República Del Ecuador

Universidad Tegnológica Empresarial De Guayaquil - UTEG

Trabajo de Titulación

para la obtención del titulo de:

Ingeniero en Sistemas Computacionales

Mención Aplicaciones Web y Multimedia

Tema:

Diseño y Creación de una Aplicación Móvil para el registro de ingreso de personas con reconocimiento facial, implementando Microsoft Azure

Autor:

Alex Alberto García Arias

Directora de Tesis:

Ing. Diana López Msc.

Julio, 2019

Guayaquil – Ecuador

Declaración expresa de autoría y sesión de derechos de autor

Yo, **Alex Alberto García Arias**

DECLARO QUE: El Trabajo de Titulación **DISEÑO Y CREACIÓN DE UNA APLICACIÓN MÓVIL PARA EL REGISTRO DE INGRESO DE PERSONAS CON RECONOCIMIENTO FACIAL, IMPLEMENTANDO MICROSOFT AZURE**, previo a la obtención del Título de **INGENIERO EN SISTEMAS COMPUTACIONALES, MENCIÓN APLICACIONES WEB Y MULTIMEDIA**, ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del Trabajo de Titulación referido.

Guayaquil, Julio del 2019

EL AUTOR

_____

Alex Alberto García Arias

Dedicatoria

Dedico este trabajo a:

Mis Padres, personas maravillosas que son ejemplo de perseverancia en cada una de las acciones que emprenden.

A mi esposa por su comprensión y apoyo constante; por permanecer a mi lado, por aportar con sus consejos a que termine mis proyectos, por ser esa compañía que brinda mucho amor.

A mis abuelos, porque ellos siempre me han dado la ayuda fundamental para seguir adelante con mi carrera.

A mis hermanos, porque de esta forma puedo ser un ejemplo a seguir para ellos y continúen con sus carreras como yo lo he hecho.

A todos quienes de manera directa e indirecta aportaron al desarrollo y conclusiòn de mi gran sueño: Ser un profesional.

_____

Alex Alberto García Arias

Agradecimiento

Agradezco a Dios por ser mi guía perenne en todo momento de mi vida

A la Universidad Tecnológica Empresarial de Guayaquil UTEG por darme la oportunidad de estudiar y alcanzar mi título universitario

A los Directivos de la UTEG por el apoyo a la gestión administrativa de la carrera

A los Docentes que en el transcurso de todos los años de profesionalización aportaron sin egoísmo, con sus conocimientos al enriquecimiento como estudiante

A mi Tutor, por el tiempo y la dedicación para la culminación de este trabajo de titulación

Gracias a todos

_____

Alex Alberto García Arias

Resumen

En la investigación del trabajo de tesis “Diseño y creación de una aplicación móvil para el registro de ingresos de personas con reconocimiento facial implementando Microsoft Azure”, se consideraron los objetivos de diseñar, desarrollar e implementar la base de datos y Web Services. En el proyecto se tomó en cuenta el conocimiento de varios autores especializados en el tema de autenticación facial. Los métodos inductivo y deductivo dieron la opción de investigar de lo particular a lo general, mientras que la parte analítica logro identificar los pasos para la implementacion del proyecto. La investigación fue de tipo explorativo y descriptivo; el enfoque fue cualitativo – experimental logrado por medio de la técnica de la observación de los participantes en el experimento de la aplicación móvil. En el proceso se recurrió a la fotografía como fuente primaria del experimento, los participantes que intervinieron en las pruebas tuvieron la predisposición de aportar con el tiempo para el desarrollo de la propuesta. En la creación del *App* se tuvo acceso a las teorías de reconocimiento facial y en las pruebas se usó el teléfono móvil para la toma y registro secuencial de las fotografías, las mismas que dieron lugar al análisis de las facciones y expresiones de la cara, lo que conllevó al análisis de cada una de las fotografías de la cara de los participantes. La propuesta tuvo como finalidad implementar la aplicación, recurriendo a las fotografías que comprobaron lo positivo y negativo del experimento; al final se establecieron las conclusiones y recomendaciones.

Palabras Clave: Aplicación Móvil – Reconocimiento facial - Aplicación Android - Lenguaje de Programación - Microsoft Azure

Abstract

In the research of the thesis work "Design and creation of a mobile application for the registration of income of people with facial recognition implementing Microsoft Azure", the objectives of designing, developing and implementing the database and Web Services were considered. The project took into account the knowledge of several authors specialized in facial authentication. The inductive and deductive methods gave the option to investigate from the particular to the general, while the analytical part managed to identify the steps for the implementation of the project. The research was explorative and descriptive; the focus was qualitative - experimental achieved through the technique of observing the participants in the mobile application experiment. In the process photography was used as the primary source of the experiment, the participants who took part in the tests had the predisposition to contribute with time for the development of the proposal. In the creation of the App we had access to facial recognition theories and in the tests the mobile phone was used for the sequential taking and recording of the photographs, which gave rise to the analysis of facial expressions and facial expressions. That led to the analysis of each of the photographs of the participants' faces. The purpose of the proposal was to implement the application, using the photographs that proved the positive and negative aspects of the experiment; in the end the conclusions and recommendations were established.

Keywords: Mobile Application - Face Recognition - Android Application Programming Language - Microsoft Azure

Índice de Contenido

Declaración expresa de autoría y sesión de derechos de autor ii
Dedicatoria iii
Agradecimiento iv
Resumen v
Abstract vi
Índice de Contenido vii
Índice de Tablas ix
Índice de Figuras x
Introducciòn 1
CAPÌTULO I. MARCO TEÓRICO CONCEPTUAL 3
1.1 Antecedentes de la investigación 3
1.2 Planteamiento del problema de investigación 4
1.2.1 Formulación del problema 4
1.2.2 Sistematización del problema 5
1.3 Objetivo General 5
1.3.1 Objetivos Específicos 5
1.4 Justificación 5
1.5 Marco de referencia de la investigación 7
1.5.1 Componentes para el reconocimiento facial 7
1.5.2 La tecnología y las aplicaciones del reconocimiento facial 8
1.5.3 El reconocimiento facial por medio de la tecnología 10
1.5.4 Software y sus definiciones 13
1.5.5 *App* y sus definiciones 14
1.5.6 Sistema Operativo Android 15
1.5.7 Desarrollo de la App.- Paquetes para el desarrollo ksoap2 16
1.5.8 Comunicación con el *Web Services* 17
1.5.9 Lenguaje de Programación *JAVA* 18
1.5.10 SQL server 19
1.5.11 En programación Lenguaje C# 20
1.5.12 Exposición sobre .Net 21
CAPÍTULO II: MARCO METODOLÓGICO 24
2.1 Tipo de diseño, alcance y enfoque de la investigación 24

2.2 Método de la investigación 25
2.3 Unidad de análisis, población y muestra 26
2.4 Variables de la investigación 26
2. 5 Fuentes, técnicas e instrumentos para la recolección de la información 26
2.5.1 Metodología de trabajo de la aplicación 26
CAPÍTULO III: RESULTADOS Y DISCUSIÓN 27
3.1 Análisis de la situación actual 27
3.2 Diseño de la aplicación móvil: Reconocimiento facial 28
3.3 Presentación de resultados y discusión 29
CAPÍTULO IV: PROPUESTA 31
4.1 Requerimientos Funcionales del Proyecto 31
4.2 Fase de Diseño 31
4.2.1 Diseño de la Base de Datos 31
4.2.1.1 Modelo relacional 32
4.2.2 Análisis de usuabilidad en el diseño de las pantallas 36
4.3 Fase de Desarrollo 41
4.3.1 Desarrollo de Web Services 41
4.3.2 Desarrollo en Visual Studio 2017 41
4.3.3 Código Fuente del Web Services 42
4.3.4 Métodos del Wed Services 43
4.3.5 Desarrollo de Aplicación Android 45
4.3.6 Pantallas de la aplicación 46
4.4 Fase de Pruebas 52
4.4.1 Prueba 1 53
4.4.2 Prueba 2 61
4.4.3 Prueba 3 69
4.4.4 Prueba 4 77
4.4.5 Prueba 5 85
4.4.6 Prueba 6 93
Conclusiones 100
Recomendaciones 101
Bibliografía 102
Anexos 1

Índice de Tablas

Tabla 1 Comparación de atributos de las tecnologías 27
Tabla 2 Aplicación Android 29
Tabla 3 Hardware de desarrollo 30
Tabla 4 Tbl_Sexo 34
Tabla 5 Tbl_Centro 34
Tabla 6 Tbl_Sala 34
Tabla 7 Tbl_Persona 35
Tabla 8 Tbl_Acceso 35

Índice de Figuras

Figura 1 Biometría facial 11
Figura 2 Delimitación de la cara y la comisura 11
Figura 3 Reconocimiento de la línea longitudinal 12
Figura 4 Proceso de autentificación facial 13
Figura 5 Diseño de la aplicación móvil para el reconocimiento facial 28
Figura 6 Modelo Relacional 32
Figura 7 Modelo Relacional Extendido 33
Figura 8 Acceso GPS y elección de la sala 36
Figura 9 Tomar foto 37
Figura 10 Reconocimiento facial 38
Figura 11 Éxito por reconocimiento y grabas acceso 39
Figura 12 Error por el reconocimiento facial 40
Figura 13 Visual Studio 2017 41
Figura 14 Web Services WS recognitionface 42
Figura 15 Salas del centro 43
Figura 16 Resultado 43
Figura 17 InsertRegistro 44
Figura 18 Resultado 44
Figura 19 Desarrollo en Android Studio 45
Figura 20 Pantalla 1 – activity_gps 46
Figura 21 Pantalla 2 – activity_foto 47
Figura 22 Pantalla 3 – activity_proceso 48
Figura 23 Pantalla 4 – activity_exito 49
Figura 24 Pantalla 5 – activity_error 50
Figura 25 Proceso de carga de personas a la nube de Azure 51
Figura 26 Prueba 1 - Pantalla 1 – Ubicación GPS 53
Figura 27 Prueba 1 - Pantalla 2 – Tomar foto 54
Figura 28 Prueba 1 - Pantalla 3 – Procesamiento de foto 55
Figura 29 Prueba 1 - Pantalla 4 – Detectando la cara 56
Figura 30 Prueba 1 - Pantalla 5 – Identificación de persona 57

Figura 31 Prueba 1 - Pantalla 6 – Identificando persona..... 58
Figura 32 Prueba 1 - Pantalla 7 – Persona Identificada 59
Figura 33 Prueba 1 - Pantalla 8 – Identificación exitosa..... 60
Figura 34 Prueba 2 - Pantalla 1 – Ubicación de GPS..... 61
Figura 35 Prueba 2 - Pantalla 2 – Tomar foto 62
Figura 36 Prueba 2 - Pantalla 2 – Procesamiento de la foto 63
Figura 37 Prueba 2 - Pantalla 4 – Detectando la cara..... 64
Figura 38 Prueba 2 - Pantalla 5 – Identificación de persona 65
Figura 39 Prueba 2 - Pantalla 6 – Identificando persona..... 66
Figura 40 Prueba 2 - Pantalla 7 – Persona identificada 67
Figura 41 Prueba 2 - Pantalla 8 – Identificación exitosa..... 68
Figura 42 Prueba 3 - Pantalla 1 – Ubicación de GPS..... 69
Figura 43 Prueba 3 - Pantalla 2 – Tomar foto 70
Figura 44 Prueba 3 - Pantalla 3 – Procesamiento de la foto 71
Figura 45 Prueba 3 - Pantalla 4 – Detectando cara 72
Figura 46 Prueba 3 - Pantalla 5 – Identificación de persona 73
Figura 47 Prueba 3 - Pantalla 6 – Identificando persona..... 74
Figura 48 Prueba 3 - Pantalla 7 – Persona identificada 75
Figura 49 Prueba 3- Pantalla 8 – Identificación exitosa..... 76
Figura 50 Prueba 4 - Pantalla 1 – Ubicación de GPS..... 77
Figura 51 Prueba 4 - Pantalla 2 – Tomar foto 78
Figura 52 Prueba 4 - Pantalla 3 – Procesamiento de foto 79
Figura 53 Prueba 4 - Pantalla 4 – Detectando cara 80
Figura 54 Prueba 4 - Pantalla 5 – Identificación de persona 81
Figura 55 Prueba 4 - Pantalla 6 – Identificando persona..... 82
Figura 56 Prueba 4 - Pantalla 7 – Persona identificada 83
Figura 57 Prueba 4 - Pantalla 8 – Identificación exitosa..... 84
Figura 58 Prueba 5 - Pantalla 1 – Ubicación GPS 85
Figura 59 Prueba 5 - Pantalla 2 – Tomar foto 86
Figura 60 Prueba 5 - Pantalla 3 – Procesamiento de la foto 87
Figura 61 Prueba 5 - Pantalla 4 – Detectando cara 88
Figura 62 Prueba 5 - Pantalla 5 – Identificación de persona 89
Figura 63 Prueba 5 - Pantalla 6 – Identificando persona..... 90

Figura 64 Prueba 5 - Pantalla 7 – Persona identificada 91
Figura 65 Prueba 5 - Pantalla 8 – Identificación exitosa..... 92
Figura 66 Prueba 6 - Pantalla 1 – Ubicación GPS 93
Figura 67 Prueba 6 - Pantalla 2 – Toma de foto..... 94
Figura 68 Prueba 6 - Pantalla 3 – Procesamiento de foto 95
Figura 69 Prueba 6 - Pantalla 3 – Detectando cara 96
Figura 70 Prueba 6 - Pantalla 5 – Identificación de persona 97
Figura 71 Prueba 6 - Pantalla 6 – Identificando persona..... 98
Figura 72 Prueba 6 - Pantalla 7 – Error persona no identificada 99

Introducciòn

El avance y desarrollo de la tecnología permite que aparezcan cada vez màs opciones digitales que favorezcan a la realización de actividades en las empresas y en la vida personal del ser humano. Con la aparición del sistema de reconocimiento facial realizado a través de las diversas aplicaciones en teléfonos móviles como los Smartphone, se puede identificar a los usuarios, en los que una cámara integrada pueda detectar la cara. Este tipo de tecnología digital poco a poco va convirtiéndose en una herramienta útil para agilizar los diferentes process en los que se requiere la identificación inmediatamente de la persona, evitando posibles suplantaciones o fraudes en instituciones pùblicas y privadas.

El reconocimiento biométrico, en el caso especifico de instituciones escolares tiene la ventaja de identificar si la persona ha sido registrada más de una vez en un sistema o si es quien afirma ser, por lo que se convierte en un instrumento veraz para la verificación personal de los participantes en distintas áreas escolares; lo mismo ocurre en las empresas que requieren identificar si sus colaboradores asisten puntualmente en el horario que se les ha fijado, por lo tanto se convierte en una herramienta útil y eficiente para las empresas que requieran conocer la participación de sus colaboradores.

El Sistema de Reconocimiento Facial lo compone un software que funciona de manera automatica de identificación biométrica que verifica a una persona en la que compara en tiempo real, el análisis de cada uno de los modelos humanos que se basan en cada uno de los contornos y rasgos faciales. Lo más importante es que al reconocer facialmente a las persona lo realiza por medio de un dato biométrico, el mismo que debe coincidir con la identificación de la persona. El sistema de reconocimiento facial introduce la imagen que se codifica de manera automática y que empieza a reconocer los perfiles de cada uno de los participantes dentro del sistema de registro y lo califica de ser correcto y lo rechaza sino pertece a la imagen.

El sistema de reconocimiento facial identifica en el mismo momento a una persona, el proceso comienza con un análisis de las características biométricas del rostro, en el que se incluyen procesos matemáticos y algoritmos que coincidan con el contorno del rostro por lo que se requiere seguir un proceso, compuesto por fases; primeramente es la fase de detección, utilizando una imagen del rostro del usuario, utilizando una cámara fotográfica o puede ser a través de una cámara de vídeo.

Después de la identificación del rostro, empieza el proceso de la imagen para extraer toda la información biométrica, en la que incluye la alineación de la cara, sus propiedades geométricas de cada una de las fotografías que contiene la información biométrica y se almacena un patrón biométrico facial, llegando a la fase en que se compara y se coteja toda la información biométrica que s encuentra almacenada en la base de datos, siendo así empoeza con la comparación 1, pasa a la siguiente y así sucesivamente en la que se obtenienen los resultados que indican el porcentaje de similitud y que están almacenados en cada una de la base de datos. Obteniendo el porcentaje de similitud aparece el visto de aceptación.

La investigación tiene como finalidad reconocer los aspectos positivos del reconocimiento facial para empresas pùblicas y privadas, compañías o instituciones escolares. Consta de los siguientes capítulos:

Capìtulo I: Planteamiento de la situación problemática, línea y sublìnea de investigación, objetivos general y específicos, delimitación y justificación.

Capitulo II: Marco teórico con las conceptualizaciones de tecnología, reconocimiento facial, Aap, programación, entre otros temas,

Capìtulo III: Marco metodológico con el enfoque, métodos, población, muestra, técnicas e instrumentos de investigación.

Capìtulo IV: Propuesta de investigación, las características del desarrollo.

CAPÌTULO I. MARCO TEÓRICO CONCEPTUAL

1.1 Antecedentes de la investigación

Para el desarrollo del trabajo se consultó varias investigaciones entre las que constaron tesis, artículos científicos, libros y diferentes publicaciones relacionadas con el reconocimiento facial por medio del uso de dispositivos móviles. En México, en la Universidad Iberoamericana el reconocimiento de rostros, utilizando análisis de componentes principales, limitaciones del algoritmo, con el método que requiera una muestra de los pixeles que configura cada rostro, lo cual redituará en una menor cantidad de almacenamiento y tiempo de procesamiento computacional (Villegas, 2018).

El tema se vincula con el hecho de perseguir objetivos orientados al reconocimiento de las personas por la observación del rostro y de sus principales características que contribuyan a evitar fraudes o estafas, así como agilar procesos de cumplimiento de acciones en el que se identifiquen a las personas, considerando los pixeles en cada fotografía que identifica al individuo con sus características faciales, generando la ventaja de poco almacenamiento computacional.

En la Universidad Politécnica Nacional del Ecuador, se desarroló un tema de reconocimiento facial y localización GPS, en un raspberry PIB Plus, el sistema presenta una eficiencia y eficacia aceptable, se adicionó un código de programación para evitar errores que inicidan en que el sistema falle por desconexión de internet o por desconexión de la cámara (Espinoza, 2018). La investigación se relaciona con el sistema de seguridad, vincula al reconocimiento del rostro y el desarrollo del *App* para conservar los datos, a pesar de las fallas que puedan ocurrir en la cámara y en el internet.

1.2 Planteamiento del problema de investigación

Las organizaciones sin fines de lucro (OSAL) tienen como finalidad llevar a cabo actividades de filantropía apoyada en los aportes económicos de personas que cuentan con los recursos financieros para brindar ayuda social a individuos de escasos recursos económicos en los campos de educación vivienda y alimentación; sin embargo el mayor problema que se presenta es brindar a los aportantes el seguimiento o registro de quienes asisten a las capacitaciones, detectando problemas en el monitoreo que se llevan a cabo en estas instituciones.

Por lo general, las ONG sin fines de lucro no cuentan con suficientes recursos económicos para invertir en programas de registro que certifiquen las visitas, participación y cumplimiento de horarios, lo que evidencia la carencia de un método que controle puntualmente a los escolares en los programas de estudio en determinadas OSAL, lo que pone en riesgo a la continuidad de recibir las diferentes aportaciones que permiten el impulso de estas instituciones.

En síntesis, en determinadas OSAL tienen limitado número de equipos tecnológicos como tabletas, teléfonos inteligentes que permiten el seguimiento y registro de los participantes en los diferentes cursos de capacitaciones que evidencien la adecuada inversión de los recursos recibidos. La propuesta de desarrollar una aplicación móvil (*App)* que tenga la opción de reconocimiento facial conlleva a crear evidencias del registro de los participantes de manera oportuna para las ONG sin fines de lucro y todas las personas que tengan interés en el control de los participantes.

1.2.1 Formulación del problema

¿De qué manera el diseño y creación de una aplicación móvil *App* de autenticación facial, controlará la asistencia de usuarios en una ONG sin fines de lucro?

1.2.2 Sistematización del problema

¿Cómo el diseño de la aplicación móvil, la base de datos y un Web Services se comunicará entre la aplicación y la base de datos?

¿De qué manera se puede desarrollar la base de datos, el Web Services y la aplicación móvil para el reconocieinto facial?

¿Qué aspectos contiene la implementación de la aplicación móvil, la base de datos y el Web Services para el reconocimiento facial?

1.3 Objetivo General

Crear una aplicación móvil *App* para autenticación de usuarios implementando Microsoft Azure

1.3.1 Objetivos Específicos

Diseñar la aplicación móvil, la base de datos y un Web Services que se comunique entre la aplicación y la base de datos.

Desarrollar la base de datos, el Web Services y la aplicación móvil.

Implementar la aplicación móvil, la base de datos y el Web Services.

1.4 Justificación

El presente estudio se justifica desde varios puntos de vista, entre ellos la parte teórica, en la que se aplica la recopilación bibliográfica de varios libros, de diferentes autores que son reconocidos en el mundo de la multimedia y el diseño web por la redacción de textos, investigaciones y aportaciones académicas relacionados con el reconocimiento facial, publicados en los últimos cinco años.

En la parte metodológica, el estudio fue experimental por las diferentes pruebas que se desarrollaron en la creación del *App* para el reconocimiento facial y registrar la asistencia en las diferentes capacitaciones en las determinadas OSAL. El proceso del reconocimiento facial corresponde a un sistema que permite mantener la seguridad de evaluar las características de cada una de las personas por medio de sus rasgos faciales de cada individuo y permite mantener su información resguardada y lo hace con la precisión de la tecnología en el que se da la identificación biométrica. El proceso experimental contó con la colaboración de niños y niñas participantes del programa de la OSAL, se contó con le permiso confirmado de los padres y se incluyó el experimento del autor de la investigación para el reconocimiento facial.

En la parte pràctica, el uso del *App* servirá para brindar seguridad en el registro continuo de las personas que participan en los diferentes programas de capacitaciòn de estudios en las diferentes ONG sin fines de lucro, además el proyecto puede utilizarse como referente en otras instituciones no ONG que no cuenten con este recurso de uso de medios digitales. La delimitación de la investigación tiene las siguientes características: Campo Empresarial, área del desarrollo tecnológico, aspectos relacionados con la Web y Multimedia, considerando la delimitación geo – temporo espacial a la ciudad de Guayaquil, en el Ecuador, siendo los aspectos temporo – espacial, desde el año 2018 .2019.

Entre las líneas de investigacion corresponde a la investigacion, gestión del conocimiento, tecnología de la informática y sus comunicaciones. La sublínea corresponde a las tecnologías de la información y las comunicaciones al servicio de la efectividad de los sistemas educativos y de gestión para la sociedad ecuatoriana.

1.5 Marco de referencia de la investigación

1.5.1 Componentes para el reconocimiento facial

El reconocimiento a las personas se vincula principalmente con la observación primaria cara a cara cuando se establece el primer encuentro, en el que se identifica por medio de las características faciales y se construye el precepto visual del rostro, el mismo que es percibido y reconocido como totalmente nuevo, todo por el almacenamiento de memorias de caras o lo que se conoce como nodos de identidad personal, acompañado del sentimiento de familiaridad que es corresponsable con las expresiones perceptuales faciales, cuando se estudia la memoria de rostros, se requiere tomar en cuenta las características de las caras, las mismas que permiten que sea recordado por quienes en determinado momento lo reconocen; todo depende de las diferencias entre las caras que dependen de matices, porque en esencia el rostro es igual y corresponde al analizado (Broche & Rodríguez, 2014).

El rostro es la primera característica que identifica al ser humano, siendo los rasgos faciales de los ojos, nariz, cejas, pestañas y las facciones de la cara que los diferencian unos de los otros. El rostro humano puede dividirse en tres áreas que dan la opción a otra persona de determinar el temperamento y el modo de ser de quien se observa, debido a estas características el rostro se ha convertido en una herramienta para el reconocimiento de la persona (Zuñiga, 2014).

Se considera que una fotografía del rostro da la oportunidad a otros de reconocer a alguien cuando lo conoce, en algunos casos es el factor clave para identificar si es la persona que buscan, en la parte pericial sirve de base para la resolución de casos en los que solo se cuenta con una dibujo facial o una fotografía del rostro, lo mismo ocurre en los documentos de identidad como cèdula, carnet, pasaporte, pases personalizados, entre otros.

Las características diferenciadoras en el rostro se sitúan en los laterales de la cara, las cejas compuestas por vello situado en la parte superior justo encima de los ojos, su localización puede estar modificada por la expresión; los ojos, es uno de los rasgos biométrico muy importante para el reconocimiento, están situados en la mitad superior de la cara y lo componen las pestañas, párpados y el globo ocular, que conforman la córnea, iris y pupila. La nariz se ubica en el centro de la cara y tiene dos orificios nasales suelen ser un buen punto característico cuando se miden distancias. La boca situada en la parte inferior de la cara, es otro rasgo que facilita información del individuo. Los labios son el componente que siempre está visible y define el aspecto de la boca (Dominguez, 2015). Allí radica la importancia de considerar el reconocimiento facial como punto de partida para el desarrollo de aplicaciones digitales que aporten al registro de participaciones en eventos

1.5.2 La tecnología y las aplicaciones del reconocimiento facial

La tecnología permite el uso de dispositivos como los teléfonos inteligentes para el desarrollo de aplicaciones móviles como el reconocimiento facial, las mismas que brindan opciones de control de actividades, considerando la seguridad de mantener la información actualizada de aquellos eventos que son indispensables y que evaluan los beneficios, entre ellos que hayan suplantaciones de identidad, acceso a mayor precisión de la identificación y se evita que existan manipulaciones de los datos del usuario real, denominàndose innovación tecnológica.

Las actividades diarias requieren de innovación en los servicios e información para alcanzar los objetivos de manera directa y evidencia la realidad de lo que acontece, por lo que los dispositivos como los teléfonos móviles permiten realizar actividades que ejecutan acciones beneficiosas para quienes las llevan a cabo. Los dispositivos móviles ayudan a llevar a cabo las actividades cotidianas que maravillan a quienes la utilizan por todo lo que son capaces de hacer (Atkinson, 2015).

En la utilización de la tecnología se reconocen ventajas que aportan al desarrollo de acciones que brindan la oportunidad de dar confianza en cada una de las aplicaciones, brindando fiabilidad y seguridad al contar con unos cuantos programadores, evaluando el mismo trabajo simultáneamente que da la opción de detectar los errores y corregir con anterioridad (Martín-Ávila & López, 2015). Lo mismo se puede realizar cuando se diseñan y crean aplicaciones del l*App* para los teléfonos móviles que tienen como finalidad detectar en las pruebas preliminares los posibles errores y de esa forma tener la oportunidad de corregir cada una de las fallas que puede presentar la aplicación y de así obtener las herramientas necesarias para innovar los procesos, entre ellos registros de asistencias de estudiantes a las diferentes actividades que realizan como parte del proceso de aprendizaje en las diferentes instituciones educativas.

En la evolución de las tecnologías, quienes usan con regularidad aparatos y dispositivos móviles buscan sacar partido en los que se actualice las diferentes labores que llevan a cabo, entre las cuales se reconoce que fomentan la seguridad de los datos, participan directamente con otras personas que deseen verificar la identidad para evitar suplantaciones, interactúan constantemente de la información que tienen y por lo general, segmentan el grupo con el que se busca trabajar directamente para tener fiabilidad de las acciones que se ejecutan a nivel personal.

En síntesis, la utilización de los datos de la persona como fuente de información primaria da la oportunidad de orientar y personalizar las distintas actividades que realizan, lo que conlleva a que se fortalezca la personalización y el acompañamiento perenne, brindando la oportunidad de desarrollar otras actividades que favorezcan a las instituciones por la personalización de su experiencia en todo lo que ejecutan y se lo hace estableciendo la omnipresencia y la coherencia de la experiencia de saber lo que realizan y lo comunican, sin que quede la duda de que no es la persona a la que está dirigida el reconocimiento facial por medio un dispositivo móvil (Dufrosne, 2015).

La seguridad que brindan los dispositivos móviles al implementar la *App* del reconocimiento facial en el que se puede observar las acciones que los usuarios realizan en forma simultánea y puede convertirse en una prueba que genera confiabilidad en todo lo que comunica, por lo que es una herramienta útil, porque establece en tiempo real que puede inmediatamente verificar con precisión las asistencia a determinadas actividades como las escolares en el que se mejore el registro de participación, se evita la manipulación de parte del escolar porque solo debe ponerse frente a la cámara y empieza el proceso de reconocimiento, además se evidencia menos posibilidad de que se roben la identidad.

1.5.3 El reconocimiento facial por medio de la tecnología

La aparición del sistema de reconocimiento facial por medio de aplicaciones en teléfonos móviles como los Smartphone, surgió ante la necesidad de realizar la identificación de los usuarios, con la ayuda de la cámara integrada, a través de la fotografía se detectan las caras, convirtiéndose en una herramienta que ejecuta la acción de identificar inmediatamente a la persona, dando la ventaja de evitar y prevenir, en ciertos casos posibles suplantaciones o fraudes.

El reconocimiento biométrico tiene varias ventajas bien definidas, entre las más importantes es que permite determinar si una persona ha sido registrada más de una vez en un sistema o si es quien afirma ser, por lo que se convierte en una herramienta veraz para la verificación personal de los participantes en distintas áreas laborales o escolares (Ortega, Alonso, & Coomonte, 2015) Con la aplicación en un teléfono móvil, el proceso de autenticación se vuelve rápido, impidiendo que surja la idea de que reemplacen con otra persona, así mismo, en determinado momento puede convertirse en un medio disuasivo que advierte a que una persona desee registrarse en más de una ocasión.

La autenticación facial reconoce los puntos preponderantes de la cara, tales como nariz, cejas, pómulos, mentón, orejas, boca, frente, ojos, permitiendo identificar a la persona de manera inmediata.

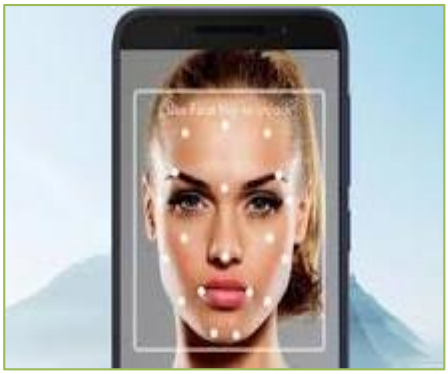

Figura 1 Biometría facial
Tomado de Alcatelmobile

Al momento de evaluar e identificar las características se consideran los rasgos de cada una de las partes del rostro; en las cejas se considera la delineación, separación y grosor, el mentón permite evidenciar lo ancho y la altura hasta los labios inferiores, en la nariz se toma en cuenta la medida, si es larga o corta, si es delgada o ancha y la altura, en frente se observa si es amplia y lo alto, en los párpados se puede establecer a la distancia entre las cejas y las pestañas, en la boca es la anchura, la forma de los labios, su comisura y en la cara se ven las líneas longitudinales.

Las líneas longuitudianales al momento del reconocimiento facial es clave para lograr que la aplicación funcione.

Figura 2 Delimitación de la cara y la comisura
Tomado de scielo.org

En la imagen se evidencia la forma en que normalmente se ejecuta el reconocimiento facial, consideran la línea longitudinal del rostro de manera imaginaria en el centro de la cara. Así la comisura de la boca si esta hacia la derecha o izquierda, o si tiene forma de corazón, la distancia entre los ojos y las cejas o a su vez la altura de la frente.

Figura 3 Reconocimiento de la línea longitudinal
Tomado de docentes.unal.edu.co

El uso del software para reconocimiento facial presenta soluciones fiables mediante los diferentes dispositivos móviles y en ocasiones pueden utilizarse dispositivos portátiles, con actualizaciones del rostro, considerando que el ser humano cambia a medida que pasa el tiempo por cuestión de la edad o en ocasiones porque decidieron realizarse alguna cirugía estética o muchos hombres se dejan crecer la barba, en estos casos se consideran la toma de una nueva fotografía que autentifique que esa imagen corresponde a la persona que se debe reconocer.

El reconocimiento facial es una técnica por el que se reconoce a una persona a partir de una imagen o fotografía. Se utilizan programas de cálculo que analizan imágenes de rostros humanos; los aspectos clave empleados para la comparación se encuentran mediciones como la distancia entre los ojos, la longitud de la nariz o el ángulo de la mandíbula (Instituto Nacional de Ciberseguridad, 2015).

La importancia del reconocimiento facial biométrico como técnica digital que utiliza una *App* que pasa por varias fases, que comienza desde la detección de la cara de forma conjunta, esto es ojos, mentón, labios, cejas, parpados, oreja, forma lineal del rostro, las mismas que son procesadas, luego se llega al reconocimiento supervisado, que al terminar la autentificación la certifica mediante un visto bueno ☑ o con una ☒ cuando es incorrecta.

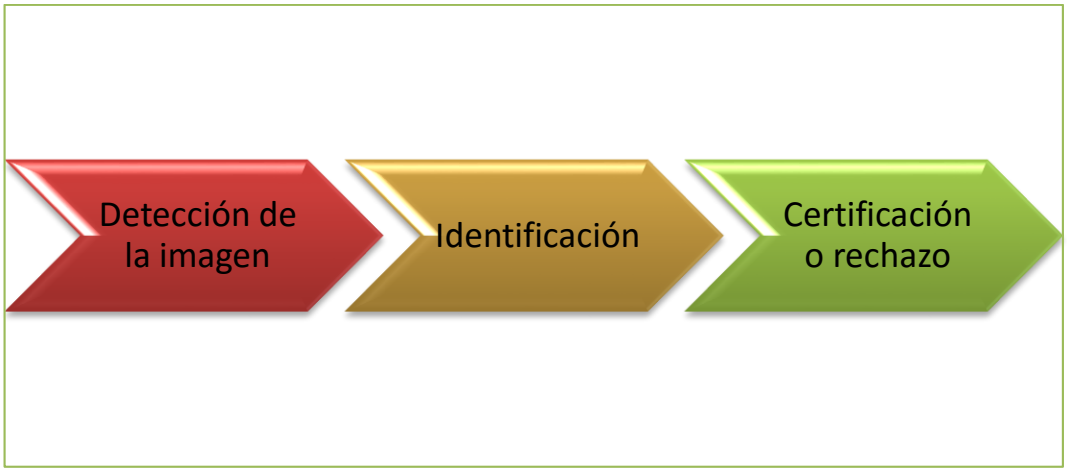

Figura 4 Proceso de autentificación facial
Elaborado por Alex García

1.5.4 Software y sus definiciones

En la tecnología, el software tiene varios conceptos relacionados con la funcionalidad que les brinda a los usuarios de equipos como las computadoras, con la gran ventaja de que hay muchos software libres en el mercado. Se reconoce que el software libre ha penetrado en prácticamente todos los nichos informáticos, convirtiéndose en una opción habitual en varios de ellos (servidores, sistemas empotrados, o últimamente, móviles)" (Gonzalez J. , 2018). Por lo tanto, el software forma parte de cualquier sistema informático que tiene muchos componentes que dan la posibilidad de que se puedan realizar diversas tareas que son específicas y apoyan al diseño de proyectos que benefician a los empresarios y otras entidades públicas o privadas.

El software de fuentes abiertas, que tiene funcionalidad de estudiar el funcionamiento de los programas y de cómo llevan a cabo su función, permitiendo así la posibilidad de detectar posibles errores, solucionándolos de la manera más rápida y segura posible". (Gonzalez & Riera, 2018). Lo que corresponde la parte lógica de todo el sistema operativo para que realice lo que piden de manera directa y ejecutarla de acuerdo a las órdenes, considerando los aspectos que automatizan las acciones con la finalidad de aumentar la calidad de los aspectos que se ejecutan, entre ellos el proceso de reconocimiento facial por medio del teléfono móvil, lo que logra es automatizar aquellas tareas repetitivas de control.

La tecnología influye en la vida diaria y con ellos el uso de software se vuelve cada vez más amplio por lo positivo en el desarrollo de soluciones a los problemas de manera eficaz y eficiente de quienes lo utilizan, su uso se ha convertido en cotidiano en la parte personal y laboral. La innovación permite el impulso de conquistar nuevos espacios, por lo que mejorar el desarrollo de software da la oportunidad de brindar otros diagramas, códigos para acceder a nuevas tecnologías, es asi que el Desarrollo de Software Dirigido por Modelos MDD se ha convertido en un nuevo paradigma de desarrollo software. MDD promete mejorar el proceso de construcción de software basándose en un proceso guiado por modelos y soportado por potentes herramientas" (Pons, Giandini, & Pérez, 2014).

1.5.5 *App* y sus definiciones

La definición del término *App* se refiere la abreviatura de la palabra en inglés *application,* lo que identifica que *App* es un programa, con características puntuales; son aplicaciones destinadas a tabletas como el iPad o equipos Android o a teléfonos del tipo smartphone como el iPhone, son dinámicas, dependen de Internet para funcionar, generalmente ocupan pocos MB. Las *App e*stán disponibles en tiendas virtuales, se puede acceder directamente a ellas, se las puede encontrar en el navegador del dispositivo, se registra y se descargan las aplicaciones y pueden realizarlas desde el propio dispositivo de forma gratuita (Gutierrez, 2018).

El crecimiento y desarrollo de dispositivos móviles como los teléfonos Smartphone y tabletas permitieron obtener aplicaciones *App* con más funciones y formatos con funcionalidades que pueden ayudar a realizar diversas tareas. Las apps móviles (teléfonos y tablets), se han convertido en un elemento cotidiano de una población activa, consumista y que se interrelaciona con los demás a través de los terminales. Muchas de estas aplicaciones, ya están preinstaladas y funcionan en los móviles o tablets cuando salen a la venta, pero la mayoría de ellas deberán descargarlas e instalarlas los usuarios" (Mobile Marketing Móvil, 2015).

El desarrollo de las *App* requiere de un proceso basado en un código base sencilla, que permite el dinamismo en las aplicaciones en los equipos móviles como teléfonos o tabletas. Es importante señalar que las *App* requieren del internet para su funcionamiento, lo que en ocasiones si limita la utilización del uso. La utilización de las *App* en las empresas les permite identificar sus servicios de manera eficiente y eficaz, dando la oportunidad de agilizar el trabajo y en casos particulares de identificación de procesos y procedimientos en las empresas que lo utilizan.

1.5.6 Sistema Operativo Android

Dentro de las definiciones del sistema operativo Android se destaca la importancia en el diseño de diferentes programas que generalmente se los emplea en dispositivos móviles como teléfonos o tabletas, "Android es un sistema operativo que fue inicialmente creado para teléfonos inteligentes (Smartphones) por la empresa Android Inc. Y comprado en 2005 por la empresa Google. Su núcleo, está basado en Linux, convirtiéndolo en sistema multiplataforma, libre y gratuito" (Zanini & Hereter, 2016).

Entre los requerimientos se reconoce que "Para la instalación de Android Studio en Windows se debe tener Windows Vista 7.8 o 10 en 32 o 64. En cuanto a la memoria RAM, se requiere 2 GB como mínimo, aunque 8GB es lo más recomendable" (Zanini & Hereter, 2016). Al tomar en cuenta las especificaciones de los programas.

En el desarrollo de programas, Android forma parte de un sistema operativo que tiene tres tipos que forman parte de la misma versión, entre ellas la comercial que se la conoce como KitKat, la que corresponde a los fabricantes, que de igual manera es comercial, tiene que ver con la versión y subversión, la que corresponde al desarrollador con nivel del API, en el sistema Android (Invarato, 2015). En lo que se refiere a *Android Studio*, es el ambiente de desarrollo de Android.

1.5.7 Desarrollo de la App.- Paquetes para el desarrollo ksoap2

En el desarrollo de las diferentes aplicaciones se debe considerar que el *Ksoap2-android* tiene una biblioteca que es ligera y permite la comunicación de los datos en una plataforma Android, mientras que SOAP es uno de los protocolos para la comunicación de datos más usados auspiciado por la W3C, existiendo multitud de servicios web implementados bajo esta tecnología" (Pereira, 2018). De allí se parte para el uso de la biblioteca *Ksoap2* en Android con la gran biblioteca de clientes e incluso en determinando momento se puede utilizar en las plataformas de la biblioteca *Java* considerando el mismo lenguaje de programación

En el desarrollo del *App,* las librerías relacionadas con las aplicaciones Android *ksop2* cuenta con un mantenimiento perenne, por lo que Ksoap2-android se compone de un módulo que sirve para "parsear" XML, que permite la serialización y otro que se utiliza para el transporte y da seguridad en el proceso en la aplicación. Es importante considerar que este tipo de librería es ligera y eficiente para consumir servicios *Soap* desde *Android.* Por lo general, sigue pasos específicos en los que se define lo que se requiere (*request*), continuar con el mensaje y el procesamiento, por medio de la configuración de un sobre (*envelope*), luego se determina la manera de hacer la llamada y la forma de recoger los datos, los mismos que se procesarían de acuerdo a la función utilizada como es el caso de la *Ksoap2* que usa XML. Uno de los puntos que se debe considerar en este tipo de programación es que se requiere del internet para poder acceder al desarrollo de los diferentes programas, por lo tanto lo primero que debe considerarse es poseer la conexión a internet.

1.5.8 Comunicación con el *Web Services*

Los *Web Services* es una función que utilizan los equipos para operar en las aplicaciones de software independiente, son protocolos que generalmente, intercambian aplicaciones de software que son diseñadas y desarrolladas en lenguajes de programación diferentes y pueden utilizar los servicios web para comunicar los datos en redes de ordenadores. Sirven para intercambiar servicios entre diferentes compañías y se los puede combinar y proveer servicios integrados, son escribir programas con costos elevados, que se comunican de aplicación en aplicación.

