Creado por:

Daniel Herrero Berengué.

Tutor:

Raúl Sánchez Reillo.

SISTEMA DE IDENTIFICACIÓN DE PERSONAS POR IRIS MEDIANTE FILTRADO DE GABOR EN PLATAFORMA ANDROID.

Proyecto Fin de Grado de Ingeniería Electrónica Industrial y Automática.

Universidad Carlos III de Madrid

Agradecimientos

Este proyecto culmina cuatro años de esfuerzo para ser ingeniero. Durante este tiempo he podido llegar a ver la electrónica y todos sus campos contiguos con diferentes ojos. He podido comprobar como cada vez más intento razonar, entender y solucionar situaciones, ya sean de origen tecnológico o no, de una forma completamente diferente a como lo hacia antes de entrar en esta carrera, cuando comprendo el fenómeno que lo produce aplicando sistemas de análisis o procedimientos usados durante la carrera me produce una gran satisfacción.

Este proyecto es en definitiva una de mis grandes satisfacciones de trabajo realizada en mi vida.

He de dar las gracias a mi tutor Raúl Sánchez y a Jaime Uriarte, por su disponibilidad, motivación y buen carácter. También al grupo de investigación GUTI, donde impartirnos clases de orientación en los campos que debíamos tocar para la realización de nuestros trabajos.

Sobretodo he de agradecer a mis padres María José y José Emilio, y mi hermano Jorge, por su apoyo incondicional, la confianza y la esperanza depositada en mí a lo largo de muchos años.

No me puedo olvidar de mi grupo de mejores amigos de toda la vida. Por su comprensión sobretodo durante estos últimos meses de mi indisponibilidad, de su apoyo y su facilidad para hacer que pueda olvidarme de todas mis preocupaciones y relajarme. He de mencionar también a mis compañeros de clase.

En especial a mi novia Flavia por estar conmigo animando a seguir estudiando y seguir trabajando con este proyecto a pesar de las adversidades, si no fuera por ella estoy seguro que no hubiese llegado a alcanzar mis metas.

Resumen

Las facilidades que nos ofrece la tecnología ha hecho que cada vez dependamos más de ella y por tanto necesitemos que todo lo que usamos cotidianamente esté en conexión con dispositivos tecnológicos. Es por eso que los dispositivos móviles de última generación, llamados smartphones constituyen una herramienta prácticamente indispensable para la mayoría de las personas. Estos dispositivos al contener una gran información del usuario es necesario que dispongan de un sistema de seguridad eficaz y cómodo, siendo esta demanda cada vez más importante y especialmente al ampliarse los usos y aplicaciones de estos dispositivos. De esta demanda ha surgido este proyecto, el cual se basa en implementar en un dispositivo móvil Android, un sistema de seguridad por reconocimiento de sujetos a través del iris. Este proyecto está desarrollado para poder demostrar que la implementación de sistemas de seguridad de alto procesado es posible en dispositivos Android, con lo que puede originar proyectos futuros de protección de datos privados, tales como contraseñas o tarjetas de crédito, o de bloqueo y desbloqueo del dispositivo.

Abstract

The possibilities offered by current technology make us more and more dependable or it and therefore we need to establish everything in connection with technological devices that we use daily. That's why the latest generation of mobile devices, called smartphones, is practically an indispensable tool for most people. These devices contain a lot of private information needing an effective and comfortable security system, this demand is more important day by day. From this demand has born this project, which has been based on a mobile device implemented on Android, a security system for iris recognition subject. This project has been developed to demonstrate that the implementation of high security systems processing is possible on Android devices. This can lead to future projects for the protection of sensitive data such as passwords or credit card, or lock and unlock the device.

Índice de contenido:

Índice de figuras

Índice de tablas

Índice de acrónimos y referencias.

AM	Amplitude Modulation (Amplitud modulada)
BMP	Bit Mapped Picture (Imagen mapeada de bits)
CDMA	Code Division Multiple Access (Código de acceso múltiple por división)
EDGE	Enhanced Data Rates for Global Evolution (Velocidades de datos mejoradas para la evolución global)
EMS	Environmental Media Services (Servicio de medios ambientales)
FM	Frecuency Modulation (Frecuencia modulada)
GSM	Global System for Mobile Communications (Sistema global para comunicaciones móviles)
GPRS	General Packet Radio System (Sistema general de paquete de radio)
GPS	Global Positioning System (Sistema de posicionamiento global)
HSCSD	High Speed Circuit Switched Data (Circuito de alta velocidad de datos con conmutación)
MMS	Multimedia Message Service (Servicio de mensajes multimedia)
NFC	Near Field Communication (Comunicación por campo cercano)
PDC	Personal Digital Communications (Comunicaciones digitales personales)
PNG	Portable Network Graphics (Gráficos de Red Portátiles)
RDSI	Red Digital de Servicios Integrados (Red digital de servicios integrados)
SDR	Software Defined Radios (Software de radios definidos)
SMS	Short Message Service (Servicio de mensajes cortos)
TACS	Total Access Communication System (Sisteme de acceso total a la comunicación)
TIFF	Tagged Image File Format (Formato de archivo de imagen etiquetado)
UMTS	Universal Mobile Telecommunications System (Sistema Universal de telecomunicaciones móviles)

Diccionario

ActionBar: (Término informático) Función de ventana que identifica la aplicación y la ubicación el usuario, y proporciona las acciones del usuario y modos de navegación.

Apk: Un archivo con extensión .apk es un paquete para el sistema operativo Android. Este formato es una variante del formato JAR de JAVA y se usa para distribuir e instalar componentes empaquetados para la plataforma Android

App: Aplicación de software.

Array: (Término informático) recurso en cadena de variables del mismo tipo.

Arraylist: (Término informático) Implementación de lista respaldado por un array. Todas las operaciones opcionales, incluyendo agregar, quitar y remplazar elementos son compatibles.

Bitmap: Mapa de bits, también conocida como imagen matricial, es una estructura o fichero de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada matriz, que se puede visualizar en papel, monitor u otro dispositivo de representación.

Bundle: (Término informático) Mapeo de valores de cadena a diferentes tipos parcelables.

Canvas: (Término informático) Clase que mantiene la función de Draw (dibujar). Para usarlo es necesario un bitmap para mantener los píxeles de destino, un bitmap de origen y la pintura o matiz de color a aplicar.

Cast: (Emitir) Término informático referido al cambio de declaración del tipo de una variable por otra.

Checkbox: (Término informático) Casilla de verificación por accionamiento por un botón, que especifica dos estados, activado o desactivado.

Dithering: Técnica usada en computación gráfica para crear la ilusión de profundidad de color en imágenes con una paleta de colores limitada (reducción de color).

Eye Locker: Nombre de la aplicación que es solución del presente proyecto.

Hackear: Acción de explorar y buscar las limitantes de un código o de una máquina. Tambien se refiere a la acción de irrumpir o entrar de manera dorzada a un sistema de cómputo o a una red.

Histograma: En este contexto es una representación binaria de una variable en función de la frecuencia de los valores de color representados en cada una de las posiciones.

Inmutable: No es posible su cambio.

Intent: Se trata del elemento básico de comunicación entre los distintos componentes de Android. Son mensajes o peticiones enviados entre distintos componentes.

IrisBiometrics: Es el nombre de las bibliotecas usadas en el lenguaje C#.

Item: (Término informático) Descripción de un solo elemento en un dato archivado. Este tipo puede tener como datos texto, intent y Uri.

Listener: (Término informático) Sentencia o función que responde frente a un estímulo declarado.

ListView: (Término informático) Vista que muestra los elementos de una lista en un desplazamiento vertical.

Mat: (Término informático) Representa una matriz n-dimensional de un solo canal o varios. Se puede utilizar para almacenar vectores reales o valores complejos y matriciales, campos vectoriales, nubes de puntos, tensores, histogramas. Ha sido usado para implementar las funciones de bibliotecas en C++.

Mutable: Que es posible su cambio.

Null: (Término informático) Referencia a la nada. Resulta ser un valor especial aplicado a un puntero (o referencia) usado para indicar que el puntero no apunta a un objeto a dato válido.

Portrait: Nombre atribuido a la presentación vertical de una aplicación Android.

Root: Es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (mono o multi usuario). Normalmente esta es la cuenta de administración, también llamado superusuario, el cual puede cambiar permisos de archivos y enlazar a puertos de numeración pequeña.

Smartphones: Dispositivos móviles de última generación (Teléfonos inteligentes) que proporcionan un procesado y una administración de hardware y software similar a la de un ordenador.

Splash: (Término informático) Presentación inicial en una aplicación o programa, donde se suele disponer las productoras y desarrolladoras del software.

Sting: (Término informático) Secuencia inmutable de caracteres representados como array.

Switch: (Término informático) Instrucción de programación en la cual se introduce cuando un valor determinado es cambiado. Dentro de la instrucción se dirige a una parte del código u otra según se declare en cada caso (case).

Toast: (Término informático) Vista que contiene un fugaz y pequeño mensaje para el usuario. Esta clase es usada para notificar acciones momentáneamente.

Uri: (Término informático) Clase necesaria para obtener datos de otras actividades o procesos específicos. No realiza poca o ninguna validación. Esta clase es muy indulgente, devolverá basura (de datos, es decir, datos inservibles) antes de lanzar una excepción a no ser que se le especifique lo contrario.

Views: Son las unidades básicas de la interfaz de usuario. Los componentes como campos de texto, botones, etc.… Heredan de View.

Widgets: Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor (Widget engine).

1. Introducción

El presente proyecto describe el estudio y creación de una aplicación en un dispositivo móvil con plataforma Android. Dicho proyecto parte de uno anterior por lo que se precisó desde el principio disponer de la biblioteca de funciones desarrollada en dicho proyecto anterior, denominada IrisBiometrics. Dicha biblioteca fue desarrollada en lenguaje C# y venía acompañada por un proyecto Test ejecutable en Visual Studio 2010. La necesidad de esta biblioteca radica en usarla para observar si los resultados, tanto parciales como totales, son semejantes o no a los obtenidos en Android.

En este documento se presentaran todos los temas referentes a las bibliotecas obtenidas y su implementación en Android, así como las dificultades en su ejecución y modificación de las mismas para poder optimizarlo en dicha plataforma. A su vez se incluirá desde el entorno de aplicación de las bibliotecas con sus vistas (view en inglés) y el uso de los widgets hasta su protección contra errores del usuario para evitar fallos inesperados de la aplicación.

Nota para el lector: El problema de incluir ciertos elementos de Android es que son conocidos generalmente por su nombre en inglés y la mayoría no tienen una traducción lógica al español. Para evitar anglicismos se ha procedido a nombrar ciertos elementos en su traducción más cercana al español, pero otros elementos no han sido posibles debido al uso nulo o inexistencia de tal traducción. Para las palabras que no se sepa su significado, por favor, diríjase al diccionario.

1.1. Motivación

El principal motivo de realizar dicho proyecto era el deseo de aprender un nuevo lenguaje de programación (el lenguaje Java) y su implementación en la plataforma Android. A lo largo del proyecto se ha podido demostrar como con este lenguaje se es capaz de desarrollar aplicaciones complejas y de alto procesado hasta ahora accesible por otros lenguajes y sistemas.

Esta ilusión estaba acompañada de una idea fundamentada ya por millones de personas de que los dispositivos Android de última generación se ha convertido hoy en día en elementos indispensables para nuestro día a día, y esto es debido a la inmensa cantidad de aplicaciones y usos que pueden tener. Y es que pasamos de tener móviles que únicamente realizaban y recibían llamadas, a los denominados actualmente como smartphones los cuales disponen de mapas mundiales con conexión GPS, conexión 3G en todo el mundo para el acceso a internet, etc…. Estos dispositivos son como pequeños ordenadores de bolsillo. A continuación de los smartphones también aparecieron las tabletas (también conocidas por su nombre en inglés, tablets) las cuales, aunque no tengan un tamaño tan reducido y por tanto presenten menor movilidad, disponen de mayor procesado y mayor pantalla, por lo cual son más cómodas de visualizar que cualquier dispositivo móvil.

Hay que tener en cuenta que gracias a la política de desarrollo utilizada en plataforma Android podemos acceder a todas las prestaciones de nuestros smartphones de forma libre y gratuita, haciendo que muchos usuarios expertos o principiantes en la programación se adentren al mundo del desarrollo de aplicaciones para esta plataforma.

Durante la realización del proyecto, la mayor motivación encontrada era la inmensidad de aplicaciones que podría tener la aplicación desarrollada, debido a que cada día que pasamos con nuestro móvil incluimos más y más datos personales, como contraseñas y sincronización de paginas sociales y correos, contactos, documentos, etc…. Toda esta información queremos que este protegida, puesto que el acceso a toda esa información personal por parte de un usuario no deseado podría ser catastrófico. Es por eso que el autor de este documento cree que un sistema de identificación SEGURO, COMODO y RÁPIDO es necesario para estos dispositivos. Esta idea se ve afianzada por la decisión de, Google de incluir un sistema de identificación facial en la versión Ice Cream Sandwich de la plataforma Android.

Además los sistemas de identificación fiables abren un gran campo a otras aplicaciones y usos de los dispositivos móviles que ahora es difícil predecir.

En la realización de este proyecto se ha necesitado aplicar conocimientos adquiridos durante la formación universitaria, y se han presentado problemas los cuales cualquier ingeniero se ha de enfrentar durante su vida laboral.

Sin duda alguna, la motivación y la ilusión de desarrollar tal proyecto han sido claves para que al final se vea realizado y funcionando correctamente.

1.2. Objetivos

El objetivo principal de este proyecto es el estudio de la implementación de los algoritmos de identificación del iris en dispositivos Android y el estudio de su posible aplicación en otros campos de la protección de datos de la telefonía. Dicha aplicación ha de ser funcional en cualquier dispositivo, sin errores de compilación o de ejecución, configurable, autónomo frente otras aplicaciones (que no comparta datos con otras aplicaciones instaladas en el dispositivo) y con posibilidad de acceso a la memoria privada del dispositivo (memoria accesible únicamente desde la aplicación que crea dicha dirección de memoria).

Los objetivos finales obtenidos se comentan y desarrollan en el capítulo de conclusiones de este documento, donde se expone la experiencia personal, así como la comparación de la implementación obtenida en Android frente a la de C#.

1.3. Planificación y estructuración

Para la realización de este proyecto se han tenido en cuenta los siguientes puntos para su correcta realización y dentro de su correspondiente tiempo establecido:

- Redactar una introducción a los sistemas y entornos de desarrollo usados para el tribunal ya que puede que no estén familiarizados con esos conceptos.
- Se cuenta con un tiempo limitado para su realización y redacción.
- Desarrollar la aplicación para una versión de Android disponible para la mayor cantidad de dispositivos posibles.
- Se ha de tener en cuenta el tiempo necesario para la adquisición de información para la realización del proyecto, puesto que se presenta desde el principio por parte del alumno el no disponer de los conocimientos necesarios para poder trabajar en dicha plataforma.
- El proyecto ha de presentar en su interfaz con una vista sencilla y con mecanismos que indiquen al usuario como ha de proceder para el análisis del sujeto.
- En la documentación se han de tener en cuenta las dificultades, y se ha de explicar las modificaciones del código para su optimización aunque este funcione correctamente.
- Se ha de implementar una configuración sencilla e intuitiva para cualquier usuario así como una explicación de cada uno de los procesos utilizados para aquellos que no estén familiarizados con la identificación biométrica de iris.

2. Estado del arte

A continuación se van explicar los dos elementos que contribuyen a este proyecto, que es la telefonía móvil y la identificación biométrica. Además se hará una mención de la contribución de Android a la telefonía tanto a su desarrollo como a su aplicación de las nuevas prestaciones ofrecidas por los más recientes smartphones.

2.1. Evolución de la telefonía

Las tecnologías inalámbricas están teniendo un gran desarrollo en los últimos años, sobretodo la telefonía móvil y esto no es algo que sorprenda y es debido a que han revolucionado enormemente las actividades que realizamos diariamente. La telefonía móvil se ha convertido en una herramienta indispensable para gente de negocios, extendiéndose a sectores que hoy llegan incluso a la pre-adolescencia, es decir, niños menores de 12 años.

Pese a que en sus orígenes la telefonía móvil fue brindada únicamente con voz debido a las limitaciones tecnológicas de la época, se ha sido capaz de avanzar con el paso de los años a incluir cada vez más prestaciones y servicios a los usuarios. Esto hace que cada vez dependamos más de esta tecnología, haciendo que cuando no precisemos de dichas herramientas nos sintamos incomunicados e indefensos.

El primer dispositivo móvil se remonta a los inicios de la Segunda Guerra Mundial, en esos duros momentos se pudo ver que era necesario la comunicación a distancia. Es por eso que la compañía Motorola (empresa estadounidense especializada en la electrónica y las comunicaciones) creó el Handie Talkie H12-16. Éste era un equipo de comunicación por ondas de radio que les permitían a las tropas establecer contacto con la base con una banda de frecuencia que no superaba los 60MHz.

Comenzaron a perfeccionarlo debido al enorme potencial que se presagiaba a este tipo de dispositivos y en los años 1980 se creo un dispositivo que contenía las mismas prestaciones que el Handie Talkie pero iba destinado a empresarios que necesitaban estar comunicados en todo momento. Esto determinaba un hito en la historia pese a que los precios de los primeros dispositivos móviles de uso comercial eran prohibitivos para la época. Más adelante se fueron haciendo más accesibles al público hasta hoy en día que cualquier niño tiene uno.

2.1.1. Generación de la telefonía

La evolución de la telefonía se puede observar en una clasificación simple según la conexión móvil, puesto que este factor es más general que los elementos adicionales de los que puede disponer un móvil. Esta conexión se clasifica en generaciones, de la 0 a la actual que es la 4.

0G – Generación 0: Esta es la perteneciente a los primeros sistemas de telefonía móvil civil que se desarrollaron por primera vez a principios de los años 40 en Estados Unidos, conocidos como telefonía de radio móvil. Esto dispositivos se conectaban por ondas de radio que estaban moduladas en AM aunque posteriormente se modulaban en FM. Este cambio de modulación es debido a que la modulación FM ofrecía una calidad de audio notablemente superior frente a AM, además no resultaba tan sensible a las interferencias. Estos primeros dispositivos eran grandes y pesados, lo que hizo que tuvieran que ser instalados en vehículos para su transporte. No era un servicio muy extendido puesto que era muy caro. Se uso esta comunicación del 1946 al 1985 donde se vio finalmente desbancada por la siguiente generación de conexiones móviles.

1G – Primera generación: Esta generación da comienzo entre el 1979 y el 1981 de la mano del fabricante Ericsson con su dispositivo NMT 450. Pese a que siguiera usando canales de radio analógicos en modulación FM, estos alcanzaban frecuencias de 450MHz. Se siguió intentando aumentar la frecuencia de datos, hasta que en 1986 Ericsson consiguió aumentarla a 900MHz. Esta solución, en su formato conocido como sistema TACS, fue la que llego a España con el nombre de MoviLine que duró en el mercado hasta el 2003. De todas formas, el sistema que habría que destacar en esta generación es el AMPS en Estados Unidos, puesto que permitía la creación de canales de frecuencia para hacer posible el aumento de usuarios en una red. En cualquiera de los casos, esta comunicación carecía de seguridad y la transferencia de celdas era muy imprecisa.

2G – Segunda generación: Se da comienzo en la década de los 90 utilizando sistemas como GSM, IS-136 o IS-95. Dependiendo del país se usaron unas frecuencias u otras, en el caso de Europa se utilizaban dos bandas de frecuencias. La de 900 y la de 1800MHz. La principal diferencia frente a las demás comunicaciones es que ésta ofrecía una conexión digital. Las comunicaciones digitales ofrecen una mejor calidad de voz y mayor seguridad gracias a la posibilidad de introducir mecanismos de cifrado. Esta seguridad se aplica a distintos niveles para diferentes sistemas. Se aplican distintos estándares de comunicación en diferentes regiones como es la AMPS en Estados Unidos, PDC en Japón, IS-95 en Estados Unidos y Asia y GSM en el resto del mundo. Al final el más reconocido fue el GSM y fue aplicado en Europa debido a su buena calidad de voz, deseo de implantación internacional, ser compatible con RDSI y, lo más importante, para triunfar en el mercado con terminales a buen precio y de peso reducido. Esta comunicación además proporcionaba servicios auxiliares tales como datos, fax y SMS.

2.5G – Generación de transición: Muchos proveedores acceden a esta generación como transición al 3G aunque países como Japón pasan directamente de la 2G a la 3G. Se presentan sistemas de comunicación como el GPRS, HSCSD, EDGE, IS-136B y IS-95B entre otros. El motivo es que estos sistemas son más rápidos que los de la 2G pero mucho más baratos que la tecnología 3G. Como es obvio, ofrecen servicios que extienden las características de los dispositivos de 2G como es el uso de MMS y el de EMS. Para ello era necesario usar sistemas de alta transferencia de datos como la que ofrecía el GPRS con de 56 a 114Kbps y el HSCSD hasta 384Kbps.

3G – Tercera generación: Esta generación es demandada para poder usar servicios que requieren una transferencia de datos superior como era el uso de la televisión, videoconferencia y la descarga de archivos con el acceso inalámbrico a internet. El sistema de comunicación más conocido en esta generación es el UMTS, que alcanza velocidades de 144 Kbps hasta 7.2Mbps. Pero este sistema no terminó de triunfar comercialmente, puesto que la mayoría de los usuarios tenían suficiente con la transmisión de voz y la transferencia de datos ofrecida por GPRS y EDGE. Sin embargo plantó las bases para una mayor demanda, como es la que se tiene en la actualidad y que ha dado lugar a la siguiente generación.

4G – Cuarta generación: Se llega a la generación más actual. El aumento de banda de datos es usado únicamente para la recepción de televisión en alta definición. Hay cambios como el uso de SDR para optimizar el acceso radio, red prevista en IP. Las velocidades de acceso son superiores a 100Mbps en movimiento y 1Gbps en reposo. Lo más reconocible de esta generación es la compatibilidad de sistemas de generaciones anteriores, que hacen que los usuarios ahora utilicen indistintamente distintos sistemas, adaptándose a las capacidades de la red en cada momento y situación.

2.2. La identificación biométrica por Iris

La biometría informática está en pleno desarrollo e implantación. Esto de da por la necesidad de gobiernos y corporaciones de disponer de sistemas de seguridad más seguros y precisos. Aunque hoy en día es una tecnología que no ha alcanzado su madurez, son muchas las aplicaciones que ofrece desde un punto de vista industrial, como por ejemplo, el control de acceso a instalaciones.

En este apartado se hablará de los conceptos más importantes de la Biometría y se acabará profundizando en la identificación por iris.

2.2.1. ¿Qué es la biometría?

Básicamente la biométrica es la ciencia que se dedica al estudio estadístico de las características biológicas o de comportamiento de los seres vivos, mediante el uso de diferentes métodos. Hoy en día este término se ha adaptado a las tecnologías de la información, para referirse a los métodos automáticos para reconocer a personas a partir de uno o más rasgos físicos intrínsecos o rasgos conductuales con el fin de identificar o autenticar a las personas.

Con esta implementación en seguridad pasamos a añadir un nuevo concepto, ya que el usuario puede acceder a un sistema protegido mediante algo que sabe como una contraseña, algo que el usuario tiene como puede ser una tarjeta personal o llave de acceso, pero gracias a la biometría puede mediante algo que el usuario es.

Para que un sistema de reconocimiento biométrico sea clasificado como tal, ha de cumplir una serie de requisitos básicos.

