObjectModel*) para manipular estos ficheros desde Android.

Uso de xml con Sax. Se recomienda usar cuando queremos que la aplicación sea simple y se quiere reducir el consumo de memoria, muy conveniente para dispositivos móviles con Android.

Uso de xml con Dom. Permite manipular dinámicamente documentos xml y html. Dom permite cargar cualquier documento xml y manipularlo en memoria, para posteriormente almacenarlo en un fichero o mandarlo o internet.

Uno de sus inconvenientes es que carga todo el documento en la memoria y consume muchos recursos del dispositivo móvil.

Base de datos. En Android se cuenta con soporte para utilizar Sqlite, la misma que permite crear y manipular bdd de una manera muy sencilla utilizando el lenguaje sql.

Para poder manipular las bases de datos en android se utiliza la clase SQLiteopenHelper que facilita la creación de las bdd, y se encarga de abrir o crear la base si no existen, además de actualizar las versiones si queremos crear una nueva estructura.

La clase SQLiteopenHelper tiene métodos como getReadableDatabase() que abre la base en modo solo lectura y el método getWritableDatabase() que abre en modo lectura y escritura.

Si queremos gestionar la creación, apertura de las bases de datos directamente, en lugar de utilizar el SQLiteopenHelper, se puede utilizar la aplicación de Contexto como se muestra a continuación.

SQLiteDatabaseBdd = context.openOrCreateDatabase(Nombre_bdd, Context.MODE_PRIVATE,null)

Después de la creación de la base de datos se debe usar los manipuladores de Oncreate, onUpgrade de la clase SQLiteopenHelper, para modificar las tablas según sea necesario. (Meier R. ,2012)

Proveedores de contenido. Es un componente adicional de la aplicación que dispone el acceso de lectura y escritura de datos de las aplicaciones, está sujeto a las opciones de seguridad que se quieran imponer. Con los proveedores de contenido podremos acceder a datos de otras aplicaciones como la lista de contactos, registro de llamadas, etc.

Cuando se hace una consulta al proveedor de contenido nos devuelve un cursor, si le comparamos con una base de datos nos devuelve los datos en forma de una tabla, donde cada fila es un registro y cada columna un tipo de dato con un significado particular.

Para acceder a un proveedor de contenido en específico se utiliza las Uri, que son una cadena de texto para identificar un recurso de información.

Para hacer que nuestro Proveedor de contenido sea visible para otras aplicaciones hay que declararlo dentro del AndroidManifest.xml en la etiqueta <application>.

Internet. Como los dispositivos Android cuentan con conexión a internet, nos permite utilizar el internet no solo para almacenar datos, sino también compartir con otros usuarios.

Al igual que en los sistemas de escritorio, también se usa los conceptos de cliente-servidor, sockets, servicios web, etc.

2.7.6 SEGURIDAD DE APLICACIONES ANDROID

El núcleo del sistema operativo Android se basa en el kernel de Linux. Las aplicaciones Android son escritas en su mayoría en lenguaje de programación java aunque también se pueden escribir en código nativo. Las aplicaciones se instalan desde un único archivo con la extensión de archivo apk.

Modelo de permisos android y acceso protegido por apis

De manera predeterminada una aplicación Android sólo puede tener acceso a cierto número limitado de los recursos del sistema. El sistema permite el acceso de aplicaciones de Android a recursos que si no se utiliza correctamente podrían afectar a los datos del usuario, la red y los datos del dispositivo.

Las apis protegidas pueden ser: funciones de la cámara, los datos de localización (GPS), funciones bluetooth, funciones de telefonía, funciones de SMS / MMS, las conexiones de red y datos.

Todos estos recursos son accesibles únicamente a través del sistema operativo, para hacer el uso de la apis protegidas el usuario debe definir en el archivo AndroidManifest.xml para cuando el usuario se disponga a instalar alguna aplicación, se le muestre un cuadro donde pueda conceder los permisos de la misma, el usuario deberá aceptar todos los permisos en conjunto ya que no tiene la opción de elegir los permisos individuales**.** (Android Open Source Project, 2013)**.**

Limitaciones de Android

A continuación se detallan un par de limitaciones de Android en la parte de seguridad y la política de control de acceso.

1. Android carece de un mecanismo de personalización de permisos y tiene un modelo de todo o nada de aplicación de las políticas. Esto significa que el usuario tiene que aceptar todos los permisos requeridos por una aplicación

durante la instalación para poder lograr una instalación exitosa, ya que el usuario no tiene la opción de otorgar o negar ciertos permisos solicitados por una determinada aplicación.

2. Otra limitación de Android es que no investiga el comportamiento de ejecución de las aplicaciones para evitar su mal funcionamiento, garantizar la integridad del sistema y la protección de los datos del usuario. (Banuri et al., 2011).

2.8 MODELO DE PROTOTIPOS

Al desarrollar un sistema o programa, el cliente detalla los requisitos de entrada, proceso y salida que espera, pero de manera muy general. En otros casos el programador no puede estar completamente seguro de la eficacia del algoritmo. En estos y muchos casos el modelo de prototipos es uno de los más adecuados ya que permite la interacción con el usuario para refinar los requisitos.

A continuación en la Figura 10 se muestran los pasos en el modelo de construcción de prototipos.

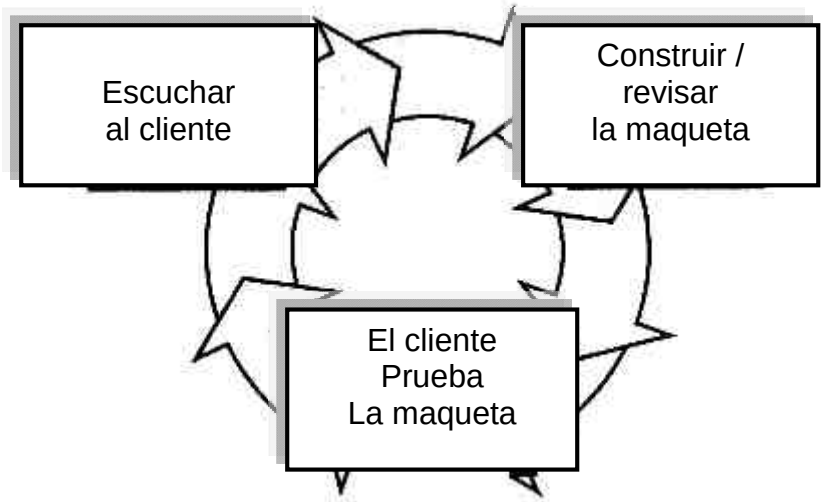

Figura 10. Modelo construcción prototipos

(Pressman, 2002)

Escuchar al Cliente

En esta etapa se realiza una reunión entre el desarrollador y el cliente, donde se definen los objetivos y especificaciones para llevar a cabo el desarrollo del proyecto.

Construir la maqueta o prototipo

En esta etapa se realiza una representación de los aspectos del software que serán visibles para el usuario o cliente, como la interfaz de usuario, formularios, etc. Estos aspectos sirven para la construcción del prototipo.

El cliente prueba la maqueta

El cliente evalúa el prototipo para refinar los requerimientos del software, permitiendo que el desarrollador logre satisfacer las necesidades del usuario.

Ventajas

- Hace el software amigable al usuario.
- Reducción de tiempo y de costos
- El método de prototipos nos da la ventaja de que el cliente o usuario final pueda ir observando cada una de las etapas o fases durante el desarrollo y pueda ir dando sus opiniones para la mejoras del mismo antes de la fase final.
- Mejor orientado cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, ya sea la adaptabilidad del sistema operativo o de la forma que debería interactuar el usuario con la aplicación.

Desventajas

- El usuario quiere que el programa funcione totalmente desde el inicio para solucionar sus problemas, sin entender que el prototipo no es el producto final.
- Cuando se van a realizar los ajustes finales al prototipo para que funcione correctamente, el usuario no lo entiende y quiere que solo se cambien pequeños detalles para utilizar el prototipo indefinidamente. (Pressman, 2002).

2.9 LENGUAJE UNIFICADO DE MODELADO (UML)

El lenguaje unificado de modelado (UML, Unified Modeling Language), se utiliza para especificar, visualizar, construir y documentar los diferentes procesos de un sistema. Se lo puede utilizar con todos los métodos de desarrollo y las etapas del ciclo de vida del sistema, además permite agrupar los modelos en paquetes, permitiendo dividir grandes sistemas en pequeñas piezas para trabajar de mejor manera.

El lenguaje unificado de modelado también permite representar las decisiones de implementación y organizar elementos de tiempo de ejecución en componentes. (Rumbaugh, Jacobson & Booch, 2007).

METODOLOGÍA

3 METODOLOGÍA

La metodología elegida para el desarrollo de la aplicación de reconocimiento de rostros se basa en el método orientado a prototipos, ya que se ajusta mejor para este tipo de sistema.

3.1 ALCANCE

Esta aplicación realizará el reconocimiento facial utilizando la cámara del dispositivo móvil, el rostro del usuario debe estar previamente registrado y guardado en una base de conocimiento, luego se procederá a comparar el rostro del usuario que requiera el acceso al dispositivo móvil con la base de conocimiento antes descrita, si el sistema lo reconoce le otorgará el acceso al uso de su dispositivo, caso contrario le mostrará un mensaje no identificado. El usuario también podrá elegir que aplicaciones de su celular desea bloquear con este sistema de detección de rostros, y además tendrá la opción de añadir una clave adicional para ingresar al dispositivo en caso de que por algún motivo su rostro no pueda ser reconocido.

La funcionalidad de este sistema dependerá de algunos factores externos como son la iluminación e intensidad de luz que se presente tanto al momento de registrar el rostro como cuando el usuario vaya a autentificarse.

3.2 HERRAMIENTAS Y TÉCNICAS

La elaboración de este sistema se lo va a realizar en un computador con sistema operativo Windows 7, 4 GB de memoria RAM, y procesador Intel core i5 de 2.2 GHz. En este equipo se instalará el software necesario para la elaboración del sistema de reconocimiento de rostros, como por ejemplo Android Studio, Eclipse Android u openCV, ya que permiten la interactividad con librerías de desarrollo para Android.

El dispositivo que se va a utilizar para las pruebas e implementación del sistema es el Samsumg Transform Ultra, o algún otro dispositivo móvil que cumpla con un mínimo de características, por ejemplo 512 MB de memoria RAM, un procesador de 1.0 GHz, una cámara frontal de 0.3 megapixels VGA, y que cuente por lo menos con la versión 2.3 de Android la misma que puede ser actualizada de acuerdo a las necesidades del sistema.

Este sistema trabajará en conjunto con algunas técnicas como son: inteligencia artificial, procesamiento de imágenes, y otras que permitan la comunicación con el sistema operativo Android, las mismas que en conjunto ayudarán a cumplir con el reconocimiento de rostro.

3.3 MÉTODOS

El método que se eligió para la llevar a cabo la investigación de esta tesis es el sistémico, ya que permite extraer los rasgos fundamentales del sistema como componentes, reglas, interrelaciones que van a ser parte de la aplicación.

Para el desarrollo de la aplicación de reconocimiento de rostros se adoptó el método basado en prototipos, el mismo que beneficia mucho al sistema porque se puede ir creando por fases (prototipos), lo que permitirá ir comprobando el funcionamiento y al mismo tiempo realizar los ajustes o mejoras necesarias a cada una de ellas. A continuación se detallan las actividades a seguir en cada etapa.

Escuchar al Cliente

En esta etapa se definen las especificaciones para poder llevar a cabo el desarrollo del sistema como: requisitos funcionales mediante casos de uso, requisitos no funcionales, factibilidad del sistema, factibilidad económica, requisitos de hardware y software.

Construir la maqueta o prototipo

En esta etapa se realiza el diseño e implementación de la aplicación para cumplir las especificaciones y requerimientos funcionales de la aplicación. La misma que deberá contemplar modificaciones futuras y deberá permitir añadir futuras funcionalidades. Para construcción de la aplicación de reconocimiento de rostros se realiza una combinación del perceptrón multicapa y el algoritmo de histogramas de patrones binarios.

Los pasos a seguir son: diseño de diagramas de clases, interfaces, y flujos de navegación.

El cliente prueba la maqueta

En esta etapa se lleva a cabo las pruebas pertinentes, teniendo en cuenta la funcionalidad de la aplicación y confirmando que se cumplan los requisitos definidos.

ANÁLISIS DE RESULTADOS

4 ANÁLISIS DE RESULTADOS

Durante esta fase se detallan los resultados obtenidos durante la investigación realizada para el desarrollo de este prototipo.

4.1 ESCUCHAR AL CLIENTE

En esta etapa se detallan todos los requisitos, requerimientos y funcionalidades con las que contará este sistema, para poder desarrollar una aplicación que sea de fácil uso y posea un diseño agradable para el usuario.

4.1.1 REQUISITOS FUNCIONALES

Para el desarrollo de esta aplicación se propone 4 fases. La primera fase se realizará la captura de la imagen a través de la cámara del dispositivo móvil y después se procesa la imagen para mejorar la calidad.

Para la segunda fase se agregará la función que permita entrenar todas las imágenes del rostro del usuario, previamente tomadas y pre procesadas.

La tercera fase es la creación de un prototipo el cual permita autentificar el rostro del usuario que desea ingresar al sistema y comparar con la base de conocimiento anteriormente mencionada.

La última fase se propone unir los prototipos anteriormente descritos y crear la lista de aplicaciones a bloquear, las mismas que si el usuario ha sido identificado se le permitirá el acceso.

4.1.2 CASOS DE USO DEL SISTEMA

Caso de uso general del sistema

En la Figura 11 se muestra el caso de uso general del sistema.