La WSDL (*Web Services Description Language*) indica que “Es un protocolo basado en XML que describe los accesos al Web Service. Es el manual de operación de este, porque indica cuáles son las interfaces que provee el Servicio web y los tipos de datos necesarios para su utilización” (Acevedo, 2018). Su importancia radica en que describe la funcionalidad de los servicios que se integran y comunican los datos por medio de formatos de mensajes como es el *Soap.*

El término Web Services describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuáles son los servicios disponibles. Uno de los usos principales es permitir la comunicación entre las empresas y entre las empresas y sus clientes. Los Web Services permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos Sistemas de Información (Saffirio, 2018).

1.5.9 Lenguaje de Programación *JAVA*

El desarrollo de la tecnología del internet, trajo consigo diversos tipos de programación, entre ellos el *JAVA*, que es el lenguaje de programación y un entorno de ejecución de programas escritos en java, considerado un compilador, que convierte el código fuente en instrucciones a nivel de máquina, el compilador java traduce el código fuente java en instrucciones que son interpretadas por la máquina virtual de java. Lo importante es reconocer que *Java* es un lenguaje de programación para internet que es interpretado, también lo utiliza la *World Wide Web* en particular (Cruz, 2018).

En el desarrollo de programación, Android utiliza el lenguaje Java con su compilador, por lo que soporta la programación orientada por eventos, Android la utiliza en todas sus facetas al momento de ejecutarlo desde el inicio (Invarato, 2015), Java permite entender de manera precisa cada evento que se programe y da la opción de brindar agilidad en los procesos que se encaminen al apoyo de programación dinámica para facilitar al usuario el ingreso de datos dentro de las facetas que lo requieran, utilizando la lógica con solo pulsar un botón en la cámara fotográfica o en el teléfono móvil.

Entre otras definiciones *Java* es uno de los lenguajes de programación más importantes, en el año 1995 se la comercializo por *Sun Microsystems* como una plataforma informática. Se conoce que existen muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado, se conoce que una de las características es la rapidez, seguridad, confiabilidad y fiabilidad por lo que se las instala en los diferentes equipos móviles y portátiles, incluso en consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, el uso de Java es importante como parte de la programación de los diferentes equipos que permiten realizar las acciones para las programaciones correspondientes (Oracle Corporation, 2017).

La codificación de las instrucciones que debe realizar en comunicación directa con el equipo tiene realacion con el reconocimiento que *Java* es un lenguaje de programación muy riguroso en su sintaxis, pero en el tema de indentación es pobre, porque puede escribir el código en una sola línea y puede ser funcional. También el código puede ser escrito en la misma jerarquía, en otras palabras sin respetar los espacios para distinguir que bloques de código pertenece a la clase, a los métodos de control (Guamán, Guamán, Erreyes, & Torres, 2017).

1.5.10 SQL server

El *Structured Query Language SQL*, es el lenguaje relacional de programación, es diferente de otros lenguajes computacionales como C, COBOL y Java, los cuales son de procedimiento. Un lenguaje de procedimiento define las operaciones de una aplicación y el orden en el cual se realizan. Un lenguaje de no procedimiento, se refiere a los resultados de una operación; el entorno fundamental del software en que se procesan las operaciones (Beltrán, 2016).

Al manejar un lenguaje estructurado relacional *SQL* con tablas, filas y columnas se convierte en una herramienta fiable que organiza y recupera datos, “El lenguaje estructurado de consultas SQL provee sentencias para definir soportes de información (crear, modificar, eliminar tablas, constraints, etc,) y sentencias para manipular la información que contienen las tablas (consultar datos, insertar, modificar y eliminar registros, etc)” (Sznajdleder, 2015). Las características lo convierten en un completo sublenguaje acompañado de una base de datos que se puede compartir con otros o a su vez permite que la estructura del diseño o de programación resulta de menor riesgo y en ocasiones de menor costo de los programas en red.

Como parte del lenguaje de programación se tiene la oportunidad de crear, consultar datos, realizar modificaciones que permitan el acceso de datos con los componentes para la verificación de la información que se requiera, por lo que *SQL* se convierte en una herramienta que permite la introducción de datos de programación y diseño que trabaja con otros componentes que pueden modificar y eliminar los datos.

Las características del SQL Server constan el soporte de transacciones, escalabilidad, estabilidad y seguridad, soporta procedimientos almacenados, incluye un potente entorno gráfico de administración, trabaja en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información y permite administrar información de otros servidores de datos (Santamaría & Hernández, 2018).

SQL Server tiene un soporte de tres tipos, instantánea, que copia los datos tal y como aparecen en un momento determinado, transaccional en que se propaga los datos iniciales a los suscriptores y mezcla para que varios sitios funcionen en línea o desconectados de manera autónoma las modificaciones de datos realizados en un resultado único y uniforme.

1.5.11 En programación Lenguaje C#

El lenguaje C# en programación es sencilla, por lo general se reconoce fácilmente la sintaxis de sus llaves y es menos compleja que el lenguaje C y C++ y da la oportunidad de incorporar mayores opciones que *Java*, considerando que el lenguaje que utiliza es parecido, por lo que se convierte en interacción personalizada, con una amplia librería de clases, entre sus características se reconocer que acepta métodos en el desarrollo de aplicaciones para varios tipos de programas como los videos juegos.

En el artículo publicado por una revista especializada se destaca que el lenguaje de programación C# fue creado por el danés Anders Hejlsberg que diseño también los lenguajes Turbo Pascal y Delphi. El C# (pronunciado en inglés “C sharp” o en español “C sostenido”) es un lenguaje de programación orientado a objetos. Con este nuevo lenguaje se quiso mejorar con respecto de los dos lenguajes anteriores de los que deriva el C, y el C++ (larevistainformatica.com, 2016).

Como constan en las páginas autorizadas de Microsoft, en lo relacionado con el lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método *Main*, el punto de entrada de la aplicación, se encapsulan dentro de las definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que invalidan los métodos virtuales en una clase primaria requieren la palabra clave *override* como una manera de evitar redefiniciones accidentales. (Microsoft, 2018).

1.5.12 Exposición sobre .Net

Lo interesante del lenguaje de los programas es que generalmente son creados para la solución de problemas que buscan favorecer a las actividades de los usuarios para facilitarle los trabajos, de allí nace la importancia de un programador, por lo que el framework de .Net es una solución a toda la problemática en torno al desarrollo de las diferentes aplicaciones desarrolladas, que brinda grandes beneficios no solamente al desarrollador, sino también al proceso de desarrollo. En primer lugar .Net está listo, lo que permite a los usuarios usarlo sin dificultad por que es mucho más fácil de instalar" (Landa, 2016). Para los programadores .Net representa una plataforma de interfaz que el acceso a una amplia librería de sistema entre las que constan *text, .Net, IO, Inter Op, Reflection, Securuty, Remoting, Globalization, Serialization;* en la librería de datos *Common, SQL types, SQL client, OleDb* (Landa, 2016).

Las librerías en XML, *serialization, XSL, Xpxth,* libreris de Dibujo *Drawing2D, Printing Text, Imagine* Entre las librerías de Windows se encuentran varias configuraciones como *Desing, ComppomentModel.* La librería Web contiene *Discovery* (Landa, 2016). Las librerías representan una mayor accesibilidad a la informacion, por lo que .Net representa una de las plataformas que son visitadas para brindar la solución a problemas del sistema.

De acuerdo con (Microsoft , 2018) ".NET Framework consta de dos componentes principales: Common Language Runtime (CLR) y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework". Lo que beneficia en la búsqueda de las diferentes librerías que contiene en la plataforma, que da la opción para la administración de comunicación y subprocesos que brindan seguridad al momento de evaluar los diferentes sistemas.

Se trata de una plataforma para el desarrollo de software lanzada por Microsoft para fusionar su amplio catálogo de productos, que va desde sus múltiples sistemas operativos hasta herramientas de desarrollo. Desde un punto de vista tecnológico**,** lo que se pretendía con la creación de .NET era desarrollar aplicaciones y sistemas independientes de la arquitectura física y del sistema operativo sobre el que se ejecutaran.

1.5.13 Microsoft Azure

Para el desarrollo de programas se requiere de elementos que apoyan al control y expansión de todo lo que se ofrece, es así que Microsoft Azure es conjunto en constante expansión de servicios en la nube para ayudar a su organización a satisfacer sus necesidades comerciales. Le otorga la libertad de crear, administrar e implementar aplicaciones en una red mundial enorme con sus herramientas validad y marcos favoritos (Microsoft, 2018). La importancia aporta efectivamente a controlar cada uno de los procesos y los servicios que otorga la nube.

La nube de Azure solo funciona con Internet y entre los datos que se recopilan constan la autenticación facial:

a) En el primer paso se comienza la comprobación de que dos caras pertenezcan a la misma persona que se le esta tomando la fotografìa. La API devuelve una puntuación de confianza sobre cuál es la probabilidad de que dos caras pertenezcan a la misma persona (Microsoft, 2018).

b) Luego, se pone a prueba detectando una, dos o varias caras humanas en una misma imagen, la finalidad es obtener rectángulos faciales donde se encuentran las caras en la imagen, junto con atributos faciales que contienen predicciones de características faciales basadas en aprendizaje automático. Las características de atributos faciales disponibles son: edad, emoción, sexo, postura, sonrisa y vello facial, junto con 27 puntos de referencia para cada cara de la imagen de la persona seleccionada, lo que conllevaa a obtener la fiabilidad de la fotografìa (Microsoft, 2018).

c) La identificación facial o *Face API* permite buscar, identificar y asociar caras en su repositorio privado de hasta un millón de personas.

d) Se puede acceder a la búsqueda de caras parecidas con facilidad dentro de un mismo contexto. Se puede poner una colección de caras y la consulta de una cara nueva en la que se expone la visión real de la persona, esta API devuelve una colección de caras parecidas, pero involucra la constante predisposición de dar la fiabilidad de cada cara (Microsoft , 2018).

CAPÍTULO II: MARCO METODOLÓGICO

2.1 Tipo de diseño, alcance y enfoque de la investigación

Tipo Experimental: La investigación experimental se lo identifica como un proceso, en el que se somete a un objeto o grupo de individuos en determinadas condiciones, estímulos o tratamiento, para observar los efectos o reacciones que se producen, intervienen directamente la variable independiente y la variable dependiente (Arias, 2015).

En la investigación la variable independiente corresponde a la aplicación móvil para el registro de ingresos de personas y la variable dependiente es el reconocimiento facial. En la investigación experimental se manipularon las variables, en el diseño y creación de la aplicación móvil el investigador a medida que va desarrollando el proyecto, modifica las variables y se determinan los factores que afectan al experimento y observan cada una de las reacciones que se generan en el proceso experimental.

En el proceso experimental se establecieron dos grupos, a quienes se les tomaba las fotografías, con uno se logró realizar el experimento sin modificar las características del proyecto, en el otro grupo se manipulaban los factores y las variables del reconocimiento facial. El objetivo fue simple cuantificar el cambio en el proceso de la utilización de la tecnología para el reconocimiento facial y observar las variaciones que se presentaron en los grupos de estudio. La determinación de los dos grupos para la aplicación del experimento se lo hizo de forma aleatoria, tomando como base las características homogéneas que se requería para la realización del experimento y establecer claramente dos momentos.

Tipo Explorativo: dentro de la investigación, la parte exploratoria corresponde a brindar un estudio de la visión panorámica del problema investigado, consiste en buscar la información de manera general sobre las variables independiente y dependiente de un estudio con la finalidad de obtener los resultados esperados del estudio (Muñoz R. , 2013).

Tipo Descriptivo: permite la identificación de los conceptos relacionados con reconocimiento facial y todos los elementos que intervienen en el desarrollo del *App*, determinando sus partes y los elementos que lo componen, los mismos que permiten entender la problemática e identificar sus posibles soluciones (Pacheco, 2014).

2.2 Método de la investigación

Ente los métodos utilizados en el Método inductivo – deductivo: Se parte desde lo particular del conocimiento del reconocimiento facial y las conceptualizaciones del software, app, sistemas operativos entre otros y se va a lo general del lenguaje de programación Java, Microsoft Azure, “Este método de inferencia se basa en la lógica y estudia hechos particulares, aunque es deductivo en un sentido parte de lo general a lo particular y el método inductivo en sentido contrario, de lo particular a lo general de la investigación (Bernal, 2014).

Método Analítico – Sintético: Este método viene del griego que busca analizar= descomposición, es decir se da por la fragmentación de un cuerpo en sus principios constitutivos. Método que va de los compuesto a lo simple (Pacheco, 2014). El análisis está identificando las partes en que se divide el proyecto experimental, parte por parte, con todos los detalles y luego se sintetiza en las conclusiones y recomendaciones.

2.3 Unidad de análisis, población y muestra

En la investigación experimental se consideró seleccionar seis personas que vivan en la ciudad de Guayaquil, el objetivo fue tomar en cuenta que tiene y comparten un conjunto común de varias características y constituyen el total del universo, para los propósitos del problema planteado en la investigación (Malhotra, 2014). Se la determinó en forma aleatoria, es decir la muestra fue de seis personas a quienes se les hizo seguimiento del proceso de reconocimiento facial.

2.4 Variables de la investigación

Entre las variables de investigación constan:

Variable Independiente: VI: Aplicación Móvil

Variable Dependiente: VD: Reconocimiento Facial

2. 5 Fuentes, técnicas e instrumentos para la recolección de la información

La observación es una de las técnicas que se la utilizó, considerando que es el primer paso para la realización de la investigación y que permitió adquirir conocimientos sobre el tema investigado, porque de una manera simple se pudo acceder a la información desde el lugar en que se dan los hechos, se puede afirmar que puede ser de manera directa e indirecta (Pacheco, 2014). Entre los instrumentos o herramientas utilizadas se consideró la fotografía, desde varios ángulos y con diferentes significados que permitieron seguir la secuencia de la cara.

2.5.1 Metodología de trabajo de la aplicación

La metodología de trabajo para el desarrollo de la aplicación móvil del reconocimiento facial es una respuesta para cambiar el registro de asistencia de una OSAL sin fines de lucro, cuyo software permitirá un funcionamiento de control más eficiente y eficaz.

CAPÍTULO III: RESULTADOS Y DISCUSIÓN

3.1 Análisis de la situación actual

Se analizó diferentes formas de registros para las personas entre las cuales estaban biometría, código *QR* y reconocimiento facial. Se eligió reconocimiento facial porque no se tiene contacto con la persona, considerando que una de las condiciones de las OSAL es que llevan a cabo sus registros de manera privada y anónima.

Para el diseño e implementación de este trabajo se realizó la búsqueda de diferentes tecnologías de reconocimiento facial, tales como *Google Vision, OpenCV, Amazon Rekognition y Azure Face Recognition.* Luego de un estudio de las ventajas y desventajas, se tomó la decisión de escoger la tecnología de *Azure Face Recognition,* se consideró la ventajas de que la OSAL tiene convenio con *Microsoft*.

Tabla 1 Comparación de atributos de las tecnologías

	Microsoft	Google	OpenCV
Face Detection	✓	✓	✓
Face Recognition (Image)	✓	✗	✗
Face Recognition (Video)	✓	✗	✓
Emotional Depth (%)	✓	✗	✗
Emotions Present (Y/N)	✓	✓	✗
Age & Gender	✓	✗	✗
Multi-face Tracking	✓	✓	✓
SDK	✗	✗	✓
API	✓	✓	✗

Elaborado por Alex García Arias

Nota: En la gráfica se evidencia la complementariedad que tienen las tecnologías, por lo que se eligió la más completa para la creación de la aplicación móvil.

Es importante señalar que las tecnologías Google y OpenCV no se consideraron porque no tienen reconocimiento facial, solo detectan la cara de la persona, se valoró la tecnología como referencia de puntos vectoriales de la cara para poder detectar un rostro, mientras que Microsoft Azure si tiene Face Recognition y detectar la cara. *Microsoft Azure* cuenta con *Face detection*, *face recognition (image), face recognition (video), emotional depth (%), emotions present (Y/N), Age&Gender, Multi-face Tracking, API*. La única carencia es *SDK*.

3.2 Diseño de la aplicación móvil: Reconocimiento facial

El diseño de la aplicación se basa en las tecnologías que se pueden usar y que no sean de costo para la OSAL. Una de las ventajas para llevar a cabo el diseño es que la OSAL tiene convenio con *Microsoft.* Para la aplicación móvil se desarrollará en Android.

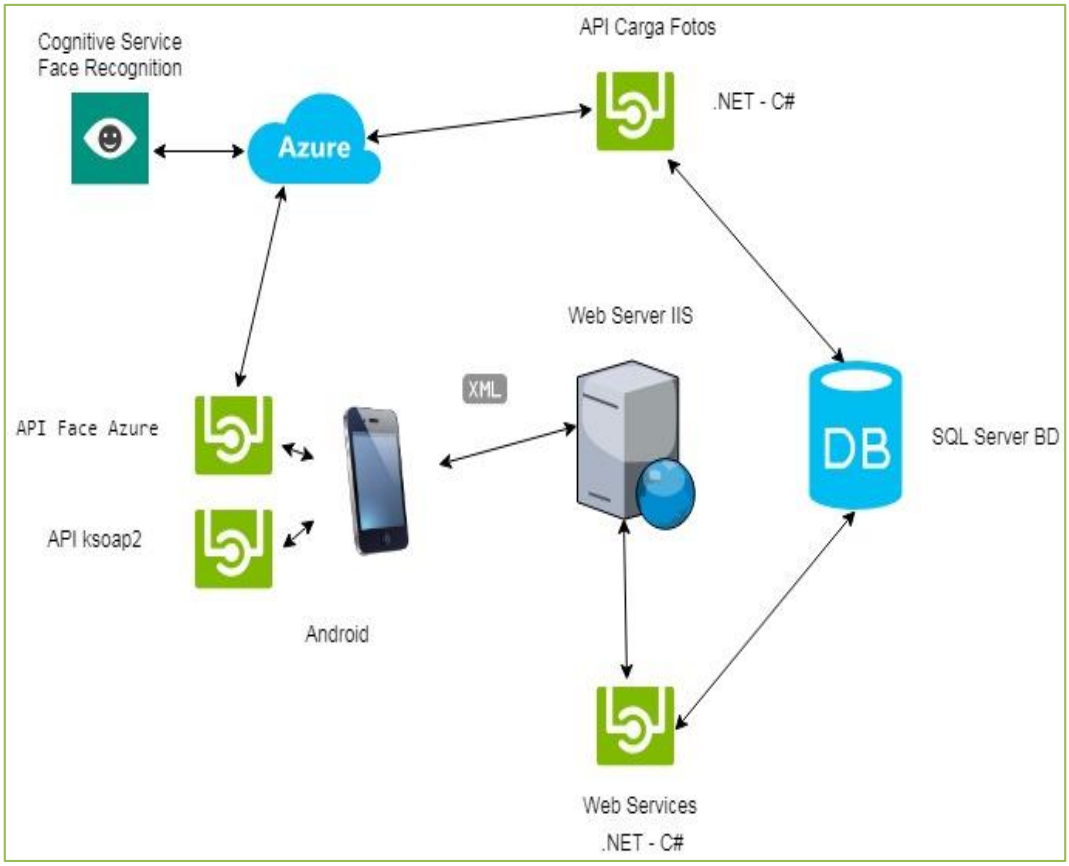

Figura 5 Diseño de la aplicación móvil para el reconocimiento facial
Elaborado por Alex García Arias

3.3 Presentación de resultados y discusión

Recursos de desarrollo: Herramientas usadas

En el desarrollo de la aplicación se utilizó las siguientes herramientas:

Tabla 2 Aplicación Android

Aplicación Android		
Herramienta	Descripción	Enlace Descarga o Instalación
Android Studio	Proporciona las herramientas más rápidas para crear apps en todas las clases de dispositivos Android.	https://developer.android.com/studio/
Java SE Development Kit 8	Es un software que provee herramientas de desarrollo para creación de programas en java.	https://www.oracle.com/technetwork/java/javase/downloads/index.html
Android SDK	Es el core de desarrollo de Android y además tiene el emulador para ejecutar la app y realizar pruebas	https://developer.android.com/studio/
Ksoap2	Api que se integra al desarrollo para comunicación con los web services	http://simpligility.github.io/ksoap2-android/
Cognitive-Face-Android	Api Android que se integra al desarrollo para reconocimiento facial de azure	https://github.com/Microsoft/Cognitive-Face-Android
Web Services		
Herramienta	Descripción	Enlace Descarga o Instalación
Visual Studio .NET	Entorno de desarrollo integrado (IDE) con todas las características para Android, iOS, Windows, la Web y la nube	https://visualstudio.microsoft.com/es/vs/
C#	lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET	https://www.microsoft.com/net
SQL Server	Es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.	https://www.microsoft.com/es-es/sql-server/sql-server-downloads
IIS Express	Servidor Web para subir el web service	Viene integrado con visual studio

Elaborado por Alex García Arias

Tabla 3 Hardware de desarrollo

Componente	Características técnicas	Tareas
Ordenador Portátil	• Dell Inspiron 15 7000 Gaming. • Procesador Core i5 7th Gen. • Memoria Ram 8Gb. • Sistema Operativo Windows 10.	• Diseño de la aplicación. • Desarrollo de la aplicación. • Pruebas de comunicación entre la aplicación móvil y el Web Services. • Redacción de la documentación.
Smartphone	• Huawei P10 Lite. • Pantalla IPS LCD 5,2 pulgadas FullHD (424ppp). • Procesador Kirin 658 de ocho núcleos a 2,3GHz. • Cámara 12 megapíxeles f/2.2, flash LED. • Versión Android 8.	Pruebas de la aplicacion móvil en el dispositivo.

Elaborado por Alex García Arias

CAPÍTULO IV: PROPUESTA

4.1 Requerimientos Funcionales del Proyecto

En el desarrollo del *App* se necesitan varios requerimientos funcionales para describir las actividades que permiten la ejecución del proyecto cuando cumplen con cada una de las condiciones, considerando el flujo que van desempeñar en el sistema. En el proyecto cada uno de los requisitos del software queda registrado en la matriz de trazabilidad de los requerimientos funcionales para la operatividad del *App* de reconocimiento facial.

La aplicación está compuesta de 4 partes fundamentales que son las siguientes:

1. Base de Datos.
2. Web Services.
3. Aplicación Móvil.
4. Programa de carga de información de la nube.

4.2 Fase de Diseño

4.2.1 Diseño de la Base de Datos

Para la base de datos se tomó en consideración las siguientes tablas para este desarrollo.

4.2.1.1 Modelo relacional

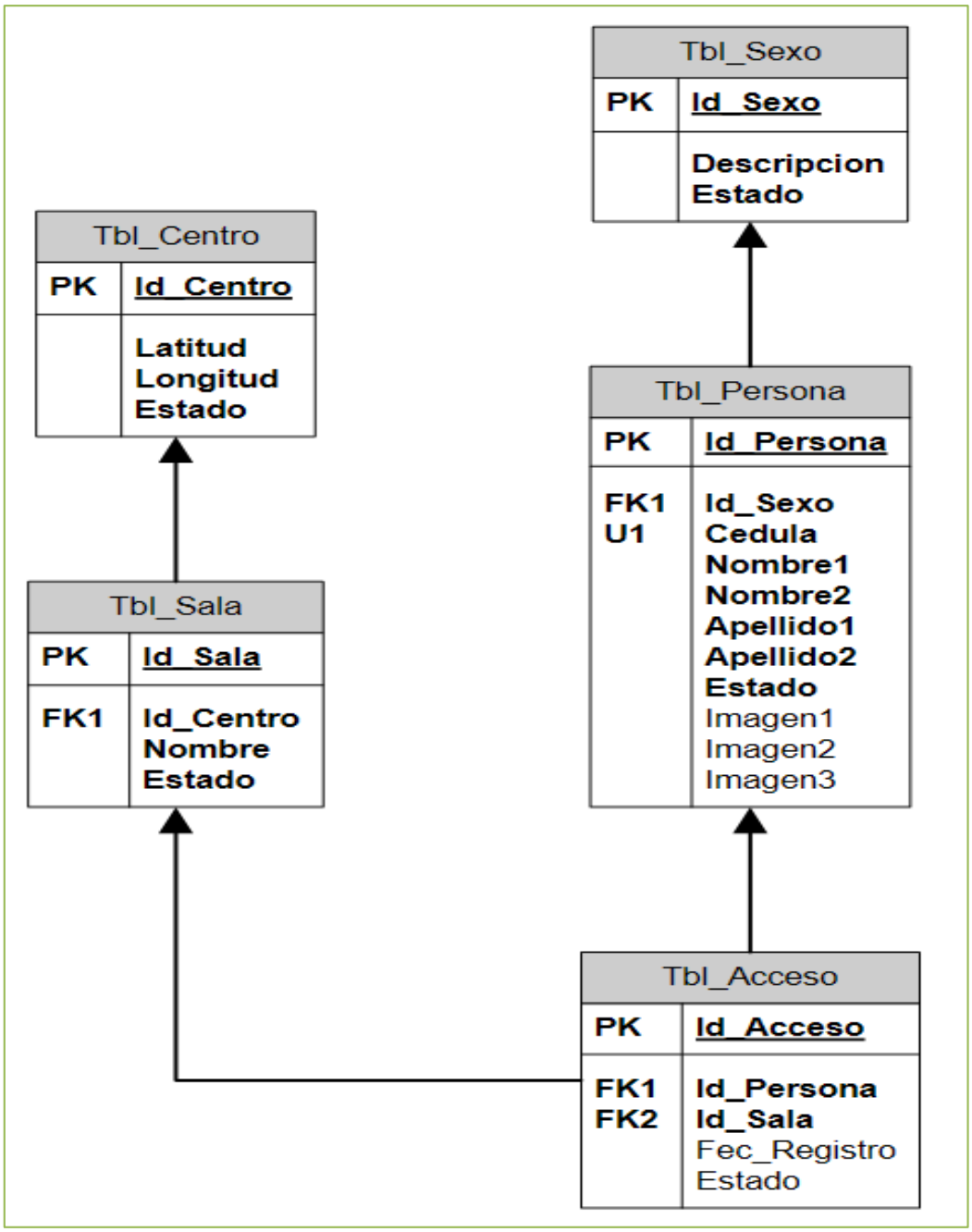

Figura 6 Modelo Relacional
Elaborado por Alex García Arias

Nota.- Para este diseño solo se tomo en consideración las tablas y campos que solo sirven para el desarrollo del APP. No se pone campo edad por consideración de seguridad de la OSAL con el manejo de información de menores de edad.

4.2.1.2 Modelo relación extendida

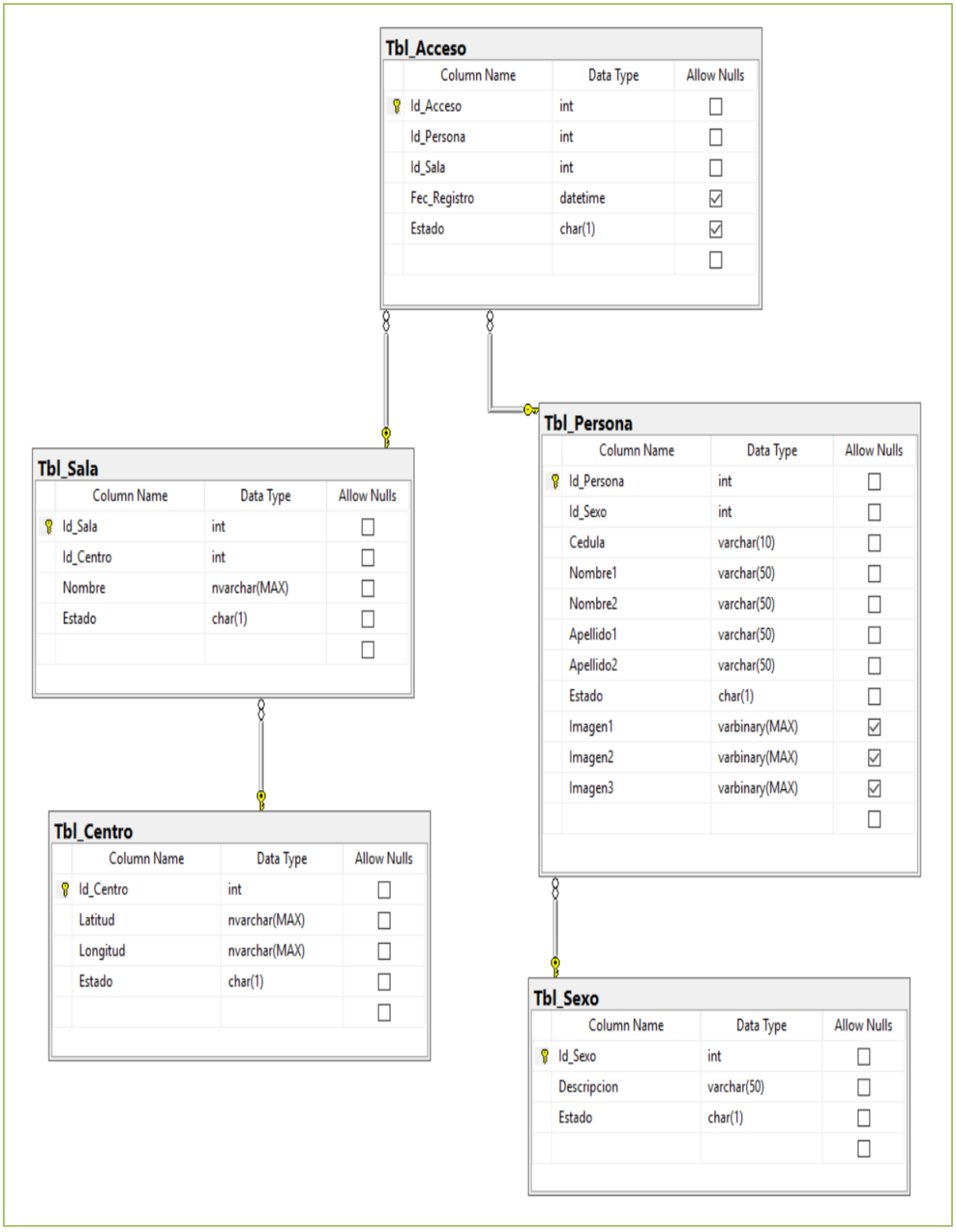

Figura 7 Modelo Relacional Extendido
Elaborado por Alex García Arias

4.2.1.3 Descripción de las tablas del modelo de base de datos

Tabla 4 Tbl_Sexo

Tbl_Sexo		
Tabla donde se guarda el sexo de la persona		
Columna	Tipo Dato / Clave	Observación
id_sexo	Int / Clave Primaria	
descripcion	Varchar(50)	
Estado	Char(1)	A, I

Elaborado por Alex García Arias

Tabla 5 Tbl_Centro

Tbl_Centro		
Tabla donde se guarda los centros de capacitación de la OSAL y su ubicación GPS		
Columna	Tipo Dato / Clave	Observación
id_centro	Int / Clave Primaria	
latitud	Nvarchar(MAX)	
longitud	Nvarchar(MAX)	
estado	Char(1)	A, I

Elaborado por Alex García Arias

Tabla 6 Tbl_Sala

Tbl_Sala		
Tabla donde se guarda las salas de capacitación que existen en los centros		
Columna	Tipo Dato / Clave	Observación
id_sala	Int / Clave Primaria	
id_centro	Int / Clave Foranea	
nombre	Nvarchar(MAX)	
estado	Char(1)	A, I

Elaborado por Alex García Arias

Tabla 7 Tbl_Persona

Tbl_Persona		
Tabla donde se guarda las personas que acceden al centro de capacitación		
Columna	Tipo Dato / Clave	Observación
id_persona	Int / Clave Primaria	
id_sexo	Int / Clave Foranea	
cedula	Varchar(10)	
nombre1	Varchar(50)	
nombre2	Varchar(50)	
apellido1	Varchar2(50)	
apellido2	Varchar2(50)	
estado	Char(1)	A, I
imagen1	Varbinary(Max)	Imagen 1 de cara
imagen2	Varbinary(Max)	Imagen 2 de cara
imagen3	Varbinary(Max)	Imagen 3 de cara

Elaborado por Alex García Arias

Tabla 8 Tbl_Acceso

Tbl_Acceso		
Tabla donde se guarda los accesos de las personas cuando ingresan al centro y son reconocidas por la tecnología de face recognition.		
Columna	Tipo Dato / Clave	Observación
id_acceso	Int / Clave Primaria	
id_persona	Int / Clave Foranea	
id_sala	Int / Clave Foranea	
fec_registro	Datetime	
estado	Char(1)	A, I

Elaborado por Alex García Arias

4.2.2 Análisis de usuabilidad en el diseño de las pantallas

Para el análisis de usabilidad, el primer diseño de pantallas se lo expuso a un grupo de programadores, quienes indicaron que las pantallas eran muy complejas para personas que son inexpertos en el manejo de internet y las diferentes tecnologías. La segunda presentación de muestra del diseño con las mejoras indicadas, aumento el nivel de aceptación, por cuanto ofrecía facilidad para el uso del diseño tecnológico, por que su exactitud y no presentar problemas de equivocación al momento de usarla.

Los diseños aceptados fueron las pantallas de la *App* realizada en Balsamiq Mockups 3. Es una aplicación de escritorio para realizar diseño de aplicaciones de escritorio, web y móviles.

4.2.2.1 Pantalla 1 – Acceso GPS y elección de la sala

1. Obtener la localización GPS.
2. Elegir el salón.

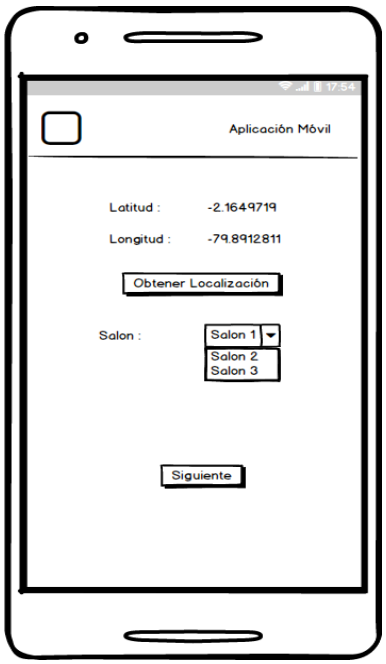

Figura 8 Acceso GPS y elección de la sala
Elaborado por Alex García Arias

4.2.2.2 Pantalla 2 – Tomar foto

Pantalla donde se toma la foto de la persona para poder después realizar las verificaciones y reconocimiento facial.

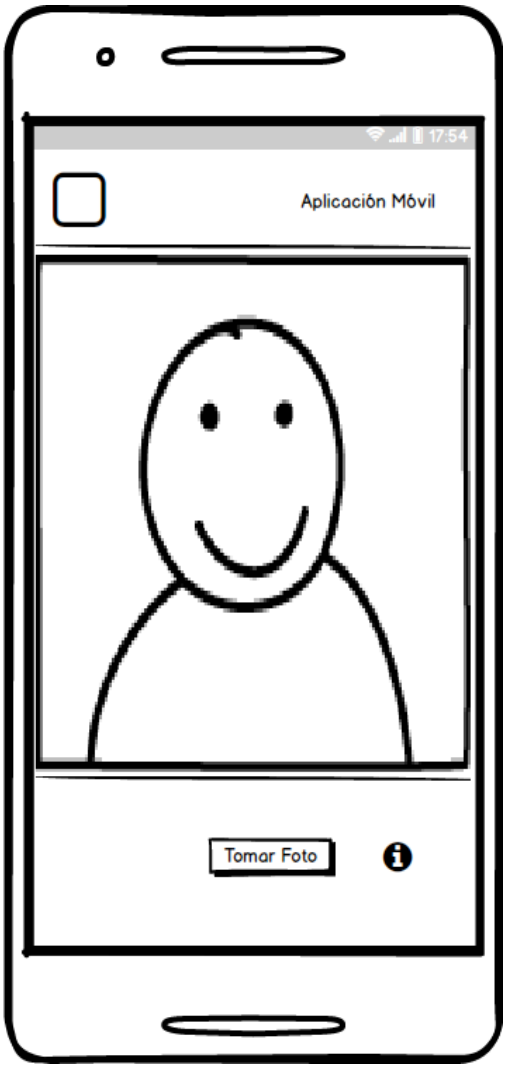

Figura 9 Tomar foto
Elaborado por Alex García Arias

4.2.2.3 Pantalla 3 – Reconocimiento Facial

1. Se detecta si existe una cara en la imagen.
2. En el caso de no detectar, se puede dar click en el botón reiniciar.
3. Se identifica la cara en la base de datos de Azure.
4. Si es exitoso sigue a la pantalla de exito.
5. Si no es exitoso sigue a la pantalla de error.

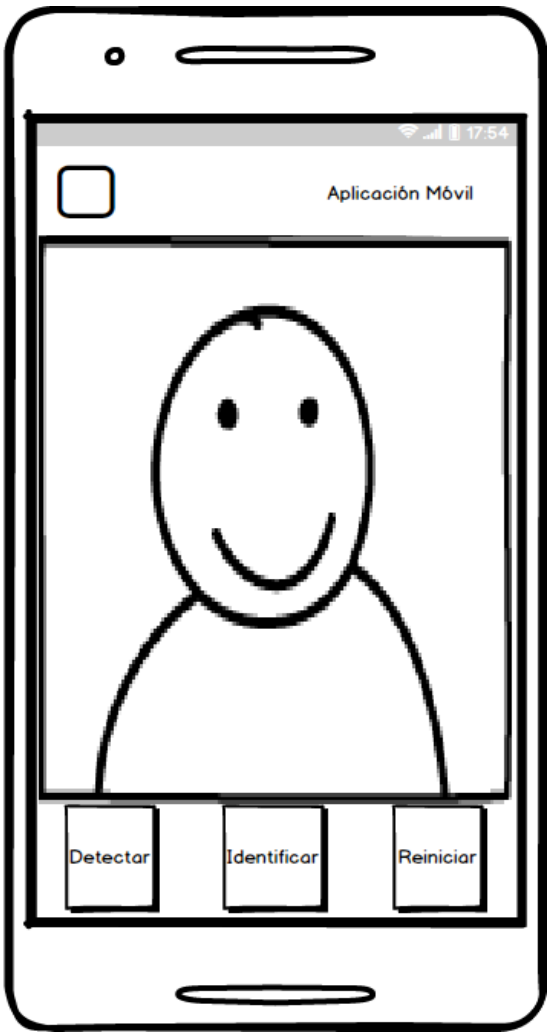

Figura 10 Reconocimiento facial
Elaborado por Alex García Arias

4.2.2.4 Pantalla 4 – Éxito por reconocimiento y grabar acceso

Si el reconocimiento fue exitoso se procede a registrar la persona su ingreso al centro de estudio en la sala a la que se dirige.

Figura 11 Éxito por reconocimiento y grabas acceso
Elaborado por Alex García Arias

4.2.2.5 Pantalla 5 – Error por reconocimiento facial

Si la persona no es reconocida, el resultado indica que esa persona no existe en la base de Azure para ser reconocida.

Figura 12 Error por el reconocimiento facial
Elaborado por Alex García Arias

4.3 Fase de Desarrollo

4.3.1 Desarrollo de Web Services

4.3.1.1 Entorno de desarrollo

Las herramientas utilizadas para el desarrollo del web services fueron los siguientes:

- Visual Studio 2017 (IDE)
- C# (Lenguaje)
- .NET (Framework)
- IIS (Servidor Web)

4.3.2 Desarrollo en Visual Studio 2017

La siguiente pantalla muestra como se ve el entorno de desarrollo para el Web Services.

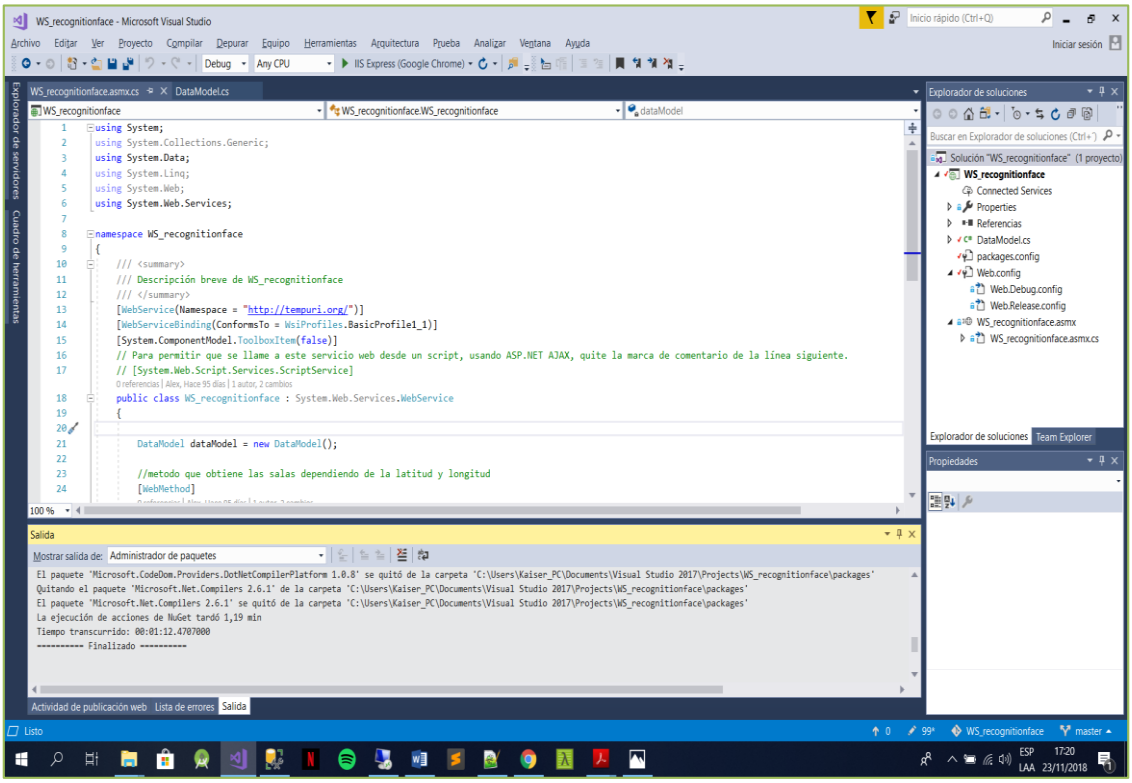

Figura 13 Visual Studio 2017
Elaborado por Alex García Arias

4.3.3 Código Fuente del Web Services

4.3.3.1 WS_recognitionface.asmx

Este código Fuente es el que invoca cuando se quiere llamar un método del Web Services (Ver anexo 1).