- ✓ Todas las personas tienen que presentar dicha característica (Universalidad).
- ✓ Dos sujetos deben de ser distinguidos lo suficiente el uno del otro sobre la misma característica (Unicidad). Por ejemplo, no valdría medir el color de piel.
- ✓ Dicha característica tiene que mantenerse constante con el paso del tiempo (Estabilidad). No valdría por ejemplo el corte de pelo.
- ✓ Mantener constante la característica en condiciones ambientales diversas (Estabilidad).
- ✓ La característica tiene que ser posible medirse.
- ✓ Tiene que ser aceptado por el sujeto, sistemas de medida dolorosos o de gran esfuerzo por parte del sujeto serán rechazados (Aceptabilidad y Usabilidad).
- ✓ El porcentaje de reconocimiento de sujeto válido ha de ser lo suficiente alto (Rendimiento).
- ✓ El sistema ha de ser capaz de resistir a técnicas de fraude, aunque en algunos casos se pueda burlar (Robustez y Seguridad).

Hay que entender que hoy en día disponemos de dos tipos de biometría:

- ➢ Estática: Estudio del conjunto de características físicas. Estas son (entre otras):
  - o Huella dactilar.
  - o Características del ojo (retina e iris).
  - o Líneas de la mano.
  - o Geometría de la mano.
  - o Poros de la piel.
  - o Características no expresivas de la cara.
  - o Composición química del olor corporal.

- Emisión térmica
- Venas de muñecas y manos.

➢ Dinámica: Estudio del conjunto de características conductuales. Estas son denominadas de esta manera puesto que son las características que se van definiendo según la edad y pueden variar con los años incluso después de la edad adulta. Estas son:

- Voz. En este caso, se trata de una modalidad que es híbrida entre estática y dinámica.
- Tecleo.
- Escritura manuscrita.
- Gesto y movimiento corporal.

El inconveniente principal de las dinámicas frente a las estáticas es que hay que ir renovando la base de datos cada poco tiempo, para evitar que un día al comparar las características obtenidas de una muestra capturada del sujeto con la de la base de datos sea tan distinta que el sistema clasifique de intruso al usuario.

Cualquier sistema biométrico debe precisar de un mecanismo de captura y procesado para obtener las características, una base de datos lo suficientemente grande como para almacenar los registros necesarios de datos de los sujetos y finalmente un procedimiento para comparar las características tomadas del sujeto con la captura con la/las que se encuentra/encuentran en la base de datos. En la comparación hay dos tipos:

- Verificación: Cuando el sujeto ha seleccionado de la base de datos las características con las cuales se va a comparar (Este es el caso de la aplicación desarrollada). Como resultado del proceso se indicará si el sujeto es o no es quien afirma ser (Ver **figura 1**).
- Identificación: Cuando la captura del sujeto es comparada con toda la base de datos disponible. Como resultado el sistema dará a conocer la persona que se asemeja con las características en la base de datos si es que hay alguno que da con el umbral de aceptación adecuado.

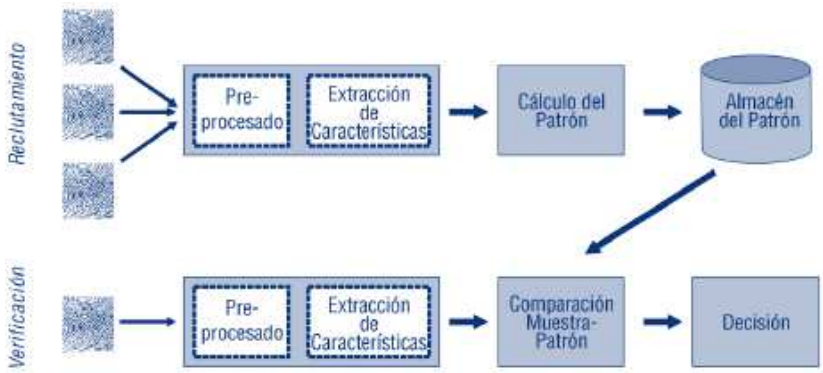

Figura 1: Procesos de verificación [30]

2.2.2. Historia

La biometría tiene sus orígenes en las culturas occidentales específicamente en China donde se remonta en el siglo VIII. En esta fecha se encuentran huellas dactilares tanto en documentos como en esculturas de arcilla. Los comerciantes chinos usaban estampaciones de tinta en papel de las huellas de la palma de las manos para identificar entre los niños jóvenes.

Sin embargo, el primer estudio científico relacionado con la biometría se realizó en el 1686 por Marcelo Malpighi, el llamado padre de la histología, aunque no se realizo con intención de la identificación individual.

Hubo que esperar hasta 1856 cuando Sir William Herschel implantara la huella del pulgar como método de identificación de documentos para personas analfabetas. Poco después en el año 1883 fue creado por Alphonse Bertillon , jefe del departamento de policía de París, un sistema de identificación antropométrico que más adelante tomó el nombre de Bertillonage. Este fue el primer sistema de la época que era preciso y que fue usado científicamente para reconocer a criminales convirtiendo así a la biometría como un campo de estudio. Funcionaba midiendo ciertas longitudes como anchura de la cabeza, distancia de los ojos, etc…. Este sistema fue desbancado más adelante por la identificación por huellas dactilares, haciendo que se volviera después de cientos de años al mismo sistema que se remontaba en China, pero esta vez con una base científica por debajo.

En los últimos años la biometría ha ido buscando nuevos métodos de identificación de personas teniendo en cuenta medidas física o de comportamiento como se especifico anteriormente. Además estos sistemas han ido evolucionando en prestaciones pero siempre con el objetivo de aumentar la seguridad.

El origen de cada uno de los distintos sistemas es obviamente diferente, pero se comentará brevemente el que concierne a este proyecto. La identificación por iris fue propuesta inicialmente por el oftalmólogo Frank Burch en 1936 pero fue olvidada hasta que en 1985 los doctores Leonard Flom y Aran Safir retomaron la idea. En 1987 consiguieron mediante su documentación e investigación patentar el concepto de Burch. Puesto que no disponían de los conocimientos suficientes para desarrollar un algoritmo se pusieron en contacto con un profesor de la universidad de Harvard llamado John G. Daugman, el cual diseñó y desarrolló un algoritmo para poder reconocer biométricamente los patrones del iris. Fue patentado en 1994 y se ha convertido en la base de todos los sistemas de reconocimiento de iris que existen. A partir de entonces la empresa creada por Flom, Safir y Daugman llamada IriScan Corp se encargó de dar licencias a otras empresas.

Por tanto como se puede ver en el caso del reconocimiento de personas por iris, se trata de una tecnología muy nueva pero que tiene un enorme potencial.

2.2.3. Implementación de la identificación biométrica en la sociedad

Este suele ser el obstáculo más difícil de saltar de cualquier sistema de identificación biométrica y esto es debido a que por naturaleza humana el sujeto es reacio a todo lo que no está familiarizado. Por tanto prefiere un sistema típico de identificación como puede ser el de usuario y contraseña, antes que cualquiera de los sistemas mencionados anteriormente, sólo por el mero hecho que es a lo que está acostumbrado en su vida cotidiana.

Por tanto como se ha podido ver por su seguridad, estos sistemas constituyen un sistema de protección mucho más avanzado pero ha de integrarse poco a poco en la sociedad.

La implementación de esta identificación se hace más forzosa con los tópicos que se forman en la sociedad.

Durante innumerables películas se ha podido observar la identificación biométrica como parte de la seguridad de grandes empresas, y ésta es, en muchos casos, la basada en el reconocimiento de alguna parte del ojo, como por ejemplo el iris. De hecho en películas de James Bond en los 80s se podían ver ya dichas medidas de seguridad incluso antes de que se crearan los algoritmos y por tanto la posibilidad de poder reconocerlo.

La industria cinematográfica afecta en gran medida en la implementación de los sistemas biométricos, pero esto puede ser positiva o negativamente. Según se ha comprobado, las personas jóvenes son menos reacias a probar estos sistemas que las personas mayores que piensan que estas soluciones pueden dañar sus ojos. La idea principal de que pueda dañarlos es por lo visto en películas como “Misión imposible” donde se ve una vistosa luz roja en forma de línea que escanea tu ojo para la obtención de la imagen. Esto es completamente falso, puesto que el sistema utilizado para la obtención de la imagen es una cámara de alta resolución, y para poder observar el ojo sin reflejos producido por una luz exterior se usa luz infrarroja que se encuentra fuera del espectro de visión humana. De hecho estas cámaras usan una cantidad de luz infrarroja ínfima en comparación a la que nosotros recibimos por el Sol.

Otro tópico que induce miedo en la sociedad por las películas es la del método drástico que hay que tomar para robar la información. Según se ha visto en “Demolition Man” o en “Ángeles y demonios” para poder entrar en una cámara protegida con un sistema de identificación por iris, el ladrón le saca el ojo y lo pone en el sistema para ser reconocido, generalmente pinchado en un elemento punzante. Esto es completamente imposible por muchos motivos. Por ejemplo, cuando se saca el ojo de la cavidad ocular mediante esos métodos, el globo ocular pierde rápidamente sus propiedades, e incluso explota al pincharlo, ya que es como un globo de humor vítreo (en el siguiente apartado se explican estos conceptos). Además, los sistemas más modernos de identificación por iris tienen numerosos controles de calidad, entre los que se encuentran los de identificación de sujeto vivo. Básicamente comprueba con una pequeña luz que el ojo

responde a este estímulo (abriendo y cerrando la pupila) haciendo así que se comprueba si el sujeto está vivo o no.

2.2.4. Anatomía básica del ojo

Para entender y saber algunas de las partes en las que se constituyen el ojo se hará a continuación una breve mención a su anatomía, al igual que se introduce un breve esquema para poder seguir de forma más clara la explicación.

Los ojos son los órganos que brindan al ser humano el sentido de la vista. Para poder facilitar una visión estereoscópica el cuerpo humano consta de dos ojos. El ojo humano es capaz de ver luz con longitud de onda comprendida entre 380 y 750nm. La cavidad esférica esta recubierta por tres capas (externa, media e interna) y está dividida en tres cámaras (anterior, posterior y vítrea) (ver **figura 2**).

La parte del globo ocular que se encuentra en contacto con el exterior, córnea y parte de la esclerótica, se encuentra protegido por los párpados y por las segregaciones que aportan las glándulas lagrimales.

La esclerótica es la capa exterior, que como se puede ver en la **figura 2**, es la zona blanca del ojo. La esclerótica se une con la córnea, la cual es la capa que comunica ópticamente el exterior con el interior del globo ocular, proporcionando una protección frente a elementos externos pero a su vez deja pasar la luz para poder captar imágenes. Esta lente proporciona el mayor poder refractivo dentro del ojo.

La parte media es la úvea y está formada hacia la parte más cercana a la cornea por el cuerpo ciliar y el iris. Para poder colocar el cristalino y mantenerlo estable es necesario que funcione la parte muscular del cuerpo ciliar.

Por otro lado el iris consta de un estroma con células pigmentadas y de un epitelio, que es el que actúa como diafragma ocular gracias a los músculos esfínter y dilatador del iris. El iris forma la barrera entre la cámara anterior y posterior del glóbulo ocular.

En la parte central del ojo se encuentra la pupila, que es la apertura y constituye una barrera entre las cámaras anterior y posterior del glóbulo ocular, encontrándose entre la córnea y el cristalino.

En el interior del globo ocular se encuentra el cristalino, que es una lente biconvexa transparente, avascular y carente de nervios y sirve de frontera entre la cámara vítrea y las cámaras anterior y posterior.

En el interior de la cámara vítrea, que se encuentra entre la superficie interna de la retina y la cara posterior del cristalino, se encuentra el humor vítreo, que es un líquido gelatinoso y transparente que rellena dicho espacio.

En la cámara anterior se encuentra el humor acuoso, que un líquido transparente que sirve para nutrir y oxigenar las estructuras del globo ocular que no tienen aporte sanguíneo como la córnea y el cristalino.

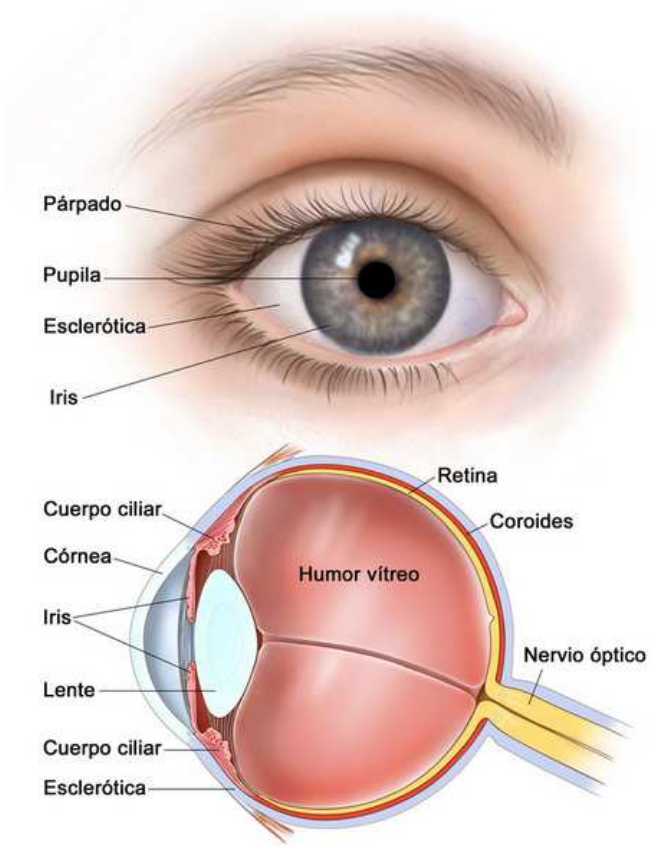

Figura 2: Anatomía del ojo humano [18]

2.2.5. Ventajas y desventajas

Como se comentó anteriormente cuando se explicaba la biometría, la principal diferencia de esta identificación frente a otras es que no es algo que sabes como una contraseña o que portas contigo como una llave de seguridad, sino que es algo que eres, lo que hace que te identifique como usuario apto. Esto hace que la seguridad, ya de por sí, aumente drásticamente, y esto es debido a que principalmente no te pueden robar esa "llave biométrica" para usarla en su beneficio, mientras que una contraseña puede ser vista por otro individuo o una llave puede ser robada.

En comparación con las contraseñas, muchos usuarios tienen la mala costumbre de poner contraseñas muy fáciles de descifrar como fechas de nacimiento, nombre de familiares como hijos, etc. Pero si el usuario ha optado por una contraseña con caracteres sin sentido números y mayúsculas, aunque sea muy seguro tiene el

inconveniente de que recordar esa contraseña será muy incómodo para la mayoría de los usuarios, haciendo que al final sea cambiada por otra más sencilla de recordar. Además los sistemas de contraseñas son los más fácilmente atacables mediante fuerza bruta, es decir, mediante el uso de algoritmos que generan contraseñas automáticamente probando todas las combinaciones posibles de letras, números y signos.

Estos sistemas tienen una ventaja muy importante y es que no se pude robar la información biométrica de los usuarios hackeando el sistema y accediendo al almacenamiento de dicho datos, puesto que estas características están registradas con una implementación diferente de cada empresa, aunque el algoritmo sea el mismo que es el de Daugman, es decir, el hacker tendría un montón de unos y ceros que no sabría que corresponde a la mascara y cual a la imagen de la captura, además de otros datos como nombre del individuo tamaño del ojo, etc…(o lo que la empresa creo oportuno almacenar). Incluso se recomienda incluir una codificación adicional para la seguridad del individuo, así como proteger todos los mecanismos de almacenamiento y transmisión de la información biométrica.

En ciertos sistemas ofrece una comodidad al usuario que llega a ser mejor que el de recordar una contraseña o el de portar con el una llave, por ejemplo, el reconocimiento fácil con el cual con solo situarse enfrente del sistema lo reconoce al momento.

A continuación se enumeran en la **tabla 1** las ventajas e inconvenientes de los diferentes sistemas de identificación biométricos.

Técnica	Ventajas	Inconvenientes
Voz	-Muy bajo coste.	-Bajo rendimiento. -Poca unicidad y estabilidad.
Huella	-Muy estudiado y desarrollado. -Alto rendimiento. -Estabilidad y unicidad. -Reconocimiento legal. -Medio coste.	-Connotaciones policiales para el usuario. -Imposibilidad de detección de dedo vivo.
Iris	-Unicidad mayor que huella. -Gran estabilidad. -Fácil detección de ojo vivo.	-Alto coste. -Inicialmente incómodo para el usuario.
Mano	-Medio coste. -Sin connotación policial. -Fácil uso y gran aceptación por el usuario.	-Unicidad y estabilidad no probadas. -Imposibilidad de detección de mano viva.
Rostro	-Medio coste. -Cómodo e incluso inapreciable para el usuario.	-Sensible a cambios del sujeto. -Imposibilidad de detección viva. -Si es captura en 2D es fácilmente engañada.

Tabla 1: Distinciones entre sistemas biométricos [25]

A su vez también hay que tener en cuenta que dependiendo de las condiciones en las que nos encontremos de ambiente y de tiempo transcurrido desde la recogida de la última muestra hay sistemas que ofrecen más dificultades que otras como se puede observar la **tabla 2**.

Técnica	Factores causan errores	Posibles soluciones
Todas	-Envejecimiento.	-Darse de alta periódicamente.
Voz	-Enfermedad.	-Darse de alta periódicamente.
Huella	-Trabajos que fuercen los dedos y afecten a las huellas (carpintería). -Lesiones, cortes o amputaciones.	-Dar de alta varios dedos.
Iris	-Posición del ojo. -Uso de gafas.	-Facilitar posición adecuada (mediante apoyos).
Mano	-Heridas.	-Darse de alta ambas manos.
Rostro	-Iluminación. -Calidad del contraste. -Fondos.	-Entornos controlados.

Tabla 2: Posibles problemas biométricos [25]

Obviamente hay medidas para solucionar estos problemas que son generales, puesto que dependiendo de las situaciones reales puede que lleguen a ser únicas y no se podrán solucionar de la misma manera.

La mayoría de los métodos de medida contienen errores que se pueden solucionar de la misma manera, aunque hay situaciones en la que se pueden encontrar, ya sean de entorno o de situación del usuario, que condicionan a que los errores formados sean únicos y que obviamente sea necesario un estudio propio para ese fallo.

Hay que saber que un sistema nunca es infalible y que cualquier sistema es apto e incluso ideal para ciertas condiciones. Por tanto, no existe la identificación biométrica por excelencia y mucho menos el definitivo, puesto que dependiendo de la situación en la que se vaya a encontrar nuestro sistema será mejor elegir uno u otro.

2.2.6. Procesos básicos para la identificación por iris

Básicamente cualquier sistema de identificación biométrica consta de las mismas etapas principales que son:

- Captura de los datos biológicos ya sea de comportamiento o físicos.
- Pre-procesado de los datos capturados, es decir, preparar la imagen para el siguiente paso.
- Extracción de características que identifiquen unívocamente al individuo.

- Comparación de las características obtenidas con el patrón previamente almacenado.

A continuación se hará una breve descripción a cada una de estas partes explicando en el caso de la identificación por iris.

Captura: Puesto que la imagen del iris es accesible desde el exterior para su captura se pueden plantear dos posibilidades, el uso de cámara digital o el uso de cámaras de video. Se pueden usar cámaras con luz infrarroja, con lo que las imágenes ya son capturadas en blanco y negro.

Para el sistema de captura hay que tener en cuenta los siguientes puntos:

- Se debe de precisar de una cámara que tenga una gran resolución.
- Centrar la captura a la característica a medir, es decir, en este caso que salga únicamente el ojo y nada más, concentrando de esta manera toda la calidad de la imagen en la captura de lo que nos interesa.
- No debe de encontrarse la cámara demasiado cerca del sujeto a medir, puesto que esto le produciría una situación incomoda y sentirse amenazado.
- La adquisición a la distancia elegida no debe de suponer una deformación de la imagen capturada.

Una vez elegida la cámara, hay que elegir el entorno de captura. Básicamente se trata de elegir un emplazamiento el cual ofrezca una iluminación lo suficientemente buena como para asegurar que el dispositivo de captura obtenga la imagen con la mejor calidad posible.

Este paso es algo complejo debido a que la cornea tiene una gran capacidad de reflexión apuesto a que es una superficie lisa y bien lubricada. Por tanto una simple luz apuntando al iris haría que ésta se refleje y no se captaría con claridad por la cámara.

Para solucionar este problema se puede utilizar focos en la parte inferior de la cámara, de esta manera se ilumina el cono inferior del cono del iris y gracias a un polarizador, colocado en cuadratura con el del emisor de luz, se elimina de la imagen el reflejo obtenido.

Pero el inconveniente es que el uso de dicho filtro polarizador resta luminosidad y provoca pérdida de sensibilidad en la cámara y por tanto se obtiene una captura de peor calidad. Se podría solucionar aumentando la luminosidad del foco pero puede que esto provoque molestias en el usuario y llegar al rechazo por parte del sujeto.

Es por esto que la mejor opción y la que se ha considerado como válida es la de una luz infrarroja, ya que esta no produce reflejos que puedan impedir la lectura completa del iris.

Además, el uso de luz infrarroja elimina la información de color del iris, permaneciendo sólo la información sobre la textura. Esto permite que el sistema sea inmune a los cambios temporales de pigmentación que pueda presentar el iris.

Pre-procesado: Esta etapa tiene una gran importancia puesto que es necesario adaptar la captura a los requisitos de extracción de características y para ello es necesario hacer una serie de procesos:

- La detección del borde externo del iris, es decir, la frontera con la esclerótica.
- La detección del borde interno del iris, o lo que es lo mismo, el límite con la pupila.
- Eliminación de las partes de la imagen no deseadas, es decir, todo lo que no este dentro del iris.
- Adaptar la imagen a la técnica de extracción de características a realizar, es decir, recoger los datos y disponerlos según un vector (conocido como array en inglés), una matriz, etc.…

A continuación se explicará la detección de ambos bordes. La eliminación de las partes no deseadas se puede realizar simplemente leyendo lo que esta dentro de esos dos márgenes y despreciando lo demás. También hay procedimientos como el que se ha utilizado en el código de la solución propuesta en este proyecto (llamado Eye Locker) que limpia ligeramente los bordes de la captura para así facilitar el trabajo a los procesos de detección de bordes. El punto de adaptación de la imagen tampoco se comentará puesto que esto depende de la implementación del algoritmo que se haya realizado. En la explicación de la aplicación desarrollada se entrará en más detalle.

Detección del borde exterior del iris: Este es el primer paso en la detección del iris y es debido al cambio brusco de contrate entre el iris y la esclerótica, que es blanca. Antes de realizar esta detección hay que asegurarse que la imagen se encuentra en blanco y negro para su mejor procesado. Una vez cargada la imagen en escala de grises se procede a ejecutar los siguientes pasos en dicho orden:

- Se crea y se copia de la imagen obtenida cuatro veces menor para mejorar su procesado (cuando se quiera obtener la detección fina se procede a realizar la búsqueda sobre la imagen grande original).
- Se eliminan las partes de la imagen sobre-expuestas debido a puntos de luz producidos por un flash, por ejemplo. Esta eliminación se realiza mediante un umbral establecido.
- Se buscan candidatos a ser centros mediante búsqueda de puntos negros agrupados. Esto se establece mediante una cuadrícula en cuyos puntos se situarán los centros que se utilizarán en el algoritmo de detección de bordes, el punto que determine el mejor borde será tomado como el centro del iris.
- Por cada uno de los puntos de la cuadrícula, que son los posibles centros, se toma uno a uno como origen de coordenadas y a partir de él se va

incrementando el radio (Δ_r) y el ángulo (Δ_θ), y buscando el múltiplo n de Δ_r que maximiza el parámetro D en la siguiente ecuación:

$$D = \sum_{m} \sum_{k=1}^{5} (I_{n,m} - I_{(n-k),m})$$

$$I_{i,j} = I(x_0 + i \cdot \Delta_r \cdot \cos(j \cdot \Delta_\theta), y_0 + i \cdot \Delta_r \cdot \sin(j \cdot \Delta_\theta))$$

Donde m son aquellos múltiplos de Δ_θ que corresponden a puntos en conos laterales del iris e I son los valores de intensidad de la imagen.