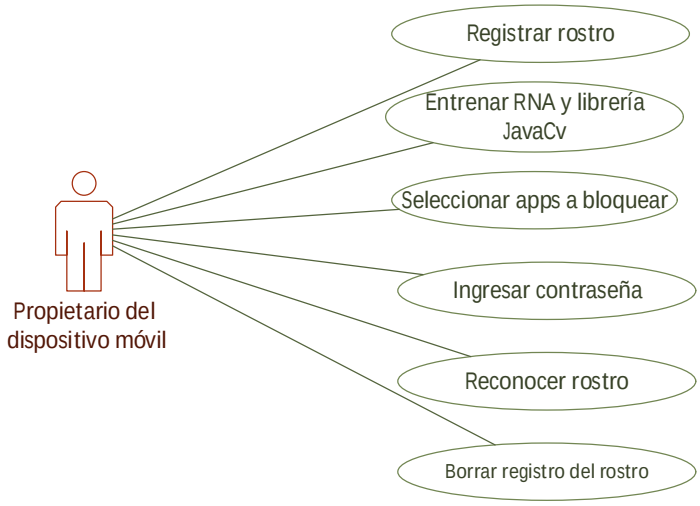

Figura 11. Caso de uso general del sistema

Este sistema solo maneja un usuario, por ende el propietario debe ingresar los datos por medio del registro del rostro, entrenar la red con los datos ingresados, para llevar a cabo el reconocimiento.

Caso de uso registrar rostro

En la Figura 12 se muestra el caso de uso del registrar rostro.

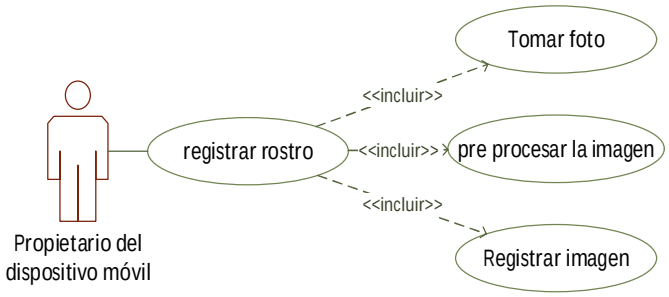

Figura 12. Caso de uso registrar rostro

Descripción caso de uso registrar rostro

A continuación en la Tabla 1 se realiza la descripción del caso de uso registrar rostro.

Nombre del caso de uso	Registrar rostro	
Descripción	Se realiza la captura de la foto del rostro, se pre procesa y registra la imagen del rostro.	
Actor	El propietario del dispositivo móvil.	
Flujo Básico		
Paso	**Actor**	**Sistema**
1	Ingresar a la aplicación.	Mostrar menú principal
2	Tomar foto del rostro	Pre procesar y registrar las fotos ingresadas.
Post condición	Entrenar RNA y librería JavaCv.	

Tabla 1. Registrar rostro

Diagrama de secuencia registrar rostro

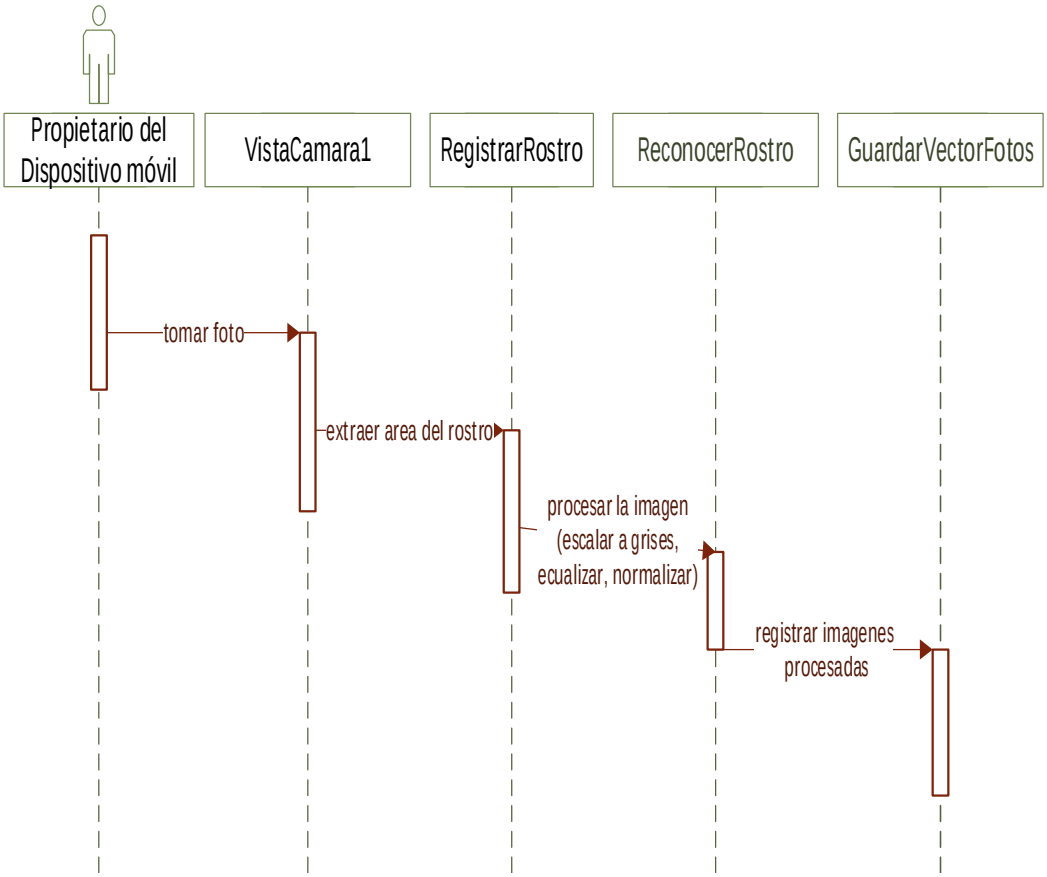

Figura 13. Diagrama de secuencia registrar rostro

En la Figura 13 se muestra el diagrama de secuencia registrar rostro, donde se observa que el usuario realiza el ingreso de la imagen del rostro, tomando una foto con la cámara frontal del dispositivo, la misma que deberá ser pre procesada y almacenada para su posterior uso.

Caso de uso Entrenar RNA y librería JavaCv

En la Figura 14 se muestra el caso de uso Entrenar RNA y librería JavaCv.

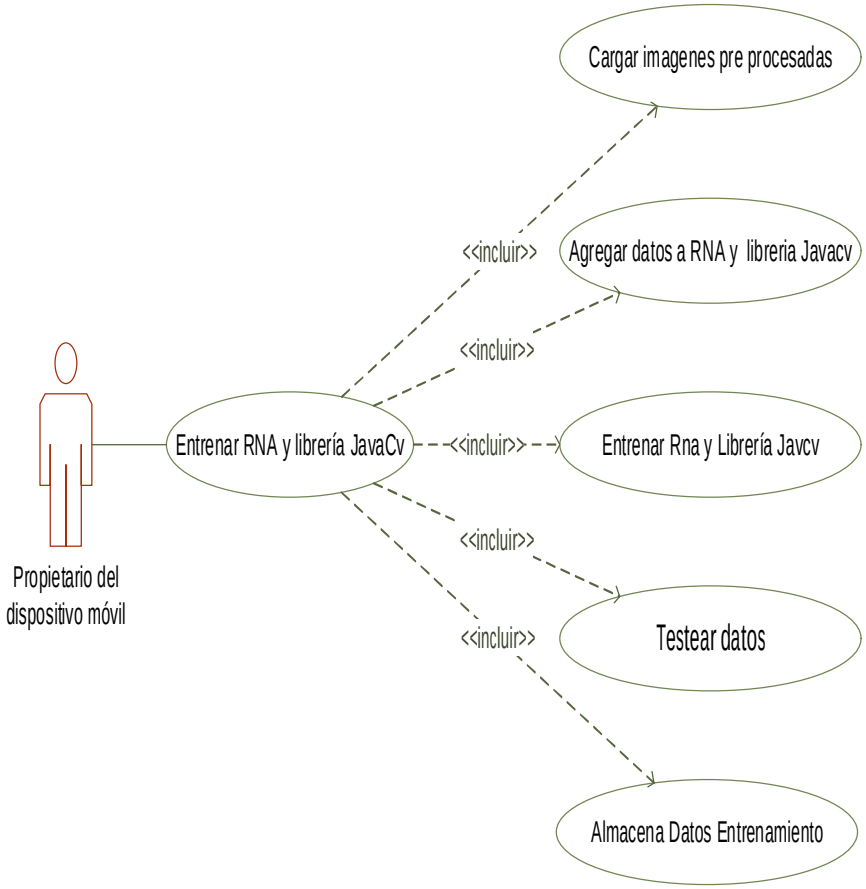

Figura 14. Caso de uso Entrenar RNA y librería JavaCv.

Descripción del caso de uso Entrenar RNA y librería JavaCv

En la Tabla 2 se observa la descripción del caso de uso Entrenar RNA y librería JavaCv.

Nombre del caso de uso	Entrenar RNA y librería JavaCv.	
Descripción	Se ingresa las imágenes del rostro pre procesadas a la RNA y librería JavaCv para su entrenamiento.	
Actor	El propietario del dispositivo móvil.	
Flujo Básico		
Paso	**Actor**	**Sistema**
1	Escoger la opción Entrenar	Cargar imágenes pre procesadas, entrenar RNA y librería JavaCv, guardar entrenamiento.
Precondición	Registrar rostro	

Tabla 2. Entrenar RNA y librería Javacv

Diagrama de secuencia entrenar RNA y librería JavaCv

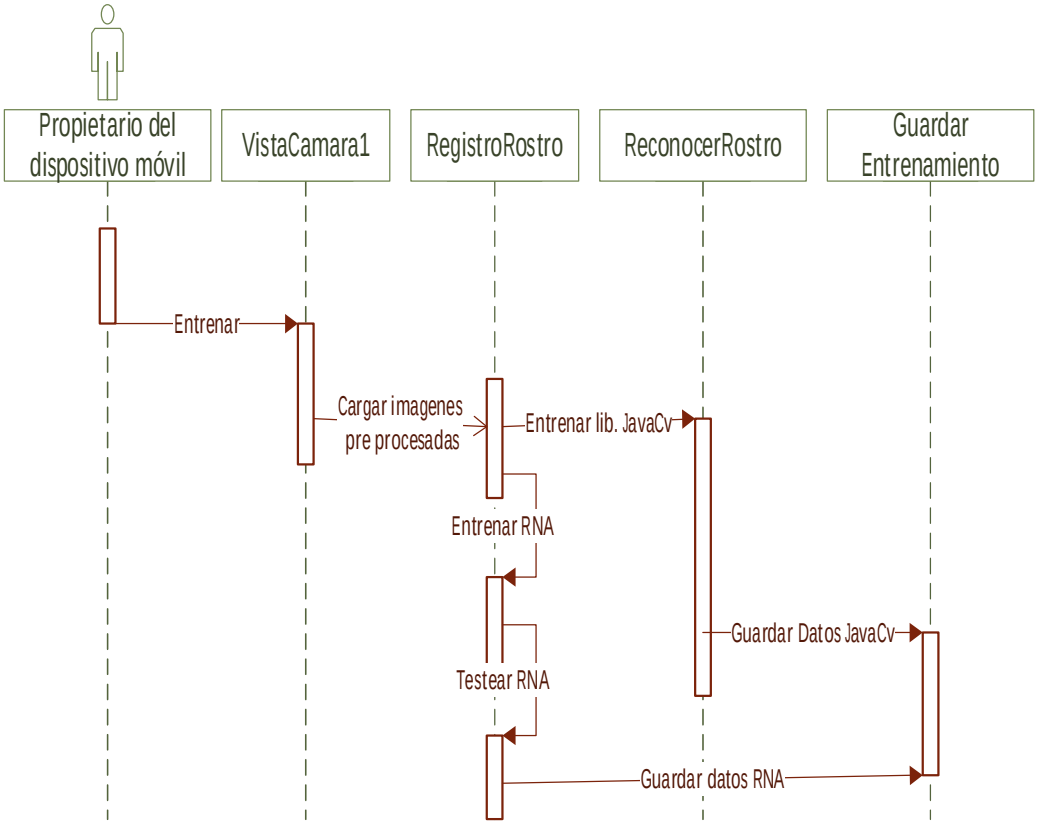

Figura 15. Diagrama de secuencia entrenamiento red neuronal y librería JavaCv

Una vez realizado el preprocesamiento de las imágenes del rostro, se procede a realizar el entrenamiento de las librerías JavaCv y Neuroph. Después se procederá a guardar el archivo de entrenamiento para usarlo en el reconocimiento de rostro tal como se muestra en la Figura 15.

Caso de uso seleccionar apps a bloquear

En la Figura 16 se muestra el caso de uso seleccionar apps (aplicaciones) a bloquear.

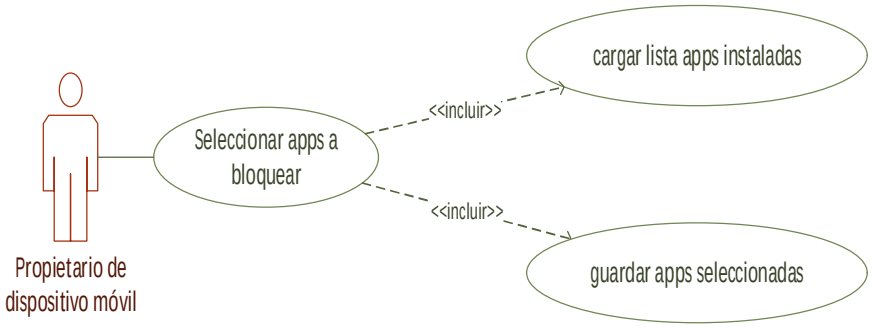

Figura 16. Caso de uso seleccionar apps a bloquear

Descripción caso de uso seleccionar apps a bloquear

En la Tabla 3 se realiza la descripción del caso de uso Seleccionar apps a bloquear.