4.3.3.2 DataModel.cs

Este código Fuente es el encargado de realizar la conexión a la base de datos y dar la información al Web Services de la consulta de salas o registrar el ingreso de la persona que fue identificada al momento del ingreso a la sala (Ver Anexo 2).

4.3.3.3 Web.config

Archivo donde se configura la conexión a la base de datos (Ver Anexo 3).

4.3.3.4 Web Services WS_recognitionface

Visualización de la ejecución del Web Service en el Browser. http://192.168.100.27/WS_recognitionface.asmx

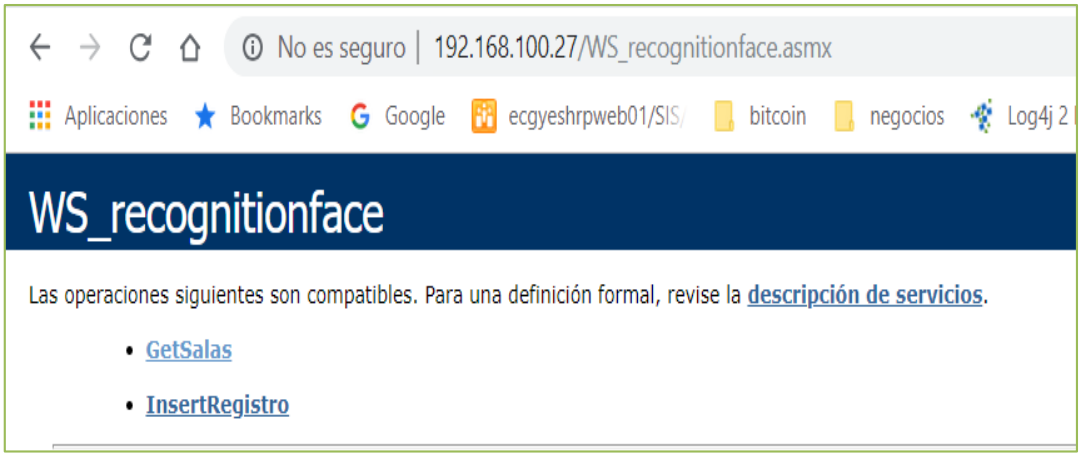

Figura 14 Web Services WS recognitionface
Elaborado por Alex García Arias

4.3.4 Métodos del Wed Services

4.3.4.1 GetSalas

Método para obtener las salas del centro

http://192.168.100.27/WS_recognitionface.asmx?op=GetSalas

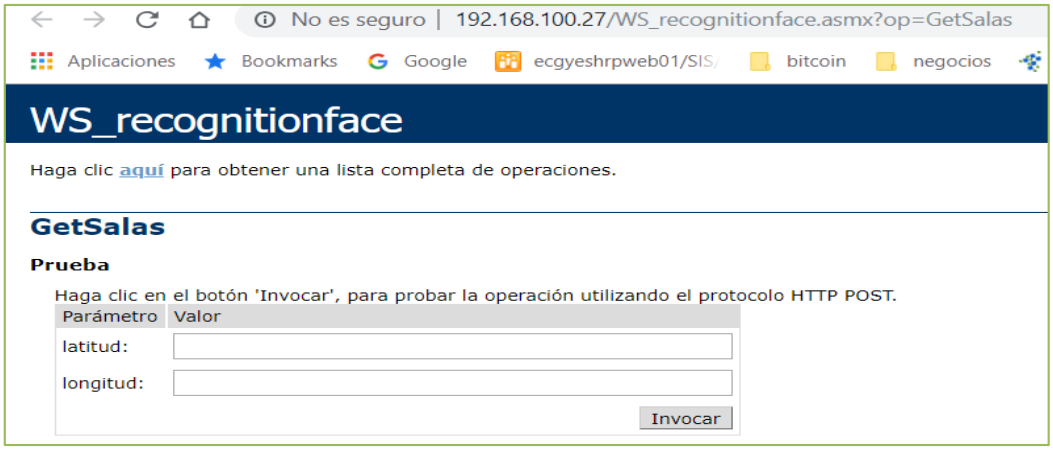

Figura 15 Salas del centro
Elaborado por Alex García Arias

4.3.4.2 Resultado

Figura 16 Resultado
Elaborado por Alex García Arias

4.3.4.3 InsertRegistro

Método para registrar a la persona que fue exitoso el reconocimiento facial

http://192.168.100.27/WS_recognitionface.asmx?op=InsertRegistro

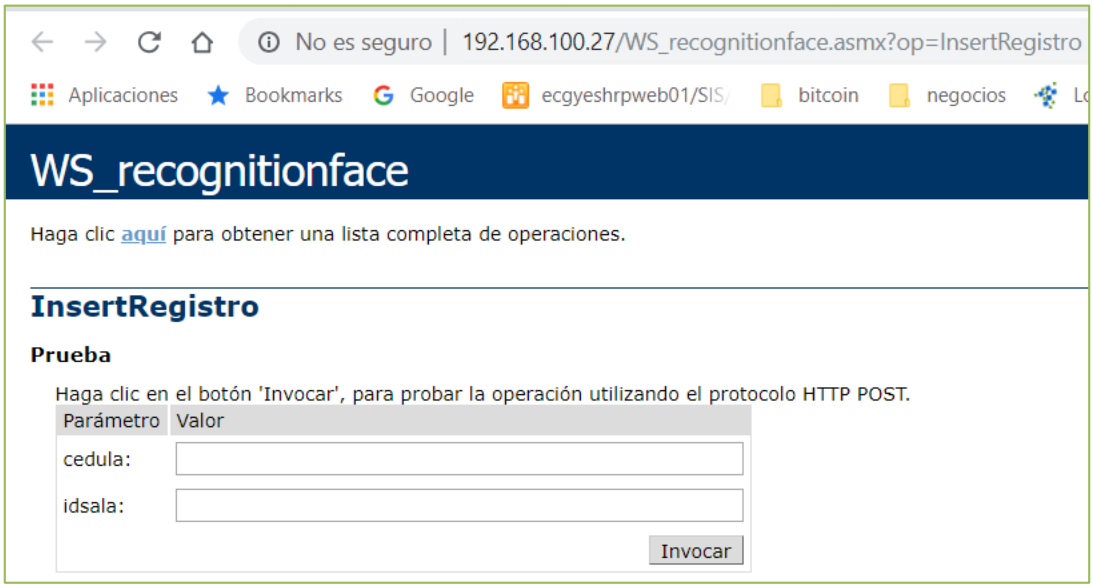

Figura 17 InsertRegistro
Elaborado por Alex García Arias

4.3.4.4 Resultado

Figura 18 Resultado
Elaborado por Alex García Arias

4.3.5 Desarrollo de Aplicación Android

4.3.5.1 Entorno de desarrollo

Las herramientas utilizadas para el desarrollo de la aplicación fueron los siguientes:

- Android Studio 3.2.1 (IDE)
- Java 8 (Lenguaje)
- Android (SDK)

4.3.5.2 Desarrollo en Android Studio

La siguiente pantalla muestra como se ve el entorno de desarrollo para la aplicación.

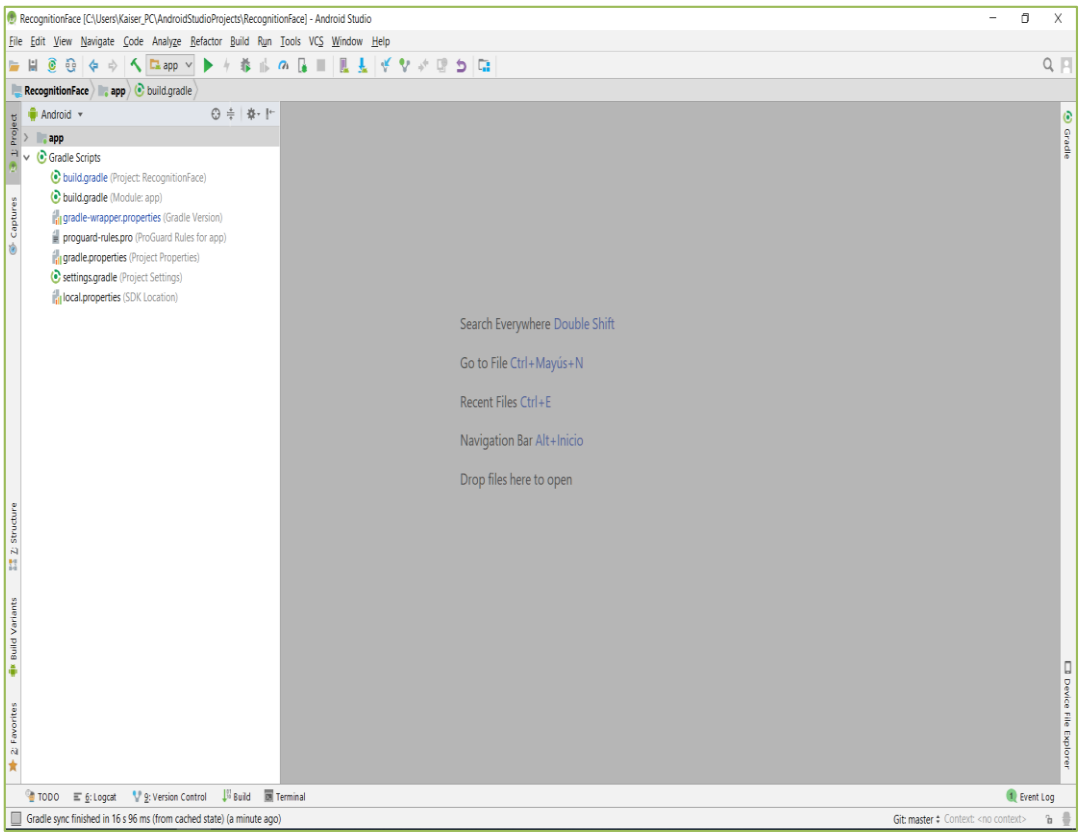

Figura 19 Desarrollo en Android Studio
Elaborado por Alex García Arias

4.3.6 Pantallas de la aplicación

4.3.6.1 Pantalla1 – activity_gps

Pantalla para obtener la ubicación GPS y seleccionar la sala a la que asiste la persona (Ver Anexo 4) código *xml* de la pantalla.

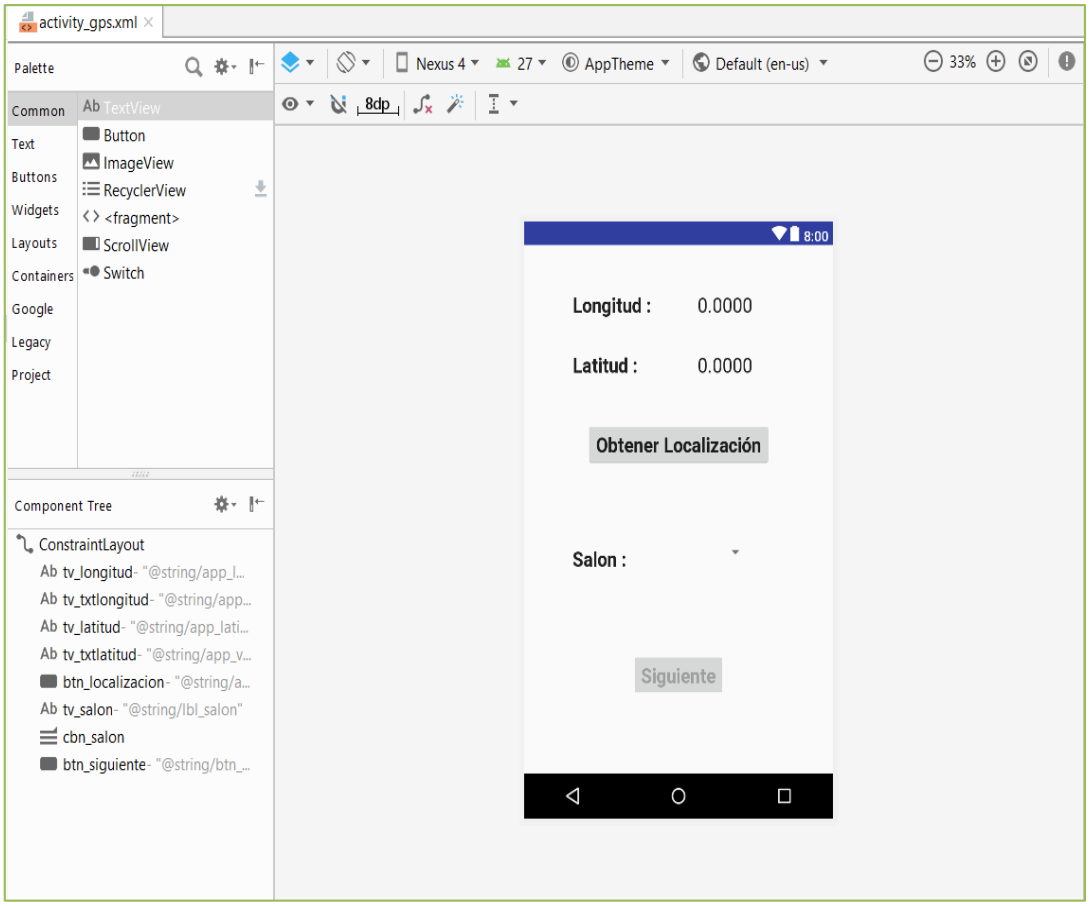

Figura 20 Pantalla 1 – activity_gps
Elaborado por Alex García Arias

4.3.6.2 Pantalla 1 – GPSActivity.java

Código fuente java de funcionamiento de la pantalla 1 (Ver Anexo 5).

4.3.6.3 Clase Salas – Salas.java

Este código es para agrupar las salas que se obtienen del Web Services (Ver Anexo 6).

4.3.6.4 Pantalla 2 – activity_foto

Pantalla para tomar la foto a la persona que va a ser reconocida para el acceso (Ver Anexo 7) código *xml* de la pantalla.

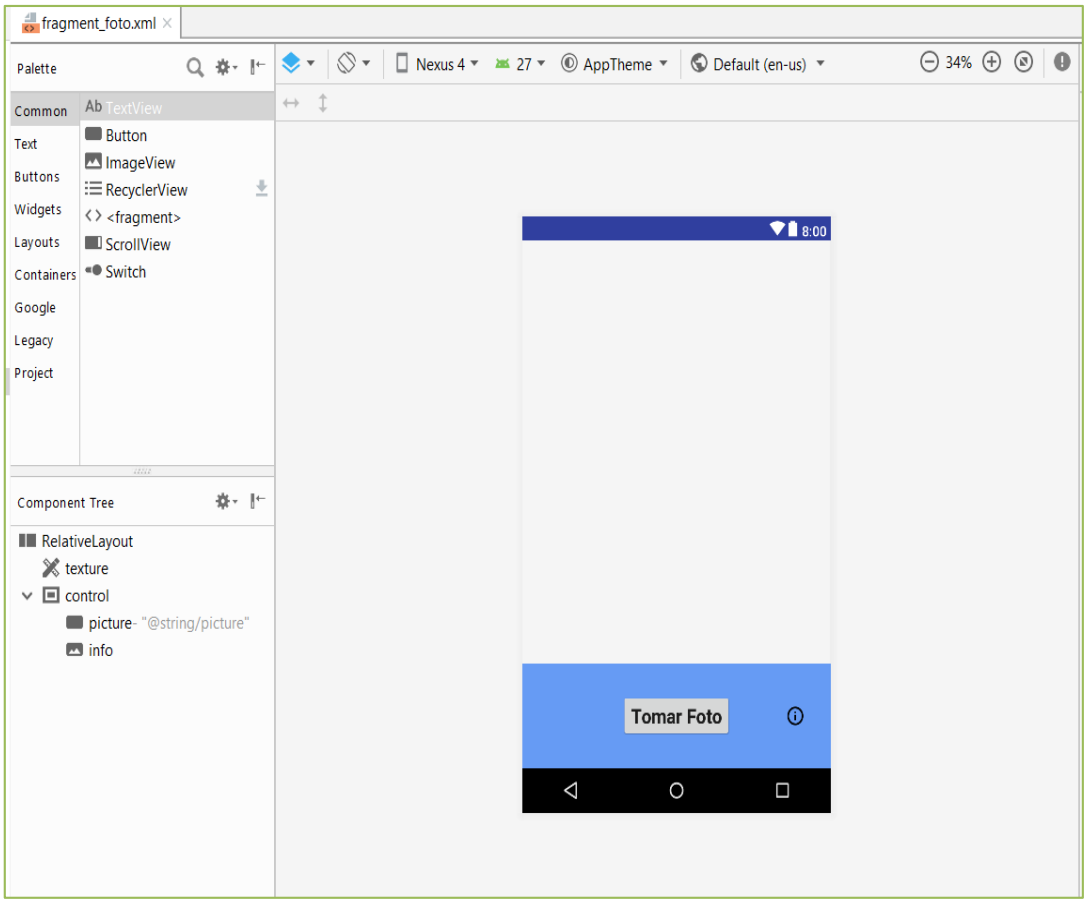

Figura 21 Pantalla 2 – activity_foto
Elaborado por Alex García Arias

4.3.6.5 Pantalla 2 – FotoActivity.java

Código fuente java de funcionamiento de la pantalla 2 (Ver Anexo 8).

4.3.6.6 Clase FotoFragment – FotoFragment.java

Código fuente que realiza el procesamiento de la foto de la pantalla 2 (Ver Anexo 9).

4.3.6.7 Clase AutoFitTextureView – AutoFitTextureView.java

Código fuente que realiza el preview de la cámara al momento de realizar la foto de la pantalla 2 (Ver Anexo 10).

4.3.6.8 Pantalla 3 – activity_proceso

Pantalla donde se procede con la detección de la cara en la foto y después se procede con la identificación de la persona por medio del reconocimiento facial. En el caso de que exista error en alguno de las 2 opciones se puede dar click en reiniciar proceso (Ver Anexo 11) código *xml* de la pantalla.

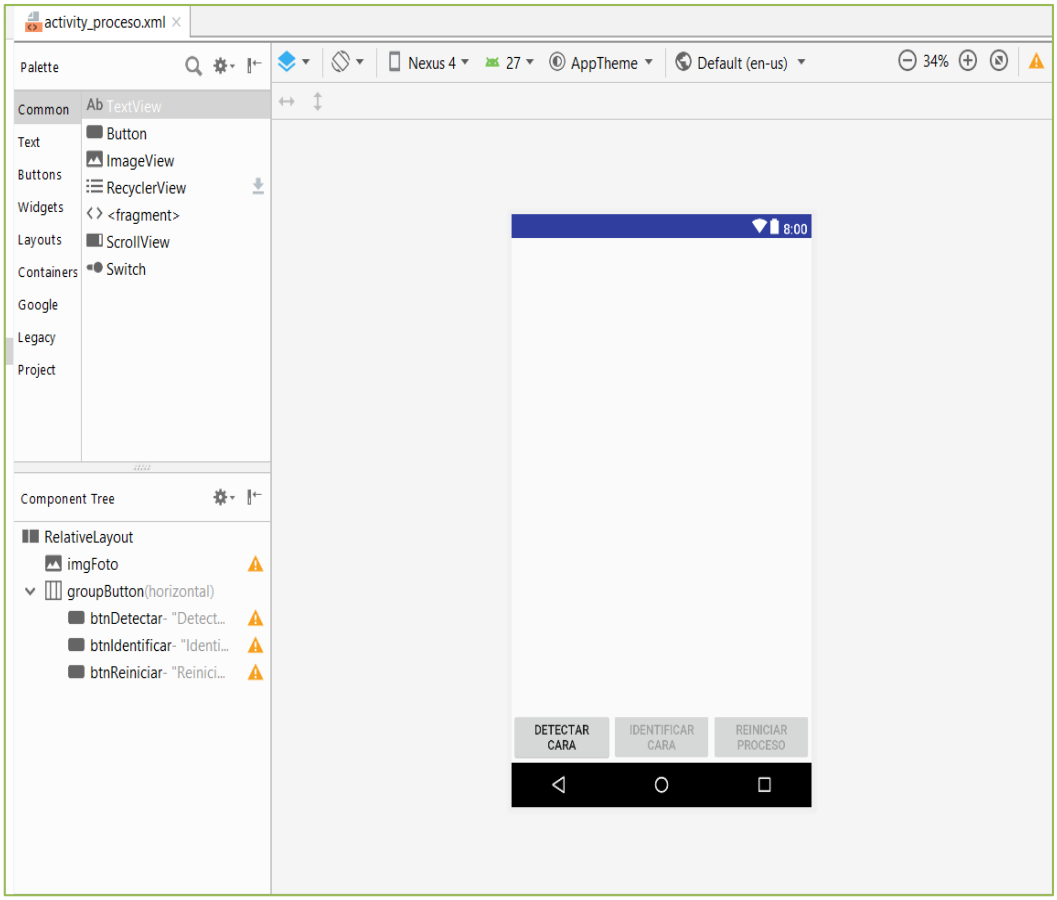

Figura 22 Pantalla 3 – activity_proceso
Elaborado por Alex García Arias

4.3.6.9 Pantalla 3 – ProcesoActivity.java

Código fuente java de funcionamiento de la pantalla 3 (Ver Anexo 12).

4.3.6.10 Pantalla 4– activity_exito

En el caso de que la persona sea identificada se envía al webservice el registro de la persona y se indica que la persona está registrada en el sistema. (Ver Anexo 13) código *xml* de la pantalla.

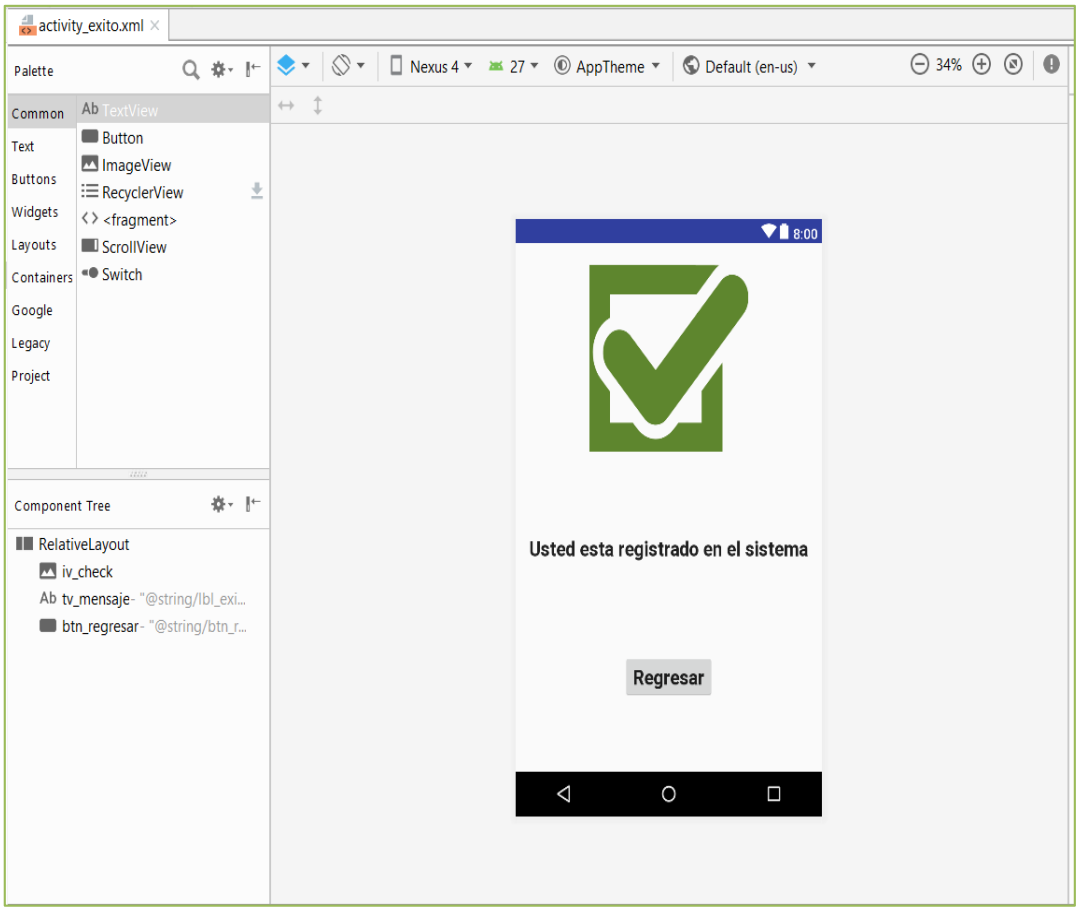

Figura 23 Pantalla 4 – activity_exito
Elaborado por Alex García Arias

4.3.6.11 Pantalla 4 – ExitoActivity.java

Código fuente java de funcionamiento de la pantalla 4 (Ver Anexo 14).

4.3.6.12 Pantalla 5 – activity_error

En el caso de que la persona no fue identificada se procede a indicar mensaje de error que no fue identificado (Ver Anexo 15) código *xml* de la pantalla.

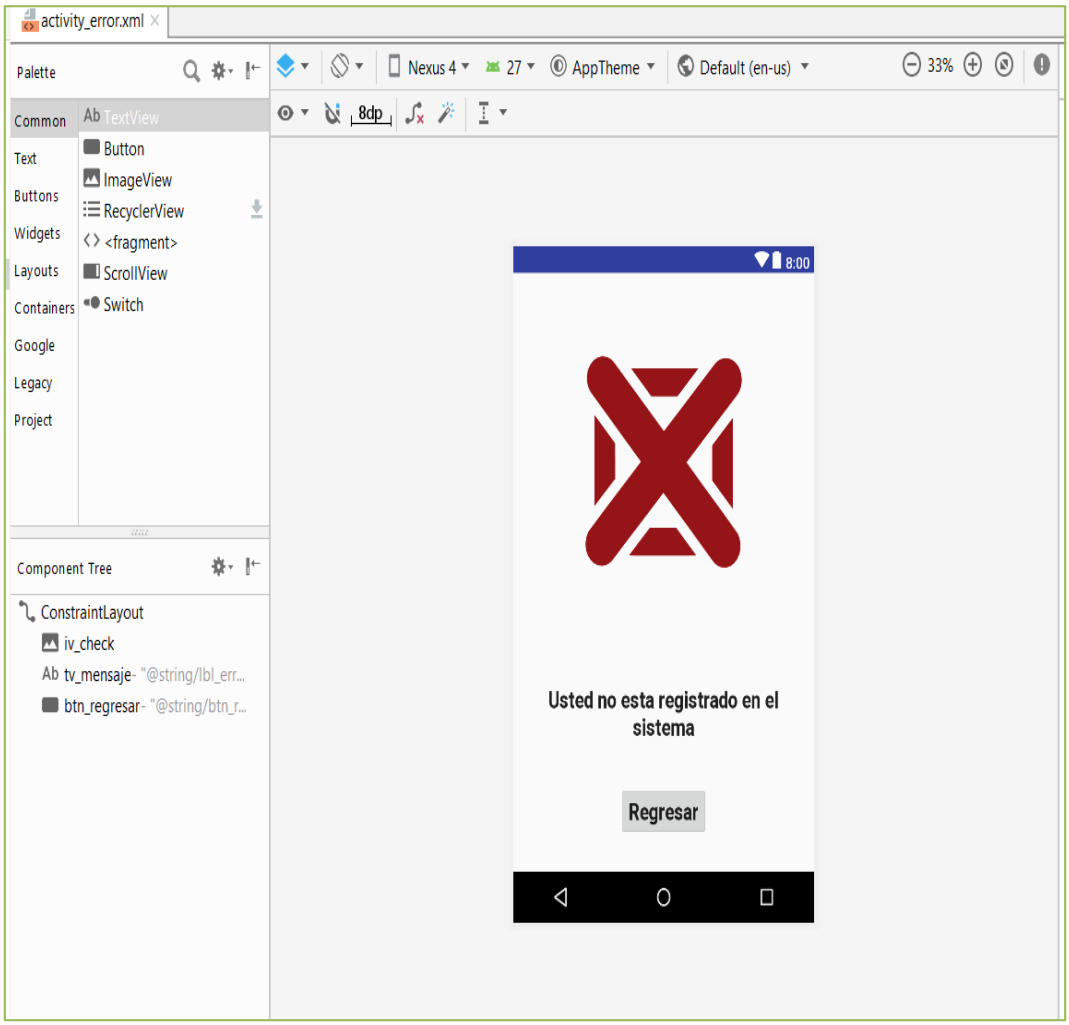

Figura 24 Pantalla 5 – activity_error
Elaborado por Alex García Arias

4.3.6.13 Pantalla 5 – ErrorActivity.java

Código fuente java de funcionamiento de la pantalla 5 (Ver Anexo 16).

4.3.6.14 Proceso de carga de personas a la nube de Azure

Este proceso es un código extra para el funcionamiento de la aplicación y pueda realizar la verificación de las personas que están registradas. Este código carga el primer nombre, primer apellido, cédula de identificación y 3 fotos que se registran en la base de datos. Esta información se la carga directamente a la nube de Azure para que al momento de identificar la persona la aplicación consulte a la nube si existe la persona.

El programa se llama AIBOTTraining. Esta hecho en .net usando c#. Consta de un solo archivo que es Program.cs.

El programa consta del siguiente menú:

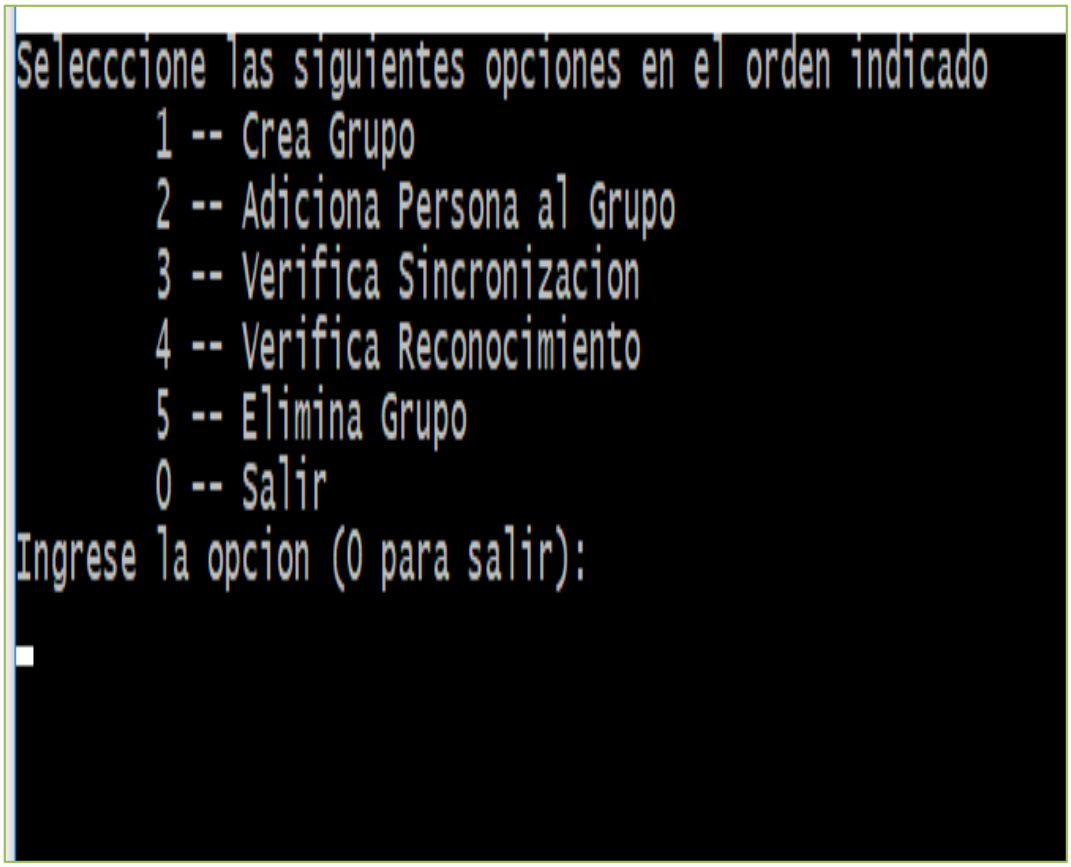

Figura 25 Proceso de carga de personas a la nube de Azure
Elaborado por Alex García Arias

- La opción uno crea el grupo en la nube de Azure. Esto agrupa a todas las personas registradas.

- La opción dos adiciona la persona al grupo que se creó previamente en la opción 1. Toma el dato de la base de datos local para subir la información que se indicó previamente.
- La opción tres realiza una verificación de sincronización de la carga realizada a la nube de Azure.

- La opción cuatro es una verificación de reconocimiento de la persona que se cargó para ver si funciono la sincronización.

- La opción cinco es para eliminar el grupo en la nube de Azure. Elimina todo lo que tenga que ver con información de las personas que se cargaron a la nube en la opción 2.

- La opción cero es para salir del programa de ejecución.

Este código fuente sirve para implementación de la parte de registro de personas cuando se realice este proyecto. Queda como código de ayuda para este proyecto (Ver Anexo 17) para código fuente.

4.4 Fase de Pruebas

Una vez finalizada la prueba de codificación y para la comprobación del correcto funcionamiento de la aplicación se realizaron 6 pruebas. 3 niños trillizos de 5 años, 1 niña de 9 años, 1 niño de 12 años y 1 adulto de 35 años. Se obtuvieron los permisos necesarios para poder hacer las pruebas con los menores de edad de sus respectivos padres.

4.4.1 Prueba 1

4.4.1.1 Prueba 1 – Pantalla 1- Ubicación GSP y Selección de la sala

Se obtiene la ubicación GPS y se selecciona la sala

Figura 26 Prueba 1 - Pantalla 1 – Ubicación GPS
Elaborado por Alex García Arias

4.4.1.2 Prueba 1 - Pantalla 2 – Tomar Foto

Se procede con la toma de foto de la persona

Figura 27 Prueba 1 - Pantalla 2 – Tomar foto
Elaborado por Alex García Arias

4.4.1.3 Prueba 1 - Pantalla 3 - Procesamiento de la foto

Se procede con la detección de la cara de la persona

Figura 28 Prueba 1 - Pantalla 3 – Procesamiento de foto
Elaborado por Alex García Arias

4.4.1.4 Prueba 1 - Pantalla 4 - Detectando cara

Se detecta la cara de la persona

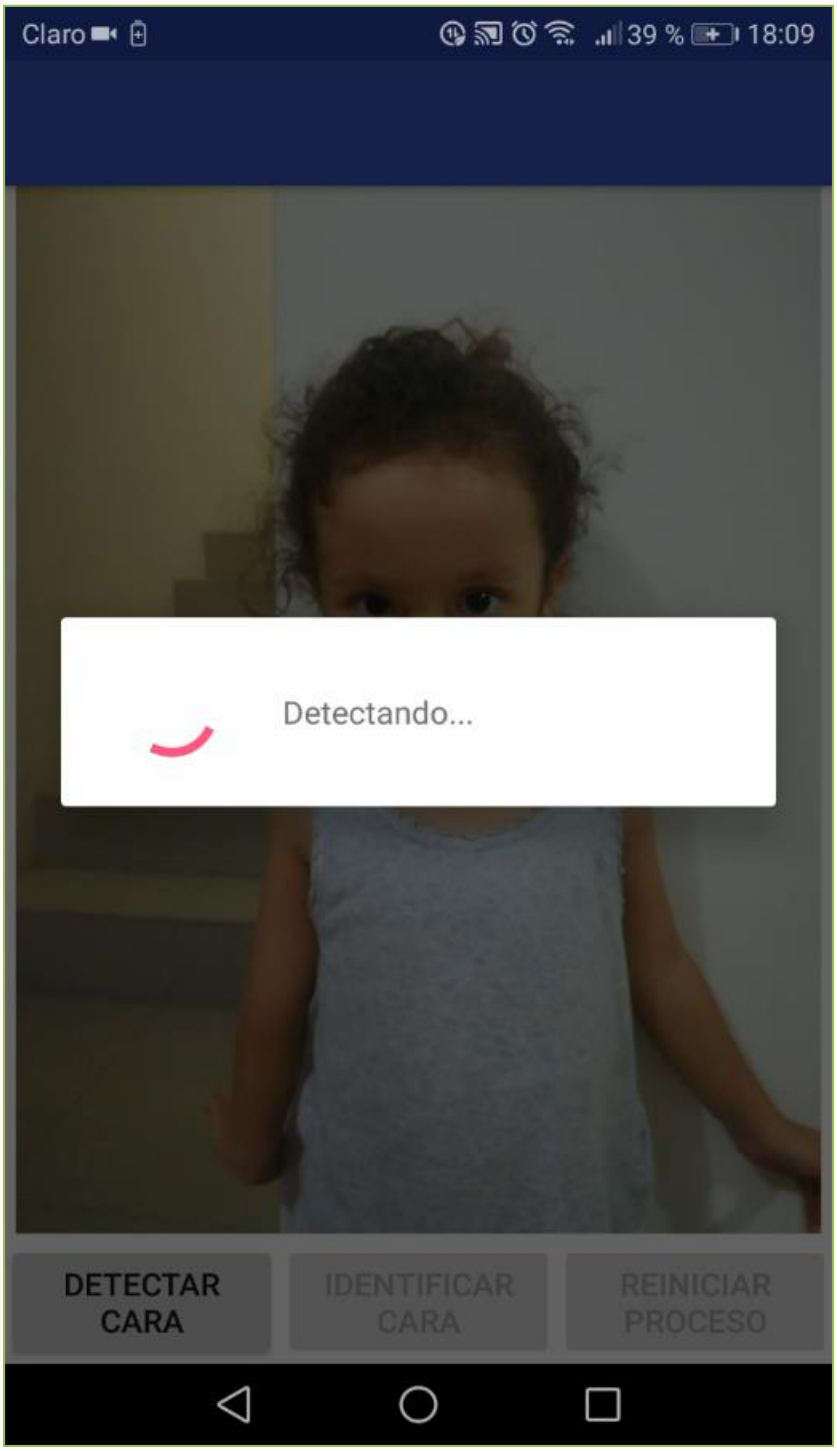

Figura 29 Prueba 1 - Pantalla 4 – Detectando la cara
Elaborado por Alex García Arias

4.4.1.5 Prueba 1 - Pantalla 5 - Identificación de persona

Una vez detectada se procede con la identificación

Figura 30 Prueba 1 - Pantalla 5 – Identificación de persona
Elaborado por Alex García Arias

4.4.1.6 Prueba 1 - Pantalla 6 - Identificando persona

Se procede con la identificación de la cara en la nube de Azure si existe la persona.