Simplificando, el sistema va incrementando en radio y ángulo hasta detectar un cambio de contraste brusco.

- Una vez que se ha encontrado el punto de la cuadrícula que proporciona el máximo D, se crea una nueva cuadrícula, con mayor resolución y que solo abarca el cuadrado formado por los puntos comprendidos entre los dos extremos del punto elegido, tanto en el eje horizontal como en el vertical.
- Se vuelve a realizar el mismo proceso de detección por cuadrícula pero sobre la imagen de gran resolución con el punto elegido y así se obtiene el centro de forma más refinado.
- Una vez localizado correctamente el centro, se disminuye el factor de Δ_r para tener una mayor precisión a la hora de sacar el borde, y se determina la distancia del centro al borde mediante $n \cdot \Delta_r$ siendo n el valor que da el máximo valor a D.

Se podría preguntar si es necesario la reducción de la imagen al principio puesto que después se realiza la misma comprobación sobre la imagen de gran resolución una vez se ha detectado el centro, pero la cuestión es que la primera vez que se pasa por la comprobación de bordes con los posibles candidatos de centros puede que tenga 10 candidatos por lo tanto tendrá que pasar 10 veces por ese paso. Por tanto la mayor reducción de resolución es necesaria, para así reducir al máximo el procesado en esa primera eliminación de candidatos.

Los puntos de muestreo para esta identificación de bordes se encuentran en los dos cuartos del iris pertenecientes a los lados derecho e izquierdo, esto es para evitar las partes del iris que están superpuestas por los párpados (o por los focos de luz que se comentaron anteriormente) y por tanto no deje producirse una buena lectura. Se puede observar con más comodidad en la siguiente figura.

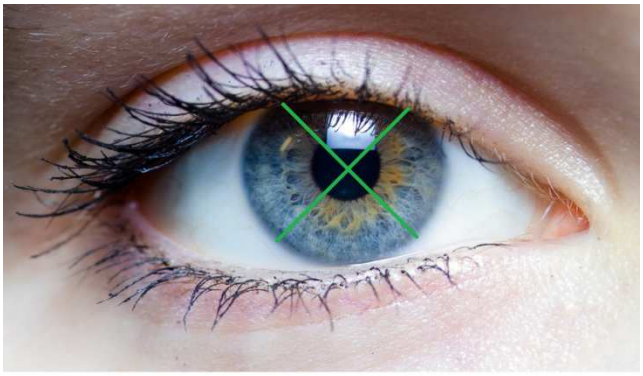

Figura 3: Sección a analizar del ojo

Como se puede observar los cuartos del iris que se encuentra arriba y abajo, se encuentran obstaculizados por el párpado o por un reflejo de luz (esto último es haciendo una iluminación por foco con filtro polarizado), es por eso que es necesario filtrar la imagen obteniendo los datos que se encuentren únicamente en zonas donde siempre sean accesibles.

Finalmente una vez detectado el borde exterior, se aplican las transformaciones oportunas para obtener el valor del centro y del radio en la imagen original, esto es tan simple como multiplicar dichos valores por el valor de la reducción de la imagen aplicada en un principio, en este caso 4.

Detección del borde interior del iris: Este proceso es similar al anterior proceso con la diferencia de que el sistema de detección ha de ser más sensible puesto que en el campo infrarrojo el primer cambio brusco se ocasiona entre pupila e iris. Por esa razón se detecta primero la pupila y luego el iris. No se puede utilizar el mismo centro puesto que la pupila y el iris no son concéntricos aunque a simple vista lo parezca, suele estar desplazada ligeramente hacia abajo y hacia la nariz. En algunos casos, esta desviación llega a ser del 15%. Por tanto no se realizará el mismo procedimiento de nuevo, basta con crear una cuadrícula alrededor del centro del iris que abarque el ±20% del tamaño del iris. En este punto se vuelve aplicar el mismo sistema que el anterior pero únicamente solo con la cuadrícula determinada anteriormente.

Suponemos que procesamos la misma imagen que la puesta en la **figura 3**, puesto que la superposición de los párpados no afectaría, ya que si es así no se vería prácticamente nada del iris, se han tomado puntos no solamente en los conos laterales como se hizo anteriormente sino que se puede tomar información del inferior y del superior. En este caso no sería recomendable coger del superior debido a la interferencia de la luz en su correcta visión completa del iris.

Como resultado dará un centro de la pupila similar al del iris y un radio menor que el del iris. Una vez realizado ambos procesos se realiza la eliminación de la información relevante que es todo lo que no se encuentre entre el borde interior y

exterior del iris. Se realiza un estiramiento del histograma, con lo que se obtiene el iris aislado del resto de la imagen.

Extracción de características de la captura: Hay varios métodos para la extracción de características, pero como se puede apreciar en el título del proyecto, se comentará el utilizado, mediante filtrado de Gabor.

Este tipo de extracción está basada en los trabajos de Daugman, que usa filtros de Gabor. Para poder usar estos filtros es necesario adaptar la imagen al algoritmo utilizado.

Adaptación del iris detectado: Una vez que tengamos el iris aislado, es decir, sin parpados o esclerótica, hay que considerar las variaciones del tamaño del iris por dilatación o contracción.

Los datos que hay que pasar para poder manipularlos con libertad han de tener dos condiciones indispensables que se impondrán independientemente del tamaño del iris:

- Estén suprimidos los conos superior e inferior.
- El tamaño de los datos sea el mismo siempre.

Para ello se realiza un muestreo de radio y de ángulo de la imagen del iris según el sistema de ecuaciones siguiente:

$$J(x,y) = I \cdot E(x_p + r \cdot cos\theta, y_p + r \cdot sin\theta)$$

$$r = r_i + (x-1)\cdot \Delta_r \qquad \forall x \in N: x \leq \frac{r_e - r_i}{\Delta_r}$$

$$si\ y \leq \frac{\pi}{2\cdot\Delta_\theta} \quad \theta = \frac{-\pi}{4} + (y-1)\cdot\Delta_\theta \qquad \forall x \in N: y \leq \frac{\pi}{\Delta_\theta}$$

$$si\ y > \frac{\pi}{2\cdot\Delta_\theta} \quad \theta = \frac{3\pi}{4} + (y-1)\cdot\Delta_\theta \qquad \forall x \in N: y \leq \frac{\pi}{\Delta_\theta}$$

De donde:

J(x, y) es la nueva imagen.

I·E(a, b) es la imagen resultante de los pasos anteriores.

El centro de la pupila está en (x_p, y_p)

r_e, r_i, Δ_r, Δ_θ y N son respectivamente radio exterior, radio interior, incremento del radio, incremento del ángulo y el conjunto de números naturales.

Para poder entenderlo mejor, se puede observar la **figura 4** donde se observa la transformación mencionada anteriormente. Estas figuras son tomadas directamente de la aplicación.

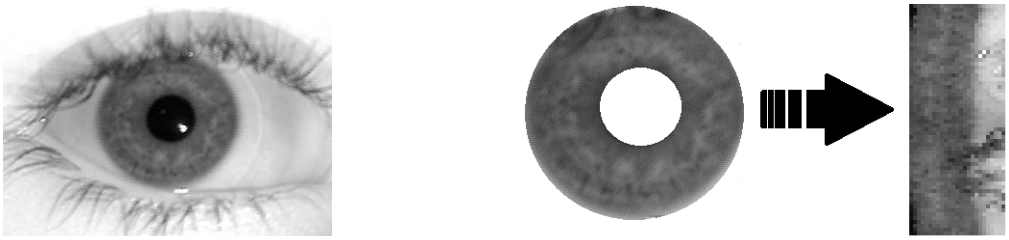

Figura 4: Extracción del iris

Como se puede ver se captura la imagen, se detecta el borde interior y exterior, posteriormente se consigue mediante el método descrito la matriz resultante, siempre del mismo tamaño. Es siempre del mismo tamaño puesto que siempre se selecciona el mismo número de puntos.

Por tanto, en este punto ya está lista la imagen para poder aplicar los filtros de Gabor que lograrán extraer las características necesarias.

Algoritmo de extracción de características: Una vez que se ha obtenido la matriz rectangular de datos de la captura de iris, se procede a sacan las características.

Para ello se aplica el filtro de Gabor para la extracción de características determinada por la siguiente ecuación:

$$g(x,y,\varphi_k,\lambda) = \exp\left\{-\frac{1}{2}\cdot\left[\frac{(x\cdot cos\varphi_k + y\cdot sin\varphi_k)^2}{\sigma_x^2} + \frac{(-x\cdot sin\varphi_k + y\cdot cos\varphi_k)^2}{\sigma_y^2}\right]\right\} \cdot exp\left\{\frac{2\pi(x\cdot cos\varphi_k + y\cdot sin\varphi_k)}{\lambda}\right\}$$

Donde:

φ_k, λ, σ_x, σ_y son respectivamente la orientación, la escala y los parámetros de dispersión de la envolvente del filtro para las coordenadas x e y.

Puesto que la idea inicial es la extracción de características, lo que estamos buscando mediante este filtro es la variación de contraste de la imagen del iris para así saber cuando cambia, es decir, cuando hay más contraste o menos dentro del iris denominando así su forma. Por tanto lo que se hace es ponderar segmentos de la imagen por los valores del filtro de Gabor. En concreto, como la parte imaginaria de un filtro de Gabor tiene media nula, se puede estudiar la variabilidad de la intensidad de zonas localizadas de una imagen, extrayendo características de dicha variabilidad. Los coeficientes obtenidos son:

$$C(i,j) = \sum_{x=1}^{N}\sum_{y=1}^{M} J\left(i+x-\frac{N}{2}, j+y-\frac{M}{2}\right)\cdot Im[g(x,y,\varphi_k,\lambda)]$$

Donde:

(i, j) son puntos en la imagen J elegida para obtener los coeficientes.

N y M son el tamaño del filtro g elegido

Por tanto, la imagen J se divide en un determinado número de secciones, estén solapadas o no. Se selecciona una a una cada sección y se multiplica por el filtro g, y se obtiene la característica que se buscaba. Es decir, por cada sección calcula la característica de distinción de intensidad de contaste frente a la media nula.

Esta operación se repite para todas las escalas y orientaciones deseadas. Por lo que generalmente se hace un desplazamiento de la imagen por si el iris capturado está girado con respecto al de la base de datos.

Por último se realiza una conversión a binario, esto se realiza asignando un 1 a los coeficientes cuyo valor sea positivo o nulo, y un 0 a aquellos que tengan un valor negativo.

Comparación: Como es lógico se practica mediante la comparación de las características obtenidas con los patrón previamente almacenados. El sistema usado para la comparación de características se basa en el cálculo de la distancia de Hamming. Este método es "prácticamente" el definitivo hoy en día. Aparte de los buenos resultados obtenidos y su simplicidad hacen que en un sistema, como el desarrollado en este proyecto, sólo precise de una imagen para identificar al usuario correctamente. Esto hace que el sistema sea de gran aceptación por el usuario.

La distancia de Hamming hace referencia a la efectividad de lo códigos de bloque y depende de la diferencia entre una palabra de código válida y otra.

Por ejemplo, la distancia de Hamming de **101111** y **100110** es 2.

Básicamente se realiza una comparación de los bits que hay de la característica tomada por captura del usuario con la de la base de datos que se considera la correcta, y por cada bit que haya diferente en la misma posición que el otro se asigna un valor de 1 en un array de la misma dimensión que el código a comparar, si el bit es el mismo, entonces se le asigna un 0. En el ejemplo anterior por tanto, sería: **001001**.

Luego ese valor puede gestionarse como se crea debido. En el caso de este proyecto se opto por un valor porcentual. Se calcula de la siguiente forma:

$$Acierto\% = 100 - \frac{valores\ distintos}{valores\ comparados} \cdot 100$$

3. Plataforma de Desarrollo: Android.

Con el desarrollo de los llamados smartphones ha sido necesario implementar sistemas operativos en móviles que operen y gestionen toda esa información que precisa estos ordenadores de bolsillo. Android fue creado durante el transcurso del año 2008 desarrollado como plataforma abierta, haciendo que sea posible modificar y mejorar de acuerdo a las diversas necesidades de los fabricantes y usuarios.

3.1. ¿Qué es Android?

Android es un sistema operativo basado en Linux para dispositivos móviles fundado en Noviembre de 2007 y liderado por Google. Desarrollado por Android Inc. (comprada por Google en 2005), pertenece a la Open Handset Alliance, en la que se engloban 84 empresas de fabricantes y desarrolladores de hardware, software y operadores. Los miembros más destacados son Google, Samsung, HTC, LG e Intel.

Gracias al código abierto de esta plataforma es fácil poder programar, implementar y distribuir mediante Google Play (distribuidor oficial de aplicaciones para Android) cualquier aplicación sin ningún tipo de restricción o espera para poder dar de alta la aplicación.

3.2. ¿Por qué Android?

Hay varios motivos por los cuales cada vez más programadores de software ya sea por hobby o por trabajo que eligen los dispositivos Android:

- Gran variedad de hardware: se puede elegir desde dispositivos de bajo precio hasta las gamas más altas, de esta manera se hace que el sistema Android sea más accesible.
- Más opciones de configuración y personalización: esto puede resultar más complicado para ciertas personas que les resultaría más sencillo un dispositivo que lo realice todo pero para el objetivo de este proyecto este en un punto a favor.
- Utilización de widgets (Ver diccionario): hay ciertos sistemas operativos que no ofrece esta función en sus aplicaciones.
- Multitarea: esto ofrece la función de poder tener operativas varias aplicaciones cumpliendo distintas tareas a la vez.

El competidor hoy en día de Android es Apple, por lo cual los dos luchan por vender más dispositivos con su sistema operativo, competencia que mejora ambos sistemas. Pero el problema es que no pueden compararse en todos los aspectos puesto que ambos sistemas están diseñados de forma distintas para plataformas distintas.

Apple es un sistema con gran experiencia en su campo, puesto que ellos desarrollan hardware y software, crean los dispositivos con la mejor compatibilidad entre sus equipos del mundo. A su vez este sistema de desarrollo de aplicaciones está muy protegido con grandes restricciones para el desarrollador, como una comprobación de la empresa de la aplicación antes de subir al comercio o un pago anual. Esto es debido a que disponen de un equipo especializado que prueban tu aplicación y comprueban si tiene alguna intención maliciosa para el dispositivo antes de lanzarlo al Apple Store. Este servicio no se ofrece en Android por lo que resulta más barato al ahorrarse ese gasto de personal pero puede ser más peligroso si el usuario no sabe lo que esta instalando.

Sin embargo, Android ofrece un código abierto completo, por lo que ofrece a los usuarios la oportunidad de poder configurar o crear aplicaciones de todo tipo pudiendo usar todas las funciones del dispositivo. Esto puede resultar una ventaja para alguien que le guste configurar cada ápice de su dispositivo o una desventaja para el que quiera tener un dispositivo con sus aplicaciones con un fácil acceso y uso.

Por tanto no es un sistema mejor que otro, son simplemente diferentes sistemas para diferentes personas. En este proyecto puesto que es necesario realizar numerosas pruebas en varios dispositivos, es mucho más recomendable usar el sistema Android.

3.3. Historia

Fue en Julio del 2005 cuando Google adquirió Android Inc, una pequeña compañía de Palo Alto (California) fundada en 2003. Los cofundadores liderados por Andy Rubin fueron a trabajar a Google y todo lo que se sabía de Android Inc. es que desarrollaban software para dispositivos móviles. Este fue la pista que indicaba que Google planeaba introducirse en la telefonía móvil.

Google pudo ponerse en contacto con varios fabricantes de hardware y software para poder implantar este sistema operativo. Todo el proceso fue llevado con el mayor secreto posible, solo se conocían datos como que Google en septiembre del 2007 solicitó diversas patentes en el área de la telefonía móvil.

La primera versión de Android fue liberada el 23 de septiembre de 2008 llamada Apple Pie.

3.4. Versiones de Android

Como se ha comentado anteriormente, la primera versión lanzada fue la 1.0 llamada Apple Pie. Como observación curiosa se puede ver que cada versión de Android contiene el nombre de un postre en ingles, y que todas las versiones en orden cronológico siguen a su vez un orden alfabético. Cada versión introduce un gran número de novedades, pero se comentarán las más significativas (todas ellas incluyen corrección de errores de sus versiones anteriores):

- **Apple Pie 1.0:** La primera versión.
- **Banana Bread 1.1:** Únicamente corrección de errores de la versión 1.0.
- **Cupcake 1.5:** Teclado virtual en horizontal y vertical. También se incorporó el widget.
- **Donut 1.6:** Posibilidad de cambiar la resolución del sistema adaptándose a nuevos tipos de pantalla.
- **Eclair 2.0:** Google Maps.
- **Froyo 2.2:** Se aumenta el número de paneles de la pantalla de inicio de 3 a 5.
- **Gingerbread 2.3:** Gestor de aplicaciones para mejorar la gestión de batería. Cambio de la interfaz.
- **Honeycomb 3.0:** Versión creada específicamente para tablets, cambio de interfaz para adaptarlo a pantallas grandes.
- **Ice Cream Sandwich 4.0:** Desbloqueo mediante reconocimiento facial. Mejoras para administrar la multitarea, ActionBar dentro de cada aplicación. Tecnología NFC.
- **Jelly Bean 4.1:** En las actualizaciones solo se descarga la parte cambiada (gran ahorro de gestión de datos). Nueva barra de búsqueda. Posibilidad de transmitir video por NFC.

A continuación se muestra la información de la cantidad de dispositivos que usan una versión u otra (Ver **figura 5** y **tabla 3**). Esta información se obtiene directamente desde la página oficial de Android developers que se actualiza cada 14 días, por lo que se puede ver en tiempo prácticamente real la versión a la que se debe implementar un nuevo proyecto. También se dispone de porcentajes de las versiones que se han metido en Google Play para nuevas aplicaciones, tamaño y densidad de pantallas, y versiones del OpenGL. En este caso solo nos hacen falta los datos de versiones de Android y si se desea comercializar en el Google Play. también los datos referentes al acceso a la tienda online.

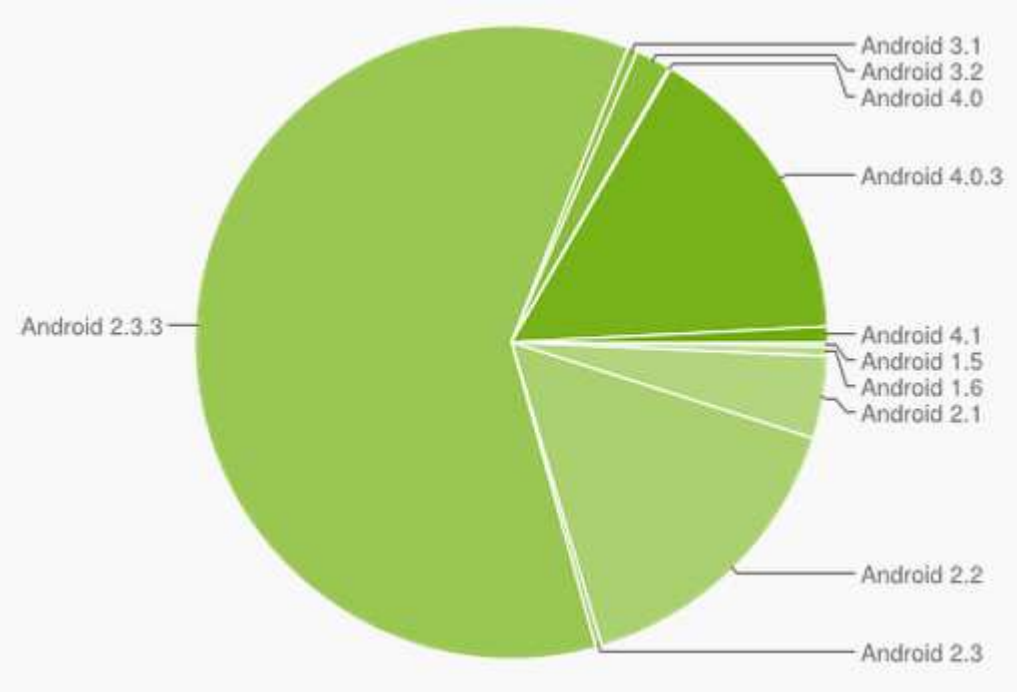

Figura 5: Distribución de versiones en los dispositivos [19]

Versión	Nombre	Nivel API	Distribución
1.5	Cupcake	3	0.2%
1.6	Donut	4	0.5%
2.1	Eclair	7	4.2%
2.2	Froyo	8	15.5%
2.3 - 2.3.2	Gingerbread	9	0.3%
2.3.3 - 2.3.7		10	60.3%
3.1	Honeycomb	12	0.5%
3.2		13	1.8%
4.0 - 4.0.2	Ice Cream Sandwich	14	0.1%
4.0.3 - 4.0.4		15	15.8%
4.1	Jelly Bean	16	0.8%

Tabla 3: Distribución de versiones en los dispositivos [19]

Como se puede observar la versión 2.3.3 es la más indicada debido a la enorme extensión del dispositivo más del 60%. Es por esto que se realizó sobre esta versión, es decir, con el nivel de API 10. El nivel de API es simplemente el número atribuido a cada versión según se van creando siendo el nivel 1 el de la versión 1.0.

3.5. Arquitectura de una aplicación Android

El archivo más elemental de una aplicación Android es una actividad, la cual es básicamente una clase. En ella se incluye lo que el usuario quiere hacer. Todas las actividades son para interactuar con el usuario por lo que hay que crear una ventana para a continuación introducir una interfaz. Esta interfaz es la llamada vista (View en Inglés).

Desde la aplicación más simple a la más compleja todas se estructuran de la misma manera. Esta estructura se crea desde el principio con la creación del proyecto. En el proyecto se puede distinguir siempre en el explorador de proyectos:

- **Src:** es la abreviatura de fuente en inglés (source), y es donde se incluyen las actividades o clases dentro del paquete de la aplicación (es el nombre en clave que tiene tu aplicación para poder interactuar aplicaciones tuyas unas con otras), son archivos .java, en los cuales se implementa toda la codificación lógica de la aplicación.
- **Gen:** son archivos de configuración y de gestión de datos gestionando las direcciones de memoria usadas para cada necesidad. Estos archivos se van generando automáticamente según se va desarrollando la aplicación en las clases del src. NO se debe modificar manualmente, el entorno de desarrollo lo hace automáticamente.
- **Android X.X.X:** En el caso de la aplicación creada ha sido Android 2.3.3. Aquí se almacenan todas las funciones de las bibliotecas de esa API.
- **Android Dependencies:** Donde se almacenan las bibliotecas implementadas como por ejemplo la de OpenCV de este proyecto.
- **Assets:** Donde se implementan nuevos cambios de la imagen predeterminada por el sistema, como cambios de tipo de letra mediante un .ttf.
- **Bin:** Donde se almacenan los archivos al construir la aplicación.
- **Res:** Es el abreviado de recursos en inglés (resources), y como su propio nombre indica almacena todos los recursos de la aplicación como sonidos, imágenes, etc.… En ella es donde se encuentran los diseños (layout en inglés) que almacenan las vistas (views en inglés) de las imágenes, archivos .XML que se diseñan para darle la apariencia deseada a la actividad, etc.
- **Android Manifest:** Manifiesto de Android en español. Es el documento más importante del programa. Se encuentra en la raíz del proyecto y actúa como descriptor de implementación de la aplicación.