Nombre del caso de uso		Seleccionar apps a bloquear
Descripción		Se carga la lista de aplicaciones instaladas en el dispositivo móvil, luego se escoge las apps a bloquear.
Actor		El propietario del dispositivo móvil.
Flujo básico		
Paso	**Actor**	**Sistema**
1	Ingresar a lista de aplicaciones	Mostrar lita de aplicaciones.
2	Seleccionar aplicaciones a bloquear	Guardar apps seleccionadas.
Precondición		Registrar rostro

Tabla 3. Seleccionar apps a bloquear

Diagrama de secuencia seleccionar apps a bloquear

Se muestra la lista de aplicaciones instaladas en el dispositivo móvil y se escoge las apps que se desee bloquear como se observa en la Figura 17.

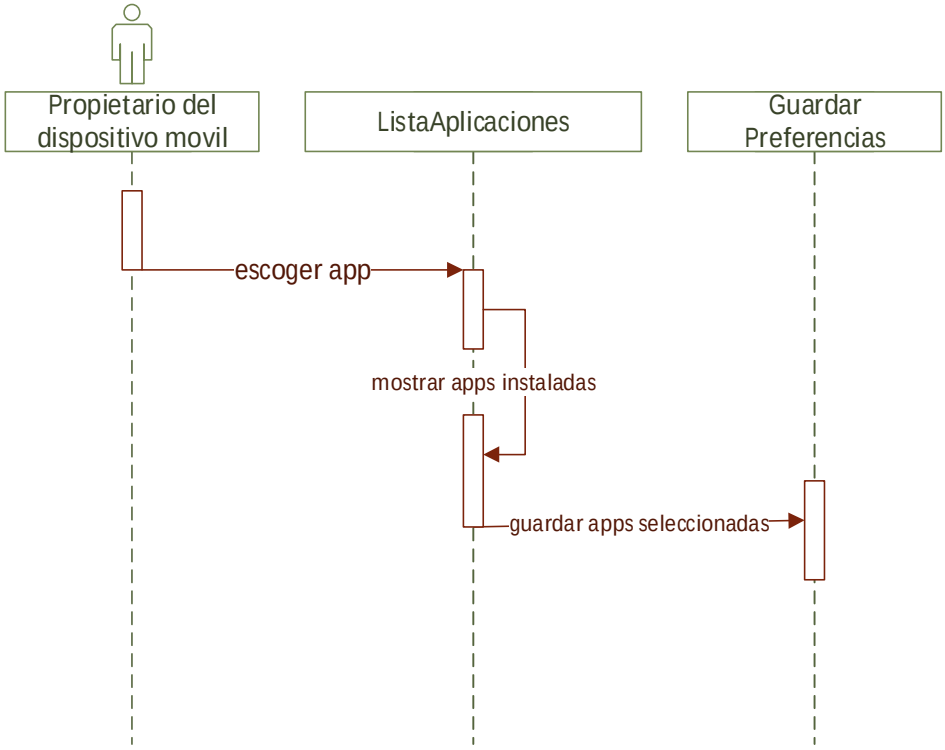

Figura 17. Diagrama de secuencia seleccionar apps a bloquear

Caso de uso reconocer rostro

En la Figura 18 se muestra el caso de uso reconocer rostro.

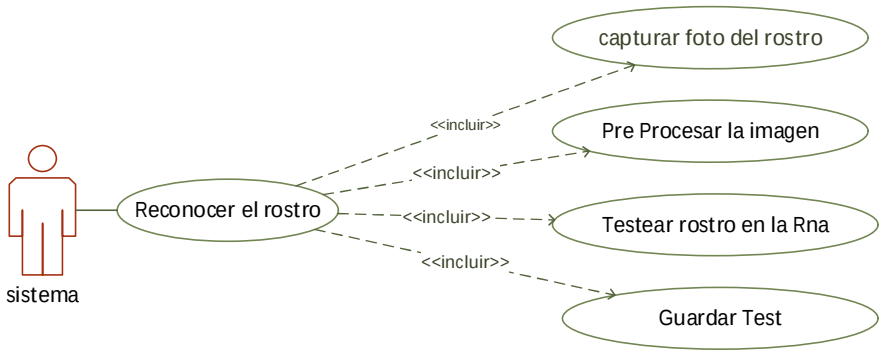

Figura 18. Caso de uso reconocer rostro

Descripción caso de uso reconocer rostro

En la Tabla 4 se realiza la descripción del caso de uso reconocer rostro.

Nombre del caso de uso		reconocer rostro
Descripción		EL sistema captura una foto del rostro, después procesa la imagen y procede a testear para realizar el reconocimiento.
Actor		El sistema
Flujo básico		
Paso	**Actor**	**Sistema**
1	Colocar rostro frente a la cámara del dispositivo móvil.	Capturar foto del rostro.
1.1		Pre procesar la imagen, testear imagen en RNA y librería Javacv, realizar el reconocimiento.
precondición		Registrar rostro, entrenar RNA y Liberia Javacv, seleccionar apps a bloquear.

Tabla 4. Reconocer rostro

Diagrama de secuencia reconocer rostro

Una vez capturada la imagen del rostro a ser reconocida se procederá a realizar un test para verificar en las librerías JavaCv y Neuroph, como se muestra en la Figura 19.

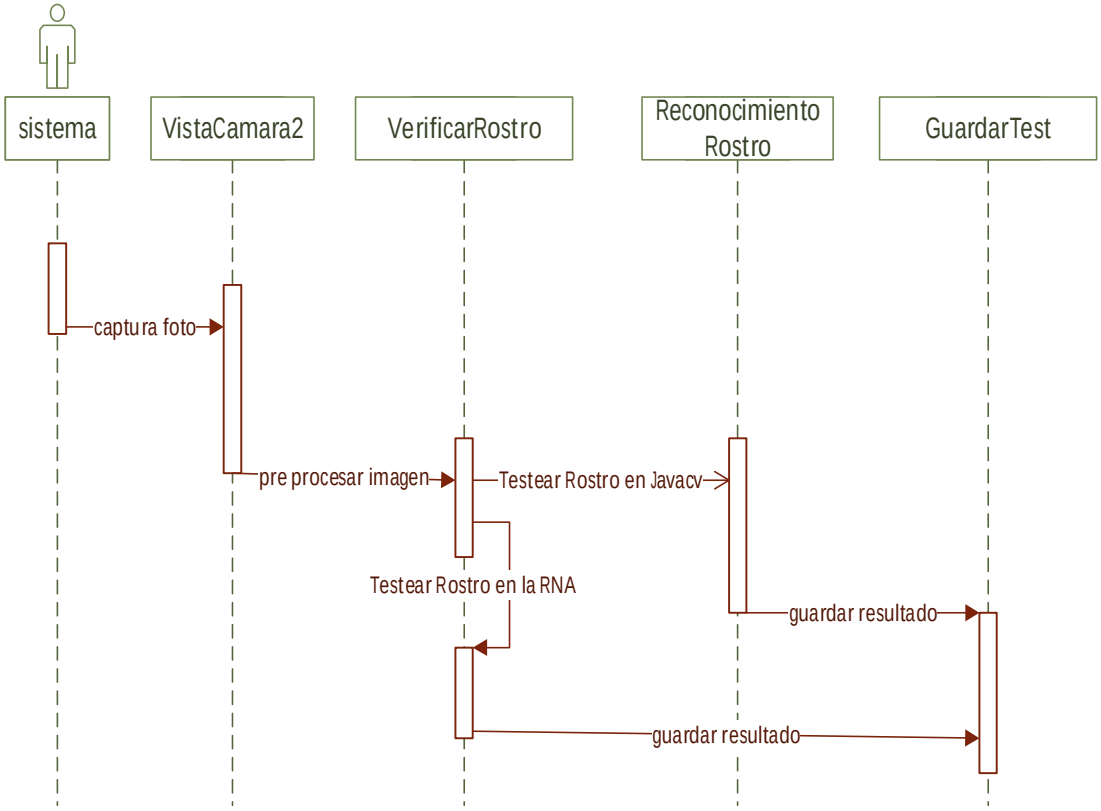

Figura 19. Diagrama de secuencia reconocer rostro

Caso de uso ingresar contraseña

En la Figura 20 se muestra el caso de uso ingresar contraseña.

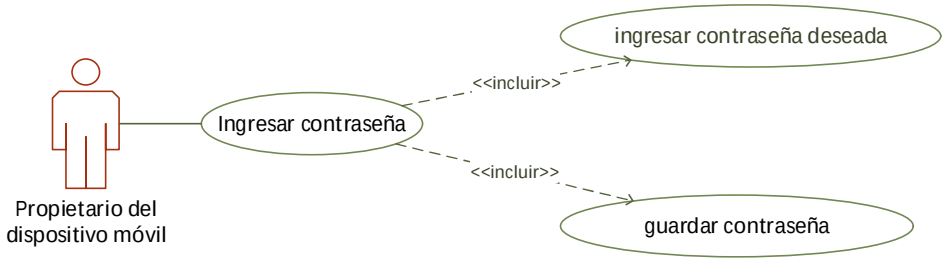

Figura 20. Caso de uso ingresar contraseña

Descripción del caso de uso ingresar contraseña

En la Tabla 5 se realiza la descripción del caso de uso ingresar contraseña.

Nombre del caso de uso		Ingresar contraseña
Descripción		Se ingresa y guarda la contraseña deseada para su posterior uso.
Actor		Propietario del dispositivo móvil.
Flujo básico		
Paso	**Actor**	**Sistema**
1	Ingresar la contraseña	Almacenar la contraseña ingresada.

Tabla 5. Ingresar contraseña

Diagrama de secuencia ingresar contraseña

El propietario del dispositivo móvil ingresa la contraseña y se almacena para su posterior uso como se muestra en la Figura 21.

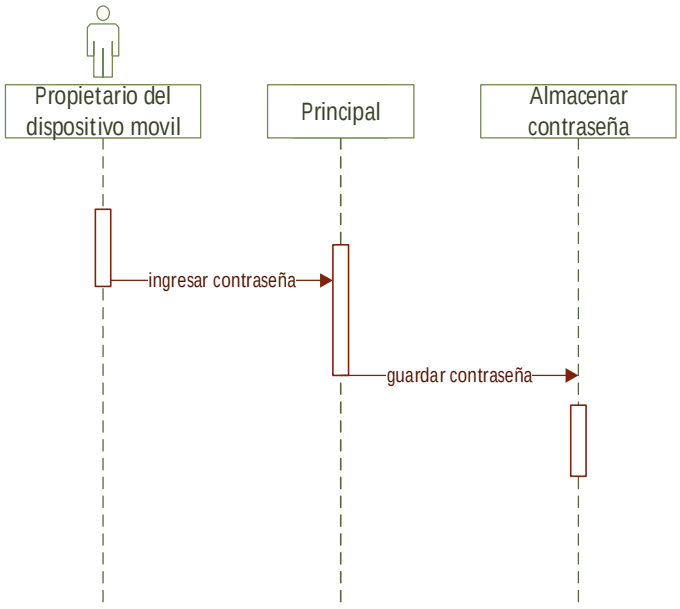

Figura 21. Diagrama de secuencia ingresar contraseña

Caso de uso eliminar registro del rostro

En la Figura 22 se muestra el caso de uso eliminar registro del rostro.

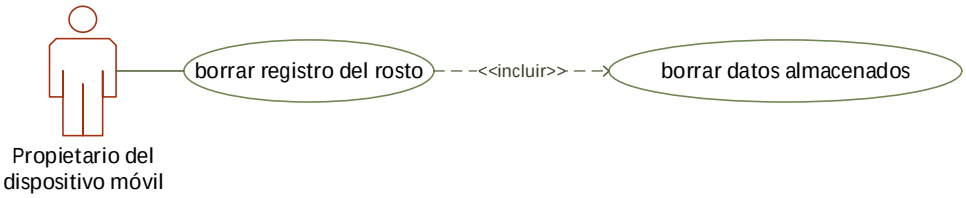

Figura 22. Caso de uso eliminar registro del rostro

Descripción del caso de uso eliminar registro del rostro

En la Tabla 6 se realiza la descripción del caso de uso eliminar registro del rostro.

Nombre del caso de uso	Eliminar registro del rostro.	
Descripción	Se elimina los datos del registro del rostro.	
Actor	Propietario del dispositivo móvil.	
Flujo básico		
Paso	**Actor**	**Sistema**
1	Seleccionar la opción borrar registro del rostro.	Eliminar datos almacenados.

Tabla 6. Eliminar registro del rostro

Diagrama de secuencia eliminar registro del rostro

El propietario del dispositivo móvil borra los datos del registro del rostro como se muestra en la Figura 23.

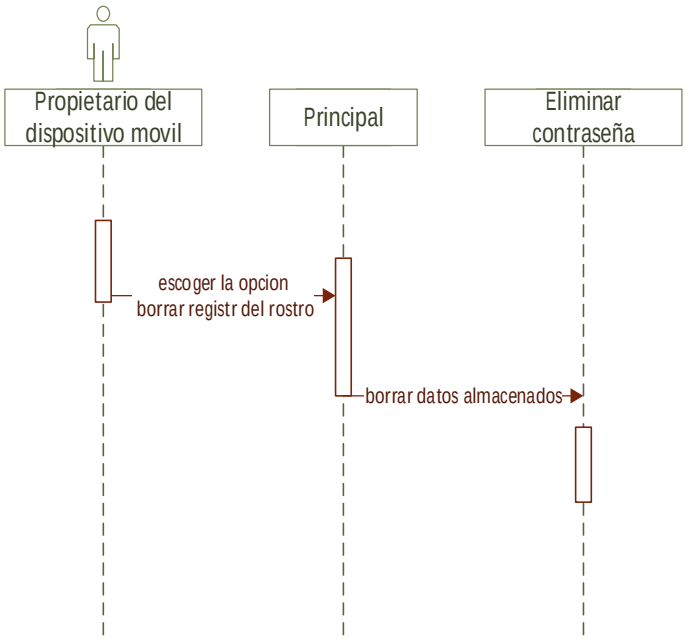

Figura 23. Diagrama de secuencia eliminar registro del rostro

4.1.3 REQUISITOS NO FUNCIONALES

Portabilidad

Este sistema de reconocimiento de rostros podrá ser instalado en cualquier versión de android a partir de la 2.3.6, ya que al ser instalado en una versión menor no permitirá usar todas sus funcionalidades, como la detección del rostro.