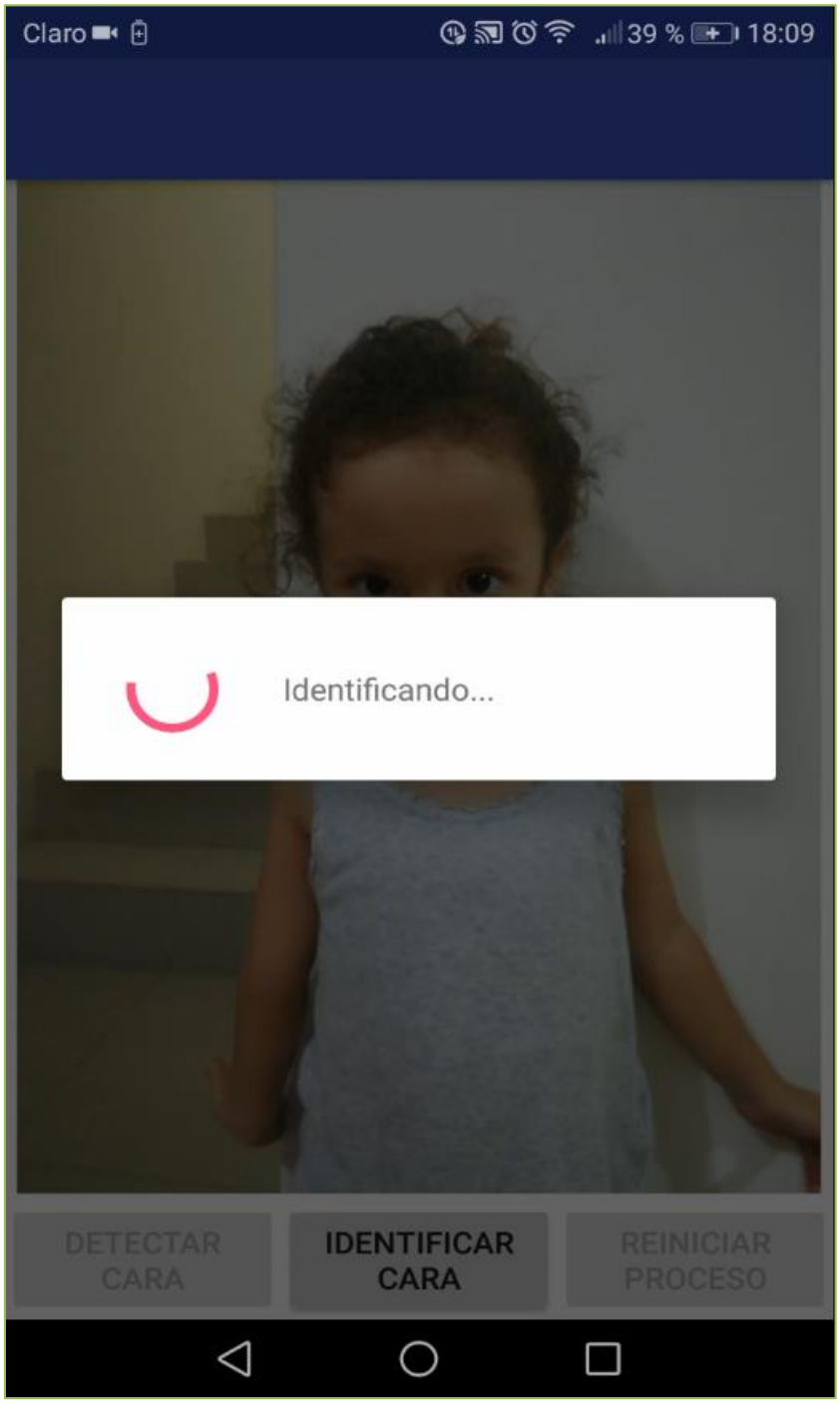

Figura 31 Prueba 1 - Pantalla 6 – Identificando persona
Elaborado por Alex García Arias

4.4.1.7 Prueba 1 - Pantalla 7 - Persona Identificada

El proceso identifico la cara de la persona

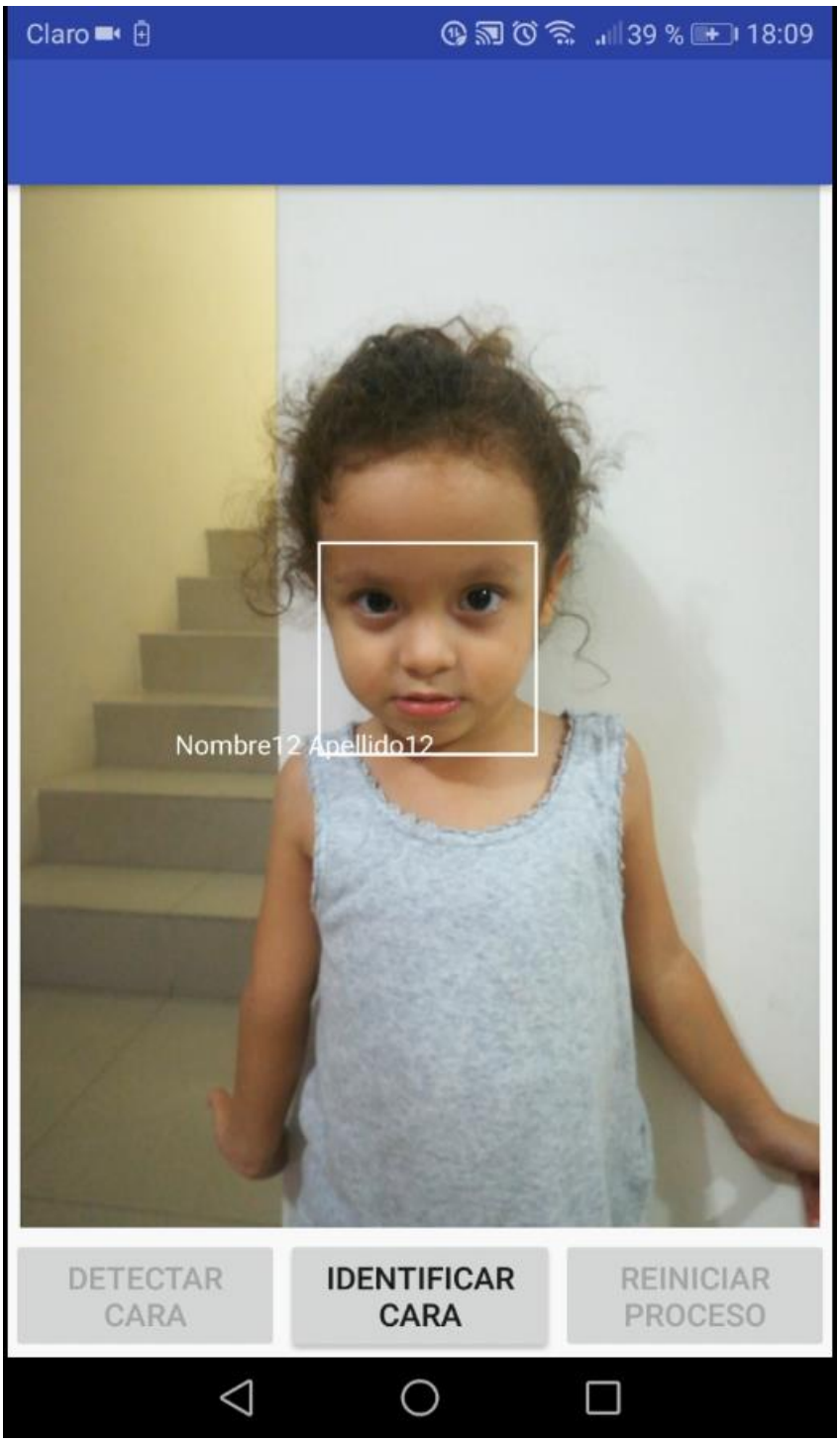

Figura 32 Prueba 1 - Pantalla 7 – Persona Identificada
Elaborado por Alex García Arias

4.4.1.8 Prueba 1 - Pantalla 8 - Identificación exitosa y registro de la persona

El proceso es exitoso porque se identificó a la persona y es registrado su ingreso

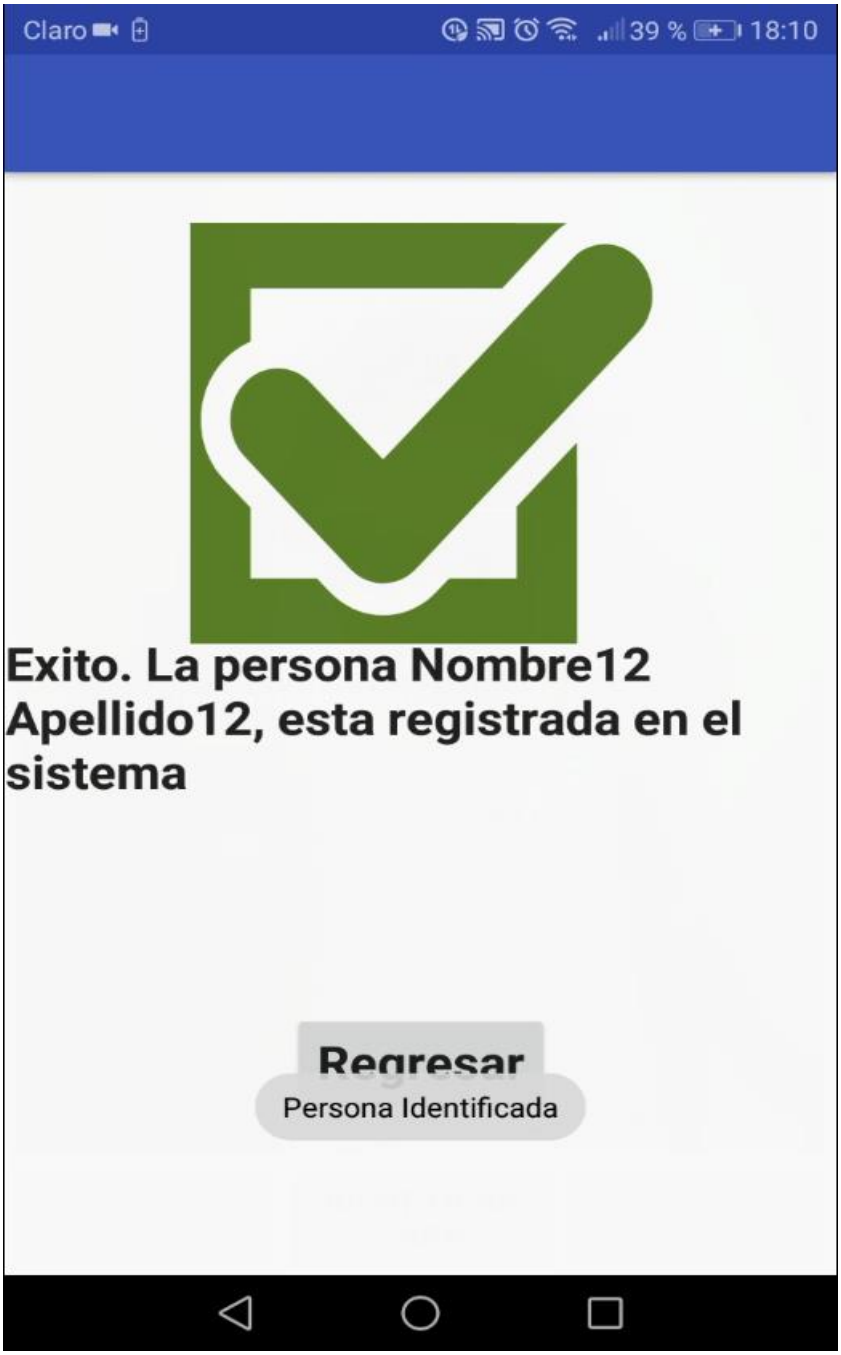

Figura 33 Prueba 1 - Pantalla 8 – Identificación exitosa
Elaborado por Alex García Arias

4.4.2 Prueba 2

4.4.2.1 Prueba 2 - Pantalla 1 - Ubicación GPS y selección de sala

Se obtiene la ubicación GPS y se selecciona la sala

Figura 34 Prueba 2 - Pantalla 1 – Ubicación de GPS
Elaborado por Alex García Arias

4.4.2.2 Prueba 2 - Pantalla 2 - Tomar Foto

Se procede con la toma de foto de la persona

Figura 35 Prueba 2 - Pantalla 2 – Tomar foto
Elaborado por Alex García Arias

4.4.2.3 Prueba 2 - Pantalla 3 - Procesamiento de la foto

Se procede con la detección de la cara de la persona

Figura 36 Prueba 2 - Pantalla 2 – Procesamiento de la foto
Elaborado por Alex García Arias

4.4.2.4 Prueba 2 - Pantalla 4 - Detectando cara

Se detecta la cara de la persona

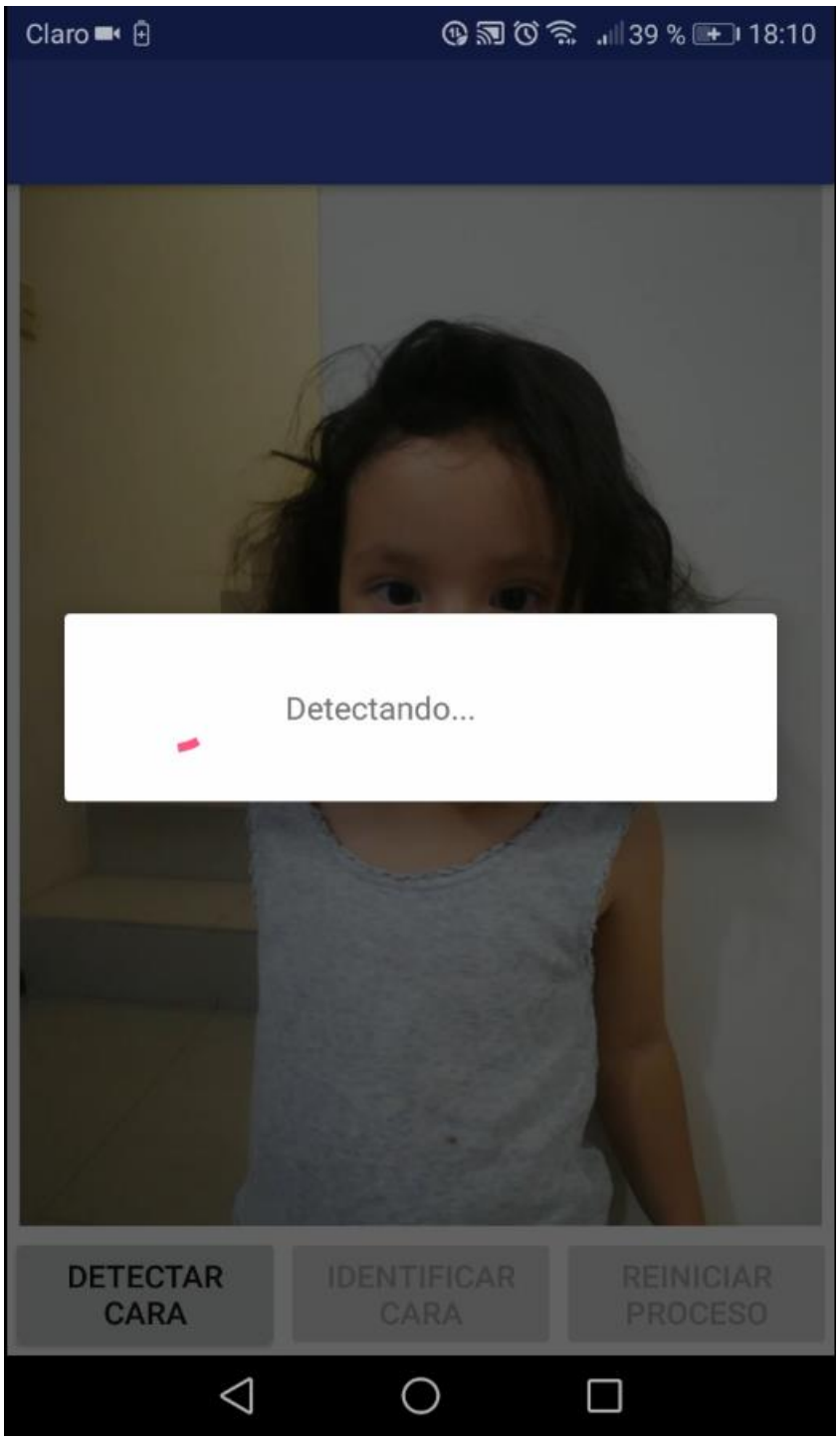

Figura 37 Prueba 2 - Pantalla 4 – Detectando la cara
Elaborado por Alex García Arias

4.4.2.5 Prueba 2 - Pantalla 5 - Identificación de persona

Una vez detectada se procede con la identificación

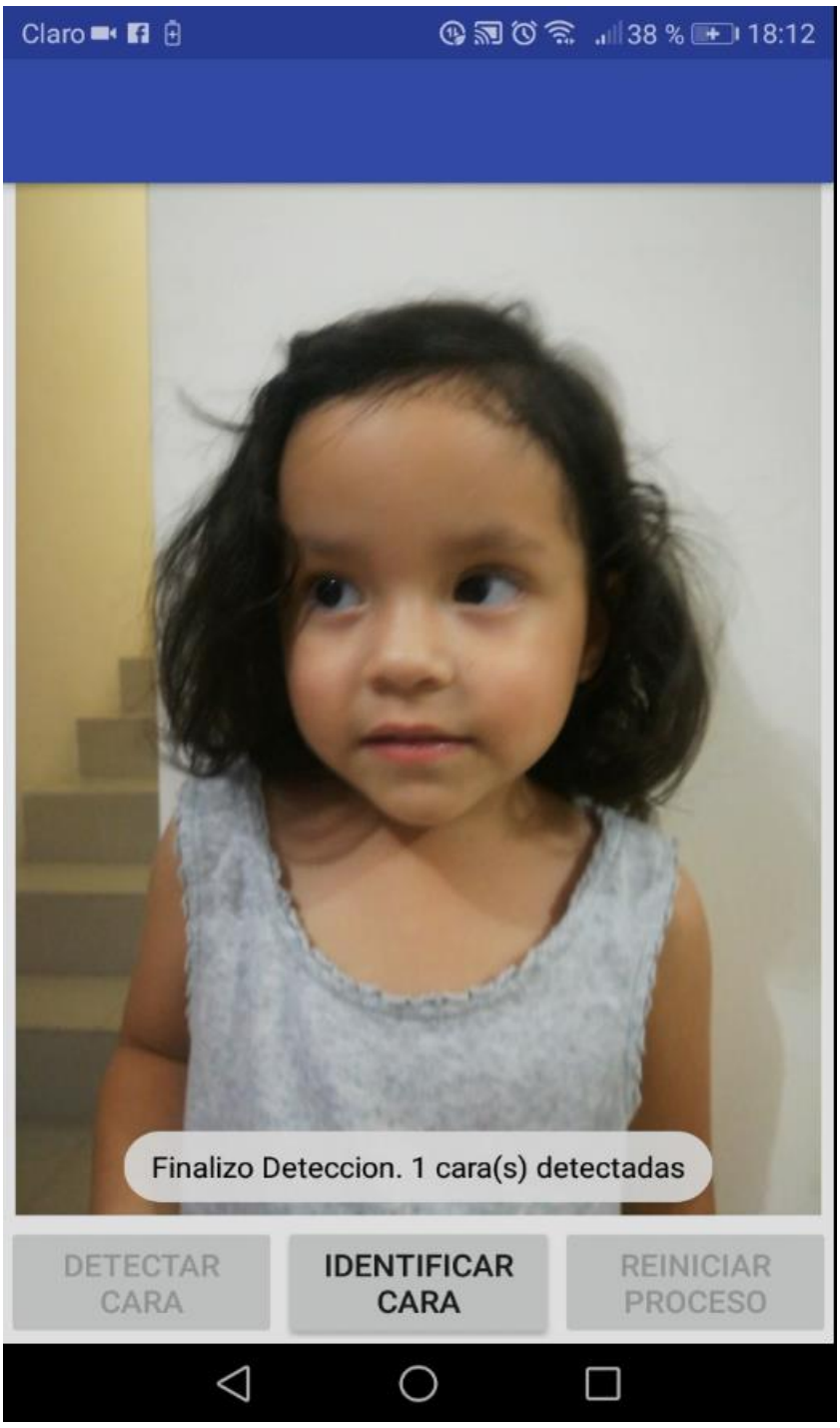

Figura 38 Prueba 2 - Pantalla 5 – Identificación de persona
Elaborado por Alex García Arias

4.4.2.6 Prueba 2 - Pantalla 6 - Identificando persona

Se procede con la identificación de la cara en la nube de Azure si existe la persona.

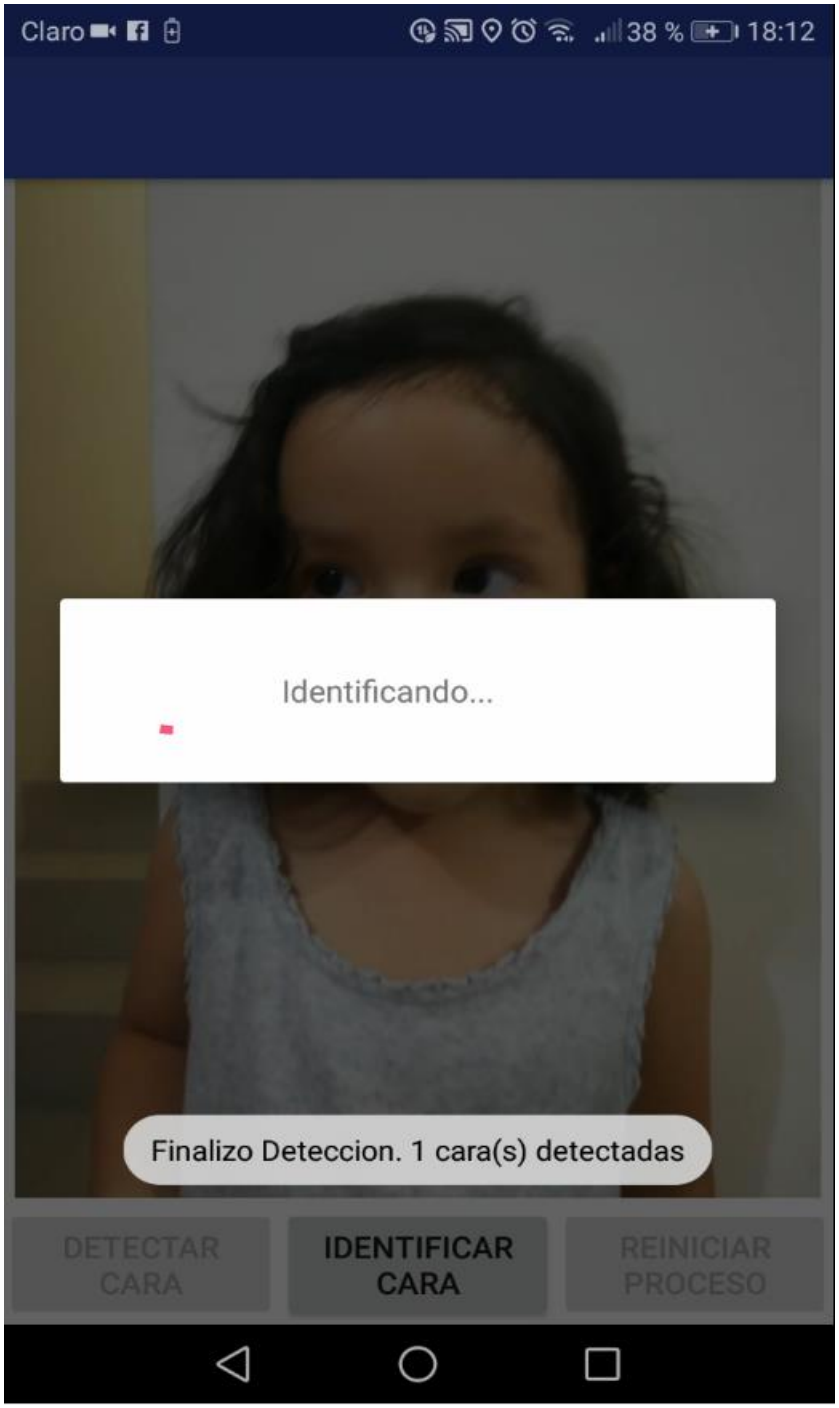

Figura 39 Prueba 2 - Pantalla 6 – Identificando persona
Elaborado por Alex García Arias

4.4.2.7 Prueba 2 - Pantalla 7 - Persona Identificada

El proceso identifico la cara de la persona

Figura 40 Prueba 2 - Pantalla 7 – Persona identificada
Elaborado por Alex García Arias

4.4.2.8 Prueba 2 - Pantalla 8 - Identificación exitosa y registro de la persona

El proceso es exitoso porque se identificó a la persona y es registrado su ingreso

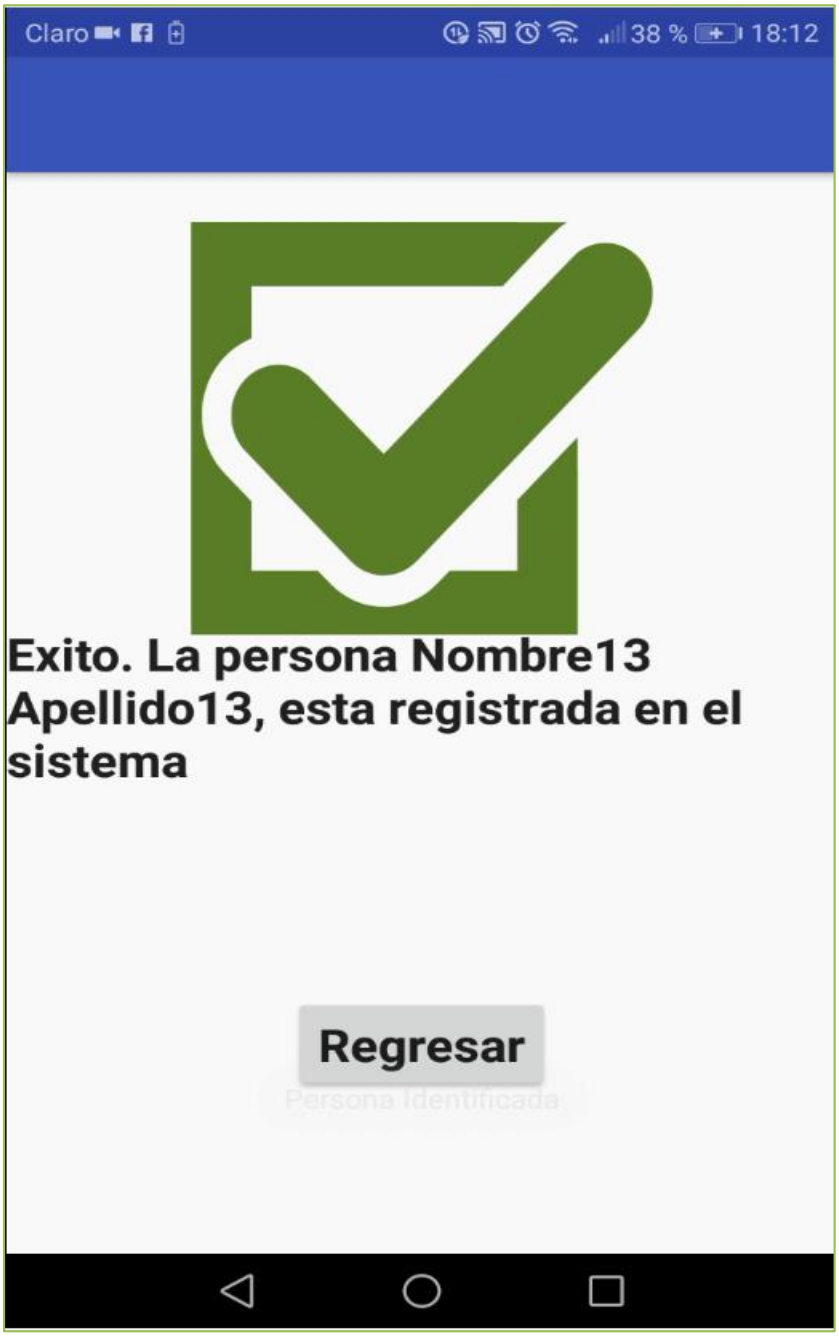

Figura 41 Prueba 2 - Pantalla 8 – Identificación exitosa
Elaborado por Alex García Arias

4.4.3 Prueba 3

4.4.3.1 Prueba 3 - Pantalla 1 - Ubicación GPS y selección de sala

Se obtiene la ubicación GPS y se selecciona la sala

Figura 42 Prueba 3 - Pantalla 1 – Ubicación de GPS
Elaborado por Alex García Arias

4.4.5.2 Prueba 3 - Pantalla 2 - Tomar Foto

Se procede con la toma de foto de la persona

Figura 43 Prueba 3 - Pantalla 2 – Tomar foto
Elaborado por Alex García Arias

4.4.5.3 Prueba 3 - Pantalla 3 - Procesamiento de la foto

Se procede con la detección de la cara de la persona

Figura 44 Prueba 3 - Pantalla 3 – Procesamiento de la foto
Elaborado por Alex García Arias

4.4.5.4 Prueba 3 - Pantalla 4 - Detectando cara

Se detecta la cara de la persona

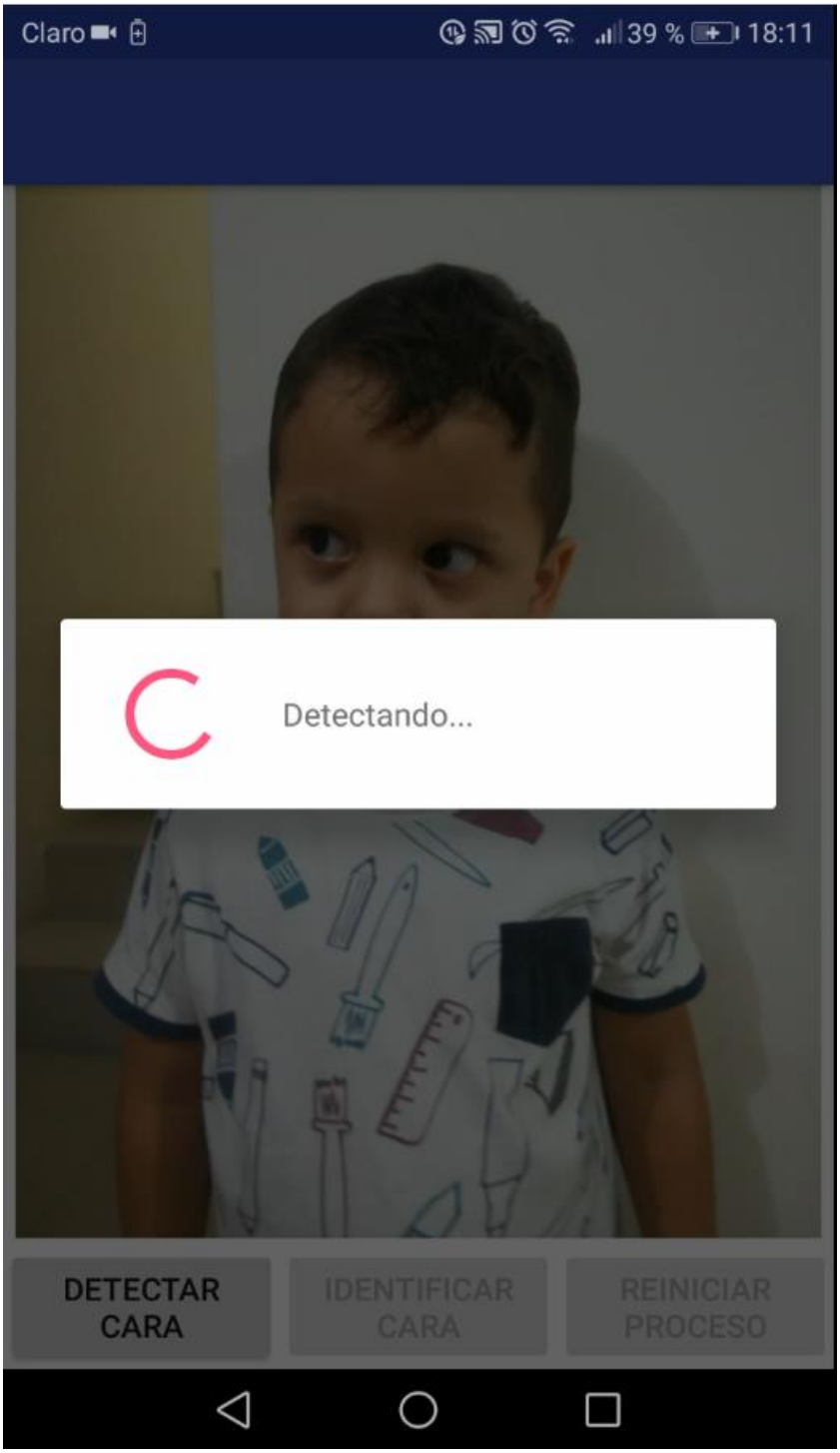

Figura 45 Prueba 3 - Pantalla 4 – Detectando cara
Elaborado por Alex García Arias

4.4.5.5 Prueba 3 - Pantalla 5 - Identificación de persona

Una vez detectada se procede con la identificación

Figura 46 Prueba 3 - Pantalla 5 – Identificación de persona
Elaborado por Alex García Arias

4.4.5.6 Prueba 3 - Pantalla 6 - Identificando persona

Se procede con la identificación de la cara en la nube de Azure si existe la persona.

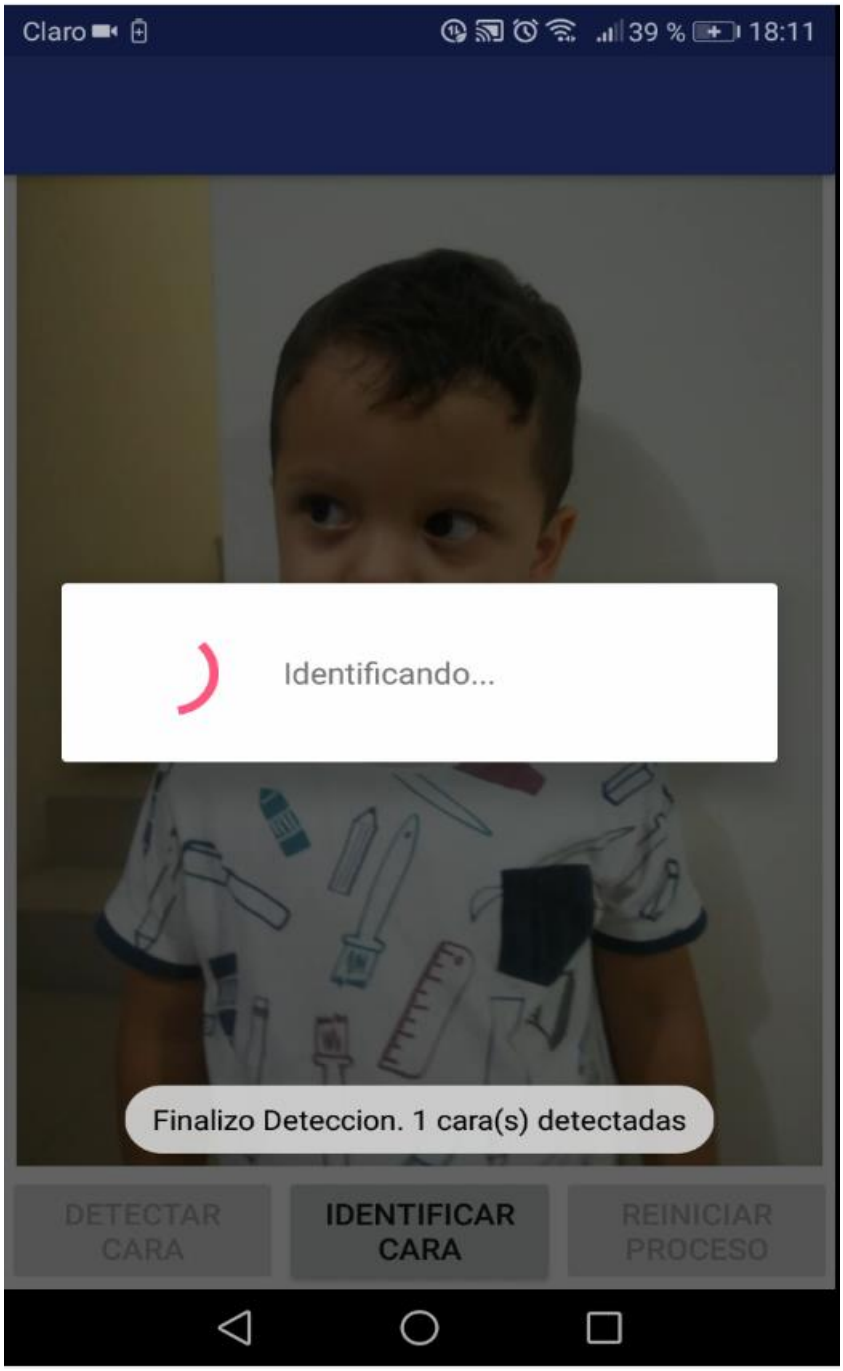

Figura 47 Prueba 3 - Pantalla 6 – Identificando persona
Elaborado por Alex García Arias

4.4.5.7 Prueba 3 - Pantalla 7 - Persona Identificada

El proceso identifico la cara de la persona

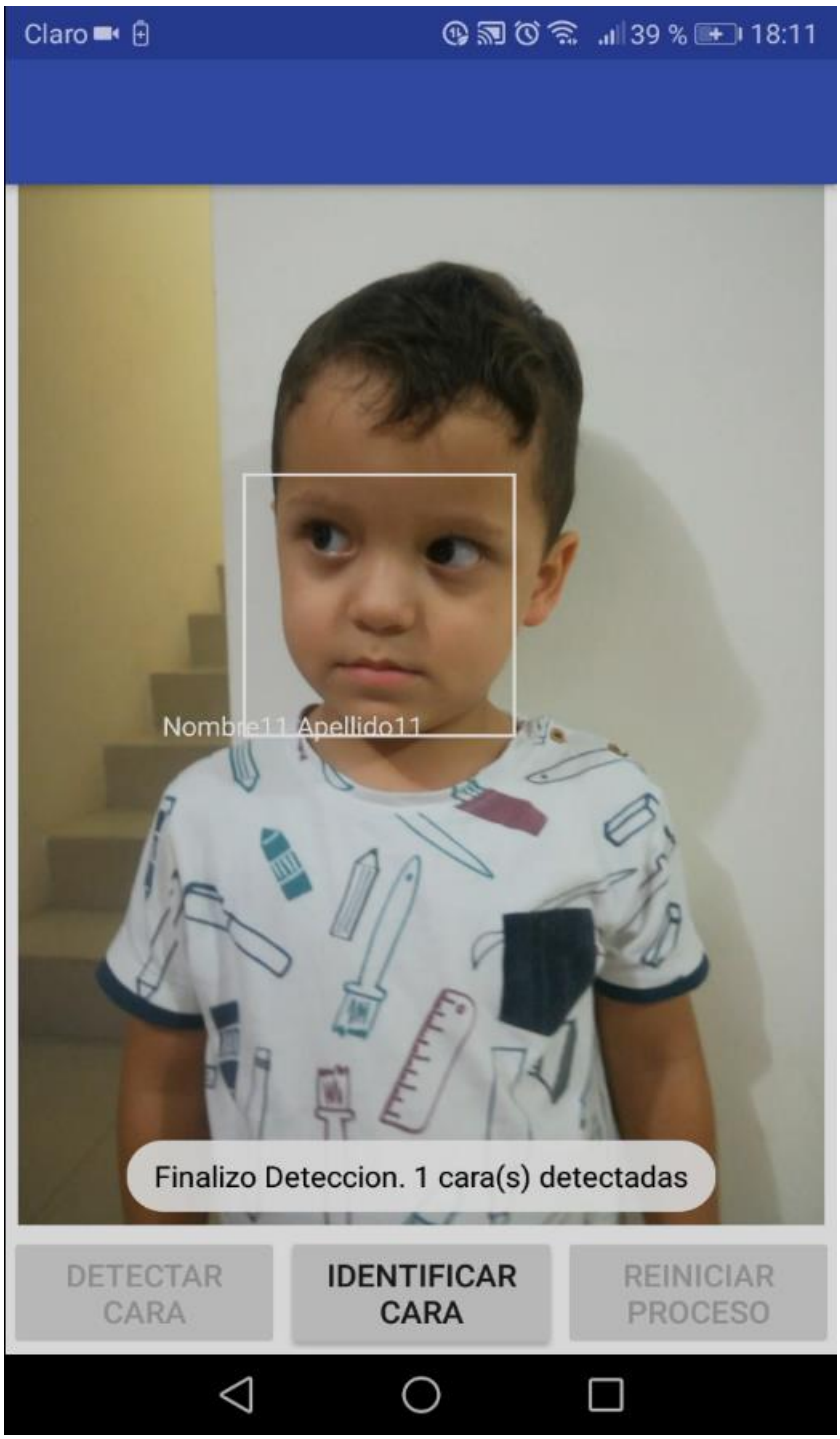

Figura 48 Prueba 3 - Pantalla 7 – Persona identificada
Elaborado por Alex García Arias

4.4.5.8 Prueba 3 - Pantalla 8 - Identificación exitosa y registro de la persona

El proceso es exitoso porque se identificó a la persona y es registrado su ingreso

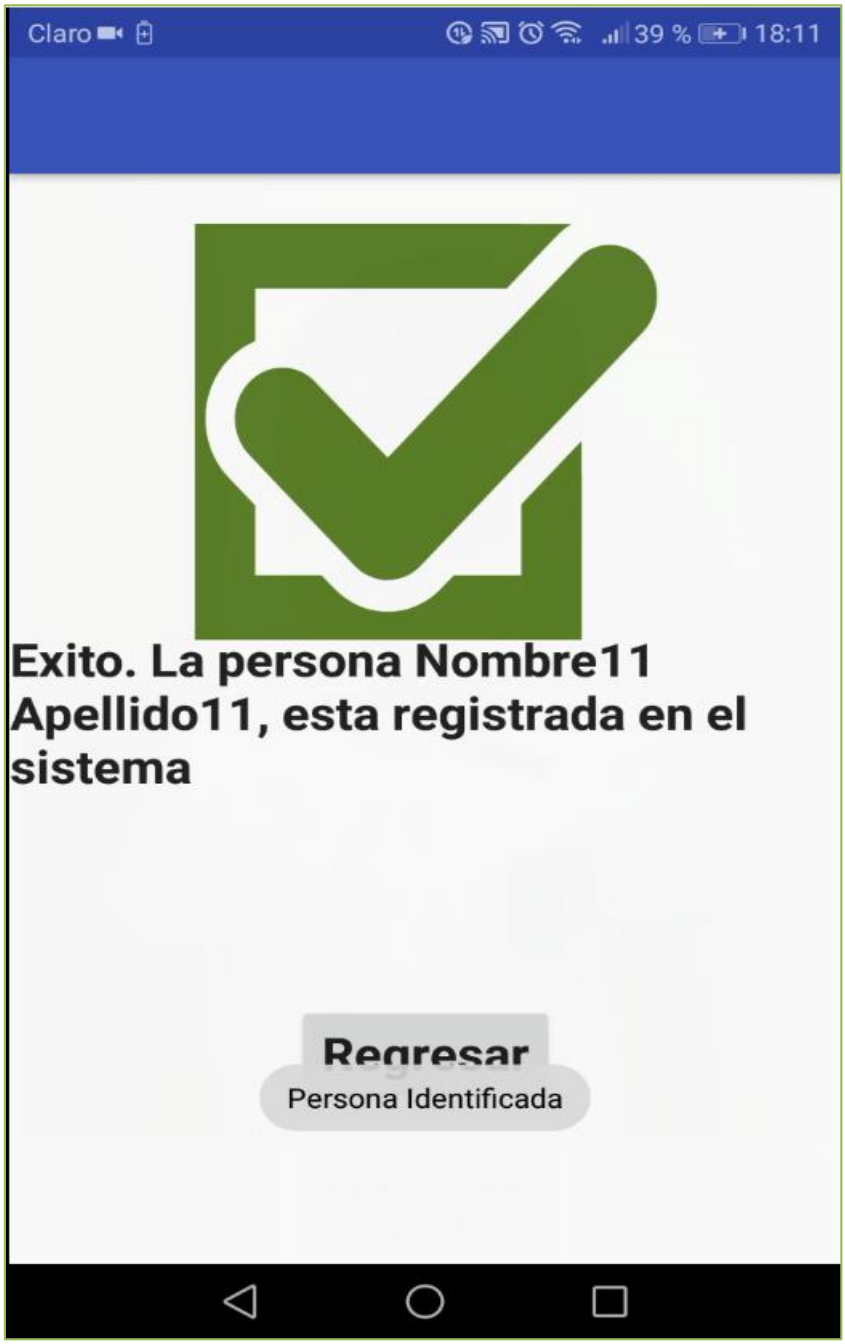

Figura 49 Prueba 3- Pantalla 8 – Identificación exitosa
Elaborado por Alex García Arias

4.4.4 Prueba 4

4.4.4.1 Prueba 4 - Pantalla 1 - Ubicación GPS y selección de sala

Se obtiene la ubicación GPS y se selecciona la sala

Figura 50 Prueba 4 - Pantalla 1 – Ubicación de GPS
Elaborado por Alex García Arias

4.4.4.2 Prueba 4 - Pantalla 2 - Tomar Foto

Se procede con la toma de foto de la persona

Figura 51 Prueba 4 - Pantalla 2 – Tomar foto
Elaborado por Alex García Arias

4.4.4.3 Prueba 4 - Pantalla 3 - Procesamiento de la foto

Se procede con la detección de la cara de la persona

Figura 52 Prueba 4 - Pantalla 3 – Procesamiento de foto
Elaborado por Alex García Arias

4.4.4.4 Prueba 4 - Pantalla 4 - Detectando cara

Se detecta la cara de la persona

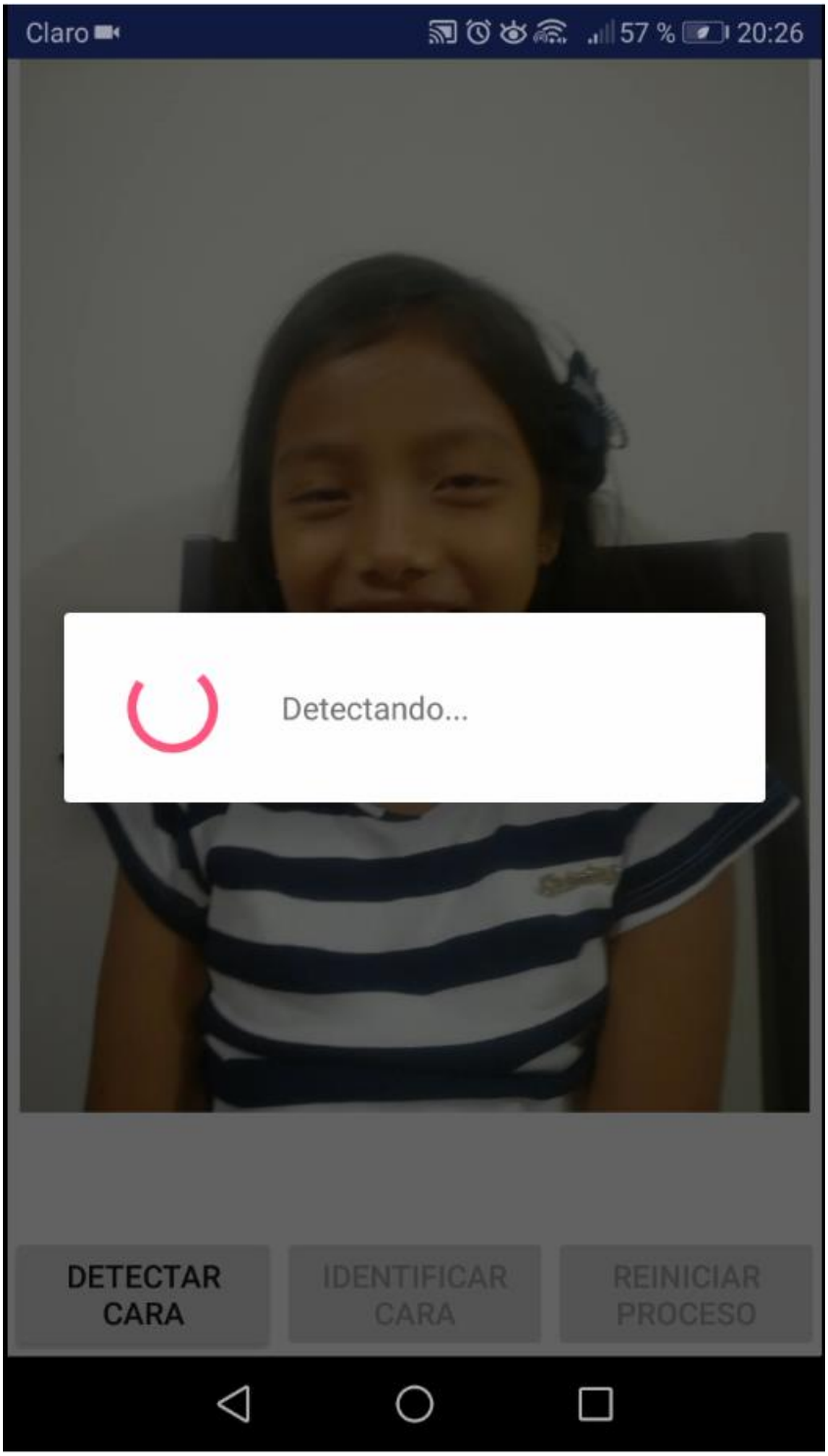

Figura 53 Prueba 4 - Pantalla 4 – Detectando cara
Elaborado por Alex García Arias

4.4.4.5 Prueba 4 - Pantalla 5 - Identificación de persona

Una vez detectada se procede con la identificación

Figura 54 Prueba 4 - Pantalla 5 – Identificación de persona
Elaborado por Alex García Arias

4.4.4.6 Prueba 4 - Pantalla 6 - Identificando persona

Se procede con la identificación de la cara en la nube de Azure si existe la persona.