Cualquier actividad tiene el mismo uso en cuanto a llamada de su arranque, pausa o finalización. Este diagrama es llamado el ciclo de vida de la actividad (ver **figura 6**):

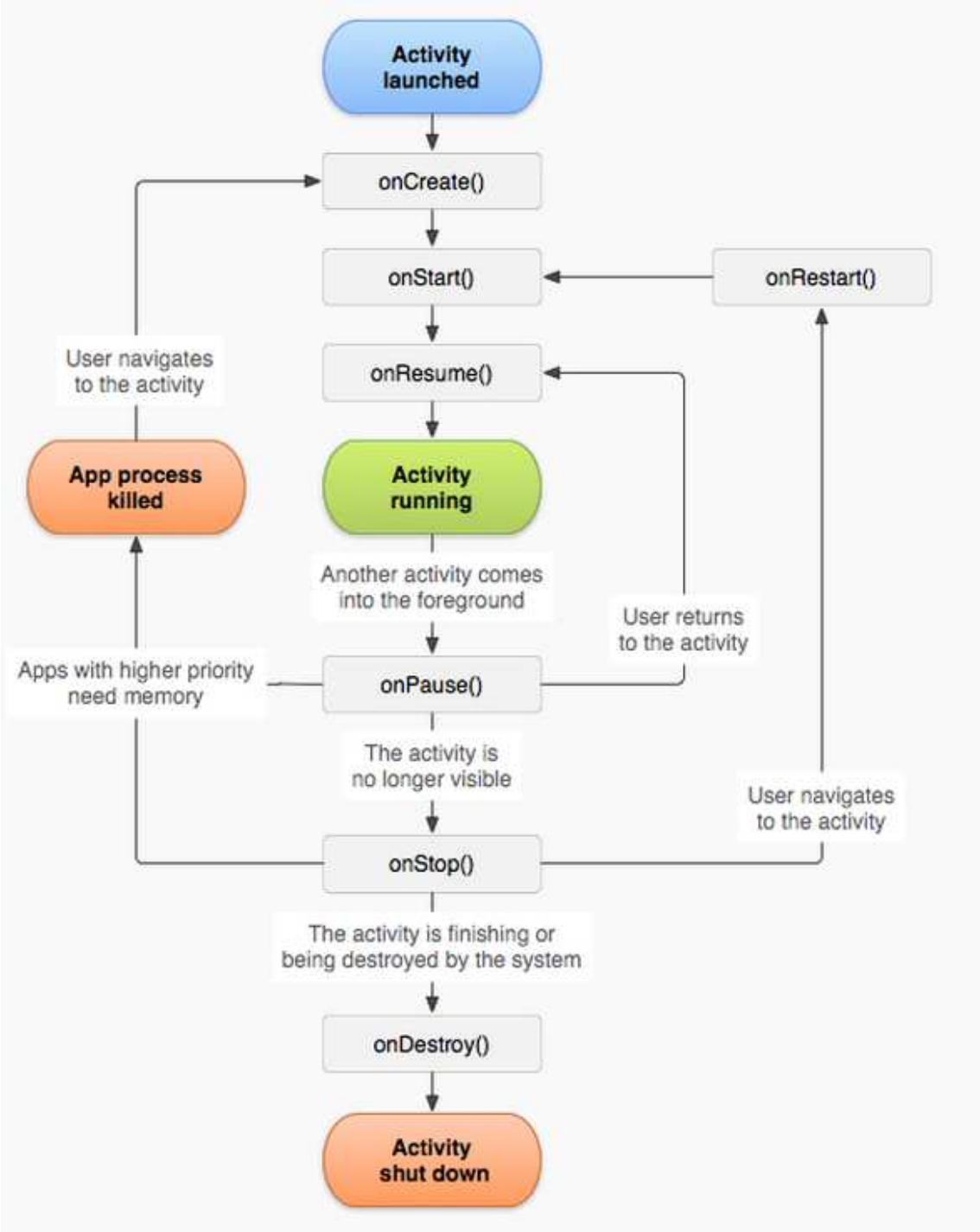

Figura 6: Ciclo de vida de una actividad [20]

3.6. Entorno de desarrollo

La plataforma de desarrollo utilizada, como se puede observar en el propio título del proyecto, es Android, que utiliza el lenguaje Java. Para poder trabajar en esta plataforma ha sido necesario utilizar el entorno de desarrollo Eclipse Indigo, desde el cual ha sido posible desarrollar la aplicación gracias a este entorno de programación en dispositivos con código abierto.

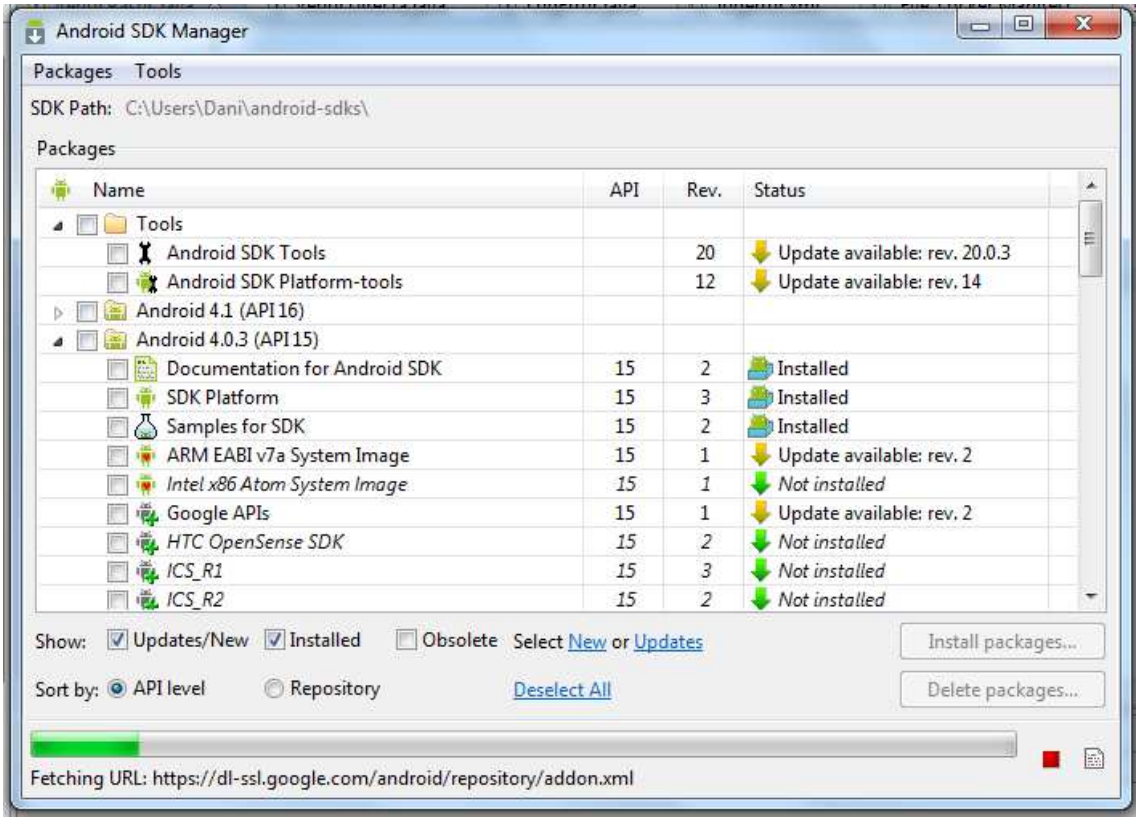

Figura 7: Gestor del SDK de Eclipse

Para poder trabajar con esta plataforma en el entorno de desarrollo Eclipse se ha necesitado la instalación del SDK de Android (Ver **figura 7**). El SDK proporciona las bibliotecas de cada API, así como herramientas de desarrollo necesarias para construir, testear y depurar aplicaciones para Android. Esto se realiza mediante un gestor del SDK (SDK Manager) que se integra en el programa, desde el cual se puede elegir la versión del API que se desea descargar para trabajar con ella.

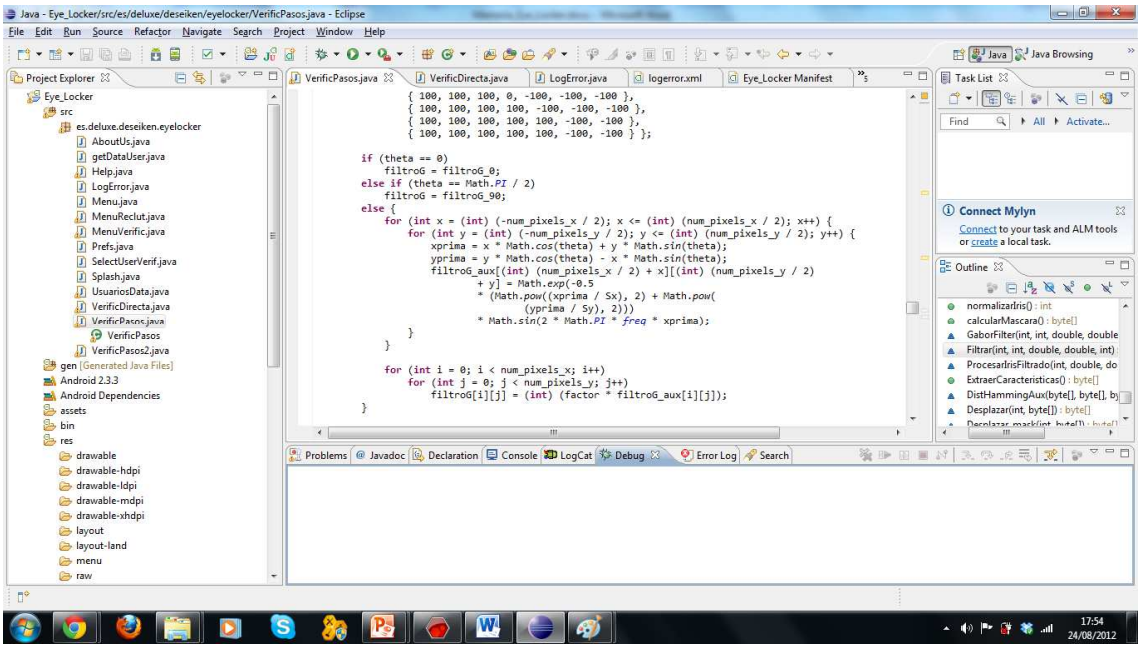

Figura 8: Eclipse Indigo

En cuanto al uso del entorno de desarrollo, tal como se puede observar en la **figura 8** la pestaña de la izquierda del todo es el explorador de los proyectos incluidos en tu espacio de trabajo. En él se puede desfragmentar el proyecto para navegar por cada una de las actividades o sus vistas. Las pestañas del medio que están recogidas en un recuadro desde el cual es posible ver y editar el código de cada uno de los archivos seleccionados en el explorador. A la derecha del todo tenemos dos ventanas, la de arriba es para crear y ver una lista de tareas que se puede organizar por días, y abajo es una recopilación de las funciones de la clase seleccionada en ese momento en la ventana del medio. Esta última ventana contiene información de cada función determinada por simbología (Ver **figura 9**) la columna de la izquierda son variables y los de la derecha

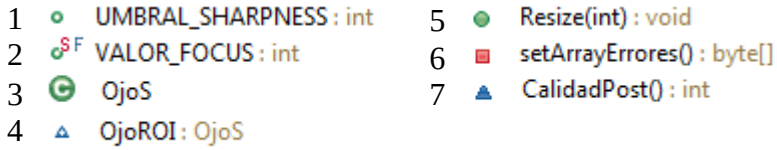

Figura 9: Simbología Eclipse

son funciones:

1. Variable global pública.
2. Variable global final estática y pública.
3. Clase.
4. Variable global.

5. Función pública.
6. Función privada.
7. Función sin declarar su privacidad.

Finalmente en la parte inferior se puede ver la ventana de información, donde se muestran desde los errores registrados al compilar la aplicación hasta los errores o excepciones producidas en una depuración. Para depurar es conveniente pasar a su vista correspondiente (Ver **figura 10**), puesto que se dispone de una ventana situada arriba a la derecha del todo que representa todas las variables globales y sus valores en el momento del punto de interrupción, la cual resulta enormemente útil para comprobar paso a paso que las funciones realizan los pasos correctamente.

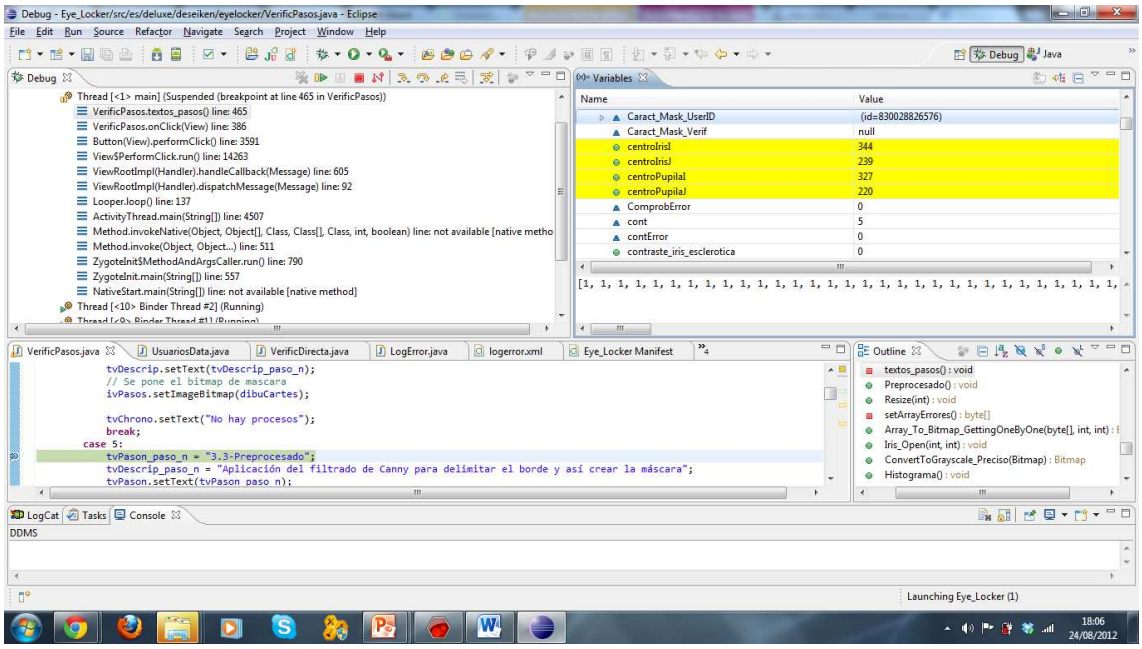

Figura 10: Modo depuración de Eclipse

3.7. Google Play

Google Play representa un bazar de aplicaciones en donde cada desarrollador puede poner en el mercado sus propias creaciones. Este método incentiva a todo tipo de programadores para poder adentrarse en el mundo de Android. Una vez preparada la aplicación basta con pagar 20$ y se dispondrá de forma vitalicia de una cuenta de desarrollador para poder subir tantas aplicaciones como desees.

4. Diseño de la solución: Eye Locker

En la creación de cualquier tipo de programa la estructuración es esencial, puesto que obliga a organizar las ideas y saber qué datos son los necesarios para almacenar, qué hay que disponer en pantalla o las opciones que son esenciales para el correcto funcionamiento del mismo. Además en la plataforma en la que se trabaja, la estructuración de las vistas (View en Inglés con extensión (.XML)) de las actividades son esenciales para un correcto guiado de la actividad y del programa en conjunto, es por eso que botones vistosos, indicadores de ayuda o cualquier elemento visual que pueda guiar al usuario ha de ser estudiado y correctamente colocado. Así mismo, hay que tener cuidado con la saturación de información mostrada en pantalla o la masiva implementación de opciones en una sola vista. El objetivo primordial de este apartado es la estructuración de una aplicación que sea fácil de seguir por lo menús con una disposición lógica de los elementos.

La solución propuesta para este trabajo ha sido la aplicación llamada Eye Locker. La aplicación está desarrollada y envuelta en un entorno vistoso y colorido, en la cual se han incluido fondos de pantalla para las actividades, una visión a pantalla completa y en algún caso comprobadores visuales de que la actividad ha sido realizada con éxito, ya sea con un toast o con un elemento visual como un check (Ver diccionario para los elementos que no se entiendan). Además se ha dispuesto en ciertos botones una imagen para la rápida asociación a su función como es el claro caso de la cámara o de la carpeta para hacer la foto o para seleccionar la imagen de la galería respectivamente.

Además se han bloqueado todas las vistas de todas las actividades para que se muestren en vertical (conocido como portrait), es decir, en vertical. Esto se incluye en el apartado del Android Manifest.

Todos los elementos dispuestos en la aplicación han sido creados personalmente en formato PNG puesto que es el formato gráfico con el cual Android trabaja mejor en sus dispositivos. Además PNG ofrece una ventaja frente a otros formatos y es la posibilidad de creación de imágenes sin fondo. Esta ventaja es esencial para el diseño de aplicación puesto que se puede de esta manera presentar botones que no presenten un cuadrado blanco como fondo sino que contiene un fondo transparente. Esto se puede observar, por ejemplo, en el icono de la aplicación que aparece en la portada del proyecto o en el icono creado al instalarla en un dispositivo Android.

Todas las creaciones visuales de elementos en la aplicación han sido creadas total o parcialmente en Photoshop CS5 de forma manual.

4.1. Planteamiento inicial

En el planteamiento inicial realizado antes de empezar a desarrollar la aplicación, fue necesaria una estructuración y un desarrollo inicial, aunque según se iba desarrollando se fueron implementando nuevas ideas y nuevos sistemas que hacían necesario una restructuración de la aplicación y de sus clases. Esta restructuración fue debida también a idea de implementar un sistema que sea más eficaz y más rápido, así como una aplicación con una estructuración más eficaz y más lógica. Más adelante en este tema se expondrán los problemas y soluciones que se encontraron en el desarrollo de la actividad que conciernen al diseño y a su estructuración.

Para el diseño aplicado en cada una de las vistas de las actividades, los elementos de dichas vistas han sido medidos en densidad de pixel en vez de pixeles de ancho y alto. Se ha realizado de esta manera para que se pueda instalar la aplicación en cualquier dispositivo y que mantenga la proporcionalidad independientemente de la resolución de la pantalla. De esta manera se puede evitar que se solapen imágenes o que pueda llegar a salir fuera de la pantalla elementos fundamentales que deben estar representados dentro de las inmediaciones del dispositivo en su totalidad.

4.2. Descripción de la aplicación

La aplicación cumple con el objetivo inicial del proyecto, por tanto puede identificar a un sujeto registrado anteriormente mediante una captura obtenida posteriormente. A continuación se explicará la aplicación de la manera en la que un usuario navegaría por lo menús de dicho programa.

Comienza la aplicación con el menú inicial en el cual se encuentra la opción de verificar o de reclutar. Puesto que suponemos que es la primera vez que se abre la aplicación, no hay datos guardados de otros sujetos por tanto vamos a reclutar para añadirlos.

Da comienzo la actividad donde se pide introducir la identificación del sujeto en un campo de texto. Esta identificación ha de ser introducida sin espacios, más adelante se explicará el porqué. Una vez introducido se añade a la lista que es guardada de inmediato. Pulsando sobre el sujeto añadido en la lista, da comienzo a la actividad donde se iniciará la introducción de los datos del sujeto.

En esta pantalla se procede a la obtención de la captura del ojo para la extracción de características. Se dispone de dos botones para la obtención de imágenes, puede ser por selección de la galería o de la captura mediante el uso de la cámara. Una vez obtenida se presenta una vista previa de la captura, en la que después de hacerle los controles de calidad pertinentes, se modificará mostrando además la detección del iris, marcando sus bordes. Posteriormente se deberá de extraer las características pulsando el botón contiguo. Por último pulsando el botón de guardado se almacenarán todas las

características en una dirección de memoria privada, accesible únicamente desde la propia actividad para mayor seguridad. Una vez guardado se sale de esta actividad y reaparece la actividad anterior con la lista de los sujetos, en cuanto se salga de esta actividad se guardarán de forma perpetua las características de dicho individuo y no podrán ser modificadas. La verificación (Check en Inglés) verde que aparece al lado del sujeto en la lista indica que están las características correctamente guardadas y no es posible modificarlas por cuestión de seguridad, la única opción es borrar al sujeto y volver a capturar el ojo.

De nuevo en el menú principal, si se selecciona la opción de verificación dará comienzo una nueva actividad donde estarán dispuestas en forma de lista, solo los sujetos que tengan guardado en la base de datos características para analizar. Una vez se seleccione un individuo de la lista se inicia la actividad de selección de captura para su verificación. Al igual que la actividad de extracción de características se dispone de dos botones para la elección de la captura, ya sea por imagen de la galería o de foto tomada desde la cámara. Se mostrará una imagen previa de la captura y se podrá realizar la verificación por 2 métodos gracias a la casilla de verificación (conocido como checkbox en inglés).

Si la verificación se realiza por pasos, dejando la casilla de verificación marcada, entonces se procederá a una actividad en la cual se expondrá paso a paso en modo de presentación los pasos que se van realizando en cada uno de los procesos, mostrando por pantalla una imagen previa de los resultados de cada uno de ellos. En la parte superior se dispone de un cronómetro que cuenta el tiempo que transcurre desde el comienzo de la operación hasta que termina sin incluir los procesos de presentación como pueden ser cargar las imágenes previas en pantalla. Con el botón de “Siguiente” se puede ir pasando de un paso al otro hasta llegar al final de los pasos si no se han detectado fallos de calidad. Donde se mostrará en pantalla el tiempo total de todos los procesos y si el resultado final indicando si el sujeto de la captura es el correspondiente al perteneciente a la base de datos de la ID seleccionada al verificar sujeto.

Si se han detectado fallos de calidad en la captura cuando ésta se esté analizando, se notificará al usuario en la pantalla que la imagen contiene errores y por tanto no se puede proceder con la verificación. Para poder ver los errores que presenta la captura, se inicia el registro de error pulsando el botón de “ERROR”, este es una actividad donde se mostrará por pantalla qué controles de calidad ha pasado la imagen y cuáles no.

De vuelta a la selección de tipo de verificación, si se deja la casilla de verificación sin marcar, entonces se inicia la verificación directa. En ésta no aparece por pantalla ninguna imagen previa como resultado de los procesos previos realizados, únicamente el resultado una vez se pulse el botón de comenzar. Además aparece el tiempo total en la parte superior, para determinar conclusiones que se comentarán más adelante

Todos los datos almacenados son guardados en la memoria interna del dispositivo móvil por lo que la retirada de la tarjeta SD no afectará en absoluto sobre los datos guardados por la aplicación.

4.3. Estructura del diseño y su función

En esta parte se hablará de la estructura del diseño y la explicación de ciertos objetos o indicadores visuales que aparecen en las actividades.