Disponibilidad

Estará disponible todo el tiempo a partir de la instalación, ya que será instalada como una aplicación nativa y además contara con un servicio disponible el 100% del tiempo para un correcto funcionamiento del sistema.

Requisitos de hardware

Los requisitos de hardware para un correcto funcionamiento son:

Cámara frontal. El dispositivo deberá contar con una cámara frontal ya que esta permitirá realizar el registro y reconocimiento del rostro.

Requisitos de Software

Los requisitos de software para un correcto funcionamiento son:

Sistema Operativo: Android 2.3.6 como mínimo.

OpencvManager 2.4.X : Podrá ser descargado desde google play store.

4.1.4 FACTIBILIDAD DEL SISTEMA

Factibilidad técnica

Se cuenta con los recursos informáticos y computacionales disponibles, tanto en software como hardware. En la Tabla 7 se detallan los aspectos relevantes de la factibilidad del sistema.

Descripción	**No**	**Si**
Experiencia en elaboración de sistemas		**x**
Experiencia en sistemas para dispositivos con s.o. Android		**x**
Experiencia con el método de prototipos.	**x**	
Experiencia en la elaboración de sistemas con Redes Neuronales	**x**	
Experiencia en manejo de librerías opencv y javacv	**x**	
Equipos necesarios		**x**

Tabla 7. Factibilidad técnica

Cabe señalar que se tiene conocimientos teóricos en el uso de redes neuronales, librerías Opencv y JavaCv, como en la elaboración de sistemas para Android básicos, pero no la experiencia en la creación de los mismos.

Factibilidad Económica

La inversión a realizar para el desarrollo de este no es mayor ya que se cuenta con los recursos de hardware y software necesario, además del recurso humano.

La elaboración del sistema se lleva a cabo con una persona, la cual está encargada de escuchar al cliente, construir y probar el prototipo. En la Tabla 8 se muestra el presupuesto general para el desarrollo del sistema**.**

Presupuesto		
Rubro	US$	Fuente financiamiento
Papel	50	Propio
Dispositivo móvil	300	Propio
Cd	10	Propio
USB	16	Propio
Lapiceros	5	Propio
Internet	80	Propio
Luz	50	Propio
Total:	511$	

Tabla 8. Factibilidad económica

4.2 CONSTRUIR LA MAQUETA O PROTOTIPO

Luego de haber definido los requisitos y funcionalidades de la aplicación, se procede con el desarrollo de la misma.

4.2.1 DIAGRAMAS DE CLASES Y CÓDIGO FUENTE

En la Figura 24 se muestra el diagrama de clases de la aplicación de reconocimiento de rostros, mismas que intervienen en el proceso de registro del rostro, entrenamiento de la red neuronal y librería javacv, ingreso de contraseña, lista de aplicaciones y verificación del rostro.

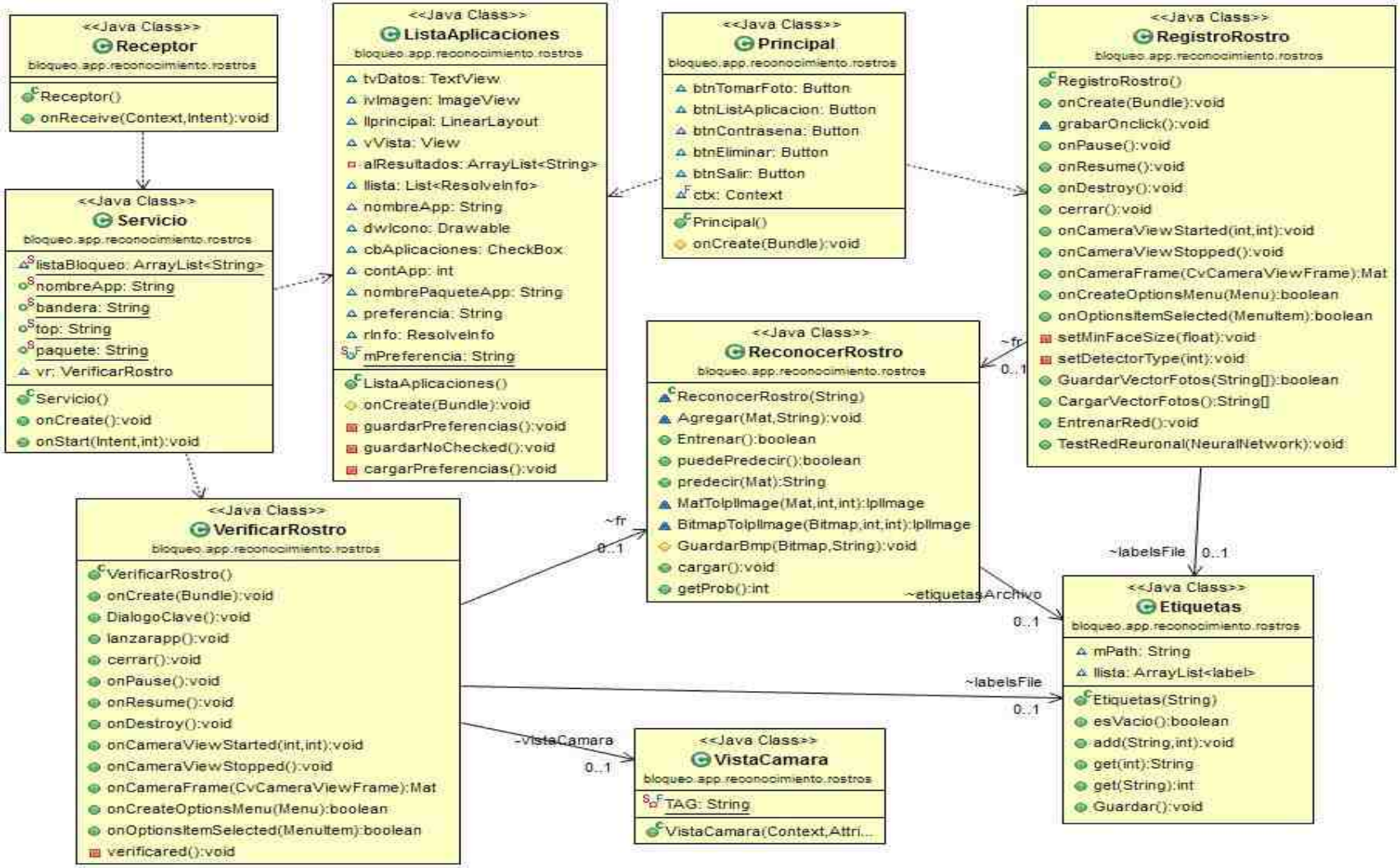

Figura 24. Diagrama de clases general

Diseño de interfaz del registrar rostro

En la Figura 25 se muestra el maquetado de la interfaz registrar rostro.

Figura 25. Diseño de la interfaz registrar rostro

Flujo de navegación registrar rostro

En la Figura 26 se puede apreciar la navegación entre el menú principal y la opción de registrar rostro. Las fechas indican la navegación entre las diferentes ventanas para llevar a cabo el registro y entrenamiento de la aplicación con las fotos del rostro.

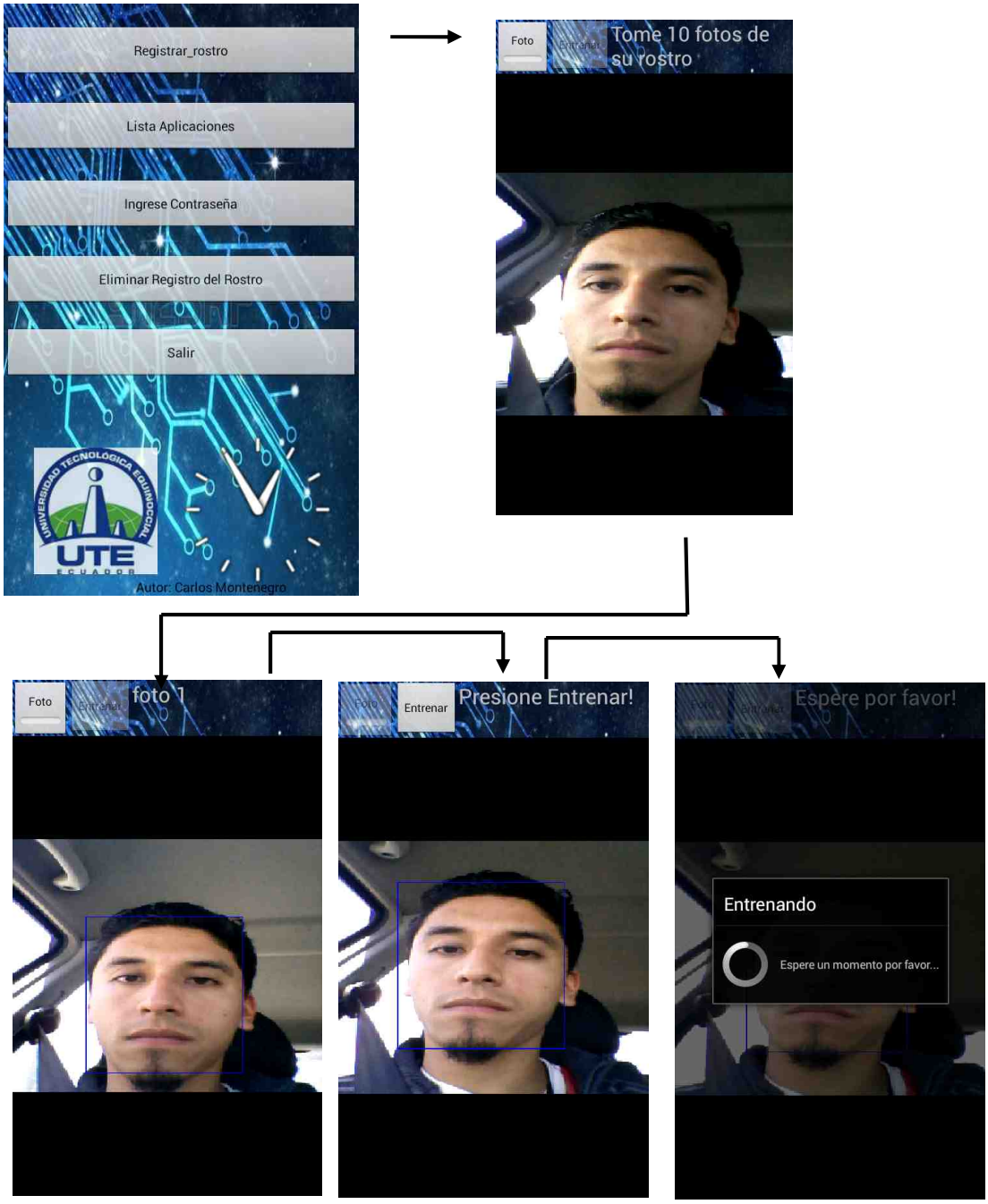

Figura 26. Flujo de navegación de registro y entrenamiento de la aplicación

Código fuente de la clase Registrar Rostro

A continuación se muestra el código fuente de los métodos y botones principales de la clase Registro Rostro.

Botón Foto

```
BtnFoto.setOnClickListener(newView.OnClickListener() {

	@Override
		publicvoidonClick(View v) {
			// TODO Auto-generated method stub
			grabarOnclick();

			tvMensaje.setText("foto "+(contadorf+1));

			if(contadorf>=9){

				GuardarVectorFotos(fr.datosred);
				btnEntrenar.setEnabled(true);
				BtnFoto.setEnabled(false);
				tvMensaje.setText("Presione Entrenar!");
			}
		contadorf++;

			}
});
```

El botón foto permite tomar y grabar 10 fotos del rostro mediante la cámara frontal del dispositivo, para luego ser utilizadas para el entrenamiento. Además activa el botón de entrenamiento.