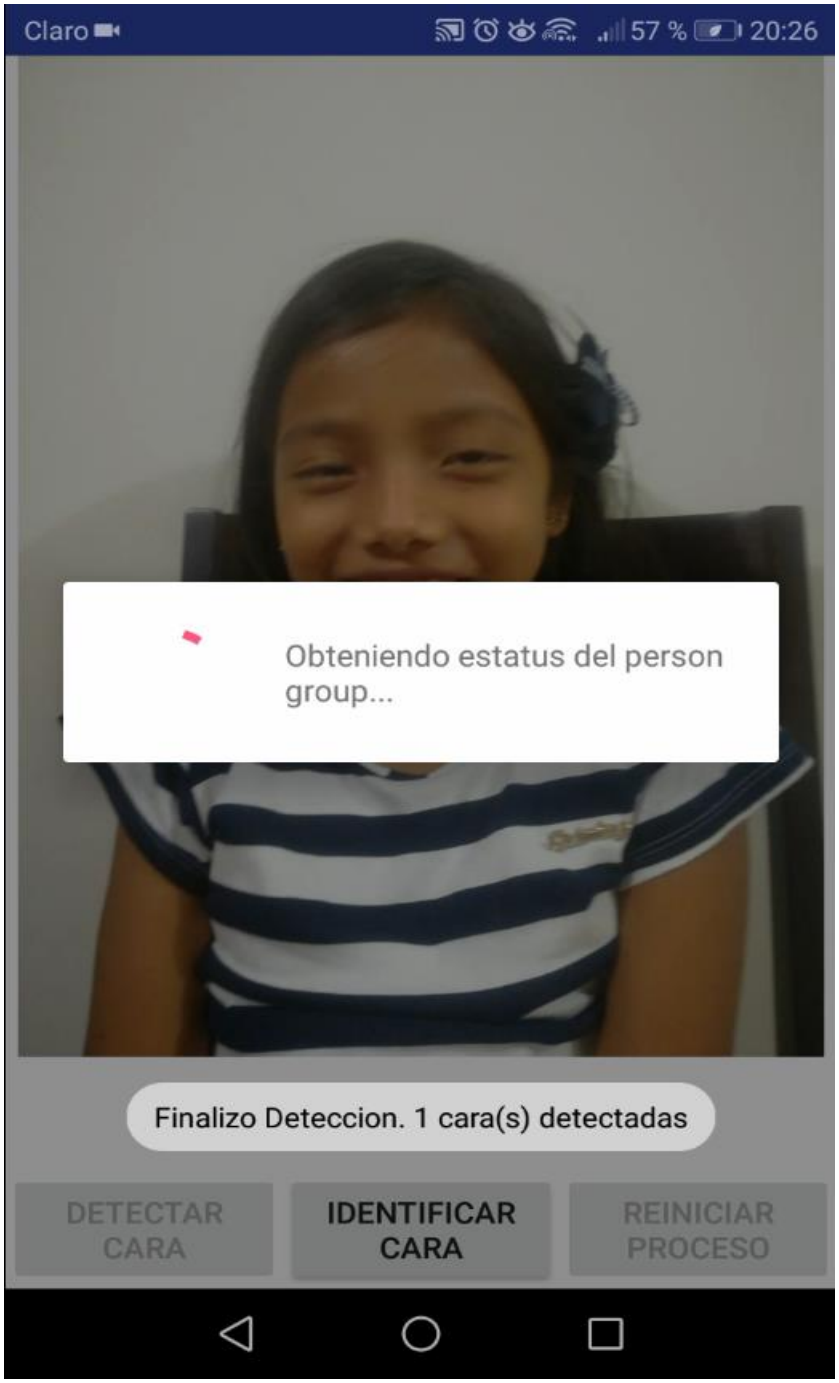

Figura 55 Prueba 4 - Pantalla 6 – Identificando persona
Elaborado por Alex García Arias

4.4.4.7 Prueba 4 - Pantalla 7 - Persona Identificada

El proceso identifico la cara de la persona

Figura 56 Prueba 4 - Pantalla 7 – Persona identificada
Elaborado por Alex García Arias

4.4.4.8 Prueba 4 - Pantalla 8 - Identificación exitosa y registro de la persona

El proceso es exitoso porque se identificó a la persona y es registrado su ingreso

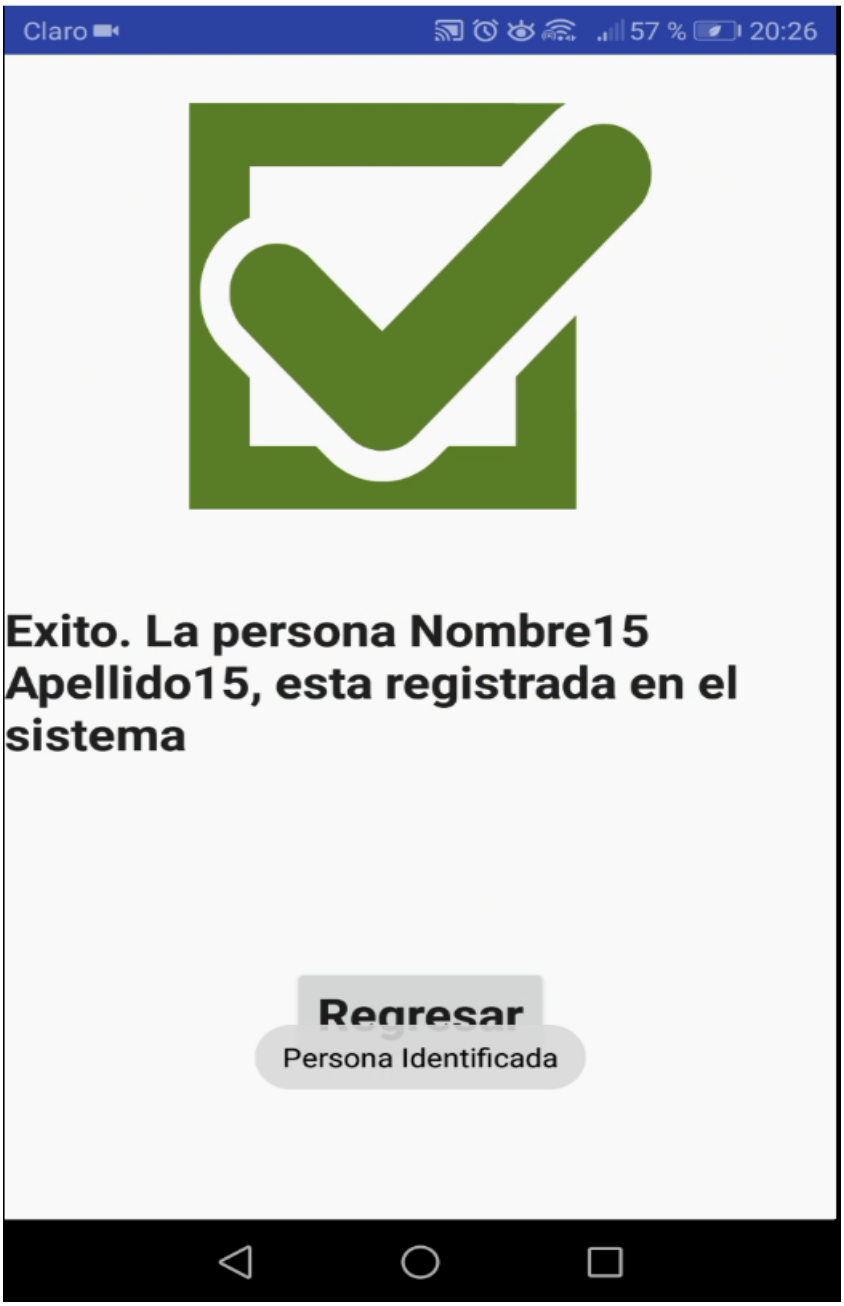

Figura 57 Prueba 4 - Pantalla 8 – Identificación exitosa
Elaborado por Alex García Arias

4.4.5 Prueba 5

4.4.5.1 Prueba 5 - Pantalla 1 - Ubicación GPS y selección de sala

Se obtiene la ubicación GPS y se selecciona la sala

Figura 58 Prueba 5 - Pantalla 1 – Ubicación GPS
Elaborado por Alex García Arias

4.4.5.2 Prueba 5 - Pantalla 2 - Tomar Foto

Se procede con la toma de foto de la persona

Figura 59 Prueba 5 - Pantalla 2 – Tomar foto
Elaborado por Alex García Arias

4.4.5.3 Prueba 5 - Pantalla 3 - Procesamiento de la foto

Se procede con la detección de la cara de la persona

Figura 60 Prueba 5 - Pantalla 3 – Procesamiento de la foto
Elaborado por Alex García Arias

4.4.5.4 Prueba 5 - Pantalla 4 - Detectando cara

Se detecta la cara de la persona

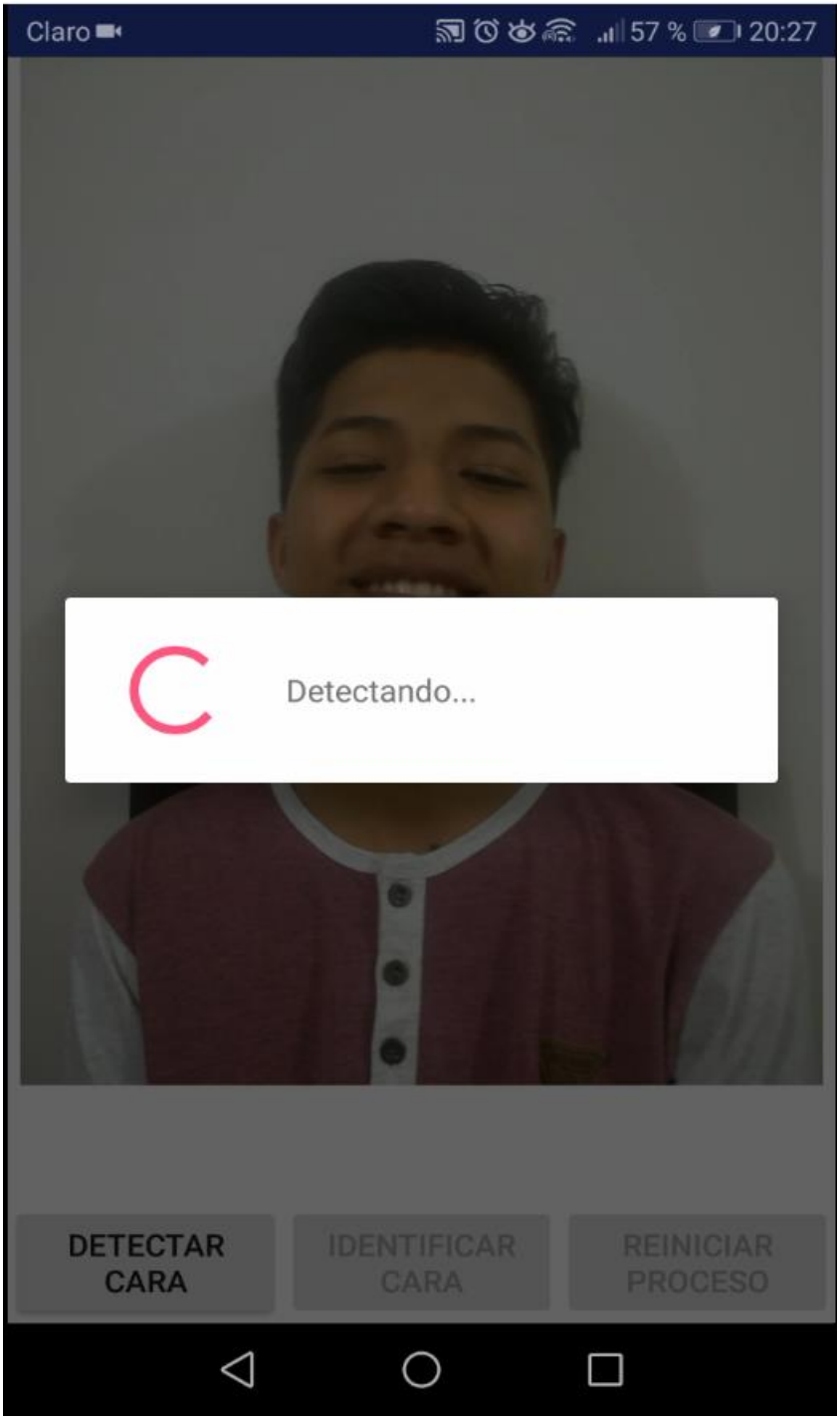

Figura 61 Prueba 5 - Pantalla 4 – Detectando cara
Elaborado por Alex García Arias

4.4.5.5 Prueba 5 - Pantalla 5 - Identificación de persona

Una vez detectada se procede con la identificación

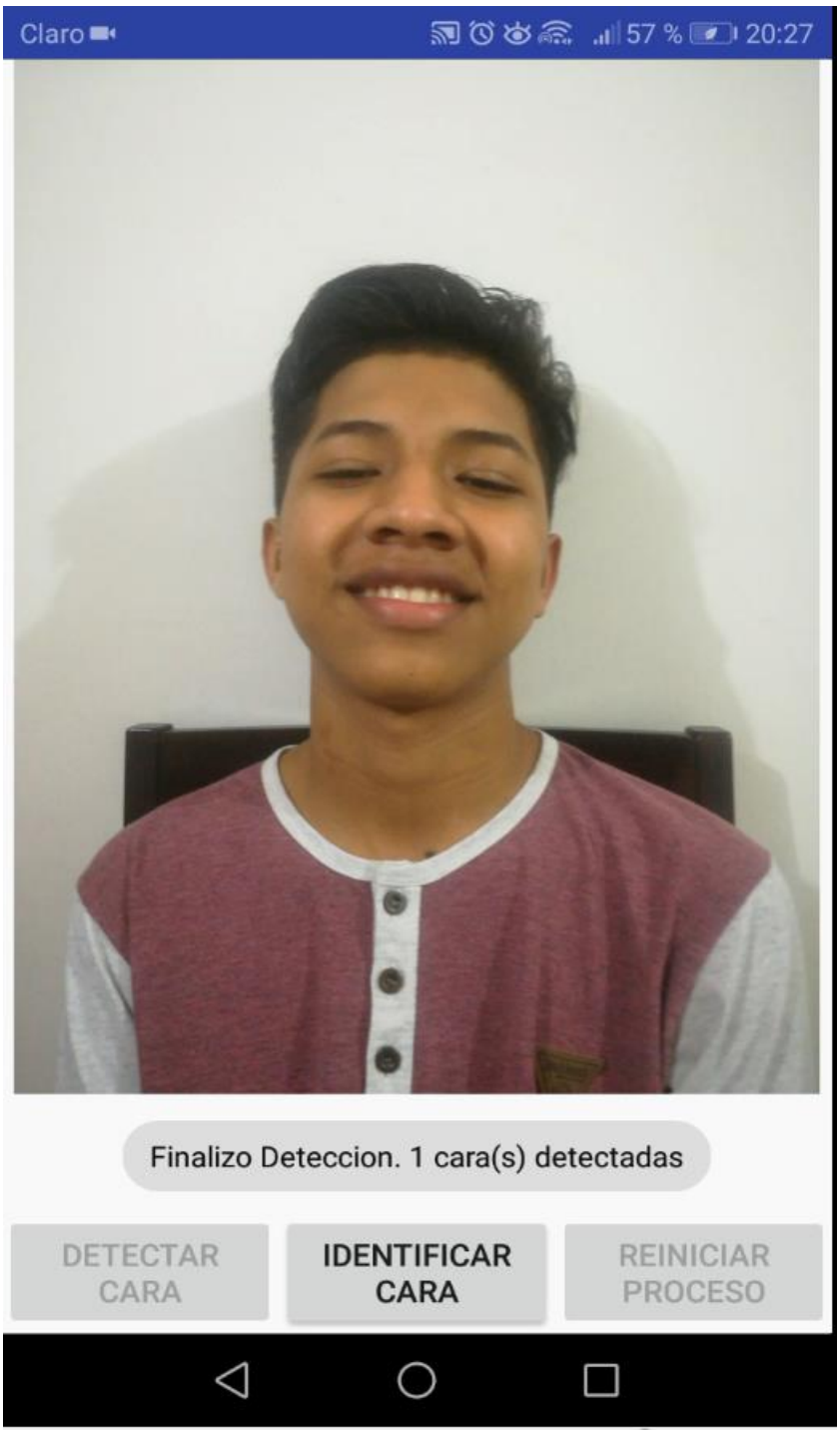

Figura 62 Prueba 5 - Pantalla 5 – Identificación de persona
Elaborado por Alex García Arias

4.4.5.6 Prueba 5 - Pantalla 6 - Identificando persona

Se procede con la identificación de la cara en la nube de Azure si existe la persona.

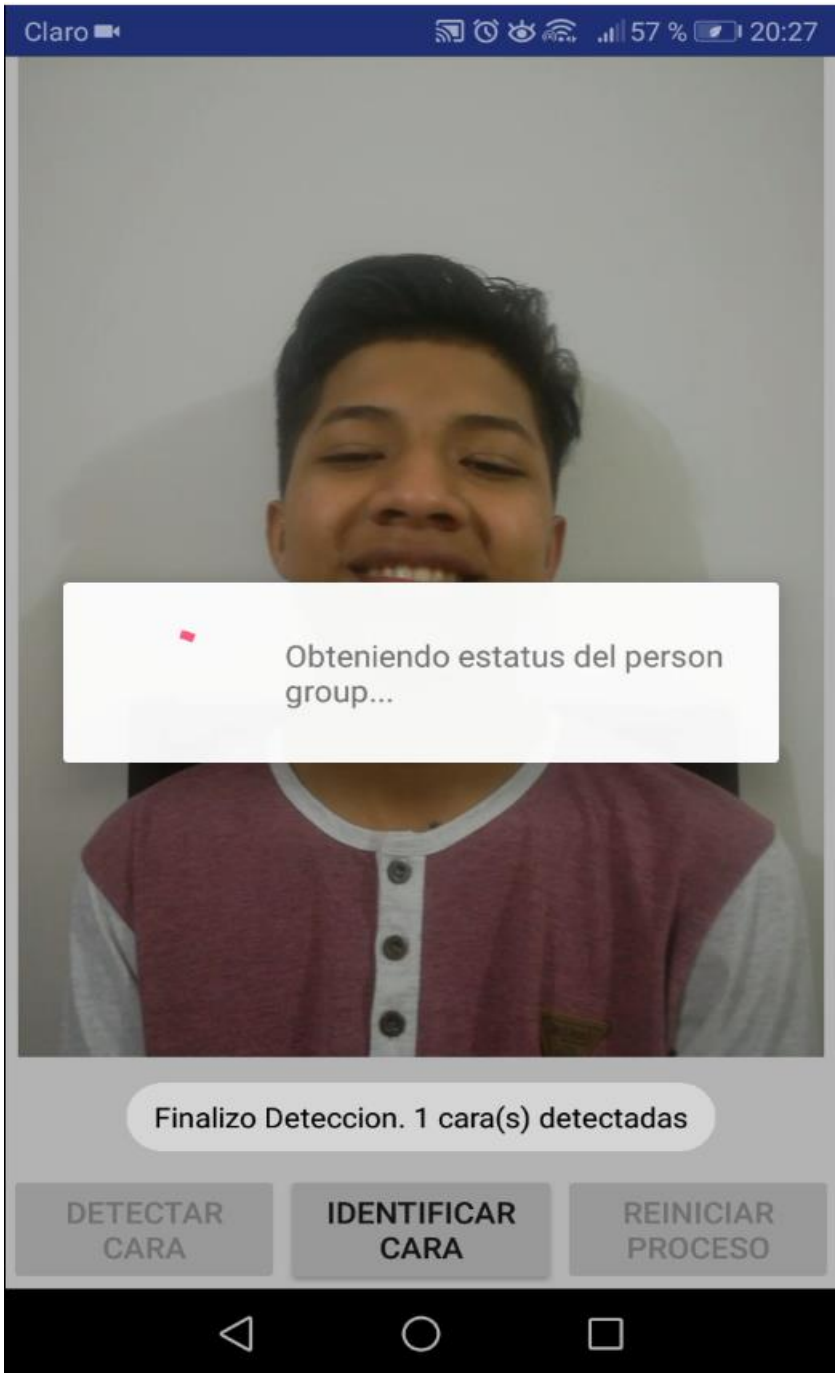

Figura 63 Prueba 5 - Pantalla 6 – Identificando persona
Elaborado por Alex García Arias

4.4.5.7 Prueba 5 - Pantalla 7 - Persona Identificada

El proceso identifico la cara de la persona

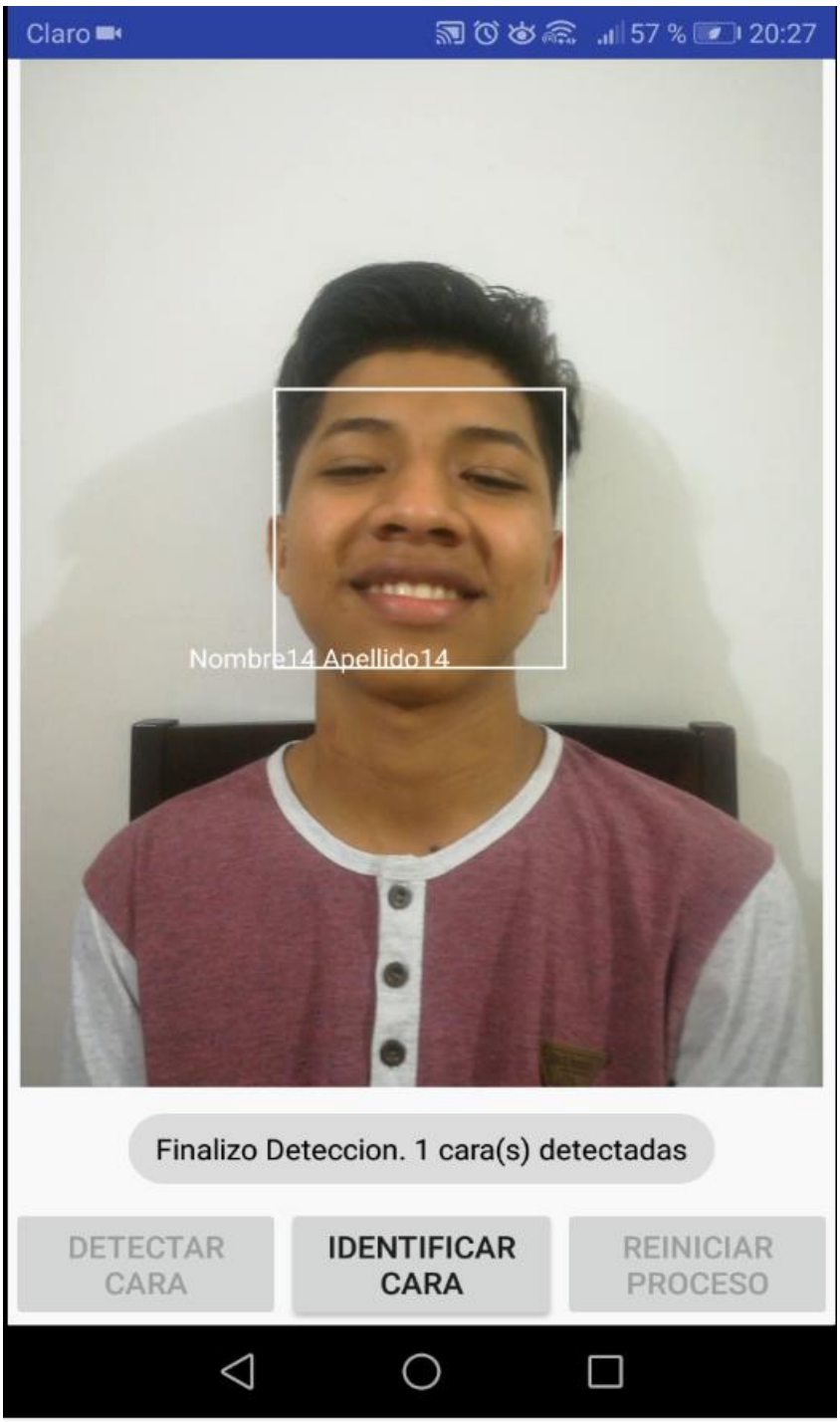

Figura 64 Prueba 5 - Pantalla 7 – Persona identificada
Elaborado por Alex García Arias

4.4.5.8 Prueba 5 - Pantalla 8 - Identificación exitosa y registro de la persona

El proceso es exitoso porque se identificó a la persona y es registrado su ingreso

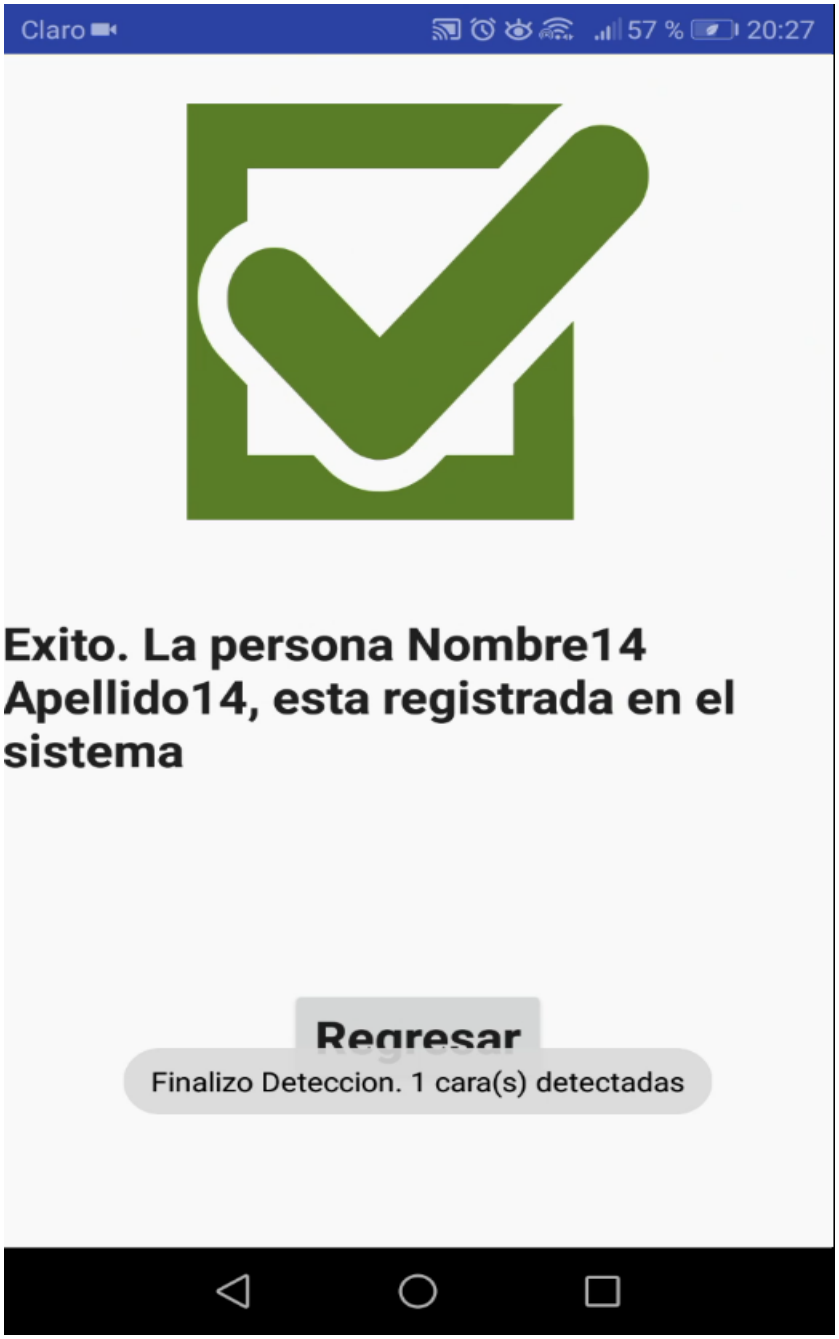

Figura 65 Prueba 5 - Pantalla 8 – Identificación exitosa
Elaborado por Alex García Arias

4.4.6 Prueba 6

4.4.6.1 Prueba 6 - Pantalla 1 - Ubicación GPS y selección de sala

Se obtiene la ubicación GPS y se selecciona la sala

Figura 66 Prueba 6 - Pantalla 1 – Ubicación GPS
Elaborado por Alex García Arias

4.4.6.2 Prueba 6 - Pantalla 2 - Tomar Foto

Se procede con la toma de foto de la persona

Figura 67 Prueba 6 - Pantalla 2 – Toma de foto
Elaborado por Alex García Arias

4.4.6.3 Prueba 6 - Pantalla 3 - Procesamiento de la foto

Se procede con la detección de la cara de la persona

Figura 68 Prueba 6 - Pantalla 3 – Procesamiento de foto
Elaborado por Alex García Arias

4.4.6.4 Prueba 6 - Pantalla 4 - Detectando cara

Se detecta la cara de la persona

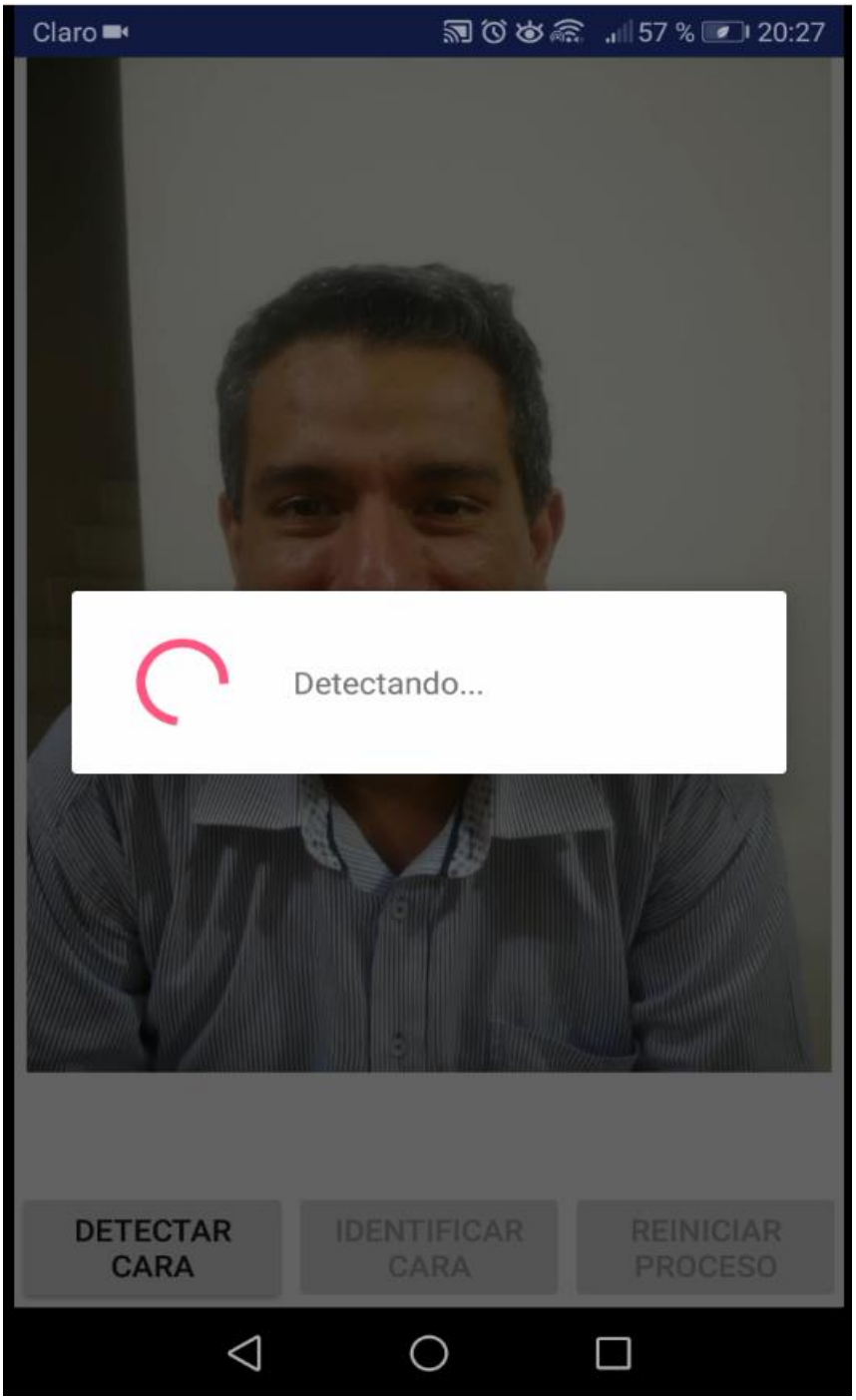

Figura 69 Prueba 6 - Pantalla 3 – Detectando cara
Elaborado por Alex García Arias

4.4.6.5 Prueba 6 - Pantalla 5 - Identificación de persona

Una vez detectada se procede con la identificación

Figura 70 Prueba 6 - Pantalla 5 – Identificación de persona
Elaborado por Alex García Arias

4.4.6.6 Prueba 6 - Pantalla 6 - Identificando persona

Se procede con la identificación de la cara en la nube de Azure si existe la persona.

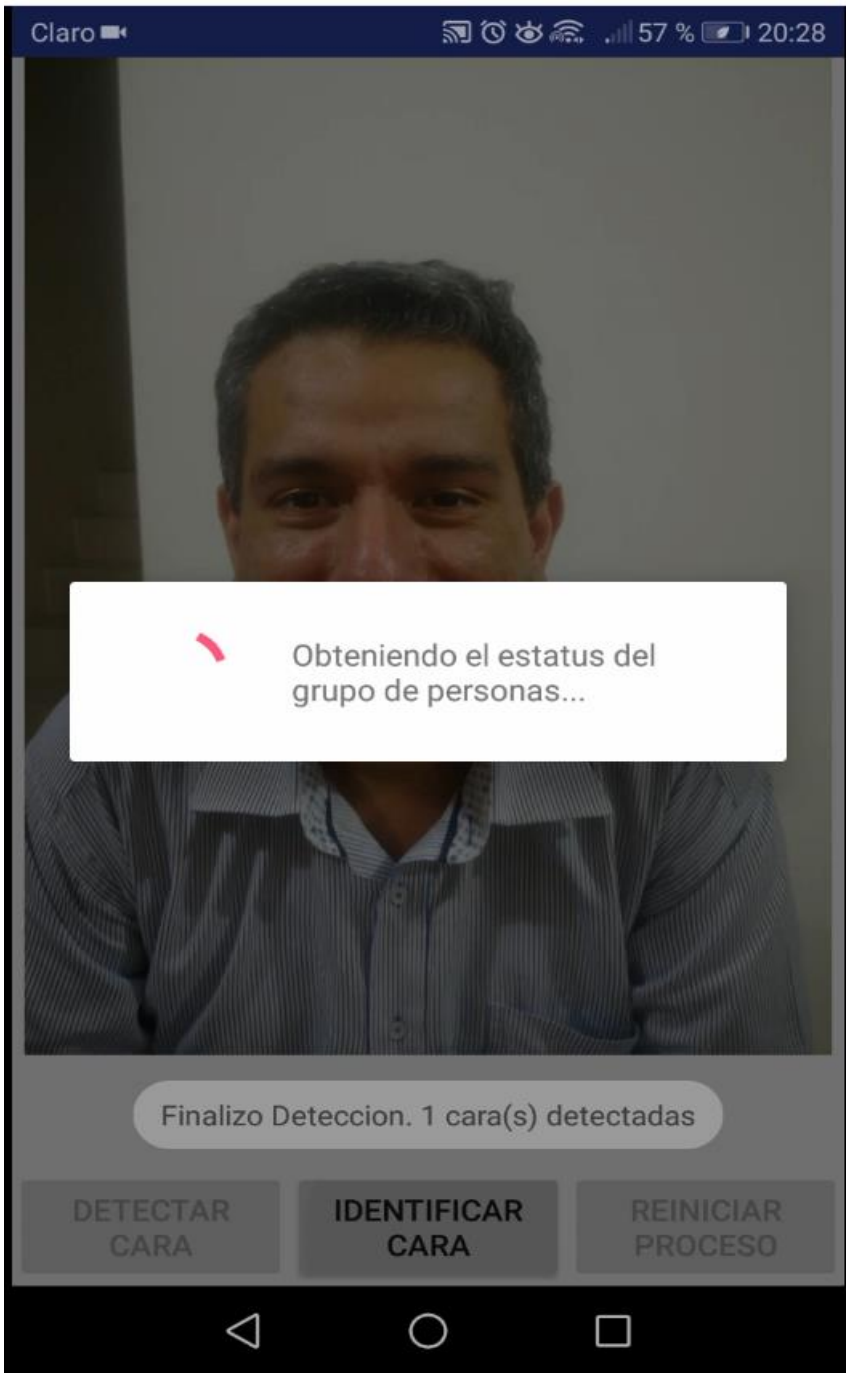

Figura 71 Prueba 6 - Pantalla 6 – Identificando persona
Elaborado por Alex García Arias

4.4.6.7 Prueba 6 - Pantalla 7 - Error persona no identificada

El proceso no identifico la cara de la persona e indica un mensaje de error que no está registrado.