En el menú principal ha optado por un diseño simple de dos botones. La imagen presente justo encima era la que anteriormente correspondía como el icono de la aplicación pero posteriormente fue sustituido por el definitivo presente en la portada de este proyecto.

La pantalla de reclutamiento de sujetos, como se puede observar en la **figura 11**, se optó por un ListView de arrayList dinámico (ver diccionario), el cual va añadiendo o eliminando elementos sin tamaño fijo y sin definir previamente. Lo bueno de tipo de disposición de ítems es que se puede manipular su vista, y como se puede ver, si la lista no cabe completa en el espacio asignado, se puede arrastrar la lista hacia arriba y hacia abajo para poder ver todos los sujetos.

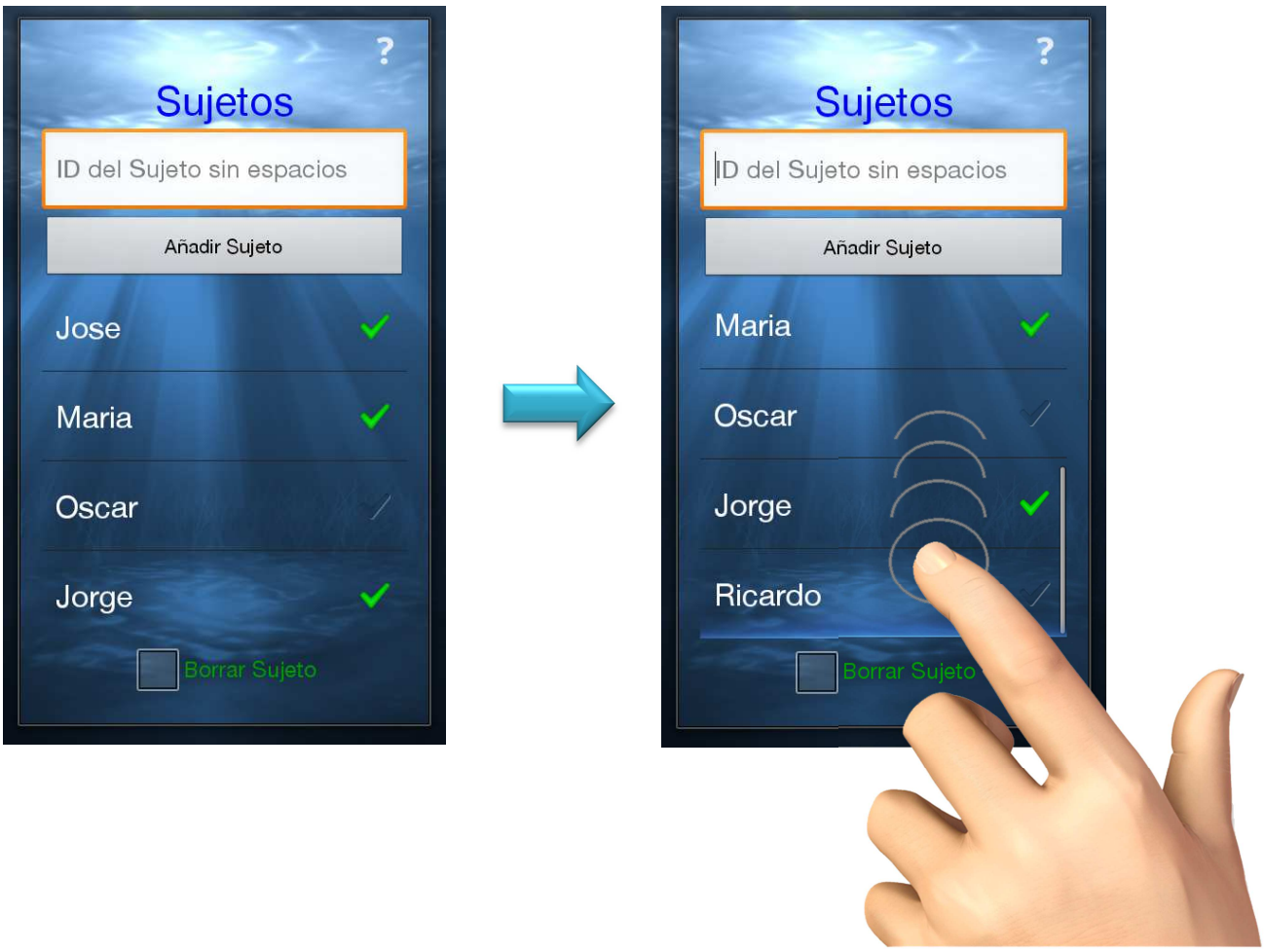

Figura 11: Desplazamiento del ListView

Además se puede disponer de un ítem listener que en cuyo caso se ha establecido que cada vez que es pulsado un sujeto se activa la actividad de recogida de características. Para poder hacer esta opción hay que implementar al ListView un adaptador que determina la disposición que tendrá, básicamente se encarga de coger ese arrayList y transformarlo en una forma visual y representable por pantalla. La forma elegida es la de "simple_list_item_checked" que básicamente pone ha disposición al lado de cada elemento de la lista un tick configurable. Este tick se usa para determinar que elementos tienen características guardadas y ya no se pueden sobrescribir.

Una vez seleccionado uno de los sujetos creados y siempre y cuando se pueda modificar el sujeto, (cuando no tenga un check a su lado) se accederá a la actividad de extracción y guardado de características que se puede ver en la **figura 12**.

Figura 12: Extracción de características y guardado

En esta actividad es donde hay que realizar un test de calidad antes de extraer las características. El check verde indica que sea realizado el proceso sin problemas, el aspa que ha ocurrido algún error. Una vez se haya confirmado que ambos procesos se han realizado con éxito se puede guardar en la dirección de memoria designada por el nombre del sujeto más "_code" para el código obtenido por el filtro de Gabor y más "_mask" para la máscara del iris.

Si se comprueba que el proceso realizado ha fallado, aparecerá un aspa. Pulsando en dicha aspa indicará el fallo en un registro de datos (Data log en Inglés). Este error se produce todas las veces al no pasar un control de calidad. Se pueden generar por dos razones:

- Porque la imagen capturada no tiene calidad suficiente para ser procesada.
- El proceso de detección de bordes no ha detectado correctamente el iris.

A continuación se muestra un ejemplo (Ver **figura 13**) de fallo de calidad del segundo tipo de los nombrados anteriormente:

Figura 13: Apertura del registro de errores

Una vez dentro de verificación, después de seleccionar uno de la lista, nos encontraremos en la actividad de obtención de la captura del sujeto. Esta actividad su disposición es muy semejante a la de obtención de características descrita anteriormente, la única diferencia es la casilla de verificación del cual disponemos en la parte inferior para poder seleccionar entre una verificación por pasos o una verificación directa.

En cuanto se pulse verificar, si la casilla de verificación está activo, inicia la verificación por pasos. El diseño según se va pulsando el botón “Siguiente” es el mismo, solo cambia los títulos y descripciones de cada paso. Cuando llega al control de

calidad de la captura, el diseño y funcionalidad es prácticamente el mismo que en el caso descrito anteriormente, con la diferencia de que el botón que hay que pulsar para ver el registro de error es el de "ERROR" el mismo que anteriormente ponía "Siguiente".

El diseño de la verificación directa es igual que la verificación por pasos, con la única diferencia de que no hay pasos intermedios entre el comienzo de la verificación y el resultado final. En la verificación directa, si el test de calidad no se pasa satisfactoriamente, ocurrirá de igual forma como se dijo anteriormente, se interrumpe el proceso y pulsando el botón "Error" se podrá ver el registro de error.

A continuación se muestran unas capturas de la actividad de verificación. Se puede ver la actividad de captura de la imagen, la verificación por pasos y la verificación directa, en la **figuras 14**.

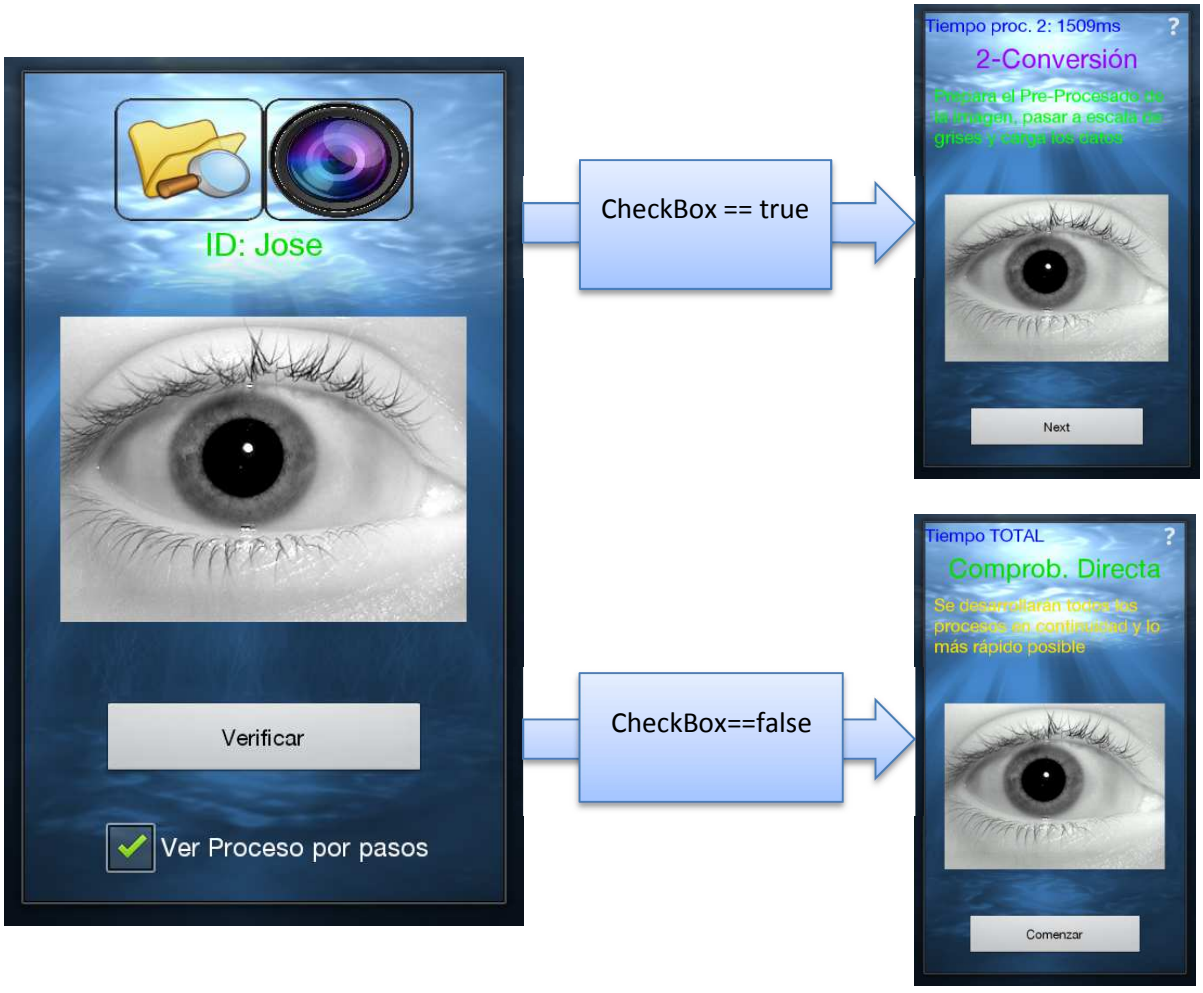

Figura 14: Verificación por pasos y directa

Como se puede observar, la disposición de botones y de imágenes es simple e intuitiva para mayor facilidad del usuario.

Además la aplicación dispone de un botón de ayuda situado en el borde superior derecho, el cual (dependiendo de la actividad en la que se encuentre el usuario) dispondrá de un texto informativo o posibles indicaciones del procedimiento a seguir según corresponda.

4.4. Diagrama de bloques general

Se expone a continuación un diagrama de bloques general de las actividades con sus respectivas vistas (Ver **figura 16**). Hay que tener en cuenta que hay algunas actividades que se pueden acceder desde dos vías posibles.

Para no sobrecargar el diagrama de bloques, el inicio de la actividad ayuda se representa aparte puesto que es simple e igual para todas las actividades que la contienen (Ver **figura 15**)**.**

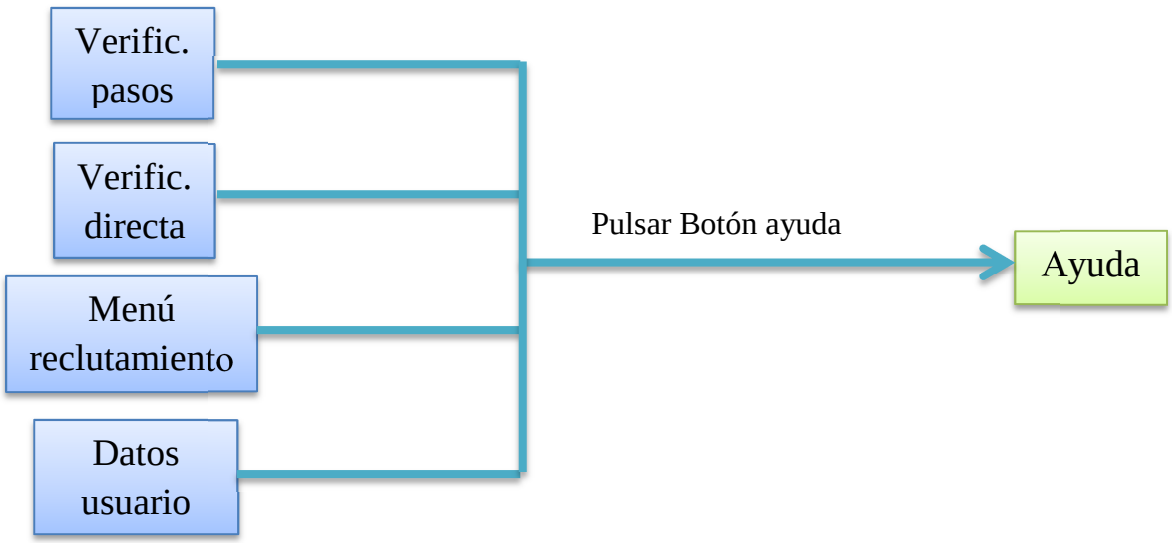

Figura 15: Diagrama de flujo de ayuda

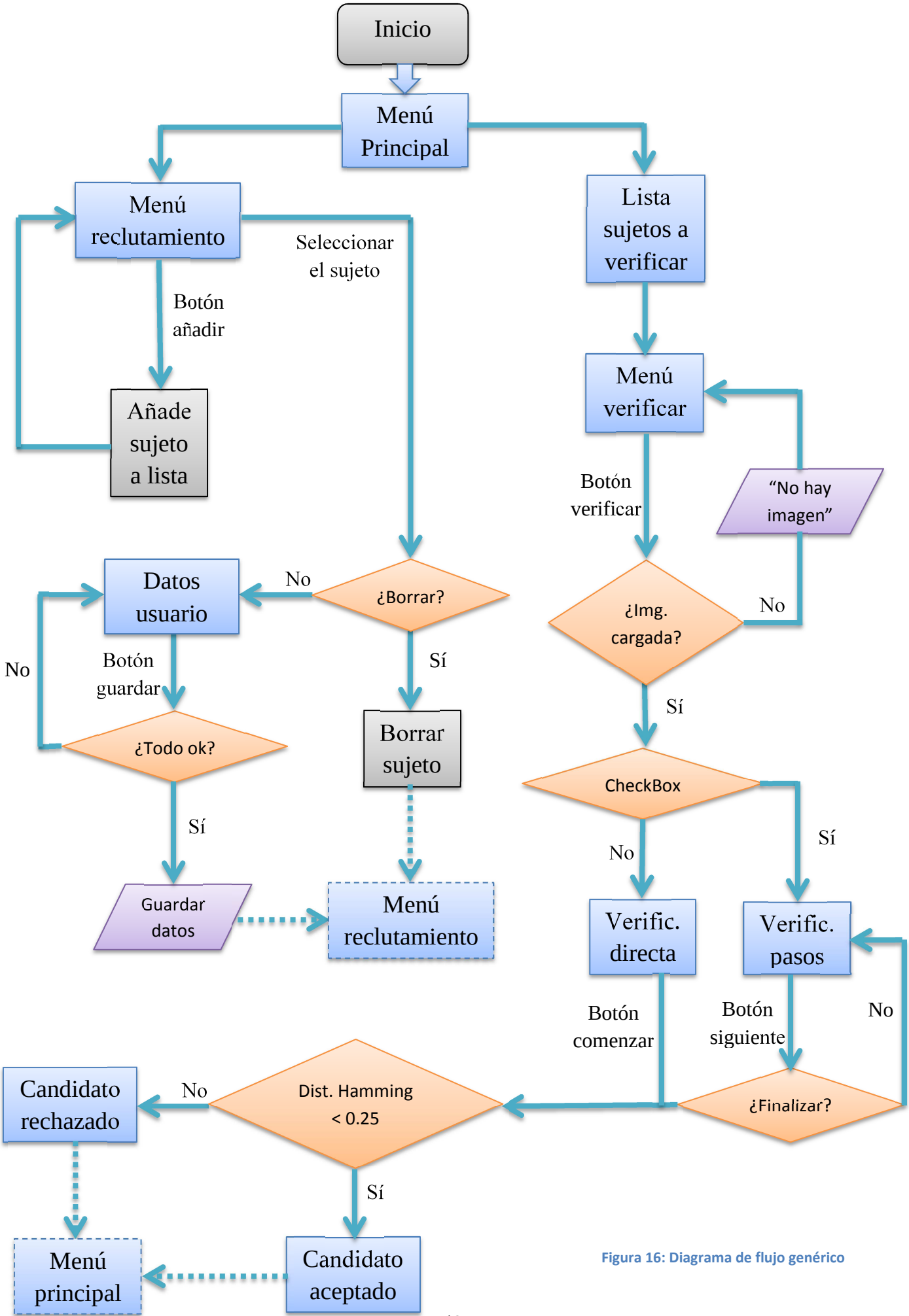

Figura 16: Diagrama de flujo genérico

5. Desarrollo

Para poder explicar el desarrollo de la aplicación se ha optado por una presentación temporal, fragmentada en etapas importantes, en cada una de ellas el desarrollo o la explicación de los problemas y su solución son expuestas según fueron surgiendo. Para terminar se expondrán los detalles y protecciones que se realizaron para dar una mejor apariencia a la aplicación aunque estos no sean imprescindible.

5.1. Prestaciones posibles

Con este proyecto se ha visualizado una gran implementación en diferentes actividades todo entorno a la seguridad del dispositivo, protegiendo datos privados del usuario. Como por ejemplo, guardar contraseñas web, números de tarjetas de crédito o bloquear y desbloquear el dispositivo móvil. Esto se ha podido observar gracias a la facilidad del reconocimiento por iris y la búsqueda continua de un sistema de reconocimiento más fácil y práctico.

Por esto es por lo que en el desarrollo de la aplicación se ha procurado la utilización del código de una manera que sea sencillo posteriormente la implementación de nuevos métodos de identificación o nuevas prestaciones. Para ello se ha realizado una división de actividades únicamente por funciones que desarrollen, es decir, aunque una actividad pueda incluir las funciones y los procesos de otra actividad, estas estarán separadas si sus objetivos no son iguales.

Por ello se puede dividir la aplicación en partes fundamentales, y cada una de ellas en objetivos particulares dentro de estos campos:

- **Selección:** Es la selección de un objeto, foto, sujeto, etc.… En la actividad se puede subdividir en actividades que, seleccionan strings de la lista de sujetos e imágenes de la galería del dispositivo.
- **Obtención de datos y su manipulación:** La obtención de los datos se puede distinguir en actividades como la captura de imágenes (como por ejemplo, la cámara) y datos guardados en memoria privada del dispositivo. Estos últimos pueden ser obtenidos para su uso en un proceso o para su manipulación, ya sea para añadir, cambiar por nueva información o simplemente borrarla.
- **Procesado de identificación:** Son las actividades que conciernen a la identificación de la captura y la comparación con los datos almacenados en memoria. Estos en la aplicación son las actividades de verificación por pasos y verificación directa.

Gracias a esta división general de las actividades por objetivos, es posible que se pueda mejorar ampliando, cambiando o eliminando actividades, las cuales solo algunas

dependen de otras, pero con pequeñas modificaciones son capaces de operar solas. De esta manera se cumple uno de los objetivos iniciales de crear una aplicación que pueda ser fácilmente modificable en el futuro.

5.2. Obtención de la imagen

La obtención de la imagen se realiza en la actividad de datos de usuario, que es la que se abre en reclutamiento al seleccionar a un sujeto de la lista creada, y la actividad de verificación que se abre para obtener la imagen del sujeto para poder comprobar que es el mismo que el de la base de datos.

Puesto que se quería obtener la captura de dos formas posibles, desde la galería el móvil y desde la cámara, hay que disponer de una codificación que haga posible la diferenciación de datos obtenidos de cada uno de los dos métodos de acceso a la captura. Estas dos formas de acceso usan una actividad genérica que dispone el dispositivo, para cada modelo la cámara y la galería es diferente, puesto que este es el que proporciona el fabricante, pero lo único diferente realmente es la forma de mostrarlo por pantalla, nombrarlos y poder usarlos se hace igual para cualquier dispositivo. Esto hace que la universalidad de la aplicación sea posible de manera muchísimo más fácil.

Esta llamada a las actividades genéricas del dispositivo se hace a través de un intent el cual sirve como intermediario y detonador de la activación. Para iniciar las actividades de captura, ya sea la de la cámara o la de la galería, se es necesaria una dirección de acceso que la aplicación entienda que sistema general desea activar. Esta dirección como es lógico es diferente para ambos casos y se puede observar a continuación la resolución propuesta finalmente:

```
switch (pressed.getId()) {
case R.id.bFolder:
	Gallery = new Intent(Intent.ACTION_GET_CONTENT);
	// Establece el modo de visión de busqueda de imagen
	Gallery.setType("image/*");
	startActivityForResult(Gallery, SELECT_IMAGE);
	break;

case R.id.bCamara:
	TakePhoto = new
	Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
	startActivityForResult(TakePhoto, CAMERA_IMAGE);
	break;
```

Como se puede observar en el apartado de cámara hace referencia a las bibliotecas de Android, pidiendo acceso a la captura de imágenes por uso del dispositivo. Este tipo de uso por parte de la aplicación hace que sea necesario, por medidas de seguridad, una aceptación por parte del usuario que instala dicha aplicación. Dicho permiso al usuario ha de ser pedido en el Android Manifest, el cual se comentará

más adelante. Al usuario le aparecerá como contrato virtual las condiciones de uso de la aplicación cuando vaya a instalarlo, de tal manera que finalizando la instalación se aceptan dichas condiciones. Posteriormente se desarrollará con mayor profundidad estos temas.

Se puede ver también una línea de código justo debajo del comentario en verde, la cual está determinando el tipo de muestra del menú que puede ser por cuadrícula, por lista, etc.…Se ha seleccionado finalmente una disposición por cuadrícula definida por defecto en la muestra de imágenes en el dispositivo usado.

Una vez utilizado uno de los dos métodos de captura de la imagen, es necesaria la obtención de ella a la aplicación en la aplicación desde la cual la hemos solicitado. Para ello es necesaria una función que recoja los datos que genere esa actividad de galería o cámara, esa función es la llamada “onActivityResult”, la cual está protegida por el sistema como un constructor especial que recupera datos de actividades que se llaman con el objetivo de obtener un resultado.