Botón Entrenar

```
btnEntrenar.setOnClickListener(newView.OnClickListener() {

@Override
publicvoidonClick(View arg0) {
	// TODO Auto-generatedmethodstub
	fr.Entrenar();

	tvMensaje.setText("Espere por favor!");

ProgressDialog progreso = ProgressDialog.show(RegistroRostro.this,
"Entrenando", "Espere un momento   por favor...", true);

progreso.setCancelable(true);

	Runnable taskre = new Runnable() {
		publicvoid run() {

			EntrenarRed();

			}
	};
	Thread threadr = new Thread(newThreadGroup("Entrenamiento"),
                                          taskre, "train", 999999999);
```

```
        threadr.start();
        //cerrar();
        btnEntrenar.setEnabled(false);
         }
});
```

El botón entrenar permite realizar el entrenamiento de la red neuronal y la librería javacv. Aquí se llama al método EntrenarRed

Método EntrenarRed

```
publicvoidEntrenarRed() {

        String []tem=CargarVectorFotos();
        //String []tem=fr.datosred;
        inttpixel=pixel1*pixel2;

TrainingSet<SupervisedTrainingElement>trainingSet = new
TrainingSet<SupervisedTrainingElement>(tpixel, 2);

int j=0;
int k=tpixel;
for(inti=0;i<10;i++){
        double []momentshud= newdouble[tpixel];

        for ( j = (k-tpixel); j < k; j++){
                int h=0;
                momentshud[h] = Double.valueOf(tem[j]);
                  h++;
          }

        trainingSet.addElement(newSupervisedTrainingElement(momentshud,
newdouble[]{1,0}));

}
intentradas = tpixel;
        int salidas = 2;
        intCapaOculta = 30;

        MultiLayerPerceptronmlPerceptron = newMultiLayerPerceptron
       (TransferFunctionType.SIGMOID, entradas, Capaoculta, salidas);

MomentumBackpropagationlearningRule = (MomentumBackpropagation)
mlPerceptron.getLearningRule();
learningRule.setLearningRate(0.001);
learningRule.setMomentum(0.7);

        mlPerceptron.learn(trainingSet);

        System.out.println("Testeando red neuronal");

        TestRedReuronal(mlPerceptron);
```

```
        File dir = new File(Environment.getExternalStorageDirectory() +
"/Android/data/nnet/");
       dir.mkdirs();

       mlPerceptron.save(rutaRed);

}
```

Este método sirve para cargar los datos de entrenamiento de la red neuronal, una vez terminado graba el entrenamiento para su posterior uso.

Método Test Red Neuronal

```
Publicvoid TestRedReuronal(NeuralNetworknnet) {

       String []tem=CargarVectorFotos();
       //pasar vector string a double
inttopix=pixel1*pixel2;
intpo=9*topix;
       //for(inti=0;i<3;i++){
       double []imagenTest= newdouble[topix];
for (int j = 0; j <topix; j++)
      {
       imagenTest [j] = Double.valueOf(tem[po+j]);
      }

nnet.setInput(imagenTest) ;
nnet.calculate();
networkOutput = nnet.getOutput()[0];

System.out.println(" Output: " + networkOutput );

       SharedPreferencesvalorred =
getSharedPreferences("valored",Context.MODE_PRIVATE);
Editor editorn = valorred.edit();
editorn.putLong("red", (long) networkOutput);
editorn.commit();

runOnUiThread(new Runnable() {
                    publicvoid run() {

       Toast.makeText(getApplicationContext(), "entrenamientoCorrecto
"+networkOutput ,Toast.LENGTH_SHORT).show();
                    }
       });


    }
```

Este método se encarga de realizar el test de la red neuronal, para poder saber su grado de eficiencia.

Código fuente de la clase Reconocer Rostro

Método Entrenar

```
Publicboolean Entrenar() {

      File root = new File(mPath);

FilenameFilterpngFilter = newFilenameFilter() {
@SuppressLint("DefaultLocale")
                  publicboolean accept(File dir, String name) {
returnname.toLowerCase().endsWith(".jpg");

        };
        };

        File[] archivoImagenes = root.listFiles(pngFilter);
MatVectorimagenes = newMatVector(archivoImagenes.length);

int[] etiqueta = newint[archivoImagenes.length];

intcontado = 0;
intetiquetaNombre;

IplImageimg=null;
IplImagegrayImg;

int i1=mPath.length();

for (File image : archivoImagenes) {
      String p = image.getAbsolutePath();
img = cvLoadImage(p);

if (img==null)
      Log.e("Error","ErrorcVCargar Imagen");
Log.i("imagen",p);

int i2=p.lastIndexOf("-");
int i3=p.lastIndexOf(".");
inticount=Integer.parseInt(p.substring(i2+1,i3));
if (contar<icount) contar++;

            String description=p.substring(i1,i2);

if (etiquetasArchivo.get(description)<0)
      etiquetasArchivo.add(description, etiquetasArchivo.max()+1);

etiquetaNombre = etiquetasArchivo.get(description);

grayImg = IplImage.create(img.width(), img.height(),
IPL_DEPTH_8U, 1);

cvCvtColor(img, grayImg, CV_BGR2GRAY);

imagenes.put(contado, grayImg);
            etiqueta[contado] = etiquetaNombre;
```

```
            contado++;
        }
if (contado>0)
      if (etiquetasArchivo.max()>0)
            reconocerRostro.train(imagenes, etiqueta);
etiquetasArchivo.Guardar();
      returntrue;
}
```

Este método permite entrenar la Liberia javacv con las diferentes fotos tomadas del rostro.

Método Agregar

```
voidAgregar(Mat m, String description) {

      Mat recorte= new Mat() ;
      Mat imagenrgb = new Mat();
      Imgproc.cvtColor(m, imagenrgb, Imgproc.COLOR_RGBA2BGR, 3);
      Imgproc.cvtColor(imagenrgb, recorte, Imgproc.COLOR_BGR2GRAY, 1);

      Bitmap bmp= Bitmap.createBitmap(m.width(), m.height(),
 Bitmap.Config.ARGB_8888);

      Utils.matToBitmap(m,bmp);
      bmp= Bitmap.createScaledBitmap(bmp, ANCHO, ALTO, false);

      FileOutputStream f;
try {
      f = newFileOutputStream(mPath+description+"-"+count+".jpg",true);
            count++;
            bmp.compress(Bitmap.CompressFormat.JPEG, 100, f);
            f.close();

      } catch (Exception e) {
            Log.e("error",e.getCause()+" "+e.getMessage());
            e.printStackTrace();

      }
      //preparacionimagen red neuronal

      Mat ecualizado = new Mat();
      Mat ntamaño = new Mat();
      Imgproc.equalizeHist(recorte, ecualizado);
      Size s1= new Size( 10, 10);
      Imgproc.resize( ecualizado, ntamaño, s1, 0, 0,
Imgproc.INTER_LANCZOS4);

      int k=0;
      ve = newdouble[pixel1*pixel2];

try {
for(inti = 0;i <pixel1;i++){
```

```
for(int j = 0;j <pixel2;j++){

       int temp;
       if(ntamaño.get(i,j)[0]>127.5){
              temp=1;
         }else{
              temp=0;
         }

       ve[k] = temp;

       k=k+1;
              }

           }
       } catch (Exception e) {
       // TODO: handle exception
}

       //añade cada foto al vector
for(int j=0;j<(pixel1*pixel2);j++){
             datosre[kf+j]=ve[j];
         }

kf=kf+(pixel1*pixel2);//suma el tamañodepixelescadavez

//pasar vector double a string
       datosred=new String[datosre.length];
       for (inti = 0; i<datosred.length; i++)
             datosred[i] = String.valueOf(datosre[i]);

       contador++;
       String filename = "photo-"+ contador +".jpg";

 File path = newFile(Environment.getExternalStorageDirectory() +
"/Android/data/imagenesRN/");
       path.mkdirs();
       File file = new File(path, filename);
filename = file.toString();
       Highgui.imwrite(filename,ntamaño);


       }
```

Este método se encarga de ir guardando en preferencias las fotos del rostro, para su posterior uso.

Diseño de la interfaz lista de aplicaciones a bloquear

En la Figura 27 se muestra el maquetado de la interfaz lista de aplicaciones a bloquear.

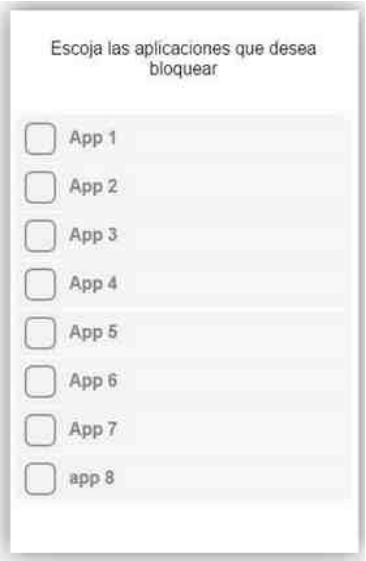

Figura 27. Diseño de la interfaz lista de aplicaciones a bloquear

Flujo de navegación de la lista de aplicaciones a bloquear

En la Figura 28 se muestra el flujo de navegación entre el menú principal y la lista de aplicaciones que van a ser bloqueadas.

Figura 28. Flujo de navegación entre el menú principal y la lista de aplicaciones

Código fuente de la clase Lista Aplicaciones

```
Protectedvoid onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.lista_apps);

       vVista = findViewById(R.id.Layout1);

       alResultados = newArrayList<String>();
PackageManager pm = this.getPackageManager();
       Intent intent = new Intent(Intent.ACTION_MAIN, null);
intent.addCategory(Intent.CATEGORY_LAUNCHER);
llista = pm.queryIntentActivities(intent,
PackageManager.PERMISSION_GRANTED);

for (ResolveInforInfo : llista) {
       nombreApp =
      Info.activityInfo.applicationInfo.loadLabel(pm).toString()
 + "\n";
nombrePaqueteApp=
rInfo.activityInfo.applicationInfo.packageName.toString();

cbAplicaciones = newCheckBox(getApplicationContext());
cbAplicaciones.setText(nombrePaqueteApp);
cbAplicaciones.setOnClickListener(this);

cbAplicaciones.setId(contApp);
// cb.setBackgroundColor(Color.parseColor("#ffffff"));

alResultados.add(rInfo.activityInfo.applicationInfo.loadLabel(pm)
                           .toString());
       Log.w("Installed Applications", rInfo.activityInfo.applicationInfo
                           .loadLabel(pm).toString());
       dwIcono = rInfo.activityInfo.applicationInfo.loadIcon(pm);
       llprincipal = newLinearLayout(getApplicationContext());
       llprincipal.setOrientation(LinearLayout.HORIZONTAL);

       cbAplicaciones.setTextColor(color.black);
       tvDatos = newTextView(getApplicationContext());
       tvDatos.setText(nombreApp);
       ivImagen = newImageView(getApplicationContext());
ivImagen.setImageDrawable(dwIcono);

LinearLayout.LayoutParamsparms = new
LinearLayout.LayoutParams(45,45);//tamañoimagenparalinearlayout
ivImagen.setLayoutParams(parms);

       ((ViewGroup) llprincipal).addView(ivImagen);
       ((ViewGroup) llprincipal).addView(tvDatos);//añade checkboxal
linear layout
       ((ViewGroup) llprincipal).addView(cbAplicaciones);
       ((ViewGroup) vVista).addView(llprincipal);
       contApp++;
cargarPreferencias();
}
}
```

Este método carga la lista de aplicaciones instaladas en el dispositivo móvil, las mismas que pueden ser bloqueadas por el usuario.

Método onClick selección de aplicaciones

```
publicvoidonClick(View view) {
              // TODO Auto-generated method stub
       cbAplicaciones = (CheckBox) view;
       if(cbAplicaciones.isChecked()){

       preferencia=String.valueOf(cbAplicaciones.getId());
       guardarPreferencias();

         }elseif(!cbAplicaciones.isChecked()){
                     guardarNoChecked();
         }

}
```

En el método onclick se guarda en preferencias todas las aplicaciones que el usuario se dispone a bloquear.

Diseño de la interfaz reconocer rostro

En la Figura 29 se muestra el maquetado de la interfaz reconocer rostro.

Figura 29. Diseño de la interfaz reconocer rostro.

Flujo de navegación reconocer rostro

En la Figura 30 se muestra el flujo de navegación del reconocimiento de rostro al momento de lanzar cualquiera nuestras aplicaciones seleccionadas para ser bloqueadas.