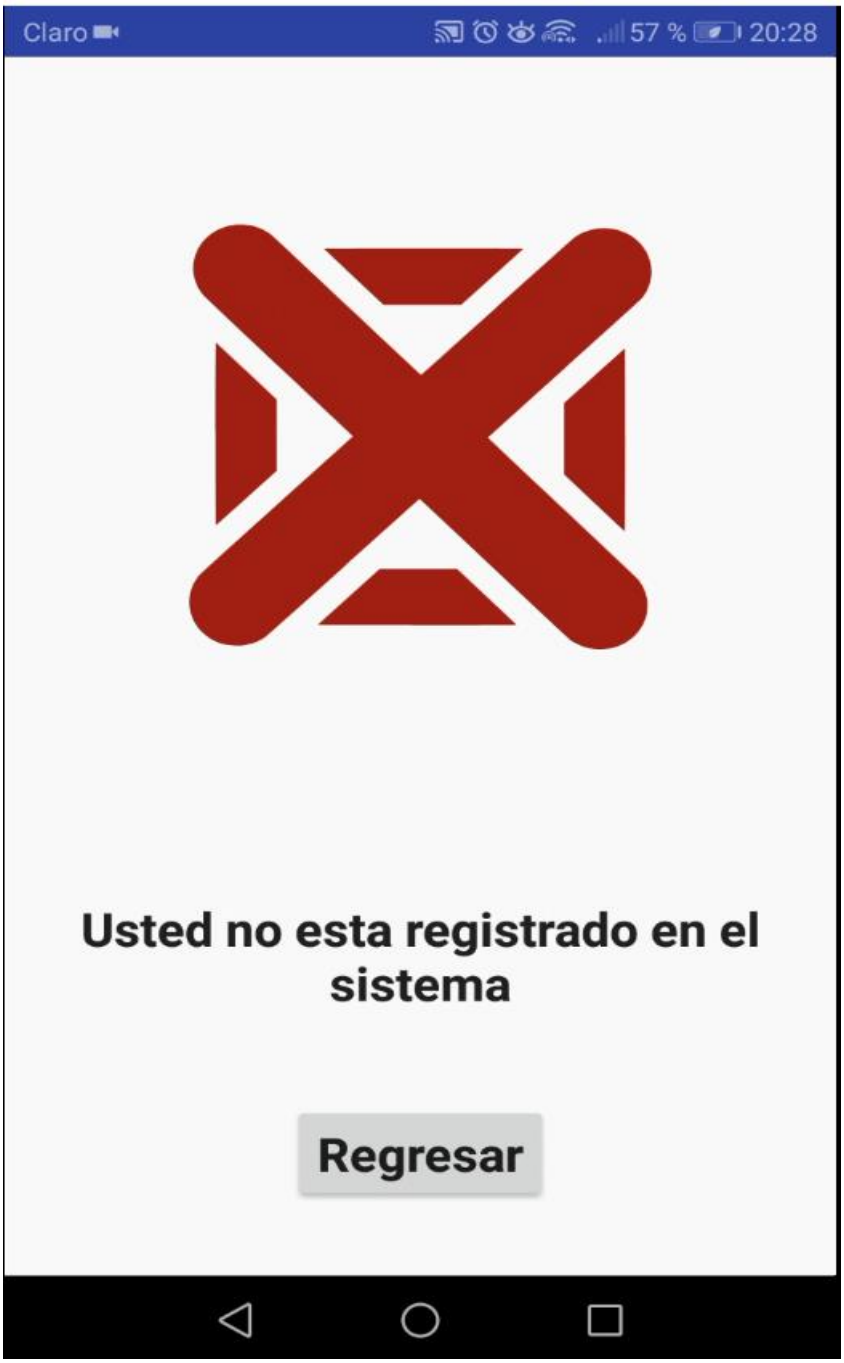

Figura 72 Prueba 6 - Pantalla 7 – Error persona no identificada
Elaborado por Alex García Arias

Conclusiones

En el desarrollo de la tesis se encontraron las siguientes conclusiones.

- La OSAL no tiene un sistema de registro de las personas que ingresan a sus instalaciones a recibir capacitaciones, requisito necesario para que los contribuyentes evidencien la inversión de sus recursos económicos en cada beneficiario directo.

- Los contribuyentes no cuentan con información de que su contribución económica mensual esta invirtiéndose adecuadamente en la educación de la persona que beneficia.

- La importancia del desarrollo de la *App* para la OSAL implica la satisfacción de las necesidades de registro de asistencia de los beneficiarios en las capacitaciones y que cada una de las fotografías se guarden en la nube, para lo que se empleó un pequeño sistema que carga las imagenes en la nube.

- Por último, como conclisión queda abierta la posibilidad de otra investigación sobre el diametro en metros del gps utilizada en el desarrollo de la *App.*

Recomendaciones

Entre las recomendaciones se detalla lo siguiente:

La *App* creada en el proyecto permite tener un registro de cada una de las personas que ingresa al centro de la OSAL para recibir capacitación, lo que se convierte en una evidencia en tiempo real de los beneficiarios de las diferentes aportantes.

Con la información del registro de ingreso del *App*, se cuenta con información veraz para demostrar a las personas que contribuyen un reporte eficaz de las personas que se benefician de su aportación.

La utilización continua de la *App* en la OSAL es recomendable para que los contribuyentes observen en cualquier momento, el registro de asistencia de los beneficiarios en cada una de las capacitaciones, porque las fotografías se guardan en en el sistema de la nube.

La investigación, en su conjunto puede servir de base para el desarrollo de otro proyecto, tomando como base esta *App,* en los que se considere los estándares de programación MVC (Vista Modelo Controlador) y la utilización del diametro en metros del gps en el desarrollo de la *App.*

Bibliografía

Acevedo, J. (7 de Julio de 2018). Obtenido de rogramacion.jias.es/2012/01/web-service-definicion-utilizacion-estructura-del-wsdl/

Arias, F. (2015). *El proyecto de investigación.* México: Epistema.

Atkinson, R. (2015). *Clictomano.* s/e.

Beltrán, P. (2016). *Fundamento de SQL.* México: McGraw Hill.

Bernal, C. (2014). *Metodología de la investigación.* Colombia: Pearson.

Broche, Y., & Rodríguez, M. (2014). Memoria de rostros y reconocimiento emocional: generalidades teóricas, bases neurales y patologías asociadas. *Revistas Académicas* , 4.

Dominguez, S. (2015). Reconocimiento facial mediante el Análisis de los Componentes Principales (PCA). *Universidad de Sevilla*, 4.

Dufrosne, M. (2015). *Las tendencias del 2016.* E - Book.

Espinoza, H. (3 de Agosto de 2018). Diseño e implementación de un sistema de seguridad y alerta para vehículos, basado en reconocimiento facial y localización GPS, en un raspberry PI B Plus. Quito, Ecuador , Sierra.

Gonzalez, C., & Riera, O. (18 de Noviembre de 2018). *ISSN 1699 - 8154*. Obtenido de Redalyc.org: http://www.redalyc.org/pdf/788/78813256008.pdf

Gonzalez, J. (16 de Noviembre de 2018). Obtenido de ISSN1578-7559: https://ddd.uab.cat/pub/tradumatica/tradumatica_a2011n9/tradumatica_a2011n9p5.pdf

Guamán, E., Guamán, J., Erreyes, D., & Torres, H. (2017). *Programación en Java.* Manabí: Mar Abierto.

Gutierrez, Á. (2018). *¿Qué es un App y cómo descargarla?* About Español.

Hernández-Sampieri, Fernández, C., & Batista, P. (2014). *Metodología de la investigación.* México: MacGraw Hill.

Instituto Nacional de Ciberseguridad. (2015). Tecnologías biométricas aplicadas a la ciberseguridad. Madrid: INCIBE.

Invarato, R. (2015). *Android 100%.* Jarroba.

La revista informática.com. (2016). Lenguaje de programación C#. *Revista Informática*, 1.

Landa, N. (2016). *C# La guía del programador.* RedUsers.

Malhotra, N. (2014). *Investigacion de Mercados.* México: Pearson Educación.

Martín-Ávila, T., & López, J. (2015). *El nuevo manifiesto de la web 2.0.* Innova.

Microsoft . (8 de Julio de 2018). Obtenido de https://azure.microsoft.com/es-es/services/cognitive-services/face/

Microsoft. (23 de Agosto de 2018). Obtenido de https://azure.microsoft.com/es-es/overview/what-is-azure/

Microsoft. (6 de Noviembre de 2018). Obtenido de https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Mobile Marketing Móvil. (2015). *Libro Blanco de las Apps.* Kioskoymás.

Muñoz, R. (2013). *La investigación científica paso a paso.* Guayaquil: Interprint.

Oracle Corporation. (6 de Dicembre de 2017). Obtenido de https://www.java.com/es/download/faq/whatis_java.xml

Ortega, J., Alonso, F., & Coomonte, R. (2015). *Biometría y Seguridad.* Madrid: EISIT.

Pacheco, O. (2014). *Fundamentación de investigación educativa.* Guayaquil: Nueva Luz.

Pereira, M. (2 de Octubre de 2018). Obtenido de Ingens-networks: http://www.ingens-networks.com/blog/2012/05/07/consumiento-web-services-soap-en-android/

Pons, C., Giandini, R., & Pérez, G. (2014). *Desarrollo del software dirigido por modelos.* Buenos Aires: McGraw Hill.

Saffirio, M. (10 de Junio de 2018). Obtenido de word press

Santamaría, J., & Hernández, J. (23 de Junio de 2018). Obtenido de SQL SERVER VS MySQL: https://iessanvicente.com/colaboraciones/sqlserver.pdf

Sznajdleder, P. (2015). *Java a fondo .* Alfaomega.

Villegas, C. (12 de Octubre de 2018). Reconocimiento de rostros utilizando análisis de compornentes principales, limitaciones del algoritmo. México, Distrito Federal, México.

Zanini, V., & Hereter, L. (2016). *Android Studio 2.* ISBN 978 - 987.

Zuñiga, O. (2014). ¿Qué dicen tus rasgos faciales sobre tu personalidad? *La mente* , 4.

Anexos

Anexo 1

Fragmento de código WS_recognitionface.asmx

```
1. using System;
2. using System.Collections.Generic;
3.
4. using System.Linq;
5. using System.Web;
6. using System.Web.Services;
7.
8. namespace WS_recognitionface
9. {
10.          /// <summary>
11.          /// Descripción breve de WS_recognitionface
12.          /// </summary>
13.          [WebService(Namespace = "http://tempuri.org/")]
14.          [WebServiceBinding(ConformsTo = WsiProfiles.BasicP
   rofile1_1)]
15.          [System.ComponentModel.ToolboxItem(false)]
16.          // Para permitir que se llame a este servicio web
   desde un script, usando ASP.NET AJAX, quite la marca de
   comentario de la línea siguiente.
17.          // [System.Web.Script.Services.ScriptService]
18.          public class WS_recognitionface : System.Web.Se
   rvices.WebService
19.          {
20.
21.              DataModel dataModel = new DataModel();
22.
23.              //metodo que obtiene las salas dependiendo de
   la latitud y longitud
24.              [WebMethod]
25.              public DataSet
   GetSalas(String latitud, String longitud)
26.              {
27.                  return dataModel.GetSalas(latitud,
   longitud);
28.              }
29.
30.              //metodo que inserta el registro de ingreso
31.              [WebMethod]
32.              public String InsertRegistro(String cedula,
   String idsala)
33.              {
34.                  return dataModel.InsertRegistro(cedula,
   idsala);
35.              }
36.          }
37.      }
```

Anexo 2

Fragmento de código DataModel.cs

```
1. using System;
2. using System.Configuration;
3. using System.Data;
4. using System.Data.SqlClient;
5. using System.Diagnostics;
6.
7. namespace WS_recognitionface
8. {
9.     public class DataModel
10.         {
11.
12.             String sql;
13.
14.             //consulta las salas por la latitud y longitud
15.             public DataSet
   GetSalas(String latitud, String longitud)
16.             {
17.                 DataSet dataTable = new DataSet();
18.
19.                 using (SqlConnection
   con = new SqlConnection(ConfigurationManager.ConnectionStri
   ngs["connectionLocal"].ConnectionString))
20.                 {
21.                     con.Open();
22.
23.                     latitud = latitud.Replace(",", ".");
24.                     longitud = longitud.Replace(",", ".");
25.
26.                     sql = "SELECT A.Id_Centro, B.Id_Sala,
   B.Nombre " +
27.                           "FROM [dbo].[Tbl_Centro] AS A,
   " +
28.                           "     [dbo].[Tbl_Sala] AS B " +
29.                           "WHERE A.Latitud =
   '" + latitud + "' AND " +
30.                           "      A.Longitud =
   '" + longitud + "' AND " +
31.                           "      B.Id_Centro =
   A.Id_Centro";
32.
33.                     using (SqlDataAdapter
   sqlAdapter = new SqlDataAdapter(sql, con))
34.                     {
35.                         sqlAdapter.Fill(dataTable);
36.                     }
37.
38.                     con.Close();
39.                     con.Dispose();
40.                 }
41.
42.                 return dataTable;
43.             }
44.
45.             //insert del registro de ingreso a la sala
46.             public String InsertRegistro(String cedula, St
   ring idsala)
```

```
47.             {
48.                 int idpersona = 0;
49.
50.                 using (SqlConnection
   con = new SqlConnection(ConfigurationManager.ConnectionStri
   ngs["connectionLocal"].ConnectionString))
51.                 {
52.                     try
53.                     {
54.                         con.Open();
55.
56.                         sql = "select a.Id_Persona " +
57.                               "from
   recognitionfacedb.dbo.Tbl_Persona a " +
58.                               "where a.Cedula =
   '" + cedula + "'";
59.
60.                         SqlCommand
   command = new SqlCommand(sql, con);
61.
62.                         using (SqlDataReader
   reader = command.ExecuteReader())
63.                         {
64.                             while (reader.Read())
65.                             {
66.                                 idpersona = reader.GetInt3
   2(0);
67.                             }
68.                             reader.Close();
69.                         }
70.
71.                         //Debug.WriteLine(idpersona);
72.
73.                         if(idpersona > 0)
74.                         {
75.                             sql = "insert into
   recognitionfacedb.dbo.Tbl_Acceso(Id_Acceso, Id_Persona,
   Id_Sala) " +
76.                                   "values(NEXT VALUE FOR
   recognitionfacedb.dbo.sq_idacceso,
   " + idpersona.ToString() + ", " + idsala + ")";
77.
78.                             command = new SqlCommand(sql,
   con);
79.                             command.ExecuteNonQuery();
80.                         }
81.
82.                         command.Dispose();
83.                         con.Close();
84.                         con.Dispose();
85.
86.                         if(idpersona == 0)
87.                         {
88.                             return "Error";
89.                         }
90.
91.                         return "OK";
92.                     }
93.                     catch (Exception ex)
94.                     {
95.                         Debug.WriteLine(ex.ToString());
```

```
                            return ex.ToString();
                        }
                    }
                }
```

Anexo 3

Fragmento de código Web.config

```
<?xml version="1.0" encoding="utf-8"?>

<!--
 Para obtener más información sobre cómo configurar la aplicación ASP.NET, visite
 https://go.microsoft.com/fwlink/?LinkId=169433
 -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.6.1"/>
          <httpRuntime targetFramework="4.6.1"/>
        </system.web>
        <connectionStrings>
          <add name="connectionLocal" connectionString="server=DESKTOP-S552A9T\SQLEXPRESS;database=recognitionfacedb;uid=sa;password=sa;"/>
        </connectionStrings>
        <system.codedom>
          <compilers>
            <compiler language="c#;cs;csharp" extension=".cs"
             type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
             warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>
            <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
             type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
             warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /define:_MYTYPE=\&amp;quot;Web\&amp;quot; /optionInfer+"/>
          </compilers>
        </system.codedom>

      </configuration>
```

Anexo 4

Fragmento de código activity_gps.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout xmlns:android="
   http://schemas.android.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context=".GPSActivity"
8.     tools:layout_editor_absoluteY="25dp">
9.
10.          <TextView
11.              android:id="@+id/tv_longitud"
12.              android:layout_width="wrap_content"
13.              android:layout_height="wrap_content"
14.              android:layout_below="@+id/tv_latitud"
15.              android:layout_marginTop="49dp"
16.              android:text="@string/app_longitud"
17.              android:textAppearance="@android:style/TextAppe
   arance.Large"
18.              android:textStyle="bold"
19.              app:layout_constraintEnd_toStartOf="@+id/cbn_sa
   lon"
20.              app:layout_constraintStart_toStartOf="@+id/tv_s
   alon"
21.              app:layout_constraintTop_toTopOf="parent"
22.              tools:layout_constraintBottom_creator="1"
23.              tools:layout_constraintLeft_creator="1"
24.              tools:layout_constraintTop_creator="1" />
25.
26.          <TextView
27.              android:id="@+id/tv_txtlongitud"
28.              android:layout_width="wrap_content"
29.              android:layout_height="wrap_content"
30.              android:layout_marginTop="35dp"
31.              android:text="@string/app_vlongitud"
32.              android:textAppearance="@android:style/TextAppe
   arance.Large"
33.              app:layout_constraintStart_toStartOf="@+id/tv_t
   xtlatitud"
34.              app:layout_constraintTop_toBottomOf="@+id/tv_tx
   tlatitud"
35.              tools:layout_constraintRight_creator="1"
36.              tools:layout_constraintTop_creator="1" />
37.
38.          <TextView
39.              android:id="@+id/tv_latitud"
40.              android:layout_width="wrap_content"
41.              android:layout_height="wrap_content"
42.              android:layout_marginTop="35dp"
43.              android:text="@string/app_latitud"
44.              android:textAppearance="@android:style/TextAppe
   arance.Large"
45.              android:textStyle="bold"
46.              app:layout_constraintStart_toStartOf="@+id/tv_l
   ongitud"
```

```
47.             app:layout_constraintTop_toBottomOf="@+id/tv_lo
    ngitud"
48.             tools:layout_constraintLeft_creator="1"
49.             tools:layout_constraintTop_creator="1" />
50.
51.         <TextView
52.             android:id="@+id/tv_txtlatitud"
53.             android:layout_width="wrap_content"
54.             android:layout_height="wrap_content"
55.             android:layout_marginEnd="100dp"
56.             android:layout_marginTop="49dp"
57.             android:text="@string/app_vlatitud"
58.             android:textAppearance="@android:style/TextAppe
    arance.Large"
59.             app:layout_constraintEnd_toEndOf="parent"
60.             app:layout_constraintTop_toTopOf="parent"
61.             tools:layout_constraintLeft_creator="1"
62.             tools:layout_constraintRight_creator="1"
63.             tools:layout_constraintTop_creator="1" />
64.
65.         <Button
66.             android:id="@+id/btn_localizacion"
67.             android:layout_width="wrap_content"
68.             android:layout_height="wrap_content"
69.             android:layout_marginStart="17dp"
70.             android:layout_marginTop="46dp"
71.             android:text="@string/app_obtener"
72.             android:textAppearance="@android:style/TextAppe
    arance.Large"
73.             android:textStyle="bold"
74.             app:layout_constraintStart_toStartOf="@+id/tv_l
    atitud"
75.             app:layout_constraintTop_toBottomOf="@+id/tv_la
    titud"
76.             tools:layout_constraintBottom_creator="1"
77.             tools:layout_constraintLeft_creator="1"
78.             tools:layout_constraintTop_creator="1" />
79.
80.         <TextView
81.             android:id="@+id/tv_salon"
82.             android:layout_width="wrap_content"
83.             android:layout_height="wrap_content"
84.             android:layout_marginBottom="216dp"
85.             android:layout_marginStart="60dp"
86.             android:text="@string/lbl_salon"
87.             android:textAppearance="@android:style/TextAppe
    arance.Large"
88.             android:textStyle="bold"
89.             app:layout_constraintBottom_toBottomOf="parent"
90.             app:layout_constraintStart_toStartOf="parent"
91.             tools:layout_constraintBottom_creator="1"
92.             tools:layout_constraintLeft_creator="1"
93.             tools:layout_constraintTop_creator="1" />
94.
95.         <Spinner
96.             android:id="@+id/cbn_salon"
97.             android:layout_width="129dp"
98.             android:layout_height="44dp"
99.             android:layout_marginEnd="97dp"
100.            android:layout_marginStart="98dp"
```

```
            app:layout_constraintBottom_toBottomOf="@+id/tv_salon"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="@+id/tv_salon"
            tools:layout_constraintBottom_creator="1"
            tools:layout_constraintRight_creator="1"
            tools:layout_constraintTop_creator="1" />

        <Button
            android:id="@+id/btn_siguiente"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="81dp"
            android:enabled="false"
            android:text="@string/btn_siguiente"
            android:textAppearance="@android:style/TextAppearance.Large"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            tools:layout_constraintBottom_creator="1"
            tools:layout_constraintLeft_creator="1"
            tools:layout_constraintTop_creator="1" />

    </android.support.constraint.ConstraintLayout>
```

Anexo 5

Fragmento de código GPSActivity.java

```
package com.uteg.recognitionface;

import android.Manifest;
import android.annotation.TargetApi;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.app.Service;
import android.content.DialogInterface;
import android.content.Intent;
      import android.content.pm.PackageManager;
      import android.location.Criteria;
      import android.location.Location;
      import android.location.LocationListener;
      import android.location.LocationManager;
      import android.os.AsyncTask;
      import android.os.Build;
      import android.os.StrictMode;
      import android.provider.Settings;
      import android.support.v7.app.AppCompatActivity;
      import android.os.Bundle;
      import android.util.Log;
      import android.view.View;
      import android.widget.ArrayAdapter;
      import android.widget.Button;
      import android.widget.Spinner;
      import android.widget.TextView;
      import android.widget.Toast;

```

```
29.       import org.ksoap2.SoapEnvelope;
30.       import org.ksoap2.serialization.PropertyInfo;
31.       import org.ksoap2.serialization.SoapObject;
32.       import org.ksoap2.serialization.SoapSerializationEnvel
   ope;
33.       import org.ksoap2.transport.HttpTransportSE;
34.
35.       import java.util.ArrayList;
36.
37.       public class GPSActivity extends AppCompatActivity imp
   lements LocationListener {
38.
39.           final String TAG = "GPS";
40.           private final static int ALL_PERMISSIONS_RESULT =
   101;
41.           private static final long MIN_DISTANCE_CHANGE_FOR_
   UPDATES = 10;
42.           private static final long MIN_TIME_BW_UPDATES = 10
   00 * 60 * 1;
43.           private Button btn_localizacion, btn_siguiente;
44.           private TextView tv_longitud, tv_latitud;
45.           private double longitud, latitud;
46.           private Spinner cbn_salon;
47.           private String[] parametros = null;
48.           LocationManager locationManager;
49.           Location loc;
50.           ArrayList<String> permissions = new ArrayList<>();
51.           ArrayList<String> permissionsToRequest;
52.           ArrayList<String> permissionsRejected = new ArrayL
   ist<>();
53.           boolean isGPS = false;
54.           boolean isNetwork = false;
55.           boolean canGetLocation = true;
56.           public static ArrayList<Salas> salas = new ArrayLi
   st<>();
57.           public static Salas sala = null;
58.           private String[] ssalas = null;
59.           private ArrayAdapter<String> combo = null;
60.
61.           @Override
62.           protected void onCreate(Bundle
   savedInstanceState) {
63.               super.onCreate(savedInstanceState);
64.               setContentView(R.layout.activity_gps);
65.
66.               btn_localizacion = (Button) findViewById(R.id.
   btn_localizacion);
67.               btn_siguiente = (Button) findViewById(R.id.btn
   _siguiente) ;
68.               cbn_salon = (Spinner) findViewById(R.id.cbn_sa
   lon);
69.               tv_longitud = (TextView) findViewById(R.id.tv_
   txtlongitud);
70.               tv_latitud = (TextView) findViewById(R.id.tv_t
   xtlatitud);
71.
72.               locationManager = (LocationManager) getSystemS
   ervice(Service.LOCATION_SERVICE);
73.               isGPS = locationManager.isProviderEnabled(Loca
   tionManager.GPS_PROVIDER);
```

```
74.             isNetwork = locationManager.isProviderEnabled(
    LocationManager.NETWORK_PROVIDER);
75.
76.             permissions.add(Manifest.permission.ACCESS_FIN
    E_LOCATION);
77.             permissions.add(Manifest.permission.ACCESS_COA
    RSE_LOCATION);
78.             permissions.add(Manifest.permission.INTERNET);
79.             permissions.add(Manifest.permission.ACCESS_NET
    WORK_STATE);
80.             permissionsToRequest = findUnAskedPermissions(
    permissions);
81.
82.             // chequeo de permisos GPS
83.             if (Build.VERSION.SDK_INT >= Build.VERSION_COD
    ES.M)
84.             {
85.                 if (permissionsToRequest.size() > 0)
86.                 {
87.                     requestPermissions(permissionsToReques
    t.toArray(new String[permissionsToRequest.size()]),
88.                             ALL_PERMISSIONS_RESULT);
89.                     StrictMode.ThreadPolicy policy = new S
    trictMode.ThreadPolicy.Builder().permitAll().build();
90.                     StrictMode.setThreadPolicy(policy);
91.                     Log.i(TAG,
    getString(R.string.msg_permiso));
92.                     canGetLocation = false;
93.                 }
94.             }
95.
96.             //evento para realizar la localizacion y
    cargar el list
97.             btn_localizacion.setOnClickListener(new View.O
    nClickListener() {
98.                 @Override
99.                 public void onClick(View view) {
100.
101.                    //verificar si la conexion gps esta
    apagada
102.                    if (!isGPS && !isNetwork)
103.                    {
104.                        Log.i(TAG,
    getString(R.string.err_conexion_off));
105.                        //solicitud de encender el gps
106.                        showSettingsAlert();
107.                        //obtiene la ultima localizacion
    gps
108.                        getLastLocation();
109.                    }
110.                    else
111.                    {
112.                        Log.i(TAG,
    getString(R.string.err_conexion_on));
113.
114.                        //metodo para obtener la
    geolocalizacion
115.                        getLocation();
116.
117.                        if(latitud == 0 && longitud == 0)
118.                        {
```

```
119.                                 Toast.makeText(getApplicationC
    ontext(),
120.                                         "No se ha podido
    obtener la localizacion del punto. Intente nuevamente",
121.                                         Toast.LENGTH_LONG).sho
    w();
122.
123.                                 Log.i("ActGpsActivity", "No se
    ha podido obtener la localizacion del punto");
124.                             }
125.                             else
126.                             {
127.                                 parametros = new String[2];
128.                                 parametros[0] = String.format(
    "%.2f",latitud);
129.                                 parametros[1] = String.format(
    "%.2f",longitud);
130.
131.                                 //se llama al metodo de
    consuta de webservice para carga de las salas en el combo
132.                                 new CallWebService(parametros)
    .execute();
133.
134.                             }
135.                         }
136.                     }
137.                 });
138.
139.                 //accion para seguir a la siguiente pantalla
    que es la de tomar la foto
140.                 btn_siguiente.setOnClickListener(new View.OnCl
    ickListener(){
141.                     @Override
142.                     public void onClick(View v) {
143.                         sala = salas.get(cbn_salon.getSelected
    ItemPosition());
144.                         Intent
    intent = new Intent(GPSActivity.this, FotoActivity.class);
145.                         intent.putExtra("sala", sala);
146.                         startActivity(intent);
147.                         finish();
148.                     }
149.                 });
150.             }
151.
152.             //metodo de consuta de webservice para carga de
    las salas en el combo, se llama al webservice GetSalas
153.             class CallWebService extends AsyncTask<Void, Strin
    g, ArrayList<Salas>>
154.             {
155.                 private String SOAP_ACTION;
156.                 private String OPERATION_NAME;
157.                 private final String WSDL_TARGET_NAMESPACE = "
    http://tempuri.org/";
158.                 //local
159.                 //public final String SOAP_ADDRESS =
    "http://10.0.2.2/WSChildrenApp.asmx";
160.                 //public final String SOAP_ADDRESS =
    "http://localhost:3030/WSChildrenApp.asmx";
161.                 //celular externo
```

```
162.             public final String SOAP_ADDRESS = "http://192
   .168.100.27/WS_recognitionface.asmx";
163.             private String[] parametros = null;
164.             private ProgressDialog
   mDialog = new ProgressDialog(GPSActivity.this);
165.
166.             public CallWebService(String[] parametros) {
167.                 this.parametros = parametros;
168.             }
169.
170.             @Override
171.             protected ArrayList<Salas> doInBackground(Void
   ... voids) {
172.
173.                 publishProgress("Procesando");
174.
175.                 SOAP_ACTION = "http://tempuri.org/GetSalas
   ";
176.                 OPERATION_NAME = "GetSalas";
177.
178.                 SoapObject
   request = new SoapObject(WSDL_TARGET_NAMESPACE,
   OPERATION_NAME);
179.
180.                 PropertyInfo pi = new PropertyInfo();
181.                 pi.setName("latitud");
182.                 pi.setValue(parametros[0]);
183.                 pi.setType(String.class);
184.                 request.addProperty(pi);
185.                 pi = new PropertyInfo();
186.                 pi.setName("longitud");
187.                 pi.setValue(parametros[1]);
188.                 pi.setType(String.class);
189.                 request.addProperty(pi);
190.
191.                 SoapSerializationEnvelope
   envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11)
   ;
192.                 envelope.dotNet = true;
193.
194.                 envelope.setOutputSoapObject(request);
195.
196.                 HttpTransportSE
   httpTransport = new HttpTransportSE(SOAP_ADDRESS);
197.                 SoapObject response = null;
198.
199.                 salas.clear();
200.
201.                 try {
202.                     httpTransport.call(SOAP_ACTION,
   envelope);
203.                     response = (SoapObject)envelope.getRes
   ponse();
204.
205.                     SoapObject
   obj1 = (SoapObject) response.getProperty(1);
206.
207.                     if(obj1.getPropertyCount() == 0)
208.                     {
209.                         sala = new Salas();
```

```
210.                            sala.setError("No existe datos
    para esta ubicacion");
211.                            salas.add(sala);
212.                            publishProgress("No existen datos
    a mostrar");
213.                            return salas;
214.                        }
215.
216.                        SoapObject
    obj2 = (SoapObject) obj1.getProperty(0);
217.
218.                        for(int i = 0; i < obj2.getPropertyCou
    nt(); i++)
219.                        {
220.                            SoapObject
    obj3 = (SoapObject) obj2.getProperty(i);
221.
222.                            sala = new Salas();
223.                            sala.setId_centro(obj3.getProperty
    ("Id_Centro").toString());
224.                            sala.setId_sala(obj3.getProperty("
    Id_Sala").toString());
225.                            sala.setNombre(obj3.getProperty("N
    ombre").toString());
226.                            salas.add(sala);
227.                            publishProgress("Finalizo
    Proceso");
228.                        }
229.
230.                        return salas;
231.                    }
232.                    catch (Exception e)
233.                    {
234.                        Log.e("Error WebService",
    e.toString());
235.                        publishProgress("Error de consulta
    WS");
236.                        return null;
237.                    }
238.                }
239.
240.                @Override
241.                protected void onPreExecute() {
242.                    mDialog.show();
243.                }
244.
245.                @Override
246.                protected void onProgressUpdate(String... valu
    es) {
247.                    mDialog.setMessage(values[0]);
248.                }
249.
250.                @Override
251.                protected void onPostExecute(ArrayList<Salas>
    salas) {
252.                    mDialog.dismiss();
253.                    if(salas != null)
254.                    {
255.                        if(salas.get(0).getError() != null)
256.                        {
```

```
257.                         Toast.makeText(getApplicationConte
    xt(),
258.                                 "Error WS
    " + salas.get(0).getError(),
259.                                 Toast.LENGTH_LONG).show();
260.                     }
261.                     else
262.                     {
263.                         ssalas = new String[salas.size()];
264.
265.                         cbn_salon.setAdapter(null);
266.
267.                         for(int i = 0; i < salas.size(); i
    ++)
268.                         {
269.                             ssalas[i] = salas.get(i).getNo
    mbre();
270.                         }
271.
272.                         //cuando obtiene los datos del
    WebService llena el elemento combo con las salas
273.                         combo = new ArrayAdapter<String>(G
    PSActivity.this, android.R.layout.simple_spinner_item,
    ssalas);
274.                         cbn_salon.setAdapter(combo);
275.                         btn_siguiente.setEnabled(true);
276.                         btn_localizacion.setEnabled(false)
    ;
277.                     }
278.                 }
279.                 else
280.                 {
281.                     Toast.makeText(getApplicationContext()
    ,
282.                                 "No existe datos para
    consultar, verificar con el administrador",
283.                                 Toast.LENGTH_LONG).show();
284.                     Log.e("Error", "Error consulta
    AsyncTask de salas");
285.                 }
286.             }
287.
288.
289.         }
290.
291.         //metodos de implementacion de la clase
    LocationListener pero no se usan
292.         @Override
293.         public void onLocationChanged(Location location) {
294.             Log.i(TAG,
    getString(R.string.on_location_changed));
295.             //updateUI(location);
296.         }
297.
298.         //metodos de implementacion de la clase
    LocationListener pero no se usan
299.         @Override
300.         public void onStatusChanged(String s, int i,
    Bundle bundle) {
301.
302.         }
```

```
303.
304.         //metodos de implementacion de la clase
   LocationListener que se llama cuando se enciende el gps
305.         @Override
306.         public void onProviderEnabled(String s) {
307.             getLocation();
308.         }
309.
310.         //metodos de implementacion de la clase
   LocationListener que se llama cuando se deshabilita el gps
311.         @Override
312.         public void onProviderDisabled(String s) {
313.             if (locationManager != null)
314.             {
315.                 locationManager.removeUpdates(this);
316.             }
317.         }
318.
319.         //metodo para obtener la geolocalizacion
320.         private void getLocation() {
321.             try
322.             {
323.                 if (canGetLocation)
324.                 {
325.                     Log.e(TAG,
   getString(R.string.err_localicazacion));
326.                     if (isGPS)
327.                     {
328.                         // from GPS
329.                         Log.i(TAG,
   getString(R.string.msg_gps_on));
330.                         locationManager.requestLocationUpd
   ates(
331.                                 LocationManager.GPS_PROVID
   ER,
332.                                 MIN_TIME_BW_UPDATES,
333.                                 MIN_DISTANCE_CHANGE_FOR_UP
   DATES, this);
334.
335.                         if (locationManager != null) {
336.                             loc = locationManager.getLastK
   nownLocation(LocationManager.GPS_PROVIDER);
337.                             if (loc != null)
338.                                 updateUI(loc);
339.                         }
340.                     } else if (isNetwork) {
341.                         // from Network Provider
342.                         Log.i(TAG,
   getString(R.string.msg_networkprovider_on));
343.                         locationManager.requestLocationUpd
   ates(
344.                                 LocationManager.NETWORK_PR
   OVIDER,
345.                                 MIN_TIME_BW_UPDATES,
346.                                 MIN_DISTANCE_CHANGE_FOR_UP
   DATES, this);
347.
348.                         if (locationManager != null) {
349.                             loc = locationManager.getLastK
   nownLocation(LocationManager.NETWORK_PROVIDER);
350.                             if (loc != null)
```

```
                                    updateUI(loc);
                                }
                            } else {
                                loc.setLatitude(0);
                                loc.setLongitude(0);
                                updateUI(loc);
                            }
                        } else {
                            Log.e(TAG,
   getString(R.string.err_localizacion));
                        }
                    } catch (SecurityException e) {
                        e.printStackTrace();
                        Log.e(TAG, e.getMessage());
                    }
                }

                //obtiene la ultima localizacion gps
                private void getLastLocation() {
                    try {
                        Criteria criteria = new Criteria();
                        String provider = locationManager.getBestP
   rovider(criteria, false);
                        Location
   location = locationManager.getLastKnownLocation(provider);
                        Log.i(TAG, provider);
                        Log.i(TAG,
   location == null ? getString(R.string.err_lastlocalizacion)
   : location.toString());
                    } catch (SecurityException e) {
                        e.printStackTrace();
                        Log.e(TAG, e.getMessage());
                    }
                }

                //busca si no se ha dado permisos
                private ArrayList<String> findUnAskedPermissions(A
   rrayList<String> wanted) {
                    ArrayList<String> result = new ArrayList<>();

                    for (String perm : wanted) {
                        if (!hasPermission(perm)) {
                            result.add(perm);
                        }
                    }

                    return result;
                }

                //verifica los permisos
                private boolean hasPermission(String permission) {
                    if (canAskPermission()) {
                        if (Build.VERSION.SDK_INT >= Build.VERSION
   _CODES.M) {
                            return (checkSelfPermission(permission
   ) == PackageManager.PERMISSION_GRANTED);
                        }
                    }
                    return true;
                }

```

```
404.         //verifica si se pregunto los permisos
405.         private boolean canAskPermission() {
406.             return (Build.VERSION.SDK_INT > Build.VERSION_
   CODES.LOLLIPOP_MR1);
407.         }
408.
409.         //si se dieron los permisos
410.         @TargetApi(Build.VERSION_CODES.M)
411.         @Override
412.         public void onRequestPermissionsResult(int request
   Code, String[] permissions, int[] grantResults) {
413.             switch (requestCode) {
414.                 case ALL_PERMISSIONS_RESULT:
415.                     Log.i(TAG,
   getString(R.string.msg_resultpermisos));
416.                     for (String perms : permissionsToReque
   st) {
417.                         if (!hasPermission(perms)) {
418.                             permissionsRejected.add(perms)
   ;
419.                         }
420.                     }
421.
422.                     if (permissionsRejected.size() > 0) {
423.                         if (Build.VERSION.SDK_INT >= Build
   .VERSION_CODES.M) {
424.                             if (shouldShowRequestPermissio
   nRationale(permissionsRejected.get(0))) {
425.                                 showMessageOKCancel(getStr
   ing(R.string.msg_okpermisos),
426.                                         new DialogInterfac
   e.OnClickListener() {
427.                                             @Override
428.                                             public void on
   Click(DialogInterface dialog, int which) {
429.                                                 if (Build.
   VERSION.SDK_INT >= Build.VERSION_CODES.M) {
430.                                                     reques
   tPermissions(permissionsRejected.toArray(
431.
    new String[permissionsRejected.size()]),
   ALL_PERMISSIONS_RESULT);
432.                                                 }
433.                                             }
434.                                         });
435.                                 return;
436.                             }
437.                         }
438.                     } else {
439.                         Log.i(TAG,
   getString(R.string.msg_permisosrechazados));
440.                         canGetLocation = true;
441.                         getLocation();
442.                     }
443.                     break;
444.             }
445.         }
446.
447.         //solicitud de encender el gps
448.         public void showSettingsAlert() {
```

```
449.             AlertDialog.Builder alertDialog = new AlertDia
   log.Builder(this);
450.             alertDialog.setTitle(getString(R.string.alert_
   titulo));
451.             alertDialog.setMessage(getString(R.string.aler
   t_mensaje));
452.             alertDialog.setPositiveButton(getString(R.stri
   ng.alert_si), new DialogInterface.OnClickListener() {
453.                 public void onClick(DialogInterface
   dialog, int which) {
454.                     Intent
   intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS
   );
455.                     startActivity(intent);
456.                 }
457.             });
458.
459.             alertDialog.setNegativeButton(getString(R.stri
   ng.alert_no), new DialogInterface.OnClickListener() {
460.                 public void onClick(DialogInterface
   dialog, int which) {
461.                     dialog.cancel();
462.                 }
463.             });
464.
465.             alertDialog.show();
466.         }
467.
468.         //muestra una venta si los permisos fueron dados
469.         private void showMessageOKCancel(String message,
   DialogInterface.OnClickListener okListener) {
470.             new AlertDialog.Builder(GPSActivity.this)
471.                     .setMessage(message)
472.                     .setPositiveButton("OK", okListener)
473.                     .setNegativeButton("Cancel", null)
474.                     .create()
475.                     .show();
476.         }
477.
478.         //setea en la pantalla la latitud y longitud
479.         private void updateUI(Location loc) {
480.             Log.i(TAG, "updateUI");
481.             longitud = loc.getLongitude();
482.             latitud = loc.getLatitude();
483.             /*Log.d("com.uteg.childrenapp",
   Double.toString(longitud));
484.             Log.d("com.uteg.childrenapp",
   Double.toString(latitud));*/
485.             tv_latitud.setText(String.format("%.6f",
   loc.getLatitude()));
486.             tv_longitud.setText(String.format("%.6f",
   loc.getLongitude()));
487.         }
488.
489.         //para remover update del localizador GPS
490.         @Override
491.         protected void onDestroy() {
492.             super.onDestroy();
493.             if (locationManager != null) {
494.                 locationManager.removeUpdates(this);
495.             }
```

```
496.          }
497.      }
```

Anexo 6

Fragmento de código Salas.java

```
1. package com.uteg.recognitionface;
2.
3. import java.io.Serializable;
4.
5. class Salas implements Serializable {
6.
7.     private String id_centro, id_sala, nombre,
   error = null;
8.
9.     public Salas()
10.          {
11.
12.          }
13.
14.          public String getId_centro()
15.          {
16.              return id_centro;
17.          }
18.
19.          public String getId_sala()
20.          {
21.              return id_sala;
22.          }
23.
24.          public String getNombre()
25.          {
26.              return nombre;
27.          }
28.
29.          public String getError()
30.          { return error; }
31.
32.          public void setId_centro(String id_centro)
33.          {
34.              this.id_centro = id_centro;
35.          }
36.
37.          public void setId_sala(String id_sala)
38.          {
39.              this.id_sala = id_sala;
40.          }
41.
42.          public void setNombre(String nombre)
43.          {
44.              this.nombre = nombre;
45.          }
46.
```