Dentro de esta función para obtener los datos ha de saber diferenciar si se quiere obtener los datos desde la actividad de la galería o la de la cámara, para ello solo tiene que comparar que tipo de dato requerido ha sido seleccionado. Esto se ha conseguido simplemente al asignar un entero en el acceso de la actividad por medio de la declaración “startActivityForResult”, como se puede comprobar en el código de arriba se llama a esta actividad gracias al intent como se ha comentado antes y se envía el valor del entero comentado que será recuperado posteriormente por “onActivityResult” cuando los datos de esa actividad existan.

Es decir, cuando la actividad de obtener imagen por galería sea seleccionada y se quiera obtener el dato, el valor del entero que recibe es el de “SELECT_IMAGE” y cuando sea por cámara será el de “CAMERA_IMAGE”. Es por eso que se optó por un comparador para gestionar los datos si viene de la cámara o de la galería.

Posteriormente se procesan los datos según sea necesario determinado por su procedencia. De esta manera fue posible que la aplicación no sufriera bloqueos producidos por el desconocimiento por parte de la actividad de la procedencia de los datos, puesto que los procedentes de la galería son diferentes de los de la cámara y han de gestionarse de manera completamente diferente.

Se ha podido observar en el transcurso de esta etapa del desarrollo de la actividad que el procesado operado para la gestión de la imagen por la selección de la galería es más complicada y enrevesada que la de la cámara. Por lo que se ha determinado que esto es debido a que la utilización de la cámara está más generalizada que el de la galería en Android, ya que en la obtención de los datos de la cámara llegan directamente codificados en bitmap con uso de un simple cast. Mientras que en la de la galería es necesario un detector de posición del cursor para obtener la imagen, en vez, de enviar directamente la seleccionada, envía la posición de la imagen pulsada y se accede a esa dirección y se obtiene desde la propia actividad que solicita los datos.

Esto es debido a que en la galería lo que se está buscando es la dirección de un fichero, por lo que cuando pulsamos sobre la imagen deseada, lo que estamos pulsando es una vista previa de la imagen que se encuentra en esa dirección de memoria, a continuación por ficheros se puede obtener la imagen deseada. Mientras que en la obtención por cámara se realiza la captura en el acto, haciendo que dichos datos puedan ser procesados directamente sin necesidad de acceder a memoria para obtenerlos.

5.3. Formato de las imágenes

Para poder probar la aplicación, son necesarias las imágenes capturadas de los ojos de distintos sujetos. Hay que precisar de imágenes de varios sujetos diferentes, con numerosas fotos del mismo ojo y del otro, así mismo si se puede disponer de imágenes con elementos como lentillas o maquillaje mejor, puesto que la aplicación es probada al máximo. El motivo de que en el desarrollo se usen estas imágenes en vez de la cámara es debido fundamentalmente de que la cámara no contiene la suficiente resolución como para poder captar las minucias del ojo. En el apartado de pruebas se especificarán los resultados obtenidos con estas imágenes y se extenderá la conclusión del uso de la cámara. Pero durante el desarrollo de la aplicación para poder probar cada procesado de la imagen era necesario cargarlas en la aplicación.

El problema surge cuando las aproximadamente 6000 imágenes que fueron proporcionadas se encuentran en el formato TIFF y BMP, y el entorno Android no reconoce el formato TIFF y no gestiona correctamente el BMP. La solución surgió al comprobar que Android trabaja en pleno rendimiento con el formato PNG.

Para poder transformar el formato de tantas imágenes se utilizó un programa de cambio de formato automático llamado “FastStone Photo Resize” el cual con solo es necesario indicar la carpeta fuente de las imágenes y la de destino de estas una vez transformadas. En unas 3 horas fue posible usarlas todas sin ningún problema en el dispositivo.

En algunos casos se implementa el formato de la imagen PNG al crear un bitmap para ayudar en la compresión de la imagen para agilizar el envío entre actividades. Este proceso aunque comprima no reduce en ningún caso la resolución, únicamente usa el algoritmo lógico de compresión, que básicamente reduce los bits cuando estos se repiten de forma continuada. Este uso se comentará más detalladamente en el siguiente punto.

Durante la aplicación, las imágenes están almacenadas en un bitmap, el cual es básicamente una matriz de direcciones donde en una determinada posición se encuentra el valor de color de la imagen. Es decir, esa disposición es igual que el de una televisión en la cual para mostrar una imagen basta con asignar un color a cada pixel. Pero el bitmap puede tener distinta codificación de color, según se desee una calidad alta o baja. Es por esto que se dispone de distintas configuraciones cuando se crea el bitmap. Estas son:

- **ALPHA_8:** Cada pixel se almacena como una sola translucidez (alfa) de canal. Esto es muy útil para almacenar eficientemente máscaras, por ejemplo. No se almacena información de color. Con esta configuración, cada pixel requiere 1byte de memoria.
- **ARGB_4444:** Debido a la mala calidad de esta configuración, se recomienda utilizar ARGB_8888 en su lugar. Cada pixel almacena 2 bytes. Los tres canales de color RGB y el canal alfa (transparencia) se almacenan con una precisión de 4bits (16 valores posibles). Esta configuración es útil sobre todo si la aplicación necesita almacenar información de translucidez pero también tiene que ahorrar memoria.
- **ARGB_8888:** Cada pixel se almacena en 4 bytes. Cada canal (RGB y alfa de transparencia) se almacena con 8 bits de precisión (256 valores posibles). Esta configuración es muy flexible y ofrece la mejor calidad. Se debe de utilizar siempre que sea posible.
- **RGB_565:** Cada pixel se almacena en 2 bytes y sólo los canales RGB están codificados. El rojo se almacena con 5 bits de precisión (32 valores posibles), verde se almacena con 6 bits de precisión (64 valores posibles) y azul se almacena con 5 bits de precisión. Esta configuración puede producir ligeros artefactos visuales en función de la configuración de la fuente. Es decir, el resultado podría mostrar un tiente verdoso. Para obtener mejores resultados se debe aplicar dithering. Esta configuración puede ser útil cuando se usan los mapas de bits opacos que no requieren alta fidelidad de color.

Puesto que en el uso de estas imágenes es necesaria una alta calidad para poder diferenciar todas las minucias del iris y no hace falta el canal alfa, se ha decidido el uso de la configuración RGB_565 durante todo el desarrollo de la aplicación. Además las contraindicaciones de esta codificación no afectan en este caso, puesto que lo primero que se realiza en la imagen es una conversión a escala de grises que será comentada más adelante, por lo que no es necesario el color sino el contraste de la imagen. Para el uso de la máscara no se usó bitmap sino un array de bytes por lo que no es necesario hacerlo en un bitmap, pero si fuera ese el caso se usaría para la máscara el ALPHA_8.

5.4. Uso de listas dinámicas y del check_item

Para poder añadir sujetos en el menú de reclutamiento, es necesaria una disposición de elementos en lista sin tamaño definido. La mejor opción que se ha encontrado para conseguir tan objetivo es un arraylist, en el cual se puede añadir y eliminar elementos con suma facilidad. Además, es posible almacenar los datos de tal manera que cada uno de los strings de los nombre, ocupen una posición haciendo que sea una buena opción para luego poder eliminar elementos de la lista sin que los demás se vean afectados.

Manipular la lista dinámica es sencillo puesto que Android dispone de numerosos elementos que facilitan enormemente su desarrollo e implementación. Pero el verdadero problema se encuentra en que esta lista ha de mantenerse constante por lo que es necesario un guardado y cargado continuo, cada vez que la lista sea modificada, por si el usuario realiza cambio e inmediatamente después cierra la aplicación. Para ello hay que acceder a memoria numerosas veces, pero puesto que ese es otro tema que llevó tiempo y fue desarrollado más tarde, se dejará para más adelante.

Otro problema encontrado es a la hora de eliminar un sujeto, se había decidido implantar un mensaje de advertencia cada vez que se quisiera borrar a un sujeto, pos si se había realizado de forma accidentada. Se ha realizado finalmente un mensaje parecido a los continuamente presentes en Windows, el cual es un simple bloque con la notificación y un botón de "Ok" y otro de "Cancelar" como se puede observar en la **figura 17**. Este mensaje se programó sin ningún tipo de vista o diseño gráfico ya que se creyó innecesario, aunque puede ser posible. Al final para poder desarrollarlo, fue necesaria la utilización de "AlertDialog.Builder" el cual inicia una subclase que puede disponer de uno, dos o tres botones como máximo con un pequeño título y descripción del proceso que conlleva. Se puede observar que es el mensaje de alerta que siempre se utiliza en sistemas operativos, los cuales estos mensajes NUNCA superan los tres botones. Al ser una subclase aparece en pantalla superpuesta a la actividad anterior, pero esta no se ha cerrado, sino que está pausada esperando la decisión que toma el usuario para realizar un proceso u otro en función de la selección.

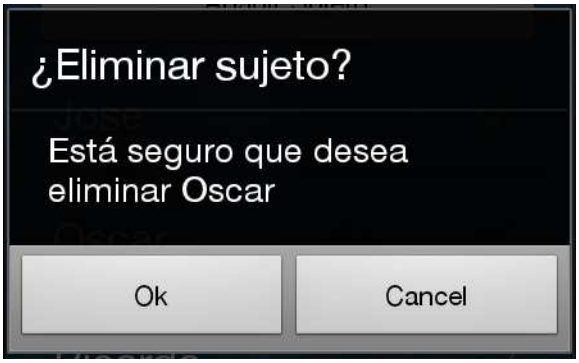

Figura 17: Mensaje de confirmación de borrado

En la construcción de un arraylist, hace falta una adaptador que disponga este elemento como una vista para poder representarlo en pantalla, esto es debido a que el arraylist de por si es solo con un almacén de datos sin ninguna visión propia definida por defecto, por lo que es necesario un elemento que le indique que tipo de imagen y presencia debe de presentar al incluirlo dentro de la vista de la actividad.

```
aaList = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_checked, Lista_ID_Sujetos);
```

Como se puede ver en la anterior línea de código, la vista se puede seleccionar directamente en la creación del adaptador, en este caso se ha accedido a los que tiene la biblioteca de Android, aunque es posible diseñar o implementar otras apariencias, esta es la gran ventaja que ofrece el código abierto, que no estas sujeto a un modelo específico.

El que se ha elegido es una lista de elementos simples, donde solo se representa un nombre, pero con una diferencia y es que esta lista contiene una opción de uso de check. Estos check se muestras como ticks al lado de cada elemento creado en la lista. Esta apariencia es la que se ha elegido puesto que resultó muy interesante para el uso de indicativos para saber si el elemento de la lista contiene ya características almacenadas y no se pueden modificar. Por tanto todos los candidatos que tengas el check en verde al lado de su ID, indica que tiene las características guardadas y por tanto listas para ser analizadas. Por lo que estos candidatos serán los que aparezcan en la lista de selección de usuarios una vez se quiera verificar.

Durante el desarrollo de la implementación y uso de los check como se ha comentado justo antes, resultó algo complicado. Esto fue debido a que el check se ha de activar no cuando el elemento es pulsado, sino cuando se ha guardado correctamente las características y se ha salido de la actividad de menú de reclutamiento. Por lo que esto complicó enormemente las cosas, ya que el check es afirmado cuando es pulsado el item. Por lo cual, se ha buscado un elemento que pudiera aportar la información necesaria al ítem para indicarle si ha de estar activado su check o no. El elemento que se ha encontrado es un arraylist, que se añade y se elimina de forma paralela al de los sujetos. De esta manera tenemos una lista que siempre contiene la misma cantidad de elementos que la lista de sujetos, si en una se elimina un elemento, en la otra también y si en la lista se crea un candidato en la otra también. La intención de la creación de ese arraylist es que trabaje como el representante del inicial (el array de los sujetos), es decir, se le asigna un valor de string llamado “false” (este no es un valor booleano aunque lo usemos como tal) en la misma posición que la del sujeto creado, y cuando este extraiga las características correctamente y se le de a guardar, entonces en el arraylist de los checks se le pone el valor “true” sustituyendo el anterior. De esta manera se sabe en todo momento, simplemente leyendo el arraylist, cuales son las posiciones de los sujetos que tienen características guardadas y cuales no.

El valor introducido, no tiene porqué ser “false” y “true” fueron los valores elegidos pero se puede poner cualquier palabra con tal de que se diferencien con al menos de una letra.

Para acceder a esta lista para obtener y cambiar el valor cuando se guarden las características del sujeto correctamente, es necesario guardarla en memoria, de esta manera se puede acceder a sus datos en cualquier actividad con solo precisar su dirección de fichero. En esquema de la **figura 18** se puede ver como cuando se borra o se añade un elemento en la lista de sujetos afecta a la otra, y cuando los datos de las

características son guardados, se modifica el valor de la misma posición que el del sujeto en la lista de checks.

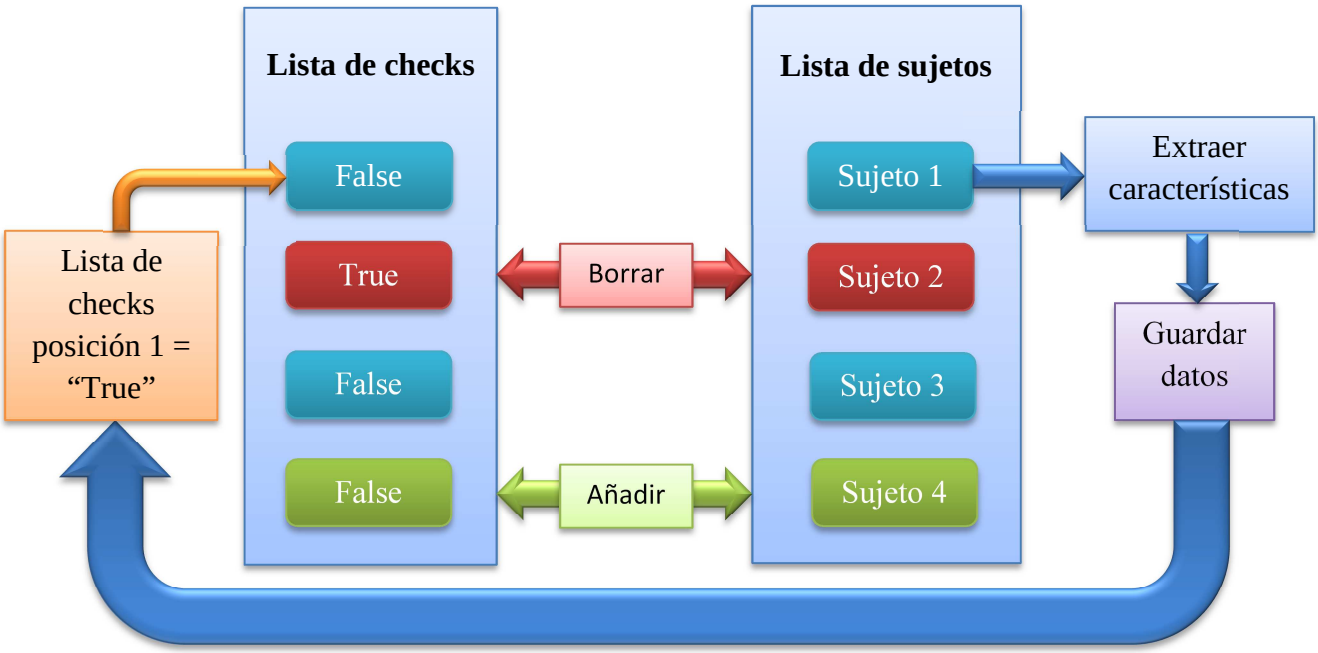

Figura 18: Esquema de interacción entre listas

Una vez que se tiene el valor de la lista de checks se puede asignar en cada elemento de la lista de sujetos el tick que indica que los datos están guardados con simplemente recorrer la lista y darle el valor según corresponda. De esta manera quedaría, en tick activado en el sujeto que tenga en su misma posición en el array de checks cuando su valor valga "true" y desactivado cuando valga "false".

5.5. Envío y recogida de datos entre actividades

Para poder enviar información de una actividad a otra, se ha utilizado dos métodos. Estos métodos han sido el uso de ficheros mediante la escritura y lectura de esos datos, y el envío mediante el uso de extensiones de datos al intent que inicia la actividad de destino.

En este punto se comentará únicamente este último puesto que más adelante se expondrá el primer caso con más detalle.

Este método se ha usado bastante durante el desarrollo de la aplicación, y es debido a que aunque se ha seccionado la aplicación en actividades que cumplen unos objetivos diferentes, todas están relacionadas unas con otras, conectadas como una red

de información. Esto es debido a que la utilización de los datos de otras actividades ha podido contribuir a desarrollar un código más estructurado y lógico.

Este uso de datos se puede ver por ejemplo en el intercambio de información entre la lista de sujetos y la extracción de características de cada sujeto marcado.

Cuando se selecciona uno de los sujetos de la lista de reclutados, este accede a la actividad de extracción de características. El problema que surgió es que se puede extraer las características del sujeto pero no resulta fácil el almacenamiento masivo de datos de una lista dinámica. Es por eso que al estar en la actividad de extracción de características resulta muy útil el nombre del sujeto, puesto que esta identificación resulta definitiva para poder usar ficheros de manera sencilla. La solución fue determinar la dirección de almacenamiento de datos para cada individuo según el nombre del sujeto.

Pero este ejemplo está muy estrechamente ligado con el apartado de almacenamiento y acceso de datos, y fue realizado posteriormente, por tanto, se desarrollará en ese punto.

El ejemplo por excelencia es el del envío de la imagen desde la actividad de captura en el menú de verificación hacia la actividad de procesado de verificación, ya sea la directa o la de por pasos.

El dato enviado es el del bitmap, pero como se comentó anteriormente, se le propicia una transformación a compresión PNG. Esta compresión se le aplica a un array, dicho array es el propio bitmap convertido. El objetivo de este cambio a array, es debido a que el envío de datos entre actividades se realiza normalmente de esta manera pues es mucho más fácil de operar y origina menos errores, además que optimizan mucho más el tiempo de traspaso que una matriz de datos como es el bitmap.

Se ha comprobado mediante pruebas la extensión del array convertido desde el bitmap con conversión en PNG, y se ha podido observar que el tamaño del array es menor que el alto por ancho de la imagen. De esta manera se comprueba efectivamente que se ha visto reducido, pero no se comprueba la calidad. Se ha realizado otra prueba que aseguraba que la calidad de la imagen se mantenía inquebrantable y con el máximo de calidad, mediante el paso de bitmap a array y continuadamente de array pasarlo de nuevo a bitmap. Además según especifica en la página de Google, Android-Development, si el formato es PNG se ignorará los valores de configuración de calidad.

El método usado para poder enviar los datos de una actividad a otra es añadiendo información extra al intent que abre la actividad. Para poder entender este proceso, hay que ver que el intent es el detonador que inicia la actividad, es como la dirección que hace que de una actividad se pueda ir a otra. Cuando se introducen datos en el intent para que sean enviados, hay que introducir una clave (key en inglés), es decir, una clave que haga diferenciar la referencia de ese dato frente a otro que envíes con él. El motivo de esto es que cuando se inicia la actividad de destino, esta puede

recoger los datos pero lo hace cogiendo todos los datos que le porta la otra actividad. Puesto que no es posible gestionar tanta cantidad de datos en una sola variable, es creado una variable para cada dato con el tipo que corresponda, y se le adjunta el valor del dato de la referencia aportada por la clave a la variable creada.

Para recoger los datos en la actividad de destino es necesario un bundle, es un mapa de memoria que puede almacenar varios tipos de datos, básicamente es como una cesta en la cual dentro precisamos de todos los datos. Posteriormente para poder diferencias un dato de otro, se hace referencia al dato mediante la clave. Es por esto que la referencia asignada al dato es tan importante, puesto que se mantiene constante independientemente de la actividad en la que se encuentre. A continuación en la **figura 19** se representa un esquema explicativo para su mejor entendimiento.

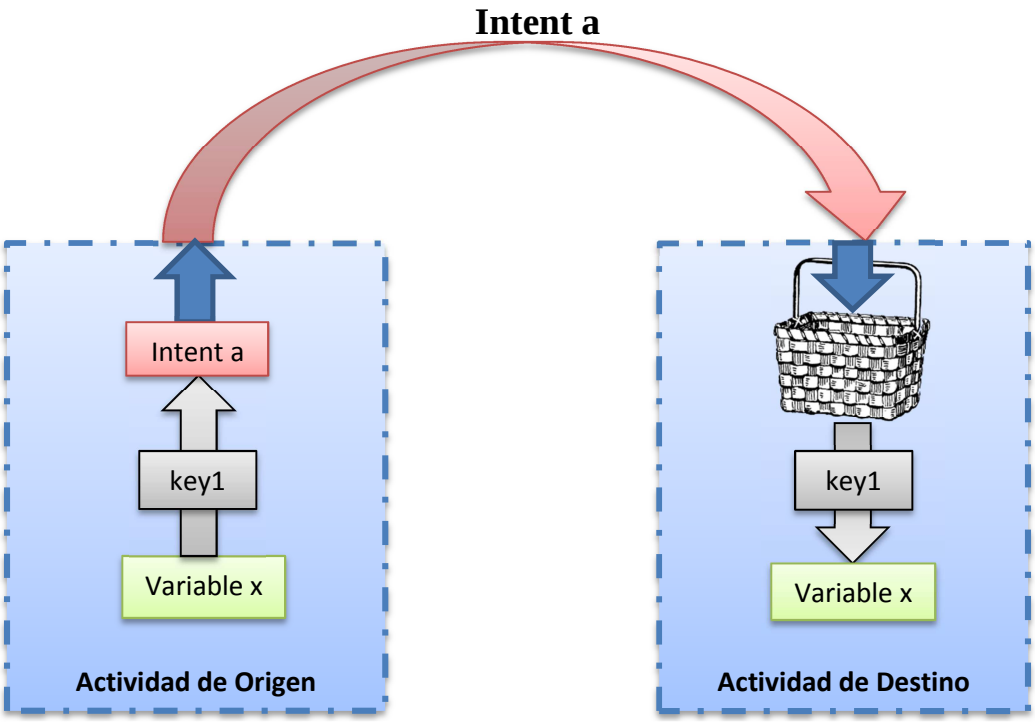

Figura 19: Esquema de envío de datos

El bundle también puede ser usado para almenar todas las variables que se ha de enviar desde la actividad de origen a la de destino. Pero se ha creído oportuno no usarlo puesto que la propia declaración ".putextras" ya dispone los datos de una manera distintiva de datos, y por tanto sería un paso innecesario aunque se comprobó que era válido.

Este método de comunicación entre varias actividades ha sido implementado para varios usos, como la gestión de los check de la lista que fue comentada anteriormente, o la ID del sujeto para mostrarlo en otras actividades y usarlo como dirección de memoria.

5.6. Manipulación de las imágenes

El uso principal de la actividad desarrollada es la manipulación de la captura para poder obtener los datos necesarios para su comparación con los de la base de datos. Es por eso que la manipulación de imágenes está incluida en gran parte del código referente a la extracción de características. Durante el comienzo del desarrollo fue necesaria la implementación de un código estable y fácilmente manipulable para poder incluir la imagen y poder procesarla a placer en las siguientes funciones. Es por todo esto que las primeras modificaciones de la imagen adaptada en un bitmap han de ser perfectas, es decir, sin minucias y sin modificaciones aparentes a la calidad de la imagen.

Varias pruebas fueron necesarias para determinar el sistema de conversión de bitmap a array y para comprobar las distintas aplicaciones a la conversión de la imagen, hasta conseguir procesos que no redujeran la calidad de la imagen.