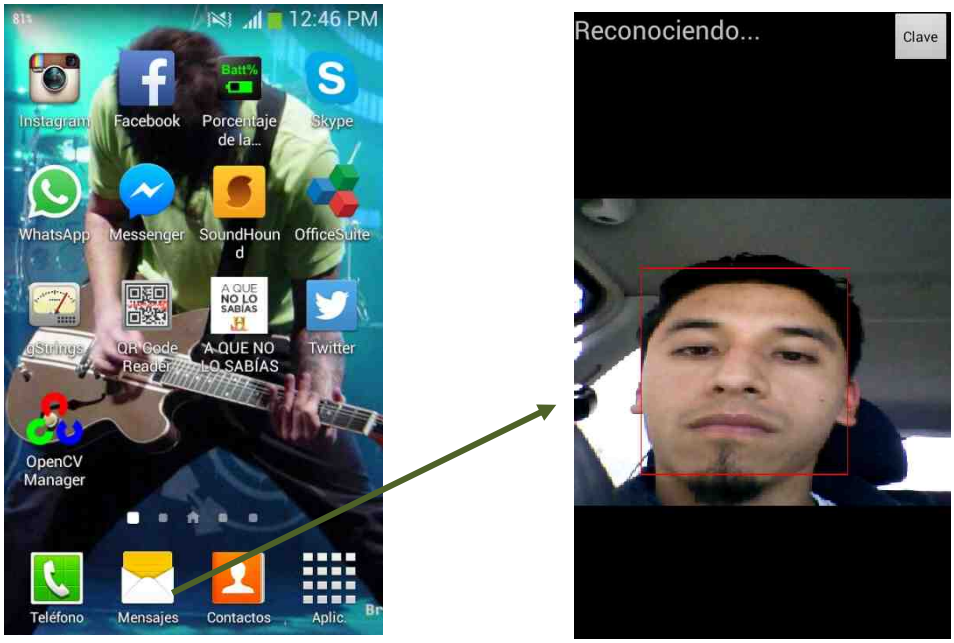

Figura 30. Flujo de navegación del reconocer rostro

Código fuente de la clase reconocer rostro

Método Oncreate

```
publicvoidonCreate(Bundle savedInstanceState) {

Log.i(TAG, "called onCreate");
super.onCreate(savedInstanceState);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

setContentView(R.layout.vista_camara2);
vistaCamara = (VistaCamara) findViewById(R.id.area_camara);
vistaCamara.setCvCameraViewListener(this);
vistaCamara.setCameraIndex(org.opencv.android.CameraBridgeViewBase.CAMERA_
ID_FRONT);

pathFoto=Environment.getExternalStorageDirectory() +
"/Android/data/facerecogOCV/";
```

```
labelsFile= newEtiquetas(pathFoto);
BotonClave= (Button) findViewById(R.id.BtnClave);
Tvmensaje = (TextView) findViewById(R.id.TvMensaje);

//boton clave
BotonClave.setOnClickListener(newView.OnClickListener() {

              @Override
              publicvoidonClick(View v) {
                     // TODO Auto-generated method stub
                     DialogoClave();
              }
       });

        Timer mitimer = new Timer();

mitimer.schedule( newTimerTask() {

       @Override
       publicvoid run() {
                           // TODO Auto-generated method stub
       try {
        Runnable task1 = new Runnable() {
                     publicvoid run() {
                                  verificared();

                           }
                     };
              Thread thread1 = new Thread(newThreadGroup("Verificacion"),
                           task1, "Verificar", 999999999);
              Thread.currentThread();
              thread1.start();
       } catch (Exception e2) {
              // TODO: handle exception
                     }
                     }
       }, 1000, 5000);

mHandler1 = new Handler() {
@Override
publicvoidhandleMessage(Message msg) {
SharedPreferencesvalorred =
getSharedPreferences("valored",Context.MODE_PRIVATE);
doublevalored= valorred.getLong("red", 0);

if (mconfiabilidad<0);
elseif (mconfiabilidad<50&&networkOutput>valored)
{
       RETANGULOROSTRO     = newScalar(0, 0, 255, 255);

lanzarapp();
Toast.makeText(getApplicationContext(), "Reconocimiento
Correcto!",Toast.LENGTH_SHORT).show();
            }
elseif (mconfiabilidad<80 &&networkOutput>valored)//||
       {
```

```
       RETANGULOROSTRO     = newScalar(0, 255, 0, 255);
lanzarapp();
       Toast.makeText(getApplicationContext(), "Reconocimiento
Correcto!",Toast.LENGTH_SHORT).show();
       }
       else
RETANGULOROSTRO     = new Scalar(255, 0, 0, 255);
       Tvmensaje.setText("Reconociendo...");
            }
        };
estadoRostro=BUSQUEDAROSTRO;

        }
    }
```

El método oncreate se encarga de encender la cámara frontal del dispositivo móvil, también se encarga de verificar si el rostro ingresado corresponde al usuario previamente registrado, tanto en la red neuronal como la librería javacv.

```
public Mat onCameraFrame(CvCameraViewFrameinputFrame) {

MatOfRect faces = newMatOfRect();

if (mDetectorType == JAVA_DETECTOR) {
if (mJavaDetector != null)
mJavaDetector.detectMultiScale(migrey, faces, 1.1, 2, 2,
new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new
 Size());
        }
elseif (mDetectorType == NATIVE_DETECTOR) {
//   if (mNativeDetector != null)
//     mNativeDetector.detect(migrey, faces);
        }
else {
Log.e(TAG, "Detection method is not selected!");
        }

Rect[] facesArray = faces.toArray();

try {

       if ((facesArray.length>0)&& (estadoRostro==BUSQUEDAROSTRO))
{
         Mat m=new Mat();
         m=migrey.submat(facesArray[0]);
       mBitmap = Bitmap.createBitmap(m.width(),m.height(),
 Bitmap.Config.ARGB_8888);

Utils.matToBitmap(m, mBitmap);
              Message msg = new Message();
              String textTochange = "IMG";
```

```
              msg.obj = textTochange;
mHandler1.sendMessage(msg);

textTochange=fr.predecir(m);
mconfiabilidad=fr.getProb();
       msg = new Message();
         msg.obj = textTochange;
       mHandler1.sendMessage(msg);
          }

        } catch (Exception e) {
                    // TODO: handle exception
              }
       // normalizarrostroparaverificar red neuronal
Mat cara= new Mat() ;
       Mat ecualizado = new Mat();
              Mat ntamaño = new Mat();
try{
       cara=migrey.submat(facesArray[0]);

       Imgproc.equalizeHist(cara, ecualizado);
       Size s1= new Size( 10, 10);
Imgproc.resize( ecualizado, ntamaño, s1, 0, 0,
 Imgproc.INTER_LANCZOS4);
              int k=0;
       ve = newdouble[pixel1*pixel2];
       for(inti = 0;i <pixel1;i++){
              for(int j = 0;j <pixel2;j++){
              int temp;
              if(ntamaño.get(i,j)[0]>127.5){
                      temp=1;
                }else{
                      temp=0;
                }

              ve[k] = temp;
              k=k+1;
                        }
}
                    } catch (Exception e) {
                     // TODO: handle exception
                    }
for (inti = 0; i<facesArray.length; i++)
Core.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(),
RETANGULOROSTRO, 1);
returnmRgba;
    }
```

Este método verifica si existe un rostro para realizar el reconocimiento, lo procesa y envía para su posterior verificación.

Método Lanzar App

```
publicvoidlanzarapp(){
```

```
       SharedPreferences  prefer1 =
getSharedPreferences("estadoapp",Context.MODE_PRIVATE);
appn=prefer1.getString("appname", "vacio");

       Intent in;
       PackageManager manager = getPackageManager();
       try {
          in = manager.getLaunchIntentForPackage(appn);
       if (in == null)
             thrownewPackageManager.NameNotFoundException();
             in.addCategory(Intent.CATEGORY_LAUNCHER);
             startActivity(in);
             } catch (PackageManager.NameNotFoundException e) {

       }
       cerrar();//finalizaestaactividad

    }
```

Este método se encarga de volver a lanar la aplicación previamente seleccionada.

Método Verificar Red

```
privatevoidverificared() {

       NeuralNetworkloadedMlPerceptron = NeuralNetwork.load(rutaRed);
System.out.println("testeo.........");

       try {
             loadedMlPerceptron.setInput(ve);
       loadedMlPerceptron.calculate();
       networkOutput = loadedMlPerceptron.getOutput()[0];
       System.out.println(" Output: " + networkOutput );
                   } catch (Exception e) {
                         // TODO: handle exception
       }

    }
```

Aquí se realiza un test del rostro ingresado para comprobar que corresponda al usuario registrado.

Código fuente clase Servicio

Método Onstart

```
publicvoidonStart(Intent intent, intstartId) {

       listaBloqueo = newArrayList<String>();
       SharedPreferencesprefs =
getSharedPreferences(ListaAplicaciones.mPreferencia,
Context.MODE_PRIVATE);
```

```
       Map<String,?> map = prefs.getAll();

       for(Map.Entry<String,?> entry : map.entrySet()){

       if(entry.getValue().toString().equals("true")){
                     String nombre=entry.getKey().toString();
                     listaBloqueo.add(nombre);

              }
       }

       List<ActivityManager.RunningTaskInfo>Infotarea =
am.getRunningTasks(1);
              top = Infotarea.get(0).topActivity.getPackageName();
              SharedPreferences  prefer1 =
getSharedPreferences("estadoapp",Context.MODE_PRIVATE);
               Editor editor1 = prefer1.edit();
       nombreApp=prefer1.getString("appname", "no1");

              SharedPreferencesbander =
getSharedPreferences("bandera",Context.MODE_PRIVATE);
              bandera=bander.getString("band", "no");
if(!top.equals(bandera)&&!top.equals
("bloqueo.app.reconocimiento.rostros")){
bander = getSharedPreferences("bandera",Context.MODE_PRIVATE);
       Editor edit = bander.edit();
       edit.putString("band",top);
       edit.commit();

       }
for(String nombre: listaBloqueo){
       //Log.d(AppLockActivity.TAG, "name = " + name);
       if(nombre.equals(top)) {

       if(!bandera.equals(top)){
              //vr.cerrar();
       bander = getSharedPreferences("bandera",Context.MODE_PRIVATE);
Editor edit = bander.edit();
       edit.putString("band",top);
       edit.commit();
       paquete=top;

       prefer1 = getSharedPreferences("estadoapp",Context.MODE_PRIVATE);
      editor1 = prefer1.edit();
      editor1.putString("appname",paquete);
      editor1.commit();

ActivityManager mane = ((ActivityManager)
getSystemService(ACTIVITY_SERVICE));

       mane.killBackgroundProcesses(paquete);

Intent i = new Intent();
       i.setClass(getBaseContext(), VerificarRostro.class);
i.setAction(Intent.ACTION_MAIN);
       i.addCategory(Intent.CATEGORY_HOME);
```

```
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(i);
                    }
                }

        }

}
```

Este servicio está ejecutándose siempre en el dispositivo, y es el encargado de verificar si la última aplicación lanzada por el usuario está en la lista de bloqueo.

Diseño de la interfaz borrar registro del rostro

En la Figura 31 se muestra el maquetado de la interfaz eliminar registro del rostro.

Figura 31. Diseño de la interfaz eliminar registro del rostro.

Flujo de navegación eliminar registro del rostro

En la Figura 32 se muestra el flujo de navegación para eliminar el registro del rostro.

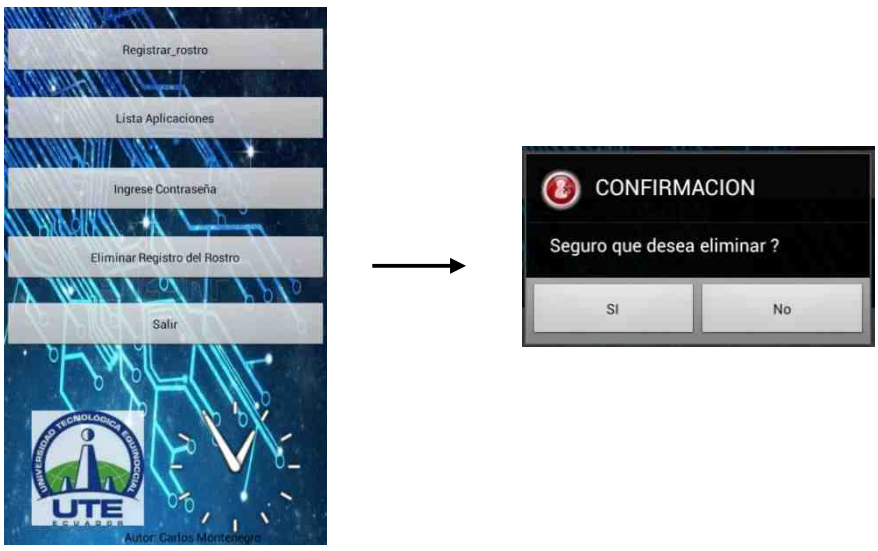

Figura 32. Flujo de navegación borrar registro del rostro.

Código fuente de la clase principal

Botón Eliminar

```
btnEliminar.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View arg0) {
                        // TODO Auto-generated method stub

 AlertDialog.Builder alertaE = new AlertDialog.Builder(ctx);
       alertaE.setTitle("CONFIRMACION"); //titulo
       alertaE.setMessage("Seguro que desea eliminar ?"); //Mensaje
 alertaE.setIcon(R.drawable.iconeliminar);

      alertaE.setPositiveButton("SI", new
DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int whichButton) {

       File directorio= new File(Environment.getExternalStorageDirectory()
+ "/Android/data/facerecogOCV/");
       if (directorio.isDirectory()) {
         String[] children = directorio.list();
         for (int i = 0; i < children.length; i++) {
             new File(directorio, children[i]).delete();
                                        }
                                    }

File directorio2= new File(Environment.getExternalStorageDirectory() +
"/Android/data/nnet/");
       if (directorio2.isDirectory()) {
         String[] children = directorio2.list();
        for (int i = 0; i < children.length; i++) {
            new File(directorio2, children[i]).delete();
                                        }
                                    }

File directorio3= new File(Environment.getExternalStorageDirectory() +
"/Android/data/imagenesRN/");
       if (directorio3.isDirectory()) {
         String[] children = directorio3.list();
         for (int i = 0; i < children.length; i++) {
             new File(directorio3, children[i]).delete();
                                        }
                                    }
                            }
                        });
alertaE.setNegativeButton("No", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialogo, int whichButton) {

                                dialogo.cancel();
                             }
                        });
                            alertaE.create();
                            alertaE.show();
```

```
                }
            });
```

Diseño de interfaz ingresar contraseña

En la Figura 33 se muestra el maquetado de la interfaz ingresar contraseña.

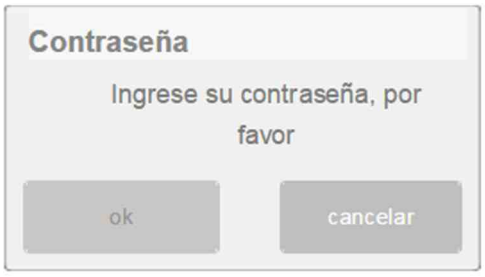

Figura 33. Diseño de la interfaz ingresar contraseña

Flujo de navegación ingresar contraseña

En la Figura 34 se muestra el flujo de navegación de la opción de ingreso de contraseña como medida alternativa al reconocimiento del rostro.