```
47.          public void setError(String error)
48.          {
49.              this.error = error;
50.          }
51.
52.          public String toString()
53.          {
54.              return nombre.trim();
55.          }
56.      }
```

Anexo 7

Fragmento de código activity_foto.xml

```
1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     tools:context=".FotoFragment">
6.
7.      <com.uteg.recognitionface.AutoFitTextureView
8.         android:id="@+id/texture"
9.         android:layout_width="wrap_content"
10.             android:layout_height="wrap_content"
11.             android:layout_alignParentStart="true"
12.             android:layout_alignParentTop="true" />
13.
14.          <FrameLayout
15.             android:id="@+id/control"
16.             android:layout_width="match_parent"
17.             android:layout_height="112dp"
18.             android:layout_alignParentBottom="true"
19.             android:layout_alignParentStart="true"
20.             android:background="@color/control_background">
21.
22.              <Button
23.                 android:id="@+id/picture"
24.                 android:layout_width="wrap_content"
25.                 android:layout_height="wrap_content"
26.                 android:layout_gravity="center"
27.                 android:text="@string/picture"
28.                 android:textAppearance="@android:style/TextAppearance.Large"
29.                 android:textStyle="bold" />
30.
31.              <ImageButton
32.                 android:id="@+id/info"
33.                 android:contentDescription="@string/description_info"
34.                 style="@android:style/Widget.Material.Light.Button.Borderless"
35.                 android:layout_width="wrap_content"
36.                 android:layout_height="wrap_content"
37.                 android:layout_gravity="center_vertical|right"
38.                 android:padding="20dp"
39.                 android:src="@drawable/ic_action_info"
40.                 tools:ignore="RtlHardcoded" />
```

```
41.
42.          </FrameLayout>
43.
44.      </RelativeLayout>
```

Anexo 8

Fragmento de código FotoActivity.java

```
1. package com.uteg.recognitionface;
2.
3. import android.content.Intent;
4. import android.support.v7.app.AppCompatActivity;
5. import android.os.Bundle;
6.
7. //clase principal para llamar el fragmento que tiene la
   camara
8. public class FotoActivity extends AppCompatActivity {
9.
10.          public static Salas sala;
11.
12.          @Override
13.          protected void onCreate(Bundle
   savedInstanceState) {
14.              super.onCreate(savedInstanceState);
15.              setContentView(R.layout.activity_foto);
16.
17.              Intent intent = getIntent();
18.              sala = (Salas) intent.getSerializableExtra("s
   ala");
19.
20.              if (null == savedInstanceState) {
21.                  getFragmentManager().beginTransaction()
22.                          .replace(R.id.container,
   FotoFragment.newInstance())
23.                          .commit();
24.              }
25.          }
26.      }
```

Anexo 9

Fragmento de código FotoFragment.java

```
1. package com.uteg.recognitionface;
2.
3. import android.Manifest;
4. import android.app.Activity;
5. import android.app.AlertDialog;
6. import android.app.Dialog;
7. import android.app.DialogFragment;
```

```
8. import android.content.Context;
9. import android.content.DialogInterface;
10.      import android.content.Intent;
11.      import android.content.pm.PackageManager;
12.      import android.content.res.Configuration;
13.      import android.graphics.ImageFormat;
14.      import android.graphics.Matrix;
15.      import android.graphics.Point;
16.      import android.graphics.RectF;
17.      import android.graphics.SurfaceTexture;
18.      import android.hardware.camera2.CameraAccessException;
19.      import android.hardware.camera2.CameraCaptureSession;
20.      import android.hardware.camera2.CameraCharacteristics;
21.      import android.hardware.camera2.CameraDevice;
22.      import android.hardware.camera2.CameraManager;
23.      import android.hardware.camera2.CameraMetadata;
24.      import android.hardware.camera2.CaptureRequest;
25.      import android.hardware.camera2.CaptureResult;
26.      import android.hardware.camera2.TotalCaptureResult;
27.      import android.hardware.camera2.params.StreamConfigura
   tionMap;
28.      import android.media.Image;
29.      import android.media.ImageReader;
30.      import android.os.Bundle;
31.      import android.os.Handler;
32.      import android.os.HandlerThread;
33.      import android.support.annotation.NonNull;
34.      import android.support.v13.app.FragmentCompat;
35.      import android.support.v4.content.ContextCompat;
36.      import android.util.Log;
37.      import android.util.Size;
38.      import android.util.SparseIntArray;
39.      import android.view.LayoutInflater;
40.      import android.view.Surface;
41.      import android.view.TextureView;
42.      import android.view.View;
43.      import android.view.ViewGroup;
44.      import android.widget.Toast;
45.
46.      import java.io.File;
47.      import java.io.FileOutputStream;
48.      import java.io.IOException;
49.      import java.nio.ByteBuffer;
50.      import java.util.ArrayList;
51.      import java.util.Arrays;
52.      import java.util.Collections;
53.      import java.util.Comparator;
54.      import java.util.List;
55.      import java.util.concurrent.Semaphore;
56.      import java.util.concurrent.TimeUnit;
57.
58.
59.      public class FotoFragment extends android.app.Fragment
60.              implements View.OnClickListener,
   FragmentCompat.OnRequestPermissionsResultCallback {
61.          // TODO: Rename parameter arguments, choose names
   that match
62.
63.          /**
64.           * Conversion from screen rotation to JPEG
   orientation.
```

```
65.          */
66.         private static final SparseIntArray
    ORIENTATIONS = new SparseIntArray();
67.         private static final int REQUEST_CAMERA_PERMISSION
    = 1;
68.         private static final String FRAGMENT_DIALOG = "dia
    log";
69.
70.         static {
71.             ORIENTATIONS.append(Surface.ROTATION_0, 90);
72.             ORIENTATIONS.append(Surface.ROTATION_90, 0);
73.             ORIENTATIONS.append(Surface.ROTATION_180, 270)
    ;
74.             ORIENTATIONS.append(Surface.ROTATION_270, 180)
    ;
75.         }
76.
77.         /**
78.          * Tag for the {@link Log}.
79.          */
80.         private static final String TAG = "FotoFragment";
81.
82.         /**
83.          * Camera state: Showing camera preview.
84.          */
85.         private static final int STATE_PREVIEW = 0;
86.
87.         /**
88.          * Camera state: Waiting for the focus to be
    locked.
89.          */
90.         private static final int STATE_WAITING_LOCK = 1;
91.
92.         /**
93.          * Camera state: Waiting for the exposure to be
    precapture state.
94.          */
95.         private static final int STATE_WAITING_PRECAPTURE
    = 2;
96.
97.         /**
98.          * Camera state: Waiting for the exposure state to
    be something other than precapture.
99.          */
100.        private static final int STATE_WAITING_NON_PRECAPT
    URE = 3;
101.
102.        /**
103.         * Camera state: Picture was taken.
104.         */
105.        private static final int STATE_PICTURE_TAKEN = 4;
106.
107.        /**
108.         * Max preview width that is guaranteed by Camera2
    API
109.         */
110.        private static final int MAX_PREVIEW_WIDTH = 1920;
111.
112.        /**
113.         * Max preview height that is guaranteed by
    Camera2 API
```

```
114.          */
115.         private static final int MAX_PREVIEW_HEIGHT = 1080
   ;
116.
117.         /**
118.          * {@link TextureView.SurfaceTextureListener}
   handles several lifecycle events on a
119.          * {@link TextureView}.
120.          */
121.         private final TextureView.SurfaceTextureListener m
   SurfaceTextureListener
122.                 = new TextureView.SurfaceTextureListener()
   {
123.
124.             @Override
125.             public void onSurfaceTextureAvailable(SurfaceT
   exture texture, int width, int height) {
126.                 openCamera(width, height);
127.             }
128.
129.             @Override
130.             public void onSurfaceTextureSizeChanged(Surfac
   eTexture texture, int width, int height) {
131.                 configureTransform(width, height);
132.             }
133.
134.             @Override
135.             public boolean onSurfaceTextureDestroyed(Surfa
   ceTexture texture) {
136.                 return true;
137.             }
138.
139.             @Override
140.             public void onSurfaceTextureUpdated(SurfaceTex
   ture texture) {
141.             }
142.
143.         };
144.
145.         /**
146.          * ID of the current {@link CameraDevice}.
147.          */
148.         private String mCameraId;
149.
150.         /**
151.          * An {@link AutoFitTextureView} for camera
   preview.
152.          */
153.         private AutoFitTextureView mTextureView;
154.
155.         /**
156.          * A {@link CameraCaptureSession } for camera
   preview.
157.          */
158.         private CameraCaptureSession mCaptureSession;
159.
160.         /**
161.          * A reference to the opened {@link CameraDevice}.
162.          */
163.         private CameraDevice mCameraDevice;
164.
```

```
165.         /**
166.          * The {@link android.util.Size} of camera
   preview.
167.          */
168.         private Size mPreviewSize;
169.
170.         /**
171.          * {@link CameraDevice.StateCallback} is called
   when {@link CameraDevice} changes its state.
172.          */
173.         private final CameraDevice.StateCallback mStateCal
   lback = new CameraDevice.StateCallback() {
174.
175.             @Override
176.             public void onOpened(@NonNull CameraDevice
   cameraDevice) {
177.                 // This method is called when the camera
   is opened.  We start camera preview here.
178.                 mCameraOpenCloseLock.release();
179.                 mCameraDevice = cameraDevice;
180.                 createCameraPreviewSession();
181.             }
182.
183.             @Override
184.             public void onDisconnected(@NonNull
   CameraDevice cameraDevice) {
185.                 mCameraOpenCloseLock.release();
186.                 cameraDevice.close();
187.                 mCameraDevice = null;
188.             }
189.
190.             @Override
191.             public void onError(@NonNull CameraDevice
   cameraDevice, int error) {
192.                 mCameraOpenCloseLock.release();
193.                 cameraDevice.close();
194.                 mCameraDevice = null;
195.                 Activity activity = getActivity();
196.                 if (null != activity) {
197.                     activity.finish();
198.                 }
199.             }
200.
201.         };
202.
203.         /**
204.          * An additional thread for running tasks that
   shouldn't block the UI.
205.          */
206.         private HandlerThread mBackgroundThread;
207.
208.         /**
209.          * A {@link Handler} for running tasks in the
   background.
210.          */
211.         private Handler mBackgroundHandler;
212.
213.         /**
214.          * An {@link ImageReader} that handles still image
   capture.
215.          */
```

```
216.         private ImageReader mImageReader;
217.
218.         /**
219.          * This is the output file for our picture.
220.          */
221.         private File mFile;
222.
223.         private Context context;
224.
225.         /**
226.          * This a callback object for the {@link
   ImageReader}. "onImageAvailable" will be called when a
227.          * still image is ready to be saved.
228.          */
229.         private final ImageReader.OnImageAvailableListener
   mOnImageAvailableListener
230.                 = new ImageReader.OnImageAvailableListener
   () {
231.
232.             @Override
233.             public void onImageAvailable(ImageReader
   reader) {
234.                 mBackgroundHandler.post(new ImageSaver(rea
   der.acquireNextImage(), mFile));
235.             }
236.
237.         };
238.
239.         /**
240.          * {@link CaptureRequest.Builder} for the camera
   preview
241.          */
242.         private CaptureRequest.Builder mPreviewRequestBuil
   der;
243.
244.         /**
245.          * {@link CaptureRequest} generated by {@link
   #mPreviewRequestBuilder}
246.          */
247.         private CaptureRequest mPreviewRequest;
248.
249.         /**
250.          * The current state of camera state for taking
   pictures.
251.          *
252.          * @see #mCaptureCallback
253.          */
254.         private int mState = STATE_PREVIEW;
255.
256.         /**
257.          * A {@link Semaphore} to prevent the app from
   exiting before closing the camera.
258.          */
259.         private Semaphore
   mCameraOpenCloseLock = new Semaphore(1);
260.
261.         /**
262.          * Whether the current camera device supports
   Flash or not.
263.          */
264.         private boolean mFlashSupported;
```

```
265.
266.         /**
267.          * Orientation of the camera sensor
268.          */
269.         private int mSensorOrientation;
270.
271.         /**
272.          * A {@link CameraCaptureSession.CaptureCallback}
   that handles events related to JPEG capture.
273.          */
274.         private CameraCaptureSession.CaptureCallback mCapt
   ureCallback
275.                 = new CameraCaptureSession.CaptureCallback
   () {
276.
277.             private void process(CaptureResult result) {
278.                 switch (mState) {
279.                     case STATE_PREVIEW: {
280.                         // We have nothing to do when the
   camera preview is working normally.
281.                         break;
282.                     }
283.                     case STATE_WAITING_LOCK: {
284.                         Integer afState = result.get(Captu
   reResult.CONTROL_AF_STATE);
285.                         if (afState == null) {
286.                             captureStillPicture();
287.                         } else if (CaptureResult.CONTROL_A
   F_STATE_FOCUSED_LOCKED == afState ||
288.                                 CaptureResult.CONTROL_AF_S
   TATE_NOT_FOCUSED_LOCKED == afState) {
289.                             // CONTROL_AE_STATE can be
   null on some devices
290.                             Integer aeState = result.get(C
   aptureResult.CONTROL_AE_STATE);
291.                             if (aeState == null ||
292.                                     aeState == CaptureResu
   lt.CONTROL_AE_STATE_CONVERGED) {
293.                                 mState = STATE_PICTURE_TAK
   EN;
294.                                 captureStillPicture();
295.                             } else {
296.                                 runPrecaptureSequence();
297.                             }
298.                         }
299.                         break;
300.                     }
301.                     case STATE_WAITING_PRECAPTURE: {
302.                         // CONTROL_AE_STATE can be null on
   some devices
303.                         Integer aeState = result.get(Captu
   reResult.CONTROL_AE_STATE);
304.                         if (aeState == null ||
305.                                 aeState == CaptureResult.C
   ONTROL_AE_STATE_PRECAPTURE ||
306.                                 aeState == CaptureRequest.
   CONTROL_AE_STATE_FLASH_REQUIRED) {
307.                             mState = STATE_WAITING_NON_PRE
   CAPTURE;
308.                         }
309.                         break;
```

```
                    }
                    case STATE_WAITING_NON_PRECAPTURE: {
                        // CONTROL_AE_STATE can be null on some devices
                        Integer aeState = result.get(CaptureResult.CONTROL_AE_STATE);
                        if (aeState == null || aeState != CaptureResult.CONTROL_AE_STATE_PRECAPTURE) {
                            mState = STATE_PICTURE_TAKEN;
                            captureStillPicture();
                        }
                        break;
                    }
                }
            }

            @Override
            public void onCaptureProgressed(@NonNull CameraCaptureSession session,
                                            @NonNull CaptureRequest request,
                                            @NonNull CaptureResult partialResult) {
                process(partialResult);
            }

            @Override
            public void onCaptureCompleted(@NonNull CameraCaptureSession session,
                                           @NonNull CaptureRequest request,
                                           @NonNull TotalCaptureResult result) {
                process(result);
            }

        };

        /**
         * Shows a {@link Toast} on the UI thread.
         *
         * @param text The message to show
         */
        private void showToast(final String text) {
            final Activity activity = getActivity();
            if (activity != null) {
                activity.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(activity, text, Toast.LENGTH_SHORT).show();
                    }
                });
            }
        }

        /**
         * Given {@code choices} of {@code Size}s supported by a camera, choose the smallest one that
         * is at least as large as the respective texture view size, and that is at most as large as the
```

```
359.          * respective max size, and whose aspect ratio
   matches with the specified value. If such size
360.          * doesn't exist, choose the largest one that is
   at most as large as the respective max size,
361.          * and whose aspect ratio matches with the
   specified value.
362.          *
363.          * @param choices           The list of sizes that
   the camera supports for the intended output
364.          *                          class
365.          * @param textureViewWidth  The width of the
   texture view relative to sensor coordinate
366.          * @param textureViewHeight The height of the
   texture view relative to sensor coordinate
367.          * @param maxWidth          The maximum width that
   can be chosen
368.          * @param maxHeight         The maximum height
   that can be chosen
369.          * @param aspectRatio       The aspect ratio
370.          * @return The optimal {@code Size}, or an
   arbitrary one if none were big enough
371.          */
372.         private static Size
   chooseOptimalSize(Size[] choices, int textureViewWidth,
373.                                                 int textureV
   iewHeight, int maxWidth, int maxHeight, Size aspectRatio) {
374.
375.             // Collect the supported resolutions that are
   at least as big as the preview Surface
376.             List<Size> bigEnough = new ArrayList<>();
377.             // Collect the supported resolutions that are
   smaller than the preview Surface
378.             List<Size> notBigEnough = new ArrayList<>();
379.             int w = aspectRatio.getWidth();
380.             int h = aspectRatio.getHeight();
381.             for (Size option : choices) {
382.                 if (option.getWidth() <= maxWidth && optio
   n.getHeight() <= maxHeight &&
383.                         option.getHeight() == option.getWi
   dth() * h / w) {
384.                     if (option.getWidth() >= textureViewWi
   dth &&
385.                             option.getHeight() >= textureV
   iewHeight) {
386.                         bigEnough.add(option);
387.                     } else {
388.                         notBigEnough.add(option);
389.                     }
390.                 }
391.             }
392.
393.             // Pick the smallest of those big enough. If
   there is no one big enough, pick the
394.             // largest of those not big enough.
395.             if (bigEnough.size() > 0) {
396.                 return Collections.min(bigEnough, new Comp
   areSizesByArea());
397.             } else if (notBigEnough.size() > 0) {
398.                 return Collections.max(notBigEnough, new C
   ompareSizesByArea());
399.             } else {
```

```
400.                 Log.e(TAG, "Couldn't find any suitable
   preview size");
401.                 return choices[0];
402.             }
403.         }
404.
405.         public static FotoFragment newInstance() {
406.             return new FotoFragment();
407.         }
408.
409.         @Override
410.         public View onCreateView(LayoutInflater inflater,
   ViewGroup container,
411.                                  Bundle
   savedInstanceState) {
412.             return inflater.inflate(R.layout.fragment_foto
   , container, false);
413.         }
414.
415.         @Override
416.         public void onViewCreated(final View view, Bundle
   savedInstanceState) {
417.             view.findViewById(R.id.picture).setOnClickList
   ener(this);
418.             view.findViewById(R.id.info).setOnClickListene
   r(this);
419.             mTextureView = (AutoFitTextureView) view.findV
   iewById(R.id.texture);
420.         }
421.
422.         @Override
423.         public void onActivityCreated(Bundle
   savedInstanceState) {
424.             super.onActivityCreated(savedInstanceState);
425.             mFile = new File(getActivity().getExternalFile
   sDir(null), "pic.jpg");
426.         }
427.
428.         @Override
429.         public void onResume() {
430.             super.onResume();
431.             startBackgroundThread();
432.
433.             // When the screen is turned off and turned
   back on, the SurfaceTexture is already
434.             // available, and "onSurfaceTextureAvailable"
   will not be called. In that case, we can open
435.             // a camera and start preview from here
   (otherwise, we wait until the surface is ready in
436.             // the SurfaceTextureListener).
437.             if (mTextureView.isAvailable()) {
438.                 openCamera(mTextureView.getWidth(),
   mTextureView.getHeight());
439.             } else {
440.                 mTextureView.setSurfaceTextureListener(mSu
   rfaceTextureListener);
441.             }
442.         }
443.
444.         @Override
445.         public void onPause() {
```

```
446.             closeCamera();
447.             stopBackgroundThread();
448.             super.onPause();
449.         }
450.
451.         private void requestCameraPermission() {
452.             if (FragmentCompat.shouldShowRequestPermission
    Rationale(this, Manifest.permission.CAMERA)) {
453.                 new ConfirmationDialog().show(getChildFrag
    mentManager(), FRAGMENT_DIALOG);
454.             } else {
455.                 FragmentCompat.requestPermissions(this, ne
    w String[]{Manifest.permission.CAMERA},
456.                         REQUEST_CAMERA_PERMISSION);
457.             }
458.         }
459.
460.         @Override
461.         public void onRequestPermissionsResult(int request
    Code, @NonNull String[] permissions,
462.                                                @NonNull in
    t[] grantResults) {
463.             if (requestCode == REQUEST_CAMERA_PERMISSION)
    {
464.                 if (grantResults.length != 1 || grantResul
    ts[0] != PackageManager.PERMISSION_GRANTED) {
465.                     ErrorDialog.newInstance(getString(R.st
    ring.request_permission))
466.                             .show(getChildFragmentManager(
    ), FRAGMENT_DIALOG);
467.                 }
468.             } else {
469.                 super.onRequestPermissionsResult(requestCo
    de, permissions, grantResults);
470.             }
471.         }
472.
473.         /**
474.          * Sets up member variables related to camera.
475.          *
476.          * @param width  The width of available size for
    camera preview
477.          * @param height The height of available size for
    camera preview
478.          */
479.         private void setUpCameraOutputs(int width, int hei
    ght) {
480.             Activity activity = getActivity();
481.             CameraManager
    manager = (CameraManager) activity.getSystemService(Context.
    CAMERA_SERVICE);
482.             try {
483.                 for (String cameraId : manager.getCameraId
    List()) {
484.                     CameraCharacteristics characteristics
485.                             = manager.getCameraCharacteris
    tics(cameraId);
486.
487.                     // We don't use a front facing camera
    in this sample.
```

```
488.                     Integer facing = characteristics.get(C
    ameraCharacteristics.LENS_FACING);
489.                     if (facing != null && facing == Camera
    Characteristics.LENS_FACING_FRONT) {
490.                         continue;
491.                     }
492.
493.                     StreamConfigurationMap
    map = characteristics.get(
494.                             CameraCharacteristics.SCALER_S
    TREAM_CONFIGURATION_MAP);
495.                     if (map == null) {
496.                         continue;
497.                     }
498.
499.                     // For still image captures, we use
    the largest available size.
500.                     Size largest = Collections.max(
501.                             Arrays.asList(map.getOutputSiz
    es(ImageFormat.JPEG)),
502.                             new CompareSizesByArea());
503.                     mImageReader = ImageReader.newInstance
    (largest.getWidth(), largest.getHeight(),
504.                             ImageFormat.JPEG, /*maxImages*
    /2);
505.                     mImageReader.setOnImageAvailableListen
    er(
506.                             mOnImageAvailableListener,
    mBackgroundHandler);
507.
508.                     // Find out if we need to swap
    dimension to get the preview size relative to sensor
509.                     // coordinate.
510.                     int displayRotation = activity.getWind
    owManager().getDefaultDisplay().getRotation();
511.                     //noinspection ConstantConditions
512.                     mSensorOrientation = characteristics.g
    et(CameraCharacteristics.SENSOR_ORIENTATION);
513.                     boolean swappedDimensions = false;
514.                     switch (displayRotation) {
515.                         case Surface.ROTATION_0:
516.                         case Surface.ROTATION_180:
517.                             if (mSensorOrientation == 90 |
    | mSensorOrientation == 270) {
518.                                 swappedDimensions = true;
519.                             }
520.                             break;
521.                         case Surface.ROTATION_90:
522.                         case Surface.ROTATION_270:
523.                             if (mSensorOrientation == 0 ||
    mSensorOrientation == 180) {
524.                                 swappedDimensions = true;
525.                             }
526.                             break;
527.                         default:
528.                             Log.e(TAG, "Display rotation
    is invalid: " + displayRotation);
529.                     }
530.
531.                     Point displaySize = new Point();
```

```
532.                     activity.getWindowManager().getDefault
    Display().getSize(displaySize);
533.                     int rotatedPreviewWidth = width;
534.                     int rotatedPreviewHeight = height;
535.                     int maxPreviewWidth = displaySize.x;
536.                     int maxPreviewHeight = displaySize.y;
537.
538.                     if (swappedDimensions) {
539.                         rotatedPreviewWidth = height;
540.                         rotatedPreviewHeight = width;
541.                         maxPreviewWidth = displaySize.y;
542.                         maxPreviewHeight = displaySize.x;
543.                     }
544.
545.                     if (maxPreviewWidth > MAX_PREVIEW_WIDT
    H) {
546.                         maxPreviewWidth = MAX_PREVIEW_WIDT
    H;
547.                     }
548.
549.                     if (maxPreviewHeight > MAX_PREVIEW_HEI
    GHT) {
550.                         maxPreviewHeight = MAX_PREVIEW_HEI
    GHT;
551.                     }
552.
553.                     // Danger, W.R.! Attempting to use too
    large a preview size could  exceed the camera
554.                     // bus' bandwidth limitation,
    resulting in gorgeous previews but the storage of
555.                     // garbage capture data.
556.                     mPreviewSize = chooseOptimalSize(map.g
    etOutputSizes(SurfaceTexture.class),
557.                             rotatedPreviewWidth,
    rotatedPreviewHeight, maxPreviewWidth,
558.                             maxPreviewHeight, largest);
559.
560.                     // We fit the aspect ratio of
    TextureView to the size of preview we picked.
561.                     int orientation = getResources().getCo
    nfiguration().orientation;
562.                     if (orientation == Configuration.ORIEN
    TATION_LANDSCAPE) {
563.                         mTextureView.setAspectRatio(
564.                                 mPreviewSize.getWidth(),
    mPreviewSize.getHeight());
565.                     } else {
566.                         mTextureView.setAspectRatio(
567.                                 mPreviewSize.getHeight(),
    mPreviewSize.getWidth());
568.                     }
569.
570.                     // Check if the flash is supported.
571.                     Boolean available = characteristics.ge
    t(CameraCharacteristics.FLASH_INFO_AVAILABLE);
572.                     mFlashSupported = available == null ?
    false : available;
573.
574.                     mCameraId = cameraId;
575.                     return;
576.                 }
```

```
577.             } catch (CameraAccessException e) {
578.                 e.printStackTrace();
579.             } catch (NullPointerException e) {
580.                 // Currently an NPE is thrown when the
    Camera2API is used but not supported on the
581.                 // device this code runs.
582.                 ErrorDialog.newInstance(getString(R.string
    .camera_error))
583.                         .show(getChildFragmentManager(),
    FRAGMENT_DIALOG);
584.             }
585.         }
586.
587.
588.         private void openCamera(int width, int height) {
589.             if (ContextCompat.checkSelfPermission(getActiv
    ity(), Manifest.permission.CAMERA)
590.                     != PackageManager.PERMISSION_GRANTED)
    {
591.                 requestCameraPermission();
592.                 return;
593.             }
594.             setUpCameraOutputs(width, height);
595.             configureTransform(width, height);
596.             Activity activity = getActivity();
597.             CameraManager
    manager = (CameraManager) activity.getSystemService(Context.
    CAMERA_SERVICE);
598.             try {
599.                 if (!mCameraOpenCloseLock.tryAcquire(2500,
    TimeUnit.MILLISECONDS)) {
600.                     throw new RuntimeException("Time out
    waiting to lock camera opening.");
601.                 }
602.                 manager.openCamera(mCameraId,
    mStateCallback, mBackgroundHandler);
603.             } catch (CameraAccessException e) {
604.                 e.printStackTrace();
605.             } catch (InterruptedException e) {
606.                 throw new RuntimeException("Interrupted
    while trying to lock camera opening.", e);
607.             }
608.         }
609.
610.         /**
611.          * Closes the current {@link CameraDevice}.
612.          */
613.         private void closeCamera() {
614.             try {
615.                 mCameraOpenCloseLock.acquire();
616.                 if (null != mCaptureSession) {
617.                     mCaptureSession.close();
618.                     mCaptureSession = null;
619.                 }
620.                 if (null != mCameraDevice) {
621.                     mCameraDevice.close();
622.                     mCameraDevice = null;
623.                 }
624.                 if (null != mImageReader) {
625.                     mImageReader.close();
626.                     mImageReader = null;
```

```
627.                 }
628.             } catch (InterruptedException e) {
629.                 throw new RuntimeException("Interrupted
   while trying to lock camera closing.", e);
630.             } finally {
631.                 mCameraOpenCloseLock.release();
632.             }
633.         }
634.
635.         /**
636.          * Starts a background thread and its {@link
   Handler}.
637.          */
638.         private void startBackgroundThread() {
639.             mBackgroundThread = new HandlerThread("CameraB
   ackground");
640.             mBackgroundThread.start();
641.             mBackgroundHandler = new Handler(mBackgroundTh
   read.getLooper());
642.         }
643.
644.         /**
645.          * Stops the background thread and its {@link
   Handler}.
646.          */
647.         private void stopBackgroundThread() {
648.             mBackgroundThread.quitSafely();
649.             try {
650.                 mBackgroundThread.join();
651.                 mBackgroundThread = null;
652.                 mBackgroundHandler = null;
653.             } catch (InterruptedException e) {
654.                 e.printStackTrace();
655.             }
656.         }
657.
658.         /**
659.          * Creates a new {@link CameraCaptureSession} for
   camera preview.
660.          */
661.         private void createCameraPreviewSession() {
662.             try {
663.                 SurfaceTexture
   texture = mTextureView.getSurfaceTexture();
664.                 assert texture != null;
665.
666.                 // We configure the size of default buffer
   to be the size of camera preview we want.
667.                 texture.setDefaultBufferSize(mPreviewSize.
   getWidth(), mPreviewSize.getHeight());
668.
669.                 // This is the output Surface we need to
   start preview.
670.                 Surface surface = new Surface(texture);
671.
672.                 // We set up a CaptureRequest.Builder with
   the output Surface.
673.                 mPreviewRequestBuilder
674.                         = mCameraDevice.createCaptureReque
   st(CameraDevice.TEMPLATE_PREVIEW);
675.                 mPreviewRequestBuilder.addTarget(surface);
```

```
676.
677.                 // Here, we create a CameraCaptureSession
    for camera preview.
678.                 mCameraDevice.createCaptureSession(Arrays.
    asList(surface, mImageReader.getSurface()),
679.                         new CameraCaptureSession.StateCall
    back() {
680.
681.                             @Override
682.                             public void onConfigured(@NonN
    ull CameraCaptureSession cameraCaptureSession) {
683.                                 // The camera is already
    closed
684.                                 if (null == mCameraDevice)
    {
685.                                     return;
686.                                 }
687.
688.                                 // When the session is
    ready, we start displaying the preview.
689.                                 mCaptureSession = cameraCa
    ptureSession;
690.                                 try {
691.                                     // Auto focus should
    be continuous for camera preview.
692.                                     mPreviewRequestBuilder
    .set(CaptureRequest.CONTROL_AF_MODE,
693.                                             CaptureRequest
    .CONTROL_AF_MODE_CONTINUOUS_PICTURE);
694.                                     // Flash is
    automatically enabled when necessary.
695.                                     setAutoFlash(mPreviewR
    equestBuilder);
696.
697.                                     // Finally, we start
    displaying the camera preview.
698.                                     mPreviewRequest = mPre
    viewRequestBuilder.build();
699.                                     mCaptureSession.setRep
    eatingRequest(mPreviewRequest,
700.                                             mCaptureCallba
    ck, mBackgroundHandler);
701.                                 } catch (CameraAccessExcep
    tion e) {
702.                                     e.printStackTrace();
703.                                 }
704.                             }
705.
706.                             @Override
707.                             public void onConfigureFailed(
708.                                     @NonNull
    CameraCaptureSession cameraCaptureSession) {
709.                                 showToast("Failed");
710.                             }
711.                         }, null
712.                 );
713.             } catch (CameraAccessException e) {
714.                 e.printStackTrace();
715.             }
716.         }
717.
```

```
718.         /**
719.          * Configures the necessary {@link
   android.graphics.Matrix} transformation to `mTextureView`.
720.          * This method should be called after the camera
   preview size is determined in
721.          * setUpCameraOutputs and also the size of
   `mTextureView` is fixed.
722.          *
723.          * @param viewWidth  The width of `mTextureView`
724.          * @param viewHeight The height of `mTextureView`
725.          */
726.         private void configureTransform(int viewWidth, int
   viewHeight) {
727.             Activity activity = getActivity();
728.             if (null == mTextureView || null == mPreviewSi
   ze || null == activity) {
729.                 return;
730.             }
731.             int rotation = activity.getWindowManager().get
   DefaultDisplay().getRotation();
732.             Matrix matrix = new Matrix();
733.             RectF viewRect = new RectF(0, 0, viewWidth,
   viewHeight);
734.             RectF bufferRect = new RectF(0, 0,
   mPreviewSize.getHeight(), mPreviewSize.getWidth());
735.             float centerX = viewRect.centerX();
736.             float centerY = viewRect.centerY();
737.             if (Surface.ROTATION_90 == rotation || Surface
   .ROTATION_270 == rotation) {
738.                 bufferRect.offset(centerX - bufferRect.cen
   terX(), centerY - bufferRect.centerY());
739.                 matrix.setRectToRect(viewRect, bufferRect,
   Matrix.ScaleToFit.FILL);
740.                 float scale = Math.max(
741.                         (float) viewHeight / mPreviewSize.
   getHeight(),
742.                         (float) viewWidth / mPreviewSize.g
   etWidth());
743.                 matrix.postScale(scale, scale, centerX,
   centerY);
744.                 matrix.postRotate(90 * (rotation - 2),
   centerX, centerY);
745.             } else if (Surface.ROTATION_180 == rotation) {
746.                 matrix.postRotate(180, centerX, centerY);
747.             }
748.             mTextureView.setTransform(matrix);
749.         }
750.
751.         /**
752.          * Initiate a still image capture.
753.          */
754.         private void takePicture() {
755.             lockFocus();
756.         }
757.
758.         /**
759.          * Lock the focus as the first step for a still
   image capture.
760.          */
761.         private void lockFocus() {
762.             try {
```

```
763.                 // This is how to tell the camera to lock
   focus.
764.                 mPreviewRequestBuilder.set(CaptureRequest.
   CONTROL_AF_TRIGGER,
765.                         CameraMetadata.CONTROL_AF_TRIGGER_
   START);
766.                 // Tell #mCaptureCallback to wait for the
   lock.
767.                 mState = STATE_WAITING_LOCK;
768.                 mCaptureSession.capture(mPreviewRequestBui
   lder.build(), mCaptureCallback,
769.                         mBackgroundHandler);
770.             } catch (CameraAccessException e) {
771.                 e.printStackTrace();
772.             }
773.         }
774.
775.         /**
776.          * Run the precapture sequence for capturing a
   still image. This method should be called when
777.          * we get a response in {@link #mCaptureCallback}
   from {@link #lockFocus()}.
778.          */
779.         private void runPrecaptureSequence() {
780.             try {
781.                 // This is how to tell the camera to
   trigger.
782.                 mPreviewRequestBuilder.set(CaptureRequest.
   CONTROL_AE_PRECAPTURE_TRIGGER,
783.                         CaptureRequest.CONTROL_AE_PRECAPTU
   RE_TRIGGER_START);
784.                 // Tell #mCaptureCallback to wait for the
   precapture sequence to be set.
785.                 mState = STATE_WAITING_PRECAPTURE;
786.                 mCaptureSession.capture(mPreviewRequestBui
   lder.build(), mCaptureCallback,
787.                         mBackgroundHandler);
788.             } catch (CameraAccessException e) {
789.                 e.printStackTrace();
790.             }
791.         }
792.
793.         /**
794.          * Capture a still picture. This method should be
   called when we get a response in
795.          * {@link #mCaptureCallback} from both {@link
   #lockFocus()}.
796.          */
797.         private void captureStillPicture() {
798.             try {
799.                 final Activity activity = getActivity();
800.                 if (null == activity || null == mCameraDev
   ice) {
801.                     return;
802.                 }
803.                 // This is the CaptureRequest.Builder that
   we use to take a picture.
804.                 final CaptureRequest.Builder captureBuilde
   r =
805.                         mCameraDevice.createCaptureRequest
   (CameraDevice.TEMPLATE_STILL_CAPTURE);
```

```
806.                 captureBuilder.addTarget(mImageReader.getS
   urface());
807.
808.                 // Use the same AE and AF modes as the
   preview.
809.                 captureBuilder.set(CaptureRequest.CONTROL_
   AF_MODE,
810.                         CaptureRequest.CONTROL_AF_MODE_CON
   TINUOUS_PICTURE);
811.                 setAutoFlash(captureBuilder);
812.
813.                 // Orientation
814.                 int rotation = activity.getWindowManager()
   .getDefaultDisplay().getRotation();
815.                 captureBuilder.set(CaptureRequest.JPEG_ORI
   ENTATION, getOrientation(rotation));
816.
817.                 CameraCaptureSession.CaptureCallback Captu
   reCallback
818.                         = new CameraCaptureSession.Capture
   Callback() {
819.
820.                     @Override
821.                     public void onCaptureCompleted(@NonNul
   l CameraCaptureSession session,
822.                                                    @NonNul
   l CaptureRequest request,
823.                                                    @NonNul
   l TotalCaptureResult result) {
824.                         showToast("Saved: " + mFile);
825.                         Log.d(TAG, mFile.toString());
826.                         unlockFocus();
827.                     }
828.                 };
829.
830.                 mCaptureSession.stopRepeating();
831.                 mCaptureSession.capture(captureBuilder.bui
   ld(), CaptureCallback, null);
832.
833.                 /*Se aumento este fragmento de codigo para
   que despues de que tome la foto pase a la siguiente pantalla
   y que tambien envie como dato extra
834.                 * la foto para ser procesada*/
835.                 Intent
   intent = new Intent(FotoFragment.this.getContext(),
   ProcesoActivity.class);
836.                 intent.putExtra("sala",
   FotoActivity.sala);
837.                 intent.putExtra("img", mFile);
838.
839.                 startActivity(intent);
840.                 activity.finish();
841.                 //hasta aqui finaliza el codigo de
   implementacion de llamada a la siguiente pantallla
842.
843.             } catch (CameraAccessException e) {
844.                 e.printStackTrace();
845.             }
846.         }
847.
848.         /**
```

```
849.          * Retrieves the JPEG orientation from the
   specified screen rotation.
850.          *
851.          * @param rotation The screen rotation.
852.          * @return The JPEG orientation (one of 0, 90,
   270, and 360)
853.          */
854.         private int getOrientation(int rotation) {
855.             // Sensor orientation is 90 for most devices,
   or 270 for some devices (eg. Nexus 5X)
856.             // We have to take that into account and
   rotate JPEG properly.
857.             // For devices with orientation of 90, we
   simply return our mapping from ORIENTATIONS.
858.             // For devices with orientation of 270, we
   need to rotate the JPEG 180 degrees.
859.             return (ORIENTATIONS.get(rotation) + mSensorOr
   ientation + 270) % 360;
860.         }
861.
862.         /**
863.          * Unlock the focus. This method should be called
   when still image capture sequence is
864.          * finished.
865.          */
866.         private void unlockFocus() {
867.             try {
868.                 // Reset the auto-focus trigger
869.                 mPreviewRequestBuilder.set(CaptureRequest.
   CONTROL_AF_TRIGGER,
870.                         CameraMetadata.CONTROL_AF_TRIGGER_
   CANCEL);
871.                 setAutoFlash(mPreviewRequestBuilder);
872.                 mCaptureSession.capture(mPreviewRequestBui
   lder.build(), mCaptureCallback,
873.                         mBackgroundHandler);
874.                 // After this, the camera will go back to
   the normal state of preview.
875.                 mState = STATE_PREVIEW;
876.                 mCaptureSession.setRepeatingRequest(mPrevi
   ewRequest, mCaptureCallback,
877.                         mBackgroundHandler);
878.             } catch (CameraAccessException e) {
879.                 e.printStackTrace();
880.             }
881.         }
882.
883.         @Override
884.         public void onClick(View view) {
885.             switch (view.getId()) {
886.                 case R.id.picture: {
887.                     takePicture();
888.                     break;
889.                 }
890.                 case R.id.info: {
891.                     Activity activity = getActivity();
892.                     if (null != activity) {
893.                         new AlertDialog.Builder(activity)
894.                                 .setMessage(R.string.intro
   _message)
```

```
                                    .setPositiveButton(android.R.string.ok, null)
                                    .show();
                        }
                        break;
                    }
                }
            }

            private void setAutoFlash(CaptureRequest.Builder requestBuilder) {
                if (mFlashSupported) {
                    requestBuilder.set(CaptureRequest.CONTROL_AE_MODE,
                            CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);
                }
            }

            /**
             * Saves a JPEG {@link Image} into the specified {@link File}.
             */
            private static class ImageSaver implements Runnable {

                /**
                 * The JPEG image
                 */
                private final Image mImage;
                /**
                 * The file we save the image into.
                 */
                private final File mFile;

                public ImageSaver(Image image, File file) {
                    mImage = image;
                    mFile = file;
                }

                @Override
                public void run() {
                    ByteBuffer buffer = mImage.getPlanes()[0].getBuffer();
                    byte[] bytes = new byte[buffer.remaining()];
                    buffer.get(bytes);
                    FileOutputStream output = null;
                    try {
                        output = new FileOutputStream(mFile);
                        output.write(bytes);
                    } catch (IOException e) {
                        e.printStackTrace();
                    } finally {
                        mImage.close();
                        if (null != output) {
                            try {
                                output.close();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
```

```
948.                     }
949.                 }
950.             }
951.
952.         }
953.
954.         /**
955.          * Compares two {@code Size}s based on their
   areas.
956.          */
957.         static class CompareSizesByArea implements Compara
   tor<Size> {
958.
959.             @Override
960.             public int compare(Size lhs, Size rhs) {
961.                 // We cast here to ensure the
   multiplications won't overflow
962.                 return Long.signum((long) lhs.getWidth() *
   lhs.getHeight() -
963.                         (long) rhs.getWidth() * rhs.getHei
   ght());
964.             }
965.
966.         }
967.
968.         /**
969.          * Shows an error message dialog.
970.          */
971.         public static class ErrorDialog extends DialogFrag
   ment {
972.
973.             private static final String ARG_MESSAGE = "mes
   sage";
974.
975.             public static ErrorDialog
   newInstance(String message) {
976.                 ErrorDialog dialog = new ErrorDialog();
977.                 Bundle args = new Bundle();
978.                 args.putString(ARG_MESSAGE, message);
979.                 dialog.setArguments(args);
980.                 return dialog;
981.             }
982.
983.             @Override
984.             public Dialog onCreateDialog(Bundle
   savedInstanceState) {
985.                 final Activity activity = getActivity();
986.                 return new AlertDialog.Builder(activity)
987.                         .setMessage(getArguments().getStri
   ng(ARG_MESSAGE))
988.                         .setPositiveButton(android.R.strin
   g.ok, new DialogInterface.OnClickListener() {
989.                             @Override
990.                             public void onClick(DialogInte
   rface dialogInterface, int i) {
991.                                 activity.finish();
992.                             }
993.                         })
994.                         .create();
995.             }
996.
```