Para ejecutar todas esas pruebas se realizaron varios procesos y se observaron sus calidades. Uno de ellos es la conversión de bitmap a array de bytes, posteriormente se ejecutaba una conversión a escala de grises y luego un escalado para comprobar que la obtención de datos de la imagen eran correctos. Se tuvo que realizar numerosas pruebas hasta determinar cual era el método más adecuado. Para explicar todos ellos se hará tres distinciones:

- Transformación a escala de grises.
- Obtención de la imagen en array de bytes.
- Pasar el array de bytes a bitmap.
- Cambio de la escala de la imagen.

Transformación a escala de grises: Una vez se ha obtenido la imagen y se almacena en un bitmap, el primero proceso que se efectúa sobre ella es la conversión a escala de grises. El objetivo de esto viene a que un ordenador puede trabajar mucho más fácilmente operando en grises, agilizando el proceso. Esto es porque el ordenador tiene que hacer frente únicamente a un cambio de contraste antes que a todo el código de color. Cualquier color es un entero comprendido entre 0 y 255, y al pasar a escala de grises lo que se consigue es tener el mismo entero en los tres códigos de color, rojo, verde y azul. Es decir, una imagen que predomine el rojo, en su código rojo, este valor será alto, mientras que el verde y azul no, pero si los tres colores son iguales pasa a ser una tonalidad de gris. Gracias a esto, es posible procesar imágenes con diferentes tonalidades de grises antes que analizar los 3 colores a la vez.

El paso a escala de grises se ha conseguido a partir de dos métodos. Ambos son aceptables pero tienen sus diferencias, según el objetivo que se desee se puede utilizar un método u otro.

El primer método fue mediante la obtención de la información de cada pixel, se desfragmentaba el código de color en rojo, verde y azul, y se almacenaba en variables.

A continuación se aplicaba un cálculo del contraste para cada pixel, que se rige por la siguiente ecuación:

$$luma = (int)(R \cdot 0.3 + G \cdot 0.59 + B \cdot 0.11)$$

Luma es el color de tonalidad de gris. Esta ecuación se obtiene al saber que si en los tres códigos de color sean cuales sean, cada uno se multiplica por un coeficiente que la suma de los tres de 1, este color pasa a escala de grises. Estos coeficientes pueden ser 1 en rojo y cero en verde y azul, obteniéndose también el color en escala de grises. La elección de los coeficientes viene ligado a la importancia del contraste de un color determinado según sea la captura de la cámara, por tanto puede ser diferente para otra cámara o sistemas, puesto que puede que capture mejor los colores azules, por ejemplo.

Este primer método es lento pero de alta calidad, puesto que se recorre uno a uno cada pixel y lo procesa independientemente.

El segundo método que se ha desarrollado, se originó debido al tiempo de procesado, por eso se ha desarrollado un sistema de paso a escala de grises más rápido. Este método básicamente se desarrolló creando una plantilla de imagen personalizada con el entorno canvas, del cual se pudo desarrollar un filtro matricial, que adaptándolo y aplicándolo al bitmap que se quiere convertir, se consigue la imagen en escala de grises. Esto se desarrolla en un tiempo inapreciable pero la calidad de la imagen es menor como se puede apreciar en la **figura 20** comparando ambos métodos (las imágenes seleccionadas no son de capturas oculares puesto que en la base de datos son todas en colores grises). En el aumento de las imágenes transformadas se puede observar como el segundo método tiene un contraste más escalonado.

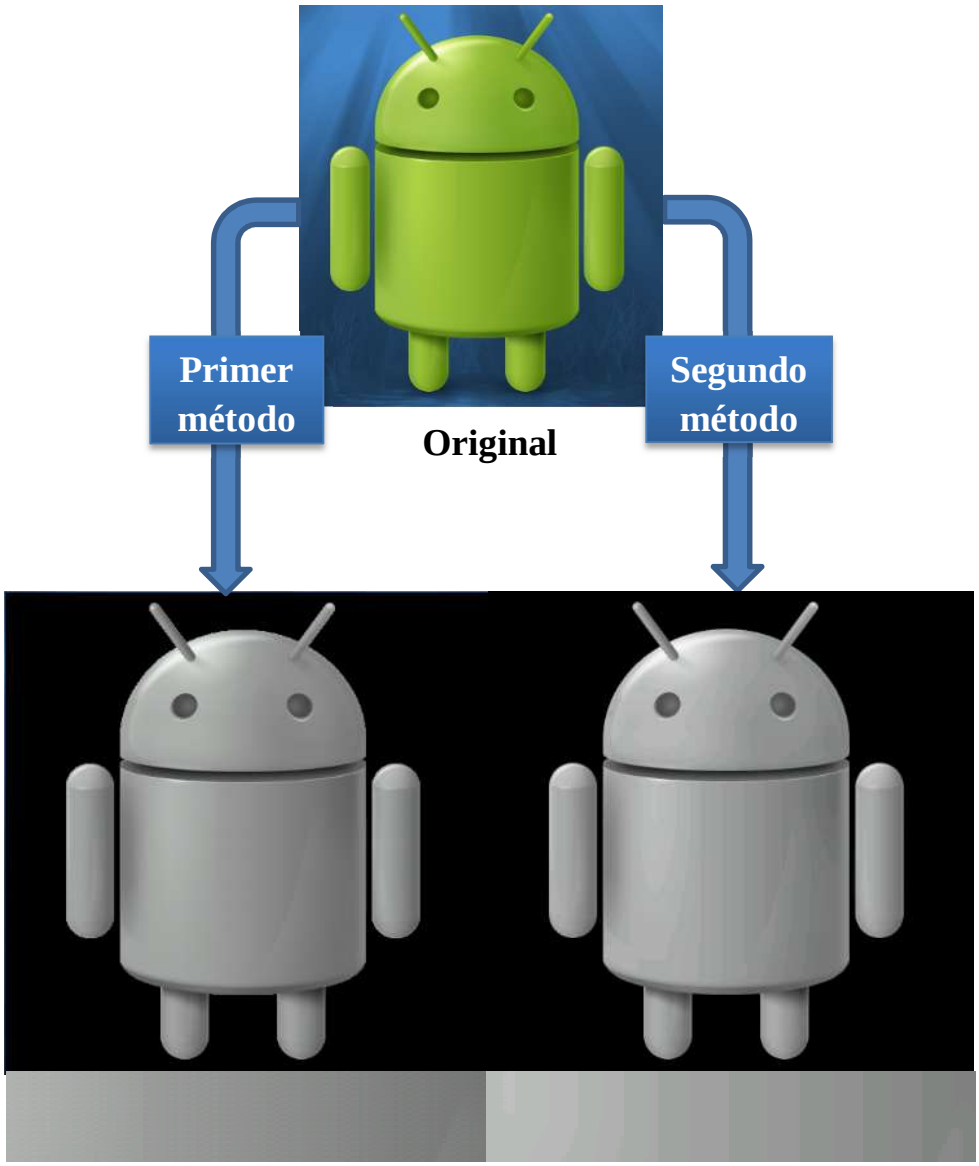

Figura 20: Métodos de escala de grises

Como se puede apreciar el primer método es con un acabado mucho más fino, mientras que el segundo se nota más brusco el cambio de tonalidad. Debido a la calidad obtenida por el primer método se descartó el segundo, puesto que si se quisiera usar este último se debería de aplicar dithering para suavizar la imagen.

Obtención de la imagen en array de bytes: Hay que determinar que no se podía realizar la transformación como se efectuó con el envío entre actividades descrito en apartado anterior. Esto es debido a que el array creado mediante el método de transformación PNG, como se comentó anteriormente, se reduce en tamaño, esto hace que no se pueda controlar la extensión del array y por tanto, no se pueda manipular aplacer. Es decir, si estoy buscando el color del byte perteneciente a la posición 630 en ancho y 250 de alto en la imagen, este color no será el mismo que el de la posición del

array $630 + ancho\_imagen \cdot 250$ ya que el tamaño del array debería de ser $ancho\_imagen \cdot alto\_imagen$ y no lo es, puesto que se le ha aplicado un algoritmo de compresión como se explico anteriormente. Esto origina una excepción al intentar acceder a la información de un dato con una dirección de memoria que no existe. Puesto que el tamaño no esta predefinido, y no era posible obtener la localización de cada pixel pues de cada imagen variaba pues fue necesario otro método.

Durante los siguientes métodos utilizados, todos se realizaron de una manera similar, la extracción de información del pixel de la imagen, mediante todo el recorrido en ancho y alto de la imagen. El recorrido de toda la imagen ocasionará aumento del tiempo de procesado, pero es el método más fiable y con menos error, en el apartado de pruebas se hablará con más detalle de este tema.

La obtención de la información de cada pixel es bastante completa en cuento a datos se refiere, es decir, dentro de cada pixel alberga el código de color que se almacena en una variable entero, para obtener cada código de color, ya sea rojo, azul, verde o la transparencia alfa, es necesario llamar a la función que desfragmente dicho entero obtenido del pixel.

En la extracción de color obtenido de cada pixel, como se ha comentado anteriormente, el sistema utilizado es el recorrido de todos los pixeles de la imagen uno a uno y se van posicionando en línea dentro del array. Puesto que el uso de pasar el bitmap a array es después de haber hecho la conversión a escala de grises, esto hace que sólo sea necesaria la obtención de un código de color, puesto que los tres son iguales. Se ha decidido obtener el del rojo e introducirlo en el array (Ver **figura 21**).

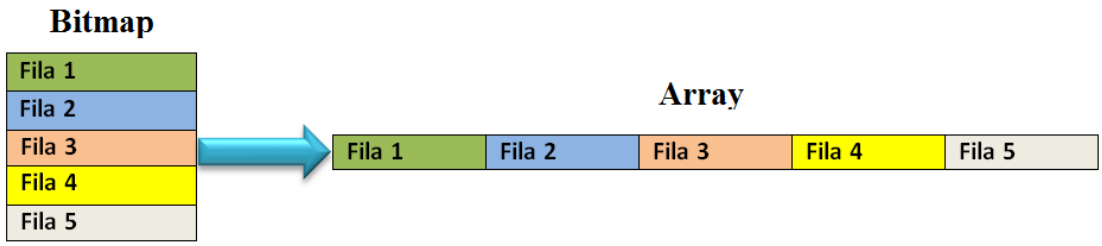

Figura 21: Conversión de Bitmap a Array

En esta figura se puede ver como se aplica la siguiente ecuación para obtener los datos del bitmap y se meten en el array:

$$array[x + ancho_{imagen} \cdot y] = color_{bitmap}(x, y)$$

Donde x e y son las coordenadas de la posición del bitmap. Las filas 1, 2, etc… son la coordenada y. Dentro de cada fila hay $ancho_{imagen}$columnas por cada una de ellas.

Pasar el array de bytes a bitmap: Para hacer esta conversión era necesario primero pasar a escala de grises, puesto que sino, no se precisa de la suficiente información por pixel. De todas maneras si se realizara la transformación únicamente cogiendo el color rojo, como se comento anteriormente, se conseguirá la escala de grises de la imagen. Esto es igual que multiplica el rojo por el coeficiente 1 y el rojo y el verde por 0.

Para hacer la conversión se han realizado varias pruebas, pero la mayoría de ellas ofrecían el mismo problema. Se generaba el bitmap a partir del array mediante el uso de bibliotecas de Android, esta transformación era rápida y prácticamente inmediata, pero en cuanto se intentaba modificar el bitmap introduciendo un valor a un pixel cualquiera se generaba una excepción en la aplicación y se bloqueaba.

El problema principal era que no se conseguía crear un bitmap mutable con el valor del array, todos los generados por biblioteca de forma automáticas era inmutables, es decir, no se podían modificar una vez fueran creados.

Se ha resulto de dos posibles formas, una de ellas es la siguiente:

```
Bitmap bmSource = BitmapFactory.decodeByteArray(arraySource, 0,
                          arraySource.length);

// Paso el Bitmap creado (Imutable) a mutable
Bitmap bmSourceMutable = bmSource.copy(Bitmap.Config.RGB_565, true);
```

Como se puede ver en el código se usa la biblioteca de bitmap para poder decodificar el array directamente a bitmap, pero el bitmap llamado bmSource es inmutable y por tanto no es útil en la aplicación. Esto se soluciona en la siguiente línea de código donde se crea un nuevo bitmap y se rellena con la información del bitmap inmutable, el valor de “true” es para indicar que se devuelva un bitmap mutable.

La otra opción es más procesada pero igual de válida:

```
// Se crea un bitmap mutable y se introduce los datos pixel a pixel.
Bitmap bmp = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
int R, pixel;
for (int y = 0; y < height; y++)
      for (int x = 0; x < width; x++) {
            pixel = data[x + width * y];
            R = pixel & 0xFF;
            bmp.setPixel(x, y, Color.rgb(R, R, R));
      }
```

El procedimiento es muy similar al utilizado para el escalado de grises que se ha explicado anteriormente. Básicamente se crea un bitmap mutable y se recorre todas las direcciones del array y se les van añadiendo al bitmap creado.

Cambio de la escala de la imagen: Finalmente cuando se ha desarrollado todos los pasos anteriores para poder comprobar finalmente si la manipulación de las imágenes es correcta y funciona a la perfección, se realiza el escalado de la imagen y se representa por pantalla, de esta manera se ha conseguido demostrar que el cambio a

escala de grises se aplica correctamente, la transformación a array desde bitmap y viceversa para conseguir un bitmap mutable, que se demuestra cambiando la escala del mismo.

Además este escalado es necesario para agilizar los procesos al máximo puesto que al tener menos pixeles se puede analizar en menos tiempo. Esto no afectará al resultado total, puesto que como se indico desde el principio, la calidad y su resolución son muy importantes, únicamente se ha usado la imagen escalada para los procesos más pesados y cuando se ha detectado y filtrado las opciones necesarias, se hace un análisis más exhaustivo con la imagen de más calidad. Todo este proceso se ha explicado anteriormente cuando se hablaba de la biometría.

5.7. Uso de bibliotecas en C#

Para el desarrollo de la aplicación ha sido necesario utilizar unas bibliotecas proporcionadas por el tutor del proyecto. Estas bibliotecas son fundamentales para el desarrollo de la aplicación puesto que son las que hacen posible el procesado de la imagen para su extracción de características.

Para implementar esta biblioteca que se encuentra en C# ha sido necesario un entorno de desarrollo necesario para su manipulación, y una transformación al lenguaje utilizado por Android, el Java. El entorno de desarrollo utilizado para la lectura del código de las bibliotecas y de la ejecución de la prueba se realizó con Visual Studio 2010. A su vez fue necesario la implementación de otra biblioteca como sustitutivo de la utilizada en C# llamada AForge, llamada OpenCV, este tema se comentará en el próximo punto.

5.7.1. Funcionalidad de la biblioteca

Junto con la biblioteca se proporcionó un proyecto Test de uso de la biblioteca, el cual únicamente estaba desarrollado para localizar los bordes interiores y exteriores y pintarlos para luego mostrarlo por pantalla. No se suministró aplicación y comprobación de la parte de extracción de características, ni de comparación.

La idea de desarrollo inicial fue la aplicación de cada uno de los pasos en la plataforma Android y comparar resultados, estos deberían de ser aproximadamente los mismos. Se observaron diferencias en cuanto la creación del histograma, en el caso de la plataforma Android, los histogramas registrados por la misma imagen se presentaban con mayor muestreo que el ofrecido por C#. Esto puede ser debido a que Android es un sistema desarrollado para el uso de imágenes y mostrado en una alta calidad, por tanto, es necesario un barrido de imagen más exhaustivo que en otra plataforma.

El problema surgió con en el momento de comparar datos más concretos, ya que en el proyecto de prueba únicamente se incluía la biblioteca como una referencia .dll, por lo que sólo era posible acceder a los datos de variables globales sin saber como reaccionaban las variables locales de cada función usada. Esto era fundamental puesto que se ha tenido que traducir todas las bibliotecas a Java, puesto que no es un lenguaje compatible con Android, mientras que otras como C y C++ sí lo son como se comprobará en el apartado de las bibliotecas usadas de OpenCV.

5.7.2. Compatibilidad con Java

La compatibilidad de las bibliotecas de C# en entorno Java son nulas, por tanto fue necesario una traducción de lenguaje. Gracias al entorno de desarrollo, Visual Studio, fue posible entender el uso de muchas funciones propias de C# con solo disponer el cursor encima de la declaración de la misma, haciendo de esta manera aparecer un menú con la descripción del proceso. Por tanto buscando las funciones en java que cumplan los mismos requisitos y objetivos, y adaptarlo al código bastaba.

El problema surge con la enorme diferencia de Java frente a C#, y es que Java no tiene unsigned (sin signo), es decir, no tiene variables de no contengan bit de signo. Esto repercute en gran medida en el desarrollo para el uso de estas bibliotecas.

5.7.3. Semejanzas y diferencias con Java

Hay una gran semejanza en el código de C# con java, generalmente suelen ser pequeñas diferencias como puede ser usar mayúsculas en C# para ciertas sentencias y declaraciones cuando en Java no. También las estructuras de las funciones que son diferentes de un código frente a otro, por lo que aunque el uso de una declaración sea la

misma puede que el orden de las variables que contiene el constructos sean diferentes y esto afecte gravemente al correcto funcionamiento del mismo. Muchas referencias a constructores son propias y únicas en el lenguaje C# por lo que es necesario buscar una semejante en Java para que cumpla la misma, e incluso se ha implementado en casos dos funciones en Java para hacer lo mismo que una en C# y viceversa.

Pero como se ha comentado anteriormente la principal diferencia de estos dos lenguajes es el uso del unsigned. Puesto que en Java no existen las variables sin signo y en el uso de todas las variables de la aplicación no vamos a usar el signo negativo, ese bit de memoria lo estamos desperdiciando.

Durante todas las bibliotecas en C# es usado un array de bytes para poder operar en todas las funciones del reconocimiento de características como se ha comentado anteriormente. Y como hemos podido comprobar, se puede pasar sin problemas de bitmap a array, pero la cosa cambia cuando especificamos que tipo de array se desea. Teniendo en cuenta que es necesario un array de una variable que contenga la capacidad necesaria para albergar un número comprendido del 0 al 255 en cada posición, se puede llegar a la conclusión que un entero podría valer pues este puede albergar hasta como máximo el número $2^{31} - 1 = 2147483647$ y como mínimo el $-2^{31} = -2147483648$.

Se podría desarrollar pues todo el uso de la biblioteca de esa manera, usando un array de enteros, pero resulta que debido a la enorme cantidad de posiciones que para las imágenes usadas son $640 \cdot 480 = 307200$ posiciones de las cuales dentro de cada una de ellas se usarán solo los 255 primeros número de los 4294967295 posibles. Esto es un derroche de memoria y ralentiza enormemente el dispositivo y el procesado, puesto que necesita una cantidad gigantesca de memoria cada vez que quiere mover el array o simplemente mantenerlo en la memoria como variable global.

Es por esto que se piensa en el byte como se usa en la biblioteca de C#. La variable byte es de 8 bits de extensión por tanto, se pude tener 256 valores, haciendo que vaya de 0 a 255. Pero en Java el último bit es de signo, haciendo que sea utilizable sólo los 7 primeros. Para poder usar esta variable y así poder utilizar únicamente la memoria que se necesite una conversión que identifique los valores negativos y los transforme en positivos cuando se tenga que extraer el valor. Solo es necesaria esa transformación al sacar el valor del array, por lo que serán valores del 0 al 127 y negativos del -128 al -1 que sumando ambos tramos es del 0 al 255 como se quería en un principio. Esta transformación es la llamada complemento a dos.

Para solucionar esto se ha desarrollado dos métodos. Uno de ellos se realizó con una concatenación de condiciones. Cuando el valor es positivo se deja como está, pero cuando es negativo se realiza la transformación de que si el valor es -128 entonces es 128 y si el valor es otro negativo entonces se hace $256 + número\_negativo$. Este razonamiento se puede ver y seguir fácilmente la **tabla 4** de verdad del complemento a dos para cuatro dígitos.

Decimal	Complemento a dos
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Tabla 4: Complemento a dos de 4 bits [6]

El segundo método, y el que se usó finalmente, es el de uso de memoria para reubicar el byte de signo y desplazarlo, haciendo esto se consigue leer directamente el valor del byte sin signo. Esto es debido a que el bit de signo se encuentra a más de la octava posición y esta es imposible de alcanzar con un valor de 1 si el byte solo llega hasta la posición 8.

```
pixel = data[x + width * y];
R = pixel & 0xFF;
bmp.setPixel(x, y, Color.rgb(R, R, R));
```

Como se puede comprobar en el código de arriba, se obtiene el valor del array de bytes llamado data, se mete en una variable entera después del desplazamiento de memoria y posteriormente se incluye en el bitmap el valor del color en sus colores rojo, verde y azul. De esta manera se soluciona el problema.

5.7.4. Entorno de desarrollo Visual Studio 2010

Para la gestión del lenguaje C# se usa el entorno de desarrollo Visual Studio 2010, en el cual ha sido posible observar los valores obtenidos con los análisis de las imágenes en el proyecto test ofrecido, mediante el uso del depurador y comparados posteriormente con el desarrollado en Android.

En este entorno no se ha trabajado mucho, únicamente observar las definiciones que aporta el programa sobre el código e implementar unas líneas de código adicional para poder obtener imágenes del resultado después de usar cada uno de los filtros usados de la biblioteca AForge, para observar su comportamiento. Estos han sido filtros de escala de grises, de implementación de Canny para detectar el borde, filtro de Threshold para cambiar el color de la imagen a negro o blanco y el filtro de eliminación de minucias. Para implementar estos filtros ha sido necesario la biblioteca OpenCV con algunos cambios a los utilizados directamente en la biblioteca de C#.

5.8. Uso de bibliotecas OpenCV

Como se comentó anteriormente se ha necesitado usar estas bibliotecas para poder aplicar los filtros necesarios para la implementación de las bibliotecas en C#. Para poder aplicar estas bibliotecas ha sido necesario descargar los proyectos de las bibliotecas desde la página de sourceforge.net, la cual contenía numerosos ejemplo de aplicación de algunos de sus métodos que resultaron muy útiles para su correcto uso.

Al implementar la biblioteca en la solución propuesta en este proyecto (Eye_Locker) se ha descubierto que existe la función de filtrado de Canny que cumple con los mismos objetivos que el filtrado de Canny junto con el Threshold de AForge en C#. Esto es debido a que en C# el filtro de Canny únicamente marca el borde con el color de la imagen y lo demás lo pasa a negro, que finalmente gracias al filtrado de Threshold depura dejando de forma más significativa el borde de la imagen, cambiando la escala de colores. Se ha descubierto en Java en la biblioteca OpenCV un filtro de Canny que cumple con esa condición final a la salida de Threshold, haciendo innecesaria su aplicación.

Finalmente en C# se aplica un filtro de eliminación de minucias llamado filtro Blob. Este filtro elimina los puntos de imagen que tengan un tamaño en conjunto menor que el designado por el programador, y con una codificación de color especificada como se puede observar en la **figura 22**. El problema es que dicho filtro no se encuentra en la biblioteca y es difícil de implementar en Java, aunque hay una biblioteca disponible en internet, pero se ha decidido no incluirla por dos motivo:

- **Por el tamaño de dicha biblioteca:** El tamaño se comprobó que era mayor que el de OpenCV y sólo iba a ser necesaria una única función, por lo que se incrementaría el tamaño de la aplicación y del proyecto.

- **No es realmente fundamental:** Este filtro es para depurar al máximo la imagen, pero no es fundamental, puesto que únicamente es para generar la máscara que determina en que intervalo hay que leer los datos.