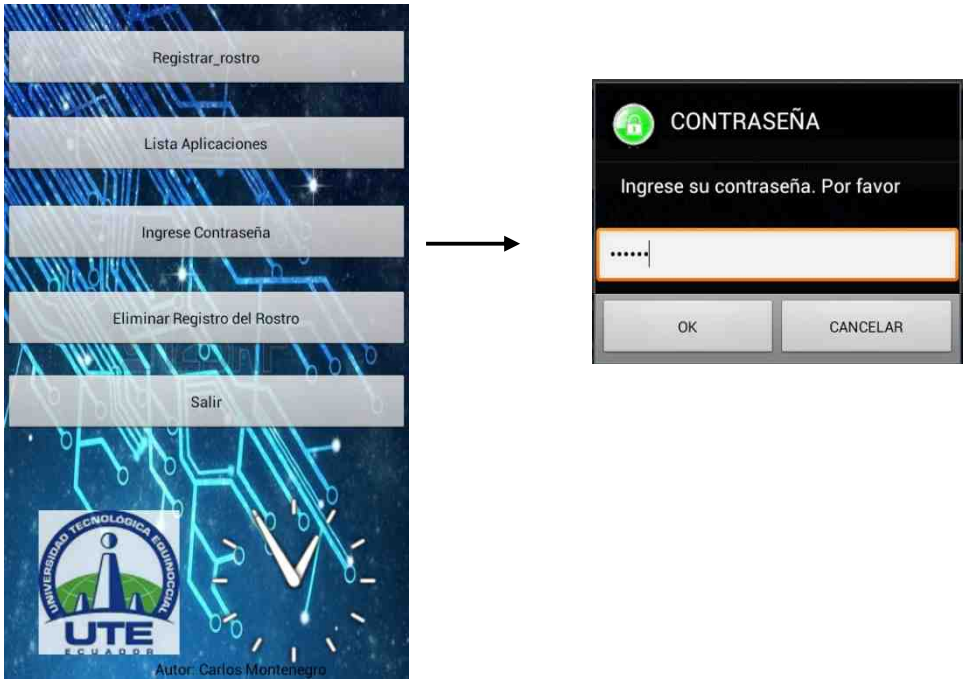

Figura 34. Flujo de navegación del ingresar contraseña

Código fuente de la clase Principal

Botón Contraseña

```
btnContrasena.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                          // TODO Auto-generated method stub

                   AlertDialog.Builder alerta = new
AlertDialog.Builder(ctx);
                   alerta.setTitle("CONTRASEÑA"); //titulo
                   alerta.setMessage("Ingrese su contraseña. Por favor");
//Mensaje
                   alerta.setIcon(R.drawable.iconclave);

                   final EditText EtIngreso = new EditText(ctx);
                   EtIngreso.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);
                   alerta.setView( EtIngreso);

                   alerta.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int
whichButton) {

                    String srt =  EtIngreso.getEditableText().toString();
                    SharedPreferences preferclave =
getSharedPreferences("contraseña",Context.MODE_PRIVATE);
                         Editor editorc = preferclave.edit();
                         editorc.putString("clave",srt);
                         editorc.commit();

                   }
               });
                   alerta.setNegativeButton("CANCELAR", new
DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialogo, int
whichButton) {

                         dialogo.cancel();
                     }
               });
                   alerta.create();
                   alerta.show();

                    }
             });
```

4.3 EL CLIENTE PRUEBA LA MAQUETA

En esta etapa se realizaron las pruebas en un dispositivo móvil real. Las mismas que ayudaron a determinar el número neuronas en cada capa de la red neuronal artificial, así como el número de fotos del rostro que serán tomadas con el dispositivo móvil.

En la Tabla 9 se muestra los diferentes resultados obtenidos durante las pruebas realizadas en la aplicación de reconocimiento de rostros, mismas que sirvieron para determinar la configuración más eficiente para esta aplicación, tomando en cuenta el número de fotos, el tiempo de respuesta de la RNA y la eficiencia de la misma.

Test	**Nro. imágenes**	**Pixeles**	**Nro. de neuronas por capa RNA**			**Tiempo respuesta RNA**	**Eficiencia RNA**
			Capa Entrada	Capa Oculta	Capa Salida		
1	5	20 x 20	400	50	1	00:01:45	65 %
2	5	15 x 15	225	40	1	00:00:58	65%
3	7	15 x 15	225	40	1	00:01:03	55 %
4	7	10 x 10	100	35	1	00:00:05	50 %
5	10	15 x 15	225	30	1	00:00:55	60 %
6	10	10x10	100	35	1	00:00:02	70 %
7	15	15 x 15	225	50	1	00:01:11	65 %
8	15	10 x 10	100	30	1	00:00:05	70 %

Tabla 9. Pruebas de la red neuronal artificial

Como se puede observar en la Tabla 9 la configuración de la RNA más eficiente es el test número 6, ya que además de tener una muy buena eficiencia, el tiempo de respuesta es menor y ventajoso para un dispositivo móvil.

La eficiencia fue dada en base a 20 muestras de cada rostro, donde cada reconocimiento correcto se contabilizo para calcular el porcentaje.

Se realizó pruebas en diferentes niveles de iluminación del ambiente al momento de tomar las fotos del rostro, y se pudo observar que mejora el nivel de reconocimiento ya que cuenta con una base de conocimiento más amplia.

También se hizo una prueba variando las distancias entre el dispositivo móvil y el usuario cuando se toma las fotos, y se determinó que tomar las fotos a diferentes distancias ayuda mucho al momento de realizar el reconocimiento del rostro.

4.3.1 MANEJO DE LA APLICACIÓN

Para una correcta instalación de la aplicación se recomienda seguir los pasos descritos en el Anexo 1. A continuación se muestra la forma de utilizar la aplicación de reconocimiento de rostros.

La pantalla Principal se muestra en la Figura 35 y tiene 5 botones, los cuales podemos seleccionar y nos presentara otra pantalla con la opción que deseamos realizar.

Figura 35. Pantalla principal de la aplicación.

Botón registrar rostro. Al seleccionar este botón nos muestra una pantalla donde vamos a registrar las 10 fotos del rostro con el botón Foto, como se muestra en la Figura 36.

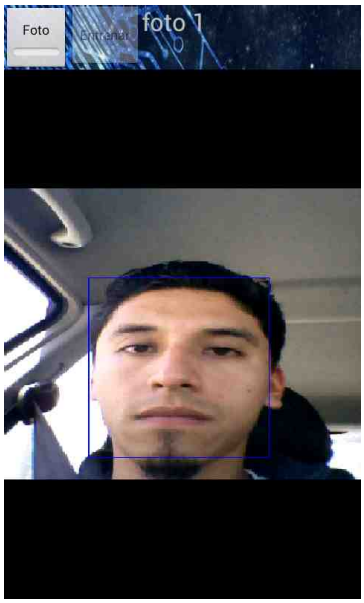

Figura 36. Pantalla registro del rostro

Una vez tomadas las fotos del rostro, se activa un botón Entrenar.

Botón Entrenar. La función de este botón es entrenar la RNA, y la librería de JavaCv, tal como se muestra en la Figura 37.

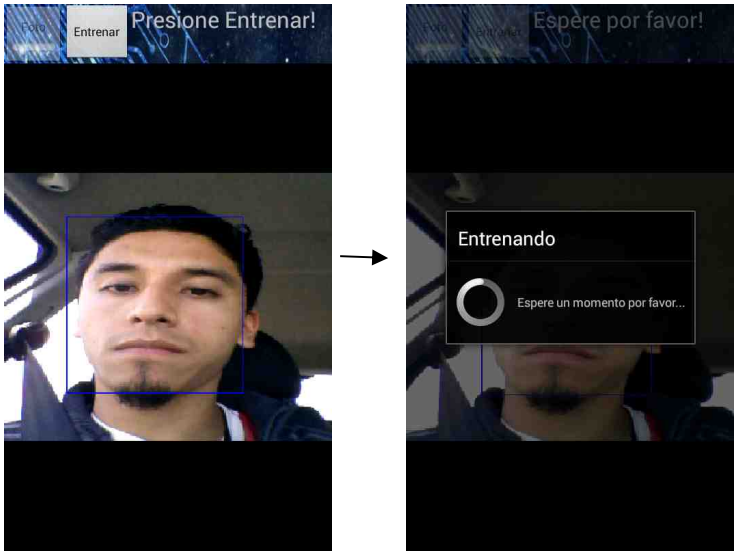

Figura 37. Pantallas entrenamiento RNA

Una vez concluido el entrenamiento nos regresara al menú principal.

Botón Lista Aplicaciones. En esta opción se muestra la lista de todas las aplicaciones instaladas en el dispositivo móvil, como se muestra en la Figura 38. Aquí debemos marcar las aplicaciones que queremos bloquear con el sistema de reconocimiento de rostros, caso contrario desmarcamos.

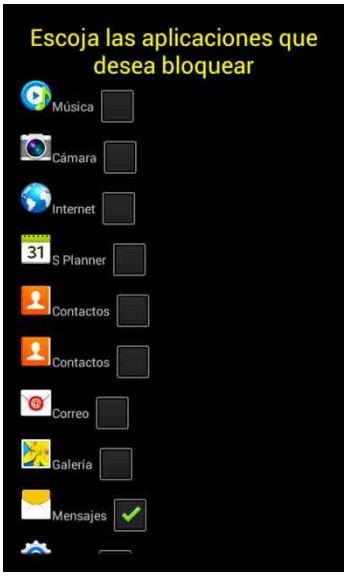

Figura 38. Pantalla lista de aplicaciones

Botón Ingrese Contraseña. Esta pantalla nos muestra un cuadro de dialogo como se ve en la Figura 39, donde vamos a ingresar una contraseña como medida adicional al reconocimiento facial, tomando en cuenta que se necesita un mínimo de luz para poder ser detectado por la cámara del dispositivo móvil. Caso contrario no podrá realizar el reconocimiento de rostros y se debe ingresar la contraseña anteriormente registrada.

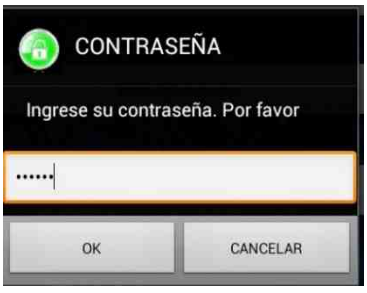

Figura 39. Pantalla Ingreso de contraseña

Botón eliminar registro del rostro. Esta opción muestra un cuadro de dialogo, donde nos realiza un pregunta de confirmación para eliminar las fotos del rostro anteriormente registradas, como se puede apreciar en la Figura 40. Esta operación se realiza con el fin de poder registrar una nueva base de conocimiento para la RNA y librería JavaCv.

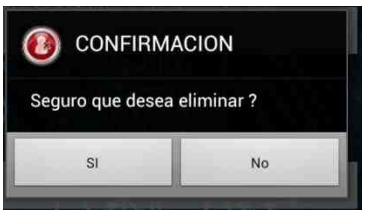

Figura 40. Pantalla eliminar registro rostro

Una vez realizados estos pasos, al momento de querer abrir cualquiera de las aplicaciones que anteriormente se hayan marcado para ser bloqueadas, se presentara una pantalla para el reconocimiento del rostro como se muestra en la Figura 41, de ser correcto el reconocimiento procederemos a mostrar la aplicación anteriormente seleccionada caso contrario no se podrá acceder.

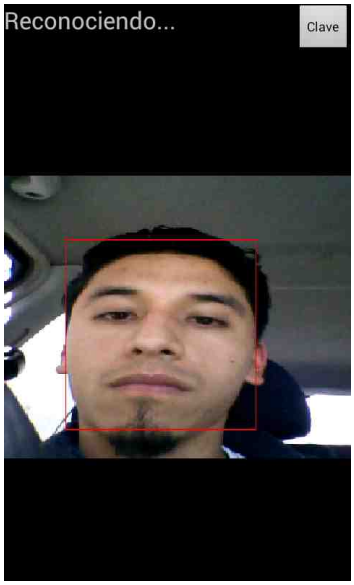

Figura 41. Pantalla de reconocimiento del rostro

CONCLUSIONES Y RECOMENDACIONES

5 CONCLUSIONES Y RECOMENDACIONES

A continuación se detalla las conclusiones y recomendaciones obtenidas durante la elaboración de la tesis.

5.1 CONCLUSIONES

- Una característica que hay que tomar en cuenta al momento de utilizar aplicaciones de reconocimiento facial es la iluminación del ambiente, ya que la cámara del dispositivo necesita un mínimo de luz para poder verificar el rostro.
- Adoptar el modelo de prototipos para el desarrollo de aplicaciones móviles ayuda mucho, ya que se requiere ir realizando pruebas en las diferentes etapas.
- Una vez realizadas las pruebas del sistema, se puede concluir que el uso de funciones de la librería javacv mejora el rendimiento al realizar el reconocimiento de rostros en un dispositivo móvil.
- El uso de librerías de visión artificial como opencv ayudan mucho al momento de realizar un reconocimiento de rostros, gracias a que cuentan con funciones muy eficientes para este tipo de procesamiento como FaceRecognizer.
- La librería neuroph provee de código muy simplificado al momento de utilizar técnicas de inteligencia artificial, en nuestro caso el perceptrón multicapa que es un modelo de red neuronal artificial.
- Demasiadas neuronas en las capas aumenta el procesamiento de la RNA y esto genera demasiado tiempo de respuesta en los dispositivos móviles que no tenga mucha capacidad de procesamiento.
- Para un correcto funcionamiento de una red neuronal artificial se recomienda entrenarla con el mayor número de datos posibles, ya que mientras más se la entrena mejores resultados nos arrojará, caso contrario la red puede presentar problemas al momento de realizar el test.

- Se puede apreciar que la seguridad de Android se maneja mediante un esquema de permisos, de tal manera que el usuario del dispositivo móvil conozca los riesgos a los que se expone al instalar alguna aplicación.

5.2 RECOMENDACIONES

- Para probar el correcto funcionamiento de una aplicación de reconocimiento de rostros, se recomienda hacerlo en un dispositivo móvil real, ya que así podemos medir el tiempo de respuesta al momento de realizar las respectivas pruebas con el usuario.
- Se recomienda procesar las imágenes del rostro antes de ingresarlas a la capa de entrada de la red neuronal, ya que esto permitirá un mejor rendimiento de la misma.
- Se recomienda realizar varias pruebas, variando el número de neuronas tanto de la capa de entrada como de la capa oculta, de esta manera podremos ver el mejor rendimiento de la red neuronal artificial en un dispositivo móvil.
- Se debe investigar sobre las técnicas de reconocimiento de rostros más utilizadas para crear este tipo de sistemas, ya que facilitara el desarrollo.
- Se recomienda utilizar un dispositivo móvil que tenga una cámara frontal con buena resolución para el buen funcionamiento de reconocimiento de rostro.
- Para futuros trabajos se recomienda utilizar funciones de la librería Opencv como Eigenfaces, ya que son muy eficientes para realizar aplicaciones de reconocimiento facial.