```
997.         }
998.
999.         /**
1000.         * Shows OK/Cancel confirmation dialog about
   camera permission.
1001.         */
1002.        public static class ConfirmationDialog extends Dia
   logFragment {
1003.
1004.            @Override
1005.            public Dialog onCreateDialog(Bundle
   savedInstanceState) {
1006.                final android.app.Fragment parent = getPar
   entFragment();
1007.                return new AlertDialog.Builder(getActivity
   ())
1008.                        .setMessage(R.string.request_permi
   ssion)
1009.                        .setPositiveButton(android.R.strin
   g.ok, new DialogInterface.OnClickListener() {
1010.                            @Override
1011.                            public void onClick(DialogInte
   rface dialog, int which) {
1012.                                FragmentCompat.requestPerm
   issions(parent,
1013.                                        new String[]{Manif
   est.permission.CAMERA},
1014.                                        REQUEST_CAMERA_PER
   MISSION);
1015.                            }
1016.                        })
1017.                        .setNegativeButton(android.R.strin
   g.cancel,
1018.                                new DialogInterface.OnClic
   kListener() {
1019.                                    @Override
1020.                                    public void onClick(Di
   alogInterface dialog, int which) {
1021.                                        Activity
   activity = parent.getActivity();
1022.                                        if (activity != nu
   ll) {
1023.                                            activity.finis
   h();
1024.                                        }
1025.                                    }
1026.                                })
1027.                        .create();
1028.            }
1029.        }
1030.
1031.    }
```

Anexo 10

Fragmento de código AutoFitTextureView.java

```
1. package com.uteg.recognitionface;
2.
3. import android.content.Context;
4. import android.util.AttributeSet;
5. import android.view.TextureView;
6.
7. class AutoFitTextureView extends TextureView {
8.
9.     private int mRatioWidth = 0;
10.          private int mRatioHeight = 0;
11.
12.          public AutoFitTextureView(Context context) {
13.              this(context, null);
14.          }
15.
16.          public AutoFitTextureView(Context context, Attribu
   teSet attrs) {
17.              this(context, attrs, 0);
18.          }
19.
20.          public AutoFitTextureView(Context context, Attribu
   teSet attrs, int defStyle) {
21.              super(context, attrs, defStyle);
22.          }
23.
24.          /**
25.           * Sets the aspect ratio for this view. The size
   of the view will be measured based on the ratio
26.           * calculated from the parameters. Note that the
   actual sizes of parameters don't matter, that
27.           * is, calling setAspectRatio(2, 3) and
   setAspectRatio(4, 6) make the same result.
28.           *
29.           * @param width  Relative horizontal size
30.           * @param height Relative vertical size
31.           */
32.          public void setAspectRatio(int width, int height
   ) {
33.              if (width < 0 || height < 0) {
34.                  throw new IllegalArgumentException("Size
   cannot be negative.");
35.              }
36.              mRatioWidth = width;
37.              mRatioHeight = height;
38.              requestLayout();
39.          }
40.
41.          @Override
42.          protected void onMeasure(int widthMeasureSpec, i
   nt heightMeasureSpec) {
43.              super.onMeasure(widthMeasureSpec,
   heightMeasureSpec);
44.              int width = MeasureSpec.getSize(widthMeasureS
   pec);
```

```
45.             int height = MeasureSpec.getSize(heightMeasur
    eSpec);
46.             if (0 == mRatioWidth || 0 == mRatioHeight) {
47.                 setMeasuredDimension(width, height);
48.             } else {
49.                 if (width < height * mRatioWidth / mRatioH
    eight) {
50.                     setMeasuredDimension(width,
    width * mRatioHeight / mRatioWidth);
51.                 } else {
52.                     setMeasuredDimension(height * mRatioWi
    dth / mRatioHeight, height);
53.                 }
54.             }
55.         }
56.     }
```

Anexo 11

Fragmento de código activity_proceso.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/ap
   k/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context=".ProcesoActivity">
8.
9.      <ImageView
10.             android:id="@+id/imgFoto"
11.             android:layout_width="match_parent"
12.             android:layout_height="466dp"
13.             android:layout_alignParentStart="true"
14.             android:layout_alignParentTop="true" />
15.
16.         <LinearLayout
17.             android:id="@+id/groupButton"
18.             android:layout_width="match_parent"
19.             android:layout_height="wrap_content"
20.             android:layout_alignParentBottom="true"
21.             android:layout_alignParentStart="true">
22.
23.             <Button
24.                 android:id="@+id/btnDetectar"
25.                 android:layout_width="0dp"
26.                 android:layout_height="wrap_content"
27.                 android:layout_weight="1"
28.                 android:text="Detectar Cara" />
29.
30.             <Button
31.                 android:id="@+id/btnIdentificar"
32.                 android:layout_width="0dp"
33.                 android:layout_height="wrap_content"
34.                 android:layout_weight="1"
35.                 android:enabled="false"
```

```
                android:text="Identificar Cara" />

            <Button
                android:id="@+id/btnReiniciar"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:enabled="false"
                android:text="Reiniciar Proceso" />

        </LinearLayout>

    </RelativeLayout>
```

Anexo 12

Fragmento de código ProcesoActivity.java

```
package com.uteg.recognitionface;

import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
    import android.os.AsyncTask;
    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.util.Log;
    import android.view.View;
    import android.widget.Button;
    import android.widget.ImageView;
    import android.widget.Toast;

    import com.microsoft.projectoxford.face.FaceServiceClient;
    import com.microsoft.projectoxford.face.FaceServiceRestClient;
    import com.microsoft.projectoxford.face.contract.Face;
    import com.microsoft.projectoxford.face.contract.FaceRectangle;
    import com.microsoft.projectoxford.face.contract.IdentifyResult;
    import com.microsoft.projectoxford.face.contract.Person;
    import com.microsoft.projectoxford.face.contract.TrainingStatus;

    import org.ksoap2.SoapEnvelope;
    import org.ksoap2.serialization.PropertyInfo;
    import org.ksoap2.serialization.SoapObject;
    import org.ksoap2.serialization.SoapPrimitive;
    import org.ksoap2.serialization.SoapSerializationEnvelope;
    import org.ksoap2.transport.HttpTransportSE;

```

```
34.      import java.io.ByteArrayInputStream;
35.      import java.io.ByteArrayOutputStream;
36.      import java.io.File;
37.      import java.io.FileInputStream;
38.      import java.io.InputStream;
39.      import java.util.UUID;
40.
41.      public class ProcesoActivity extends AppCompatActivity
   {
42.
43.          private Salas sala;
44.          private File mFile;
45.          private Button btn_Detectar, btn_Identificar,
   btn_Reiniciar;
46.          private ImageView img_Foto;
47.          private FileInputStream fis = null;
48.          private Bitmap bitmap;
49.          private ByteArrayOutputStream output;
50.          private ByteArrayInputStream inputStream;
51.          private FaceServiceClient
   faceServiceClient = new FaceServiceRestClient("https://westc
   entralus.api.cognitive.microsoft.com/face/v1.0",
52.                  "1e5465389a8246918c274709b3463799");
53.          /*private FaceServiceClient faceServiceClient =
   new
   FaceServiceRestClient("https://westcentralus.api.cognitive.m
   icrosoft.com/face/v1.0",
54.
                          "63747ea4a3154191970ddb28449ea6a3");*
   /
55.          private final String personGroupId = "estudiantest
   otal";
56.          private Face[] facesDetected;
57.          private UUID[] mFaceId = null;
58.          private Face face;
59.          private String[] persona = null;
60.
61.          @Override
62.          protected void onCreate(Bundle savedInstanceState)
63.          {
64.              super.onCreate(savedInstanceState);
65.              setContentView(R.layout.activity_proceso);
66.
67.              //se obtiene la foto que se tomo y a que sala
   se registra
68.              Intent intent = getIntent();
69.              sala = (Salas) intent.getSerializableExtra("sa
   la");
70.              mFile = (File) intent.getSerializableExtra("im
   g");
71.
72.              btn_Detectar = (Button) findViewById(R.id.btnD
   etectar);
73.              btn_Identificar = (Button) findViewById(R.id.b
   tnIdentificar);
74.              btn_Reiniciar = (Button) findViewById(R.id.btn
   Reiniciar);
75.              img_Foto = (ImageView) findViewById(R.id.imgFo
   to);
76.
```

```
77.             //procesa la imagen para ponerla en el
   imageview
78.             procesar_imagen(mFile);
79.
80.             img_Foto.setImageBitmap(bitmap);
81.
82.             //accion para detectar la cara en la foto
83.             btn_Detectar.setOnClickListener(new View.OnCli
   ckListener()
84.             {
85.                 @Override
86.                 public void onClick(View view) {
87.                     new DetectionTask().execute(inputStrea
   m);
88.                 }
89.             });
90.
91.             //accion para identificar a la persona en
   azure
92.             btn_Identificar.setOnClickListener(new View.On
   ClickListener()
93.             {
94.                 @Override
95.                 public void onClick(View view) {
96.                     mFaceId = new UUID[facesDetected.lengt
   h];
97.
98.                     for(int i = 0; i < facesDetected.lengt
   h; i++)
99.                     {
100.                        mFaceId[i] = facesDetected[i].face
   Id;
101.                    }
102.
103.                    new IdentificationTask(personGroupId).
   execute(mFaceId);
104.                }
105.            });
106.
107.            //accion para reiniciar el proceso
108.            btn_Reiniciar.setOnClickListener(new View.OnCl
   ickListener() {
109.                @Override
110.                public void onClick(View view) {
111.                    Intent
   i = getBaseContext().getPackageManager()
112.                                .getLaunchIntentForPackage( ge
   tBaseContext().getPackageName() );
113.                    i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_
   TOP);
114.                    startActivity(i);
115.                }
116.            });
117.        }
118.
119.        //metodo para procesar la imagen, para poder
   ponerla en imageview
120.        private void procesar_imagen(File lFile)
121.        {
122.            try{
123.                fis = new FileInputStream(mFile);
```

```
124.             }
125.             catch(Exception e){
126.                 e.printStackTrace();
127.             }
128.
129.             bitmap = BitmapFactory.decodeStream(fis);
130.             output = new ByteArrayOutputStream();
131.             bitmap.compress(Bitmap.CompressFormat.JPEG, 10
    0, output);
132.             inputStream = new ByteArrayInputStream(output.
    toByteArray());
133.         }
134.
135.         //accion para detectar la cara en la foto
136.         private class DetectionTask extends AsyncTask<Inpu
    tStream, String, Face[]>
137.         {
138.             private ProgressDialog
    mDialog = new ProgressDialog(ProcesoActivity.this);
139.
140.             //proceso que envia la consulta a azure para
    verificar la cara en la foto
141.             @Override
142.             protected Face[] doInBackground(InputStream...
    params)
143.             {
144.                 try
145.                 {
146.                     publishProgress("Detectando...");
147.                     Face[] results = faceServiceClient.det
    ect(params[0],
148.                             true,         // returnFaceId
149.                             false,        //
    returnFaceLandmarks
150.                             null           //
    returnFaceAttributes: a string like "age, gender"
151.                     );
152.
153.                     if (results == null)
154.                     {
155.                         publishProgress("Finalizo
    Deteccion. No detecto nada");
156.                         return null;
157.                     }
158.                     else
159.                     {
160.                         publishProgress(String.format("Fin
    alizo Deteccion. %d cara(s) detectadas", results.length));
161.                         return results;
162.                     }
163.                 } catch (Exception e) {
164.                     publishProgress("Fallo Deteccion");
165.                     Log.e("Error al detectar",
    e.toString());
166.                     return null;
167.                 }
168.             }
169.             @Override
170.             protected void onPreExecute()
171.             {
172.                 //TODO: show progress dialog
```

```
173.                 mDialog.show();
174.             }
175.             @Override
176.             protected void onProgressUpdate(String... prog
   ress)
177.             {
178.                 //TODO: update progress
179.                 mDialog.setMessage(progress[0]);
180.             }
181.
182.             /*metodo que indica cuando el proceso termina
   si encontro un rostro
183.             * en el caso de haber encontrado, habilita el
   boton de indenticacion
184.             * si no encontro habilita el boton de reinicio
   de proceso*/
185.             @Override
186.             protected void onPostExecute(Face[] faces)
187.             {
188.                 //TODO: update face frames
189.                 mDialog.dismiss();
190.
191.                 facesDetected = faces;
192.
193.                 if(faces == null || faces.length == 0)
194.                 {
195.                     Toast.makeText(getApplicationContext()
   ,
196.                             String.format("No detecto una
   cara en la imagen, volver a reiniciar el proceso"),
197.                             Toast.LENGTH_LONG).show();
198.                     btn_Reiniciar.setEnabled(true);
199.                     btn_Detectar.setEnabled(false);
200.                 }
201.                 else
202.                 {
203.                     Toast.makeText(getApplicationContext()
   ,
204.                             String.format("Finalizo
   Deteccion. %d cara(s) detectadas", faces.length),
205.                             Toast.LENGTH_LONG).show();
206.                     btn_Identificar.setEnabled(true);
207.                     btn_Detectar.setEnabled(false);
208.                 }
209.             }
210.         }
211.
212.         //accion para identificar la cara que se encontro
   en la foto si esta en la base de azure
213.         private class IdentificationTask extends AsyncTask
   <UUID, String, IdentifyResult[]>{
214.
215.             private String personGroupId;
216.             private ProgressDialog
   mDialog = new ProgressDialog(ProcesoActivity.this);
217.
218.             public IdentificationTask(String personGroupId
   ) {
219.                 this.personGroupId = personGroupId;
220.             }
221.
```

```
222.            //metodo para enviar a azure he identificar la
    cara de la persona a quien pertenece
223.            @Override
224.            protected IdentifyResult[] doInBackground(UUID
    ... uuids) {
225.                try
226.                {
227.                    publishProgress("Obteniendo el estatus
    del grupo de personas...");
228.
229.                    TrainingStatus
    trainingStatus = faceServiceClient.getPersonGroupTrainingSta
    tus(this.personGroupId);
230.
231.                    if(trainingStatus.status != TrainingSt
    atus.Status.Succeeded)
232.                    {
233.                        publishProgress("Estado training
    del grupo de personas es " + trainingStatus.status);
234.                        return null;
235.                    }
236.
237.                    publishProgress("Identificando...");
238.
239.                    return faceServiceClient.identity(pers
    onGroupId, //person group id
240.                            uuids, //face ids
241.                            1 //max number of candidates
    return
242.                    );
243.
244.                }
245.                catch (Exception ex)
246.                {
247.                    publishProgress("Fallo al Obtener
    estatus del grupo");
248.                    Log.e("Error al detectar",
    ex.toString());
249.                    return null;
250.                }
251.            }
252.
253.            @Override
254.            protected void onPreExecute() {
255.                mDialog.show();
256.            }
257.
258.            /*metodo que indica cuando el proceso termina
    si identifico una persona
259.             * en el caso de haber identificado, realiza
    el insert de registro de asistencia
260.             * y procede a la siguinte pantalla de exito
261.             * si no encontro se dirige a la pantalla de
    no exito
262.             * si no encuentra base o no existe personas
    en la base de reinicia el proceso*/
263.            @Override
264.            protected void onPostExecute(IdentifyResult[]
    identifyResults) {
265.                mDialog.dismiss();
266.
```

```
267.                 if(identifyResults != null)
268.                 {
269.                     if(identifyResults[0].candidates.isEmp
    ty()) {
270.                         Intent
    intent = new Intent(ProcesoActivity.this,
    ErrorActivity.class);
271.                         startActivity(intent);
272.                         finish();
273.                     }
274.                     else
275.                     {
276.                         for (IdentifyResult
    identifyResult : identifyResults) {
277.                             new PersonDetectionTask(person
    GroupId).execute(identifyResult.candidates.get(0).personId);
278.                         }
279.                     }
280.                 }
281.                 else
282.                 {
283.                     Toast.makeText(getApplicationContext()
    ,
284.                             "La base de datos no tiene
    personas para identificar o no ha sido creada",
285.                             Toast.LENGTH_LONG).show();
286.                     btn_Reiniciar.setEnabled(true);
287.                     btn_Detectar.setEnabled(false);
288.                     btn_Identificar.setEnabled(false);
289.                 }
290.
291.             }
292.
293.             @Override
294.             protected void onProgressUpdate(String... valu
    es) {
295.                 mDialog.setMessage(values[0]);
296.             }
297.         }
298.
299.         /*accion para marcar a la persona que aparecio en
    la foto*/
300.         private class PersonDetectionTask extends AsyncTas
    k<UUID, String, Person> {
301.
302.             private String personGroupId;
303.             private ProgressDialog
    mDialog = new ProgressDialog(ProcesoActivity.this);
304.
305.             public PersonDetectionTask(String personGroupI
    d) {
306.                 this.personGroupId = personGroupId;
307.             }
308.
309.             //obtiene la informacion de la persona
    detectada
310.             @Override
311.             protected Person
    doInBackground(UUID... uuids) {
312.                 try
313.                 {
```

```
314.                     publishProgress("Obteniendo estatus
   del person group...");
315.
316.                     return faceServiceClient.getPerson(per
   sonGroupId, uuids[0]);
317.                 }
318.                 catch (Exception ex)
319.                 {
320.                     publishProgress("Fallo al detectar a
   la persona");
321.                     Log.e("Error al detectar",
   ex.toString());
322.                     return null;
323.                 }
324.             }
325.
326.             @Override
327.             protected void onPreExecute() {
328.                 mDialog.show();
329.             }
330.
331.             /* metodo que se realiza despues de obtener
   los datos de azure
332.             * descompone esos datos para grabarlos en la
   base de datos y
333.             * regustrar su ingreso*/
334.             @Override
335.             protected void onPostExecute(Person person) {
336.                 mDialog.dismiss();
337.
338.                 persona = person.name.split("-");
339.
340.                 //img_Foto.setImageBitmap(drawFaceRectangl
   eOnBitmap(bitmap, facesDetected, person.name));
341.                 img_Foto.setImageBitmap(drawFaceRectangleO
   nBitmap(bitmap, facesDetected, persona[0]));
342.
343.                 new CallWebService(new String[]{persona[1]
   , sala.getId_sala()}).execute();
344.             }
345.
346.             @Override
347.             protected void onProgressUpdate(String... valu
   es) {
348.                 mDialog.setMessage(values[0]);
349.             }
350.         }
351.
352.         //marca la cara de la foto con los datos de la
   persona identificada
353.         private Bitmap drawFaceRectangleOnBitmap(Bitmap
   bitmap, Face[] facesDetected, String name) {
354.
355.             Bitmap
   lbitmap = bitmap.copy(Bitmap.Config.ARGB_8888, true);
356.             Canvas canvas = new Canvas(lbitmap);
357.
358.             //Rectangle
359.             Paint paint = new Paint();
360.             paint.setAntiAlias(true);
361.             paint.setStyle(Paint.Style.STROKE);
```

```
362.             paint.setColor(Color.WHITE);
363.             paint.setStrokeWidth(12);
364.
365.             if(facesDetected != null)
366.             {
367.                 for(Face face: facesDetected)
368.                 {
369.                     FaceRectangle
    faceRectangle = face.faceRectangle;
370.                     canvas.drawRect(faceRectangle.left,
371.                             faceRectangle.top,
372.                             faceRectangle.left + faceRecta
    ngle.width,
373.                             faceRectangle.top + faceRectan
    gle.height,
374.                             paint);
375.                     drawTextOnCanvas(canvas,
376.                             100,
377.                             ((faceRectangle.left+faceRecta
    ngle.width)/2)+100,
378.                             (faceRectangle.top+faceRectang
    le.height)+50,
379.                             Color.WHITE,
380.                             name);
381.                 }
382.             }
383.
384.             return lbitmap;
385.         }
386.
387.         //dibuja un cuadrado blanco en la cara de la foto
388.         private void drawTextOnCanvas(Canvas canvas, int t
    extSize, int x, int y, int color, String name) {
389.             Paint paint = new Paint();
390.             paint.setAntiAlias(true);
391.             paint.setStyle(Paint.Style.FILL);
392.             paint.setColor(color);
393.             paint.setTextSize(textSize);
394.
395.             float textWidth = paint.measureText(name);
396.
397.             canvas.drawText(name, x-(textWidth/2), y-
    (textSize/2), paint);
398.         }
399.
400.         //llamado del webservice para insertar el registro
    de ingreso a la sala de la persona detectada
401.         class CallWebService extends AsyncTask<Void, Void,
    String>
402.         {
403.             private String SOAP_ACTION;
404.             private String OPERATION_NAME;
405.             private final String WSDL_TARGET_NAMESPACE = "
    http://tempuri.org/";
406.             public final String SOAP_ADDRESS = "http://192
    .168.100.27/WS_recognitionface.asmx";
407.             private String[] parametros = null;
408.
409.             public CallWebService(String[] parametros) {
410.                 this.parametros = parametros;
411.             }
```

```
412.
413.             @Override
414.             protected void onPostExecute(String s) {
415.                 if(s != null)
416.                 {
417.                     if(s.equals("OK")) {
418.                         Toast.makeText(getApplicationConte
    xt(),
419.                                 "Persona Identificada",
420.                                 Toast.LENGTH_LONG).show();
421.
422.                         Intent
    intent = new Intent(ProcesoActivity.this,
    ExitoActivity.class);
423.                         intent.putExtra("nombre",
    persona[0]);
424.                         startActivity(intent);
425.                         finish();
426.                     }
427.                     else
428.                     {
429.                         btn_Detectar.setEnabled(false);
430.                         btn_Identificar.setEnabled(false);
431.                         btn_Reiniciar.setEnabled(true);
432.                         Toast.makeText(getApplicationConte
    xt(),
433.                                 "Volver a identificar a la
    persona por un error interno",
434.                                 Toast.LENGTH_LONG).show();
435.                     }
436.                 }
437.                 else
438.                 {
439.                     btn_Detectar.setEnabled(false);
440.                     btn_Identificar.setEnabled(false);
441.                     btn_Reiniciar.setEnabled(true);
442.                     Toast.makeText(getApplicationContext()
    ,
443.                             "Volver a identificar a la
    persona por un error interno",
444.                             Toast.LENGTH_LONG).show();
445.                 }
446.             }
447.
448.             @Override
449.             protected String doInBackground(Void... voids)
    {
450.                 SOAP_ACTION = "http://tempuri.org/InsertRe
    gistro";
451.                 OPERATION_NAME = "InsertRegistro";
452.
453.                 SoapObject
    request = new SoapObject(WSDL_TARGET_NAMESPACE,
    OPERATION_NAME);
454.
455.                 PropertyInfo pi = new PropertyInfo();
456.                 pi.setName("cedula");
457.                 pi.setValue(parametros[0]);
458.                 pi.setType(String.class);
459.                 request.addProperty(pi);
460.                 pi = new PropertyInfo();
```

```
461.                 pi.setName("idsala");
462.                 pi.setValue(parametros[1]);
463.                 pi.setType(String.class);
464.                 request.addProperty(pi);
465.
466.                 SoapSerializationEnvelope
   envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11)
   ;
467.                 envelope.dotNet = true;
468.
469.                 envelope.setOutputSoapObject(request);
470.
471.                 HttpTransportSE
   httpTransport = new HttpTransportSE(SOAP_ADDRESS);
472.                 SoapPrimitive response = null;
473.
474.                 try
475.                 {
476.                     httpTransport.call(SOAP_ACTION,
   envelope);
477.                     response = (SoapPrimitive)envelope.get
   Response();
478.                     return response.toString();
479.                 }
480.                 catch (Exception e)
481.                 {
482.                     Log.d("Error WebService",
   e.toString());
483.                     return null;
484.                 }
485.             }
486.         }
487.     }
```

Anexo 13

Fragmento de código activity_exito.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/ap
   k/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context=".ExitoActivity">
8.
9.      <ImageView
10.             android:id="@+id/iv_check"
11.             android:layout_width="200dp"
12.             android:layout_height="200dp"
13.             android:layout_alignParentTop="true"
14.             android:layout_centerHorizontal="true"
15.             android:layout_marginTop="24dp"
16.             app:srcCompat="@drawable/ic_exito"
17.             tools:ignore="ContentDescription,MissingConstra
   ints"
18.             tools:layout_editor_absoluteX="92dp"
19.             tools:layout_editor_absoluteY="79dp" />
20.
```

```
21.          <TextView
22.              android:id="@+id/tv_mensaje"
23.              android:layout_width="wrap_content"
24.              android:layout_height="wrap_content"
25.              android:layout_alignParentBottom="true"
26.              android:layout_centerHorizontal="true"
27.              android:layout_marginBottom="225dp"
28.              android:text="@string/lbl_exito"
29.              android:textAppearance="@android:style/TextAppe
   arance.Large"
30.              android:textStyle="bold"
31.              tools:ignore="MissingConstraints"
32.              tools:layout_editor_absoluteX="18dp"
33.              tools:layout_editor_absoluteY="344dp" />
34.
35.          <Button
36.              android:id="@+id/btn_regresar"
37.              android:layout_width="wrap_content"
38.              android:layout_height="wrap_content"
39.              android:layout_alignParentBottom="true"
40.              android:layout_centerHorizontal="true"
41.              android:layout_marginBottom="77dp"
42.              android:text="@string/btn_regresar"
43.              android:textAppearance="@android:style/TextAppe
   arance.Large"
44.              android:textStyle="bold"
45.              tools:ignore="MissingConstraints"
46.              tools:layout_editor_absoluteX="135dp"
47.              tools:layout_editor_absoluteY="410dp" />
48.
49.      </RelativeLayout>
```

Anexo 14

Fragmento de código ExitoActivity.java

```
1. package com.uteg.recognitionface;
2.
3. import android.content.Intent;
4. import android.support.v7.app.AppCompatActivity;
5. import android.os.Bundle;
6. import android.view.View;
7. import android.widget.Button;
8. import android.widget.TextView;
9.
10.      //pantalla de exito si identifico una persona de la
   base azure y muestra su nombre identificado
11.      public class ExitoActivity extends AppCompatActivity {
12.
13.          private TextView tv_mensaje;
14.          private String nombre;
15.          private Button btn_regresar;
16.          public static boolean respuesta = false;
17.
18.          @Override
19.          protected void onCreate(Bundle
   savedInstanceState) {
```

```
20.             super.onCreate(savedInstanceState);
21.             setContentView(R.layout.activity_exito);
22.
23.             Intent intent = getIntent();
24.             nombre = (String) intent.getSerializableExt
25.
26.             tv_mensaje = (TextView) findViewById(R.id.tv_
   mensaje);
27.             tv_mensaje.setText("Exito. La persona
   " + nombre + ", esta registrada en el sistema");
28.
29.             btn_regresar = (Button) findViewById(R.id.btn
   _regresar);
30.             btn_regresar.setOnClickListener(new View.OnCli
   ckListener() {
31.                 @Override
32.                 public void onClick(View v) {
33.                     Intent
   i = getBaseContext().getPackageManager()
34.                             .getLaunchIntentForPackage( ge
   tBaseContext().getPackageName() );
35.                     i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_
   TOP);
36.                     startActivity(i);
37.                     finish();
38.                 }
39.             });
40.         }
41.     }
```

Anexo 15

Fragmento de código activity_error.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout xmlns:android="
   http://schemas.android.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context=".ErrorActivity">
8.
9.      <ImageView
10.             android:id="@+id/iv_check"
11.             android:layout_width="200dp"
12.             android:layout_height="200dp"
13.             app:srcCompat="@drawable/ic_error"
14.             tools:ignore="ContentDescription,MissingConstra
   ints"
15.             tools:layout_constraintTop_creator="1"
16.             tools:layout_constraintRight_creator="1"
17.             app:layout_constraintRight_toRightOf="parent"
18.             android:layout_marginTop="79dp"
19.             tools:layout_constraintLeft_creator="1"
```

```
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/tv_mensaje"
            android:layout_width="0dp"
            android:layout_height="76dp"
            android:text="@string/lbl_error"
            android:textAlignment="center"
            android:textAppearance="@android:style/TextAppe
arance.Large"
            android:textStyle="bold"
            tools:ignore="MissingConstraints"
            tools:layout_constraintRight_creator="1"
            tools:layout_constraintBottom_creator="1"
            app:layout_constraintBottom_toTopOf="@+id/btn_r
egresar"
            android:layout_marginStart="23dp"
            android:layout_marginEnd="23dp"
            app:layout_constraintRight_toRightOf="parent"
            tools:layout_constraintLeft_creator="1"
            android:layout_marginBottom="20dp"
            app:layout_constraintLeft_toLeftOf="parent" />

        <Button
            android:id="@+id/btn_regresar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btn_regresar"
            android:textAppearance="@android:style/TextAppe
arance.Large"
            android:textStyle="bold"
            tools:ignore="MissingConstraints"
            tools:layout_constraintRight_creator="1"
            tools:layout_constraintBottom_creator="1"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            tools:layout_constraintLeft_creator="1"
            android:layout_marginBottom="33dp"
            app:layout_constraintLeft_toLeftOf="parent" />

    </android.support.constraint.ConstraintLayout>
```

Anexo 16

Fragmento de código ErrorActivity.java

```
package com.uteg.recognitionface;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

//pantalla de error si no identifico una persona de la base
azure
    public class ErrorActivity extends AppCompatActivity {

        private TextView tv_mensaje;
```

```
13.          private String E_FaceId;
14.          private Button btn_regresar;
15.
16.          @Override
17.          protected void onCreate(Bundle
   savedInstanceState) {
18.              super.onCreate(savedInstanceState);
19.              setContentView(R.layout.activity_error);
20.
21.              tv_mensaje = (TextView) findViewById(R.id.tv_
   mensaje);
22.              btn_regresar = (Button) findViewById(R.id.btn
   _regresar);
23.              btn_regresar.setOnClickListener(new View.OnCli
   ckListener() {
24.                  @Override
25.                  public void onClick(View v) {
26.                      Intent
   i = getBaseContext().getPackageManager()
27.                              .getLaunchIntentForPackage( ge
   tBaseContext().getPackageName() );
28.                      i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_
   TOP);
29.                      startActivity(i);
30.                      finish();
31.                  }
32.              });
33.          }
34.      }
```

Anexo 17

Fragmento de código Program.cs

```
1. using Microsoft.ProjectOxford.Face;
2. using Microsoft.ProjectOxford.Face.Contract;
3. using System;
4. using System.Collections.Generic;
5. using System.Data.SqlClient;
6. using System.IO;
7. using System.Linq;
8. using System.Text;
9. using System.Threading.Tasks;
10.
11.      namespace AIBOTTraining
12.      {
13.          class Program
14.          {
15.
16.              public static bool creaGrupo = false;
17.              public static bool creaPersona = false;
18.              public static bool eliminaGrupo = false;
19.              public static bool tStatus = false;
20.              public static bool reconocimientoPersona = fal
   se;
21.
```

```
22.             FaceServiceClient
    faceServiceClient = new FaceServiceClient("1e5465389a8246918
    c274709b3463799", "https://westcentralus.api.cognitive.micro
    soft.com/face/v1.0");
23.             //FaceServiceClient faceServiceClient = new
    FaceServiceClient("63747ea4a3154191970ddb28449ea6a3",
    "https://westcentralus.api.cognitive.microsoft.com/face/v1.0
    ");
24.
25.             //invoca el metodo de azure para creacion de
    grupo
26.             public async void CreatePersonGroup(string per
    sonGroupId, string personGroupName)
27.             {
28.                 try
29.                 {
30.                     await faceServiceClient.CreatePersonGr
    oupAsync(personGroupId, personGroupName);
31.                     Console.WriteLine("Se creo el
    personGroupId");
32.                     creaGrupo = true;
33.                 }
34.                 catch(Exception ex)
35.                 {
36.                     Console.WriteLine("Error Create Person
    Group\n" + ex.Message);
37.                 }
38.             }
39.
40.             //invoca el metodo de azure para eliminar el
    grupo
41.             public async void DeletePersonGroup(string per
    sonGroupId)
42.             {
43.                 try
44.                 {
45.                     await faceServiceClient.DeletePersonGr
    oupAsync(personGroupId);
46.                     Console.WriteLine("Se elimino el
    personGroupId");
47.                     eliminaGrupo = true;
48.                 }
49.                 catch(Exception ex)
50.                 {
51.                     Console.WriteLine("Error Delete Person
    Group\n" + ex.Message);
52.                 }
53.             }
54.
55.             //invoca el metodo de azure que adiciona la
    persona al grupo
56.             public async void AddPersonToGroup(string pers
    onGroupId, string name, Stream imagenS)
57.             {
58.                 try
59.                 {
60.                     await faceServiceClient.GetPersonGroup
    Async(personGroupId);
61.
```

```
62.                     CreatePersonResult
   person = await faceServiceClient.CreatePersonAsync(personGro
   upId, name);
63.
64.                     DetectFaceAndRegister(personGroupId,
   person, imagenS);
65.                     Console.WriteLine("Se adiciono la
   persona en el grupo");
66.                     creaPersona = true;
67.                 }
68.                 catch (Exception ex)
69.                 {
70.                     Console.WriteLine("Error Add Person to
   Group\n" + ex.Message);
71.                 }
72.             }
73.
74.             //detecta la caras en las imagenes y las
   registra en azure
75.             private async void DetectFaceAndRegister(strin
   g personGroupId, CreatePersonResult person, Stream imagenS)
76.             {
77.                 try
78.                 {
79.                     await faceServiceClient.AddPersonFaceA
   sync(personGroupId, person.PersonId, imagenS);
80.                     Console.WriteLine("Se detecto y se
   registro la persona en el grupo");
81.                 }
82.                 catch (Exception ex)
83.                 {
84.                     Console.WriteLine(ex.Message);
85.                 }
86.             }
87.
88.             //invoca metodo de sincronizacion de lo
   enviado a azure
89.             public async void TrainingAI(string personGrou
   pId)
90.             {
91.                 try
92.                 {
93.                     await faceServiceClient.TrainPersonGro
   upAsync(personGroupId);
94.                     TrainingStatus trainingStatus = null;
95.
96.                     while (true)
97.                     {
98.                         trainingStatus = await faceService
   Client.GetPersonGroupTrainingStatusAsync(personGroupId);
99.
100.                        if (trainingStatus.Status != Statu
   s.Running)
101.                            break;
102.
103.                        await Task.Delay(5000);
104.
105.                    }
106.                    tStatus = true;
107.                    Console.WriteLine("Training AI
   Completed");
```

```
108.                 }
109.                 catch(Exception ex)
110.                 {
111.                     Console.WriteLine(ex.Message);
112.                 }
113.             }
114.
115.             //invoca metodo para reconocimiento de la
    imagen que se realiza de prueba
116.             public async void RecognitionFace(string perso
    nGroupId, string imgPath)
117.             {
118.                 using (Stream s = File.OpenRead(imgPath))
119.                 {
120.                     var faces = await faceServiceClient.De
    tectAsync(s);
121.                     var faceIds = faces.Select(face => fac
    e.FaceId).ToArray();
122.
123.                     try
124.                     {
125.                         var results = await faceServiceCli
    ent.IdentifyAsync(personGroupId, faceIds);
126.
127.                         foreach(var identifyResult in resu
    lts)
128.                         {
129.                             Console.WriteLine($"Resultado
    de caras: {identifyResult.FaceId}");
130.                             if (identifyResult.Candidates.
    Length == 0)
131.                                 Console.WriteLine("No se
    identifico");
132.                             else
133.                             {
134.                                 //Get top 1 among all
    candidates returned
135.                                 var candidateId = identify
    Result.Candidates[0].PersonId;
136.                                 var person = await faceSer
    viceClient.GetPersonAsync(personGroupId, candidateId);
137.                                 Console.WriteLine($"Identi
    ficado es {person.Name}");
138.                             }
139.                         }
140.                         reconocimientoPersona = true;
141.                     }
142.                     catch(Exception ex)
143.                     {
144.                         Console.WriteLine("Error
    reconocimiento cara: " + ex.Message);
145.                     }
146.                 }
147.
148.             }
149.
150.             //metodo para crear el grupo en azure
151.             public void crearGrupo()
152.             {
153.                 //Se crea el grupo para adicionar las
    personas y sus rostros
```

```
154.                 CreatePersonGroup("estudiantestotal", "Tot
   al Estudiantes");
155.             }
156.
157.             //elimina el grupo en azure
158.             public void deleteGrupo()
159.             {
160.                 //para eliminar el grupo y volver a
   realizar la prueba
161.                 DeletePersonGroup("estudiantestotal");
162.             }
163.
164.             //adiciona las personas al grupo con su
   respectiva imagen
165.             public void adicionaPersona()
166.             {
167.                 /**
168.                  * Se lee de la base de datos las 3 fotos
   por personas para poder identificar que es la correcta
169.                  * Como observacion para pruebas se esta
   utilizando PersonGroup, pero para cantidades
170.                  * mayores se debe usar LargePersonGroup
171.                  **/
172.                 string nombre;
173.                 byte[] imagen1;
174.                 byte[] imagen2;
175.                 byte[] imagen3;
176.
177.
178.                 using (SqlConnection
   con = new SqlConnection("server=DESKTOP-
   S552A9T\\SQLEXPRESS;database=recognitionfacedb;uid=sa;passwo
   rd=sa;"))
179.                 {
180.                     con.Open();
181.
182.                     String sql = "select Nombre1 + ' '
   +  Apellido1 + '-' + Cedula as nombre, " +
183.                                         "       Imagen1, " +
184.                                         "       Imagen2, " +
185.                                         "       Imagen3 " +
186.                                         "from
   recognitionfacedb.dbo.Tbl_Persona " +
187.                                         "where Estado = 'A'";
188.
189.                     SqlCommand
   command = new SqlCommand(sql, con);
190.
191.                     using (SqlDataReader
   reader = command.ExecuteReader())
192.                     {
193.                         while (reader.Read())
194.                         {
195.                             nombre = reader.GetString(0);
196.                             imagen1 = (byte[])reader["Imag
   en1"];
197.                             imagen2 = (byte[])reader["Imag
   en2"];
198.                             imagen3 = (byte[])reader["Imag
   en3"];
199.
```

```
                                Stream
    imagen1S = new MemoryStream(imagen1);
                                Stream
    imagen2S = new MemoryStream(imagen2);
                                Stream
    imagen3S = new MemoryStream(imagen3);

                                for (int i = 0; i < 3; i++)
                                {
                                    if (i == 0)
                                    {
                                        AddPersonToGroup("estu
    diantestotal", nombre, imagen1S);
                                    }

                                    if (i == 1)
                                    {
                                        AddPersonToGroup("estu
    diantestotal", nombre, imagen2S);
                                    }

                                    if (i == 2)
                                    {
                                        AddPersonToGroup("estu
    diantestotal", nombre, imagen3S);
                                    }
                                }
                            }

                            reader.Close();
                        }

                        command.Dispose();
                        con.Close();
                    }
                }

                //sincroniza con la base de azure
                public void trainingGrupo()
                {
                    //Para saber si ya se integro todo el
    grupo de personas para la verificacion
                    TrainingAI("estudiantestotal");
                }

                //metodo para hacer una prueba de
    reconocimiento
                public void verificarReconocimiento()
                {
                    //Para verificacion de la imagen con el
    grupo creado, si es acertado o no
                    RecognitionFace("estudiantestotal", @"C:\e
    jemplo\persona4\imagen2.jpg");
                }

                //programa principal para realizar la subida a
    la base de datos azure
                static void Main(string[] args)
                {
                    bool done = false;

```

```
250.                 do
251.                 {
252.                     Console.WriteLine("Selecccione las
    siguientes opciones en el orden indicado");
253.                     Console.WriteLine("\t1 -- Crea
    Grupo");
254.                     Console.WriteLine("\t2 -- Adiciona
    Persona al Grupo");
255.                     Console.WriteLine("\t3 -- Verifica
    Sincronizacion");
256.                     Console.WriteLine("\t4 -- Verifica
    Reconocimiento");
257.                     Console.WriteLine("\t5 -- Elimina
    Grupo");
258.                     Console.WriteLine("\t0 -- Salir");
259.                     Console.WriteLine("Ingrese la opcion
    (0 para salir): ");
260.                     string strSeleccion = Console.ReadLine
    ();
261.                     int iSeleccion;
262.
263.                     try
264.                     {
265.                         iSeleccion = int.Parse(strSeleccio
    n);
266.                     }
267.                     catch(FormatException)
268.                     {
269.                         Console.WriteLine("\r\nQue?\r\n");
270.                         continue;
271.                     }
272.
273.                     Console.WriteLine("You selected
    " + iSeleccion);
274.
275.                     switch (iSeleccion)
276.                     {
277.                         case 0:
278.                             done = true;
279.                             break;
280.                         case 1:
281.                             new Program().crearGrupo();
282.                             break;
283.                         case 2:
284.                             new Program().adicionaPersona(
    );
285.                             break;
286.                         case 3:
287.                             new Program().trainingGrupo();
288.                             break;
289.                         case 4:
290.                             new Program().verificarReconoc
    imiento();
291.                             break;
292.                         case 5:
293.                             new Program().deleteGrupo();
294.                             break;
295.                         default:
296.                             Console.WriteLine("A
    seleccionado un numero invalido: {0}\r\n", iSeleccion);
297.                             continue;
```

```
                    }
                    Console.WriteLine();
                } while (!done);

                Console.WriteLine("\nGoodbye!");
            }
        }
    }
```