BIBLIOGRAFÍA

Agarwal, M., Agrawal, H., Jain, N., &Kumar, M. (2010). Face Recognition Using Principle Component Analysis, Eigenface and Neural Network. *International Conference on Signal Acquisition and Processing, 2010. ICSAP '10* (pp. 310-314).

Alvarado, G. J., Pedrycz, W., Reformat, M., &Kwak, K.-C. (2006). Deterioration of visual information in face classification using Eigenfaces and Fisherfaces. *Machine Vision and Applications*, *17*(1), 68-82.

Anam, S., Islam, M. S., Kashem, M. A., Islam, M. N., Islam, M. R., & Islam, M. S. (2009). *Face Recognition Using Genetic Algorithm and Back Propagation Neural Network*.*Proceedings of the International MultiConference of Engineers and Computer Scientists.*

Android Open Source Project (2013). Android Security Overview, recuperado el 27 de mayo del 2013, desde http://source.android.com/tech/security/#android-security-overview

Banuri, H., Alam, M., Khan, S., Manzoor, J., Ali, B., Khan, Y. Zhang, X. (2011). An Android runtime security policy enforcement framework. *Personal and Ubiquitous Computing*, *16*(6), 631-641.

Behan M. &Mansor S. , (2013). A review of face recognition methods. *International Journal of Pattern Recognition and Artificial Intelligence.*

Esqueda J, Palafox (2005). *Fundamentos de procesamiento de imágenes*. México, Rosa Galindo. 57-59.

FaceLock. (2013). *Facelock App pro*, recuperado el 10 de marzo del 2015, desde http://www.facelock.mobi/

Facial Network Inc. (2015). *Nametag*, recuperado el 15 de marzo del 2015, desde http://www.nametag.ws/

Ghorpade, S., Ghorpade, J., Mantri, S., &Ghorpade, D. (2010). *Neural Networks for Face Recognition Using Som*.

Gironés J. (2011), El gran libro de android(1ra ed.),México, Alfaomega.340,18-24.

Heseltine T., Pears N., Austin J. (2002), *Evaluation of Image Pre-Processing Techniques for Eigenface based Face Recognition*, Advanced Computer Architecture Group, Department of Computer Science, The University of York.

Isasi P. &Galvan M.(2004), Redes Neuronales Artificiales. Un Enfoque Práctico,España, Lavel s.a.

Kumar Gupta R., Kumar Sahu U. (2013*). Real Time Face Recognition under Different Conditions*. International Journal of Advanced Research in Computer Science and Software Engineering

Mayank A. Nikunj J. Manish K. &Himanshu A. (2010). Face Recognition Using Eigen Faces and Artificial Neural Network*. International Journal of Computer Theory and Engineering.*Recuperado el 22 de mayo del 2013, desde http://www.ijcte.org/papers/213-H282.pdf

Meier R. (2012), Profesional Android 4 Application Development,Indiana , Wiley J. & Sons, Inc.

Ñustes S, Hurtado J, Bedoya G&Marin L, (2013).Introducción al desarrollo de redes neuronales perceptrón multicapa aplicadas en tecnología android. Amazonia Investiga.

Opencv devteam. (2014). FaceRecognitionwith OpenCv, recuperado el 10 de noviembre del 2014, desde http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html

Pressman, R.(2002), Ingeniería del Software. Un enfoque práctico – quinta edición, Mexico, McGraw Hill.

Rabbani M, & Chellappan C., (2007). A Different Approach to Appearance based Statistical Method for Face recognition Using Median. *International Journal of Computer Science and Network Security.*

Rabunal J., Dorado J., &Pazos A.,(2009), *Encyclopedia of Artificial Intelligence. United States of America. IGI Global*.1667,639-641

Rumbaugh J, Jacobson I & Booch G,(2007). *El lenguaje unificado de modelado. Manual de referencia* –segunda edición, España, Editorial Pearson Education.

Sevarac Z, Goloskokovic I, Tait J, Carter-Greaves L, Morgan A, Steinhauer V, Koprivica M, Jovanovic I, Joksovic N,(2014). Java Neural Network framework, Recuperado el 1 de enero del 2015, desde http://neuroph.sourceforge.net/documentation.html

Taigman Y., Yang M., Ranzato M., & Wolf L. (2014). Closing the Gap to Human-Level Performance in Face Verification. Computer vision fundation.

Turk M. &Pentland A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*. Recuperado el 13 de mayo del 2013, desde http://www.facerec.org/algorithms/PCA/jcn.pdf

Visidon Ltd. (2015). *Visidon App lock*, recuperado el 10 de marzo del 2015, desde http://www.visidon.fi/en/Products#2

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2000). *FaceRecognition: A LiteratureSurvey*.

ANEXO 1

MANUAL DE INSTALACIÓN

MANUAL DE INSTALACIÓN.

A continuación se muestra la forma de instalar correctamente la aplicación del reconocimiento de rostros.

Lo primero que se debe hacer es pasar la aplicación al dispositivo móvil mediante usb o cualquier otra forma como se muestra en la Figura 42. Una vez que se tiene la aplicación de extensión .apk, pulsar sobre esta.

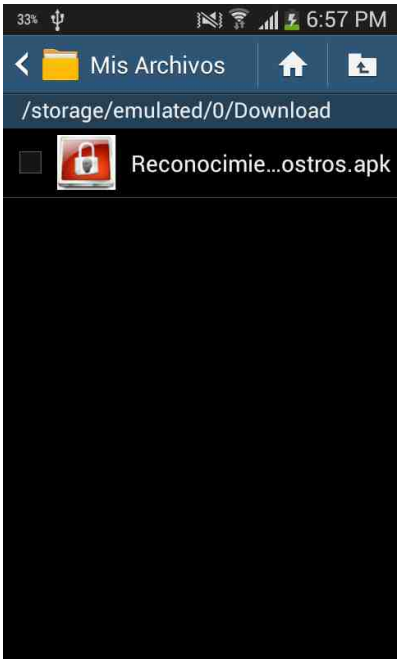

Figura 42. Pantalla instalador de la aplicación

Después aparecerá la siguiente pantalla donde pregunta si se desea instalar la aplicación y listará los permisos requeridos por la misma como se muestra en la Figura 43.

Figura 43. Pantalla permisos aplicación

Luego pulsar sobre la opción instalar y empezará con el proceso de instalación como se ve en la Figura 44.

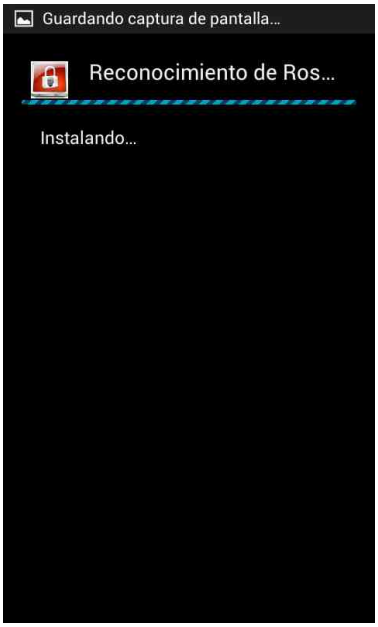

Figura 44. Pantalla instalación aplicación.

Esperar un momento hasta que presente la siguiente pantalla, donde indica que la instalación ha finalizado como se muestra en la Figura 45.

Figura 45. Pantalla finalizar instalación.

Pulsar en abrir y desplegará el menú principal de la aplicación como se aprecia en la Figura 46.

Figura 46. Pantalla menú aplicación.

Se debe escoger la opción de registrar rostro, al ejecutarse saldrá un mensaje donde pide instalar Opencv Manager que es un requisito para que funcione la aplicación, como se observa en la Figura 47.

Figura 47. Pantalla requerimiento OpenCv manager

Escoger ***Yes***, seguir el enlace donde se puede instalar el OpenCv Manager como se muestra en la Figura 48.

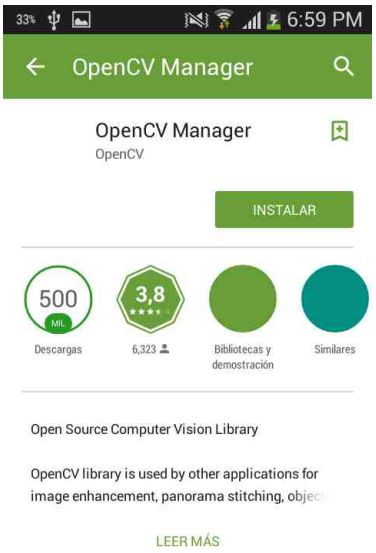

Figura 48. Pantalla instalación Opencv manager

Seleccionar la opción instalar como se observa en la Figura 49.

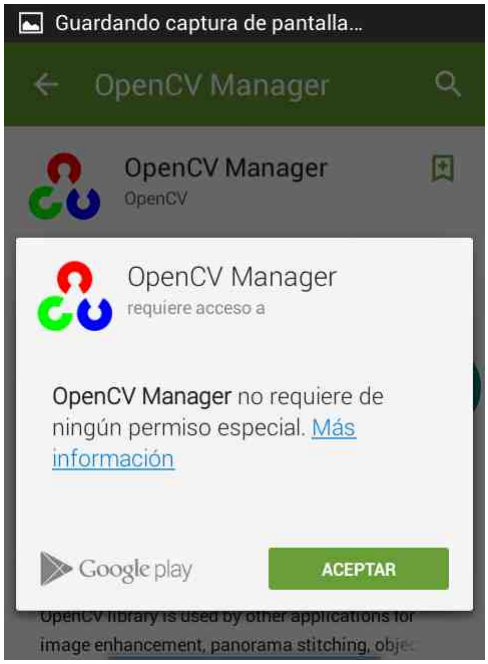

Figura 49. Pantalla aceptar instalación OpenCv manager

Pulsar aceptar y empezará el proceso de descarga como se muestra en la Figura 50.

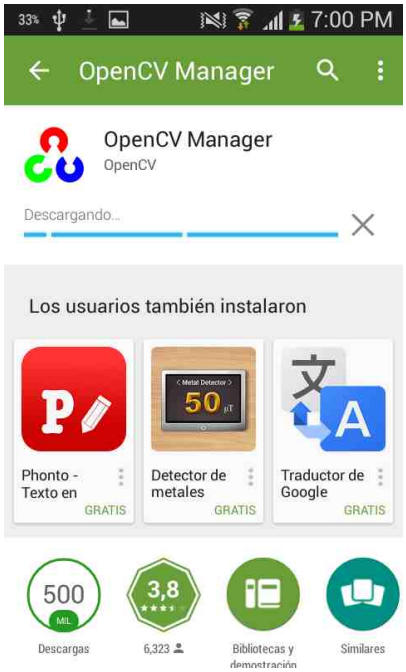

Figura 50. Pantalla descarga Opencv manager

Una vez terminada la descarga empezará automáticamente con la instalación como se observa en la Figura 51.

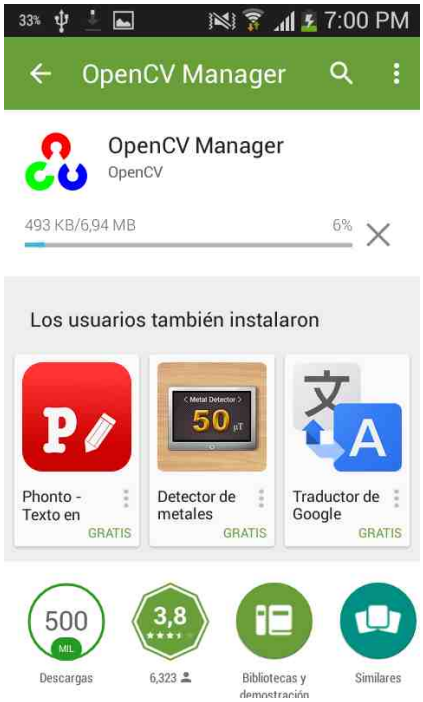

Figura 51. Pantalla instalación Opencv manager

Esperar que termine la instalación, donde mostrará la siguiente pantalla como se ve en la Figura 52.

Figura 52. Pantalla abrir Opencv Manager

Por último aparecerá una pantalla como se muestra en la Figura 53, donde se puede escoger la opción cerrar o abrir para ver acerca de Opencv Manager.

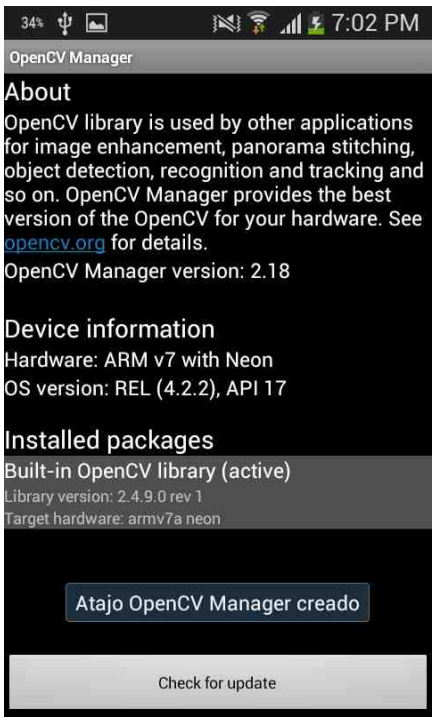

Figura 53. Pantalla información sobre Opencv manager