Como se puede ver en la **figura 22** se puede observar como el uso de filtros en las dos bibliotecas originan resultados parecidos aunque diferentes en forma, pero ambos métodos tiene las mismas especificaciones de un borde claro y usando únicamente los colores de negro (número 0) y blando (número 255).

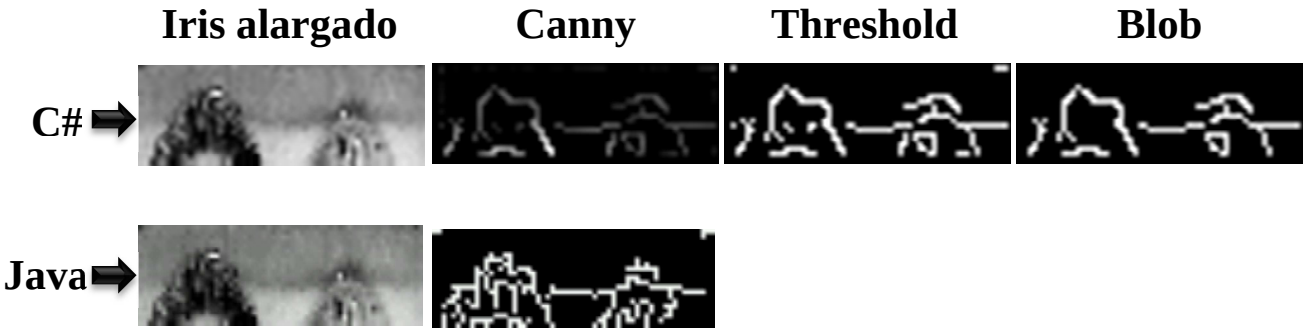

Figura 22: Distinción de funciones en C# y Java

Los valores del umbral especificados en la función de Canny que se ha usado en la aplicación han sido calculados de nuevo, puesto que los especificados en la biblioteca C# son completamente diferentes a los de OpenCV, haciendo que estos datos sean incompatibles. Los datos que mejor resultaban daban después de numerosas pruebas fueron los de 170 para el primer umbral para el procedimiento de histéresis y 270 para el segundo.

Estos umbrales determinan en el filtrado de Canny que si el valor del pixel analizado es:

- **Mayor que el valor de umbral máximo:** El pixel es considerado como borde genuino.
- **Menor que el valor de umbral mínimo:** Se elimina como borde automáticamente.
- **Entre el menor y mayor valor de umbral:** Si tiene un borde genuino o un posible pixel de borde al lado se conserva como borde.

De esta manera al final queda una imagen binaria donde cada pixel está marcado, ya sea por un pixel de borde o como uno de no borde.

5.8.1. Compatibilidad con Java

Las bibliotecas están escritas en C y C++ pero es completamente compatible con Java, puesto que dispone de sistemas de conversión de datos muy útiles y necesarios para la utilización de algunas funciones.

En el filtro de Canny usado por biblioteca de OpenCV, fue necesario convertir los bitmap al formato en C++ que es el Mat, una clase de matriz densa de n-dimensiones. Una vez transformado mediante el uso de funciones que incluye las bibliotecas de OpenCV, se pudo aplicar el filtro de Canny correctamente. Puesto que el resultado es un Mat, ha sido necesaria una nueva transformación a bitmap con el uso de otra función específica.

5.8.2. Implementación con Android

Para poder implementar estas bibliotecas al proyecto y posteriormente a la aplicación de Android, ha sido necesario varios pasos.

El primero ha sido cargar los proyectos de biblioteca en el entorno de desarrollo d Eclipse. Junto con las bibliotecas fueron cargados proyectos de ejemplo de aplicación de algunas funciones en aplicaciones Android.

Ha sido necesario cargar en el proyecto la función del otro proyecto como biblioteca de esta aplicación, para poder utilizar sin problemas sus declaraciones (Ver **figura 23**).

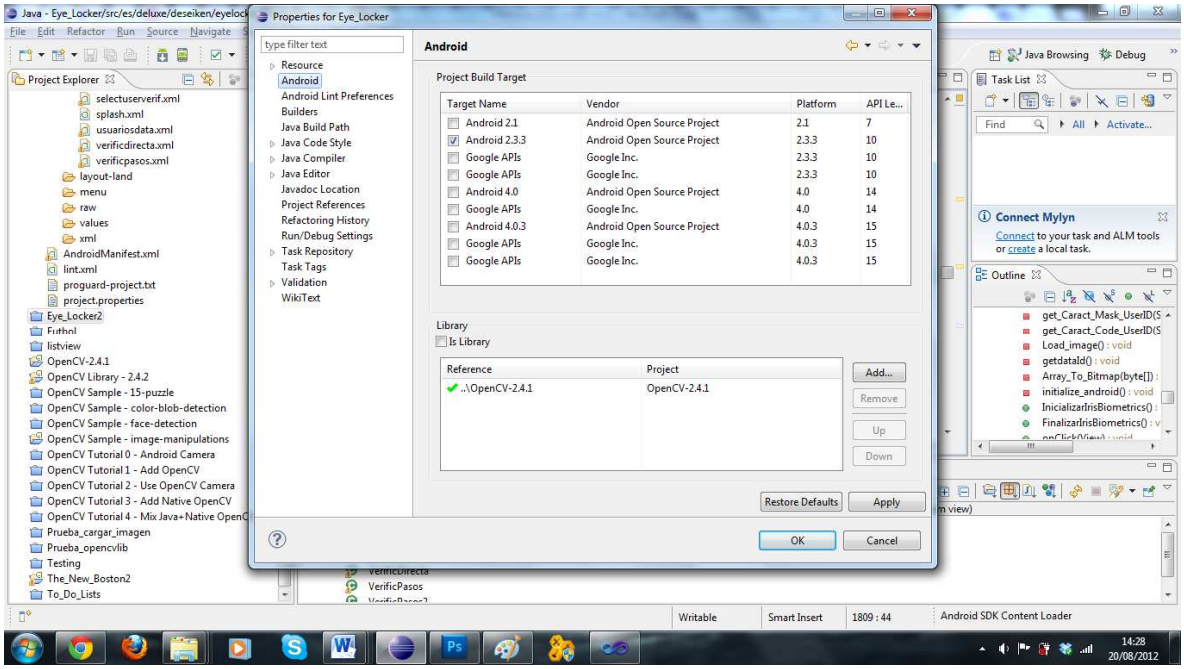

Figura 23: Implementación de bibliotecas

Una vez realizado esto, ha sido necesario importar en el código, en la actividad necesaria, las funciones de las bibliotecas a usar. Estos import en el código son las importaciones necesarias de bibliotecas propias o externas a Android que le indican la dirección de tiene que seguir para buscar ese constructor que se ha declarado en la actividad.

```
import org.opencv.imgproc.Imgproc;
import org.opencv.core.Mat;
```

De esta manera quedaba completamente incluida la biblioteca en el proyecto de la aplicación Android.

5.8.3. Uso en dispositivos móviles

Para poder trabajar con estas bibliotecas en dispositivos Android, ha sido necesaria la instalación de un controlador de OpenCV para el móvil. Este fue el OpenCV Manager que se pude encontrar en la página:

https://play.google.com/store/apps/details?id=org.opencv.engine&feature=more_from_developer#?t=W251bGwsMSwxLDEwMiwib3JnLm9wZW5jdi5lbmdpbmUiXQ..

El cual, una vez se ha arrancado en el dispositivo determina que tipo de pack se debía descargar para poder utilizar correctamente las bibliotecas.

5.9. Almacenamiento y acceso de datos.

Durante varios procesos de las diferentes actividades es necesario acceder a registros de memoria para almacenar u obtener datos. Esto se realiza de dos formas diferentes:

- De forma pasiva: No se ha pulsado ningún botón o ningún detonante para realizar la actividad de obtener datos o guardarlos, se hace de forma automática y sin que se entere el usuario.
- De forma activa: Se ha pulsado un botón o detonante para realizar la actividad ya sea de obtener datos o de guardarlos, el usuario sabe que se ha accedido a memoria.

Se realiza el acceso a memoria de forma continuada, ya sea para obtener las características de un candidato o para almacenar datos referentes al estado de la lista de sujetos.

Se podría decir que se ha utilizado exclusivamente para: la lista de sujetos y lista de checks. Dentro de cada uno de los candidatos en la lista de los sujetos se guarda también los datos y la máscara de cada uno de ellos. Por tanto queda almacenado la siguiente información de la siguiente manera (Ver **figura 24**):

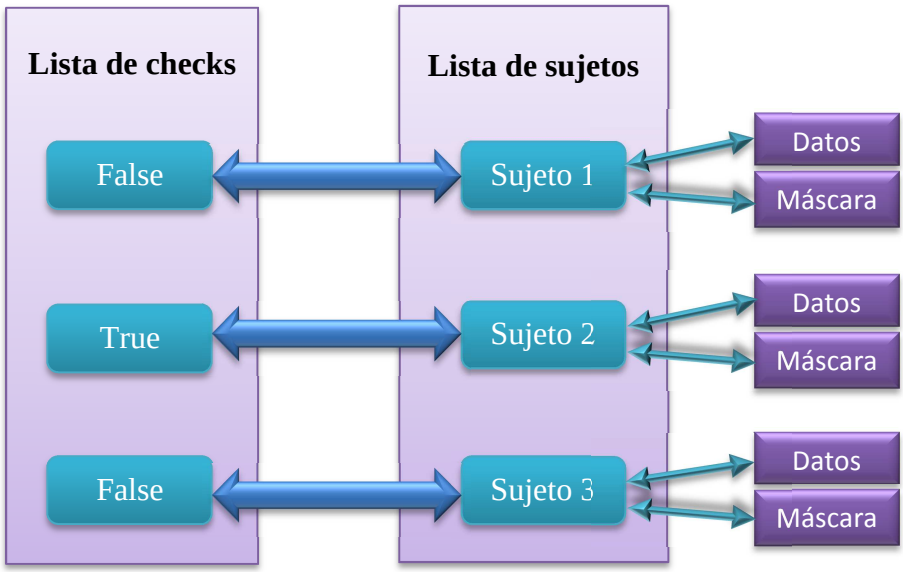

Figura 24: Esquema de guardado de características

Se ha podido desarrollar esta estructura mediante el uso reiterado de las funciones de acceso a lectura de ficheros en memoria y el guardado. Para ello es necesario usar un comando que “abra” la dirección de memoria que se quiere acceder. Primero se ha creado la función que guarda los datos:

```
private void saveToFile_code(byte[] data) {
	// Se guarda en un fichero que se encuentre en UserID
	try {
		FileOutputStream fOut = openFileOutput(FilePath_Code_UserID,
					Context.MODE_PRIVATE);
		fOut.write(data);
		fOut.close();
	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

Como se puede ver en el código, primero se crea el fichero y se abre para poder guardarse dentro. Al crearse el fichero se ha tenido que especificar la dirección y el tipo de almacenado. La dirección se especifica simplemente con un string con el nombre que se quiera. En el modo de guardado se ha elegido un almacenamiento privado de la aplicación, es decir, los datos no son visibles por ningún medio ni actividad si no es la que genera el fichero, de esta manera se asegura su protección.

Para usar esta función solo hace falta llamarla e introducir en el constructor el array de bytes que se desea guardar.

La forma de poder guardar los datos del código y la máscara de cada uno de los sujetos es mediante la asignación de una dirección de memoria diferente. Esto se consigue utilizando el nombre del sujeto como parte de la dirección de memoria. Es decir, en la aplicación, la dirección asignada para guardar la mascara y el código de datos de un sujeto cualquiera es:

- Para el código: UserID + "_code". Donde UserID es el nombre del sujeto de la lista que se ha seleccionado.
- Para la máscara: UserID + "_mask".

De esta manera se ha asegurado poder guardar y acceder fácilmente a cada uno de los datos de cada sujeto sin problemas. Además se ha conseguido de esta manera que cuando sea necesario borrar el sujeto y todos sus datos con el, se accede a esa dirección y se guarda dentro de ella null (nulo, hace referencia a la nada) sobrescribiendo todos los datos.

Para poder leer los datos guardados, es necesario solo el nombre del sujeto puesto que para sacar la dirección de la máscara o la de datos es el mismo procedimiento. A continuación se incluye el código correspondiente a la lectura del fichero:

```
public String readFile(String fileName) {
	String dataget = null;
	try {
		BufferedReader input = new BufferedReader(new InputStreamReader(
					openFileInput(fileName)));
		dataget = input.readLine();
```

```
            input.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return dataget;
}
```

Como se puede ver en el código es necesario únicamente introducir la dirección del fichero que se desea leer, y este accede y cargar todos los datos hasta un salto de línea que posteriormente se almacena en la variable que se devuelve en forma de string del constructor.

Pero los datos necesitados son en array de bytes y no en string, por lo que es necesaria una conversión usando las bibliotecas de Android es un sencillo paso. Según corresponda se usará para convertir los datos a array de datos o a lista de arrays de strings.

5.10. Detalles y protección

Para poder dar una apariencia más atractiva a la aplicación y poder hacerle un uso más comercial se han desarrollado algunos retoques. Estos han sido mensajes que le aparecen al usuario para guiarle por los menús, protección frente a contraindicaciones de la aplicación, sonidos y configuraciones no esenciales, como menús inflables y control de los sonidos de la aplicación.

5.10.1. Protección frente a posibles errores del usuario

Aunque durante todo el desarrollo de la aplicación, se intentó asegurar el correcto entendimiento de los pasos por parte del usuario, se incluyo en varios pasos significativos, una protección frente a errores del usuario.

Se han dispuesto estos sistemas únicamente en los puntos que podrían originar errores que obliguen el bloqueo de la aplicación y su posterior cierre por parte de la plataforma Android. Hay sistemas de protección, por ejemplo, cuando se extraen las características de un sujeto y se quieren guardar, solo se podrán guardar cuando se haya incluido una imagen, posteriormente se haya pasado el control de calidad y finalmente se haya extraído las características, si todos los pasos fueron correctamente procesados entonces se puede guardar en la memoria, sino en cuanto se intentara se avisaría al usuario del evento necesario para su guardado.

Estos errores son notificados y mostrados por pantalla durante un poco tiempo, si el texto es corto o durante más tiempo si es de mayor tamaño. Estos mensajes llamados toast, tienen la ventaja de que pueden ser retirados inmediatamente si se pulsa

justo encima del cuadro donde se encuentran. Además pueden desplegarse en pantalla en cualquier momento.

5.10.2. Sonido

Se ha incluido en la aplicación pequeñas reproducciones de audio para indicar cambios entre actividades, al igual que un sonido de introducción en el menú splash de la aplicación. Estos audios pueden ser reproducidos como sonidos de corta o larga duración y se pueden activar de múltiples maneras, con pulsar un botón, cuando inicia o termina un proceso, o cuando se pulsa en la pantalla sea donde sea.

5.10.3. Configuraciones no esenciales

Se han incluido en algunas actividades un menú inflable con algunas características disponibles. Una de ellas es el “About Us” donde se indica en texto el origen de la actividad, un botón de “Exit” para salir de la aplicación de inmediato y un botón de “Preferences” donde se va a un menú de preferencias que el usuario puede configurar, para por ejemplo, quitar el sonido del splash nada más iniciar la actividad. Pero todas estas configuraciones no repercuten en absoluto al transcurso normal de la actividad.

5.11. Desarrollo actividades

Es una explicación más exhaustiva desde el punto de vista del desarrollo de la actividad, donde se explica la administración de los procesos que se usan para poder llegar a ejecutarlos correctamente, en la que se tendrá más en cuenta la estructura de código usado.

Se ha creído conveniente comentar únicamente las dos actividades de verificación, puesto que son las de más desarrollo de código y las más enrevesadas donde ha resultado complicado estructurarlo para poder gestionar todas las variables globales correctamente y sin fallo.

Dentro de ambas se encuentra el sistema de calidad de la imagen, por tanto para evitar ser comentada dos veces se explicará aparte aunque pertenezca y este incluida dentro de las actividades mencionadas.

5.11.1. Verificación por pasos

Una vez se tenga todas las bibliotecas traducidas e incluidas, y todas las funciones listas para ser usadas, es necesario un código que pueda recopilar el uso de todas esas funciones y que se vayan activando según se produzcan ciertos acontecimientos que tu defines y los muestras por pantalla.

Puesto que esta actividad se trataba de una actividad dinámica, es decir, que lo mostrado por pantalla iba a cambiar durante cada uno de los procesos entonces se tenía que disponer de una codificación que fuera posible el fácil cambio de las imágenes y textos mostrados según en la etapa del proceso que se encontrara. Para realizar esto se había pensado inicialmente una concatenación de condiciones que se iban enlazando unas con otras, esto resultó, pero la codificación no estaba ordenada y era caótica. Finalmente se ha dispuesto de una estructura de switch-case, en cual es simplemente comparar una variable que se desee y en el caso de que valga el valor que aparece entonces se hace ese proceso, sino pasa al siguiente hasta encontrarlo. Una vez realizado su proceso se sale de la estructura para evitar tardar en seguir mirando.

Para poder usar esta estructura fue necesario crear una variable que estuviera inicializada a 0 y que cada vez que se pulsara el botón de “Siguiente” sumara en 1 su valor, de esa manera, en el primer proceso la variable vale 1, en el segundo vale 2 y así sucesivamente. Por lo que se podía ir remodelando la pantalla a placer, sin necesidad de crear una actividad por cada proceso que se necesitara.

5.11.2. Verificación directa

Esta actividad ha sido muy fácil de implementar una vez fue desarrollada la de verificación por pasos. En esta actividad no fue necesaria la implementación de una estructura tan compleja como la anterior para iniciar los procesos, bastó con disponer de una función llamada “MegaProceso” la cual incluía todo los pasos necesarios para la extracción y comparación de características. Esta función se iniciaba una vez se pulsara el botón de “Comenzar”, pero a mitad del proceso se ha incluido la comprobación de calidad, la cual si no la pasa, se interrumpe el proceso de verificación y da paso al registro de error.

5.11.3. Sistema de procesado de calidad

En ambos procesos de verificación y en la extracción de características de cada sujeto se han incluido un sistema de procesado de calidad. Este sistema es fundamental para aceptar la imagen como válida y no dar falsas verificaciones. Para ello se ha practicado varios análisis:

- ✓ **Calidad de sharpness (enfoque):** Se analiza si la imagen está desenfocada.
- ✓ **Calidad escala de imagen:** Analiza los bordes existentes entre el iris y la imagen.
- ✓ **Calidad contraste pupila:** Mide los valores del contraste entre la frontera del iris y pupila.
- ✓ **Calidad contraste iris:** Mide los valores del contraste entre la frontera del iris y esclerótica.
- ✓ **Calidad escala de grises:** Analiza los niveles de gris en la imagen.
- ✓ **Calidad tamaño de iris:** Analiza el diámetro del iris.
- ✓ **Calidad ratio entre pupila e iris:** Analiza la relación entre el tamaño del iris y el de la pupila.
- ✓ **Calidad distancia de centros:** Analiza la distancia entre los centros de la pupila y el del iris.
- ✓ **Comprobación contra fraude:** Analiza si la imagen es o no fraudulenta.

Todas estas comprobaciones de calidad se han practicado por varias funciones, las cuales son pasadas una a una. Pero para poder gestionarlas todas, ha sido necesario crear una variable global de cada uno de los errores posibles inicializado en 0. Si al pasar un test de calidad, este falla y no lo pasa la variable de error correspondiente con esa función adquiere el valor 1. Después de pasar todos se suman y si el valor de error es 1 ó mayor entonces se activa el registro de error.

Para poder obtener los errores causados en la actividad de representación de errores en pantalla, se ha optado por el uso de un array de bytes, del que cada posición representa un tipo de error, si el valor de esa posición es 0 es que no contiene ese error, si por el contrario es 1 esto indica que tiene ese error. Por el método de pasar datos de una actividad a otra a través del intent ha sido posible enviar esa información a la actividad final que recoge los valores de errores y muestra por pantalla lo que contenga.

5.12. Android Manifest

El manifiesto de Android es el elemento más importante de cualquier aplicación. Es como el contrato escrito en código entre programador, dispositivo y usuario. En él se dispone las actividades que van a estar activas en la aplicación y que pueden contribuir en modificaciones del dispositivo.

Si se quiere tener acceso a ciertas aplicaciones del dispositivo durante el uso de la aplicación y que esta pueda gestionarla, se ha de pedir permiso al dispositivo a través de este documento. En esta aplicación únicamente ha sido necesario pedir permiso para poder usar la cámara del móvil.

```
<uses-feature android:name="android.hardware.camera" />
```

También al ser creada la aplicación con el sdk 10 es decir para dispositivos con Android 2.3.3 (Gingerbread) o superior, debido al uso de las bibliotecas de hasta esa versión, se necesita también restringir los dispositivos que no tengan esa versión o superior mediante la siguiente línea de código:

```
<uses-sdk android:minSdkVersion="10" />
```

Como se comentó anteriormente, se decidió bloquear todas las actividades en vertical (portrait). El motivo de esto es debido a que cuando se muestra una imagen previa en la pantalla, si el dispositivo de gira, la actividad se reinicia y se muestra en la casilla de vista de la imagen, la imagen por defecto y desaparece la cargada. Esto se lograba con esta línea de código por cada actividad:

```
android:screenOrientation="portrait"
```

En todo el Android manifest se expone todas las actividades que participan en la aplicación, puesto que todas son de estructura prácticamente similar se dispone a continuación de un ejemplo de una de ellas:

```
<activity
    android:name=".Splash"
    android:label="@string/app_name"
    android:screenOrientation="portrait" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

La única diferencia de esta actividad frente a las secundarias, es que esta es la de arranque como se puede ver en el ".LAUNCHER". Las demás actividades, puesto que son activadas mediante detonadores de otras, se exponen como ".DEFAULT", especificando de esta manera que se mantengan activada o desactivada, pausada o parada, según corresponda, es decir, sin denominarlo anteriormente al inicio de la aplicación.

6. Pruebas finales

Este apartado contiene las pruebas realizadas sobre la aplicación una vez terminada, puesto que fueron numerosas las pruebas realizadas durante su desarrollo y sólo las más significativas ya han sido comentadas anteriormente.

El objetivo de esta prueba final es la correcta identificación del sujeto con todas las imágenes del mismo ojo disponibles y el rechazo de las del otro ojo (puesto que son siempre diferentes). A su vez se probará si la captura de otros sujetos en posible apariencia de contorno del ojo al original da negativo o no.

Todas las pruebas, tanto las de desarrollo como la final, han sido desarrolladas en un dispositivo Android. El dispositivo usado ha sido el Samsung Galaxy SII GT-I9100 con la versión de Android 4.0.3 Ice Cream Sandwich, aunque durante el desarrollo de la aplicación gran parte del tiempo fue con el Gingerbread 2.3.3 y posteriormente 3.2.6. Puede que debido a su doble núcleo la duración de sus procesos sean más cortos que los efectuados por un móvil de menores prestaciones. Por esto ha sido probado en diferentes dispositivos móviles, con diferentes versiones de Android.

El motivo por el cual no se ha querido hacer pruebas durante el desarrollo de la aplicación en el simulador de Android, es debido a su lenta respuesta frente a la rapidez de instalación en un dispositivo, además ofrece una interacción más fácil con el usuario gracias a su pantalla táctil frente al uso del ratón (Ver **figura 25**). Además se es necesario acceder a los archivos de imagen de las capturas, y no hay forma de cargar las imágenes en el dispositivo virtual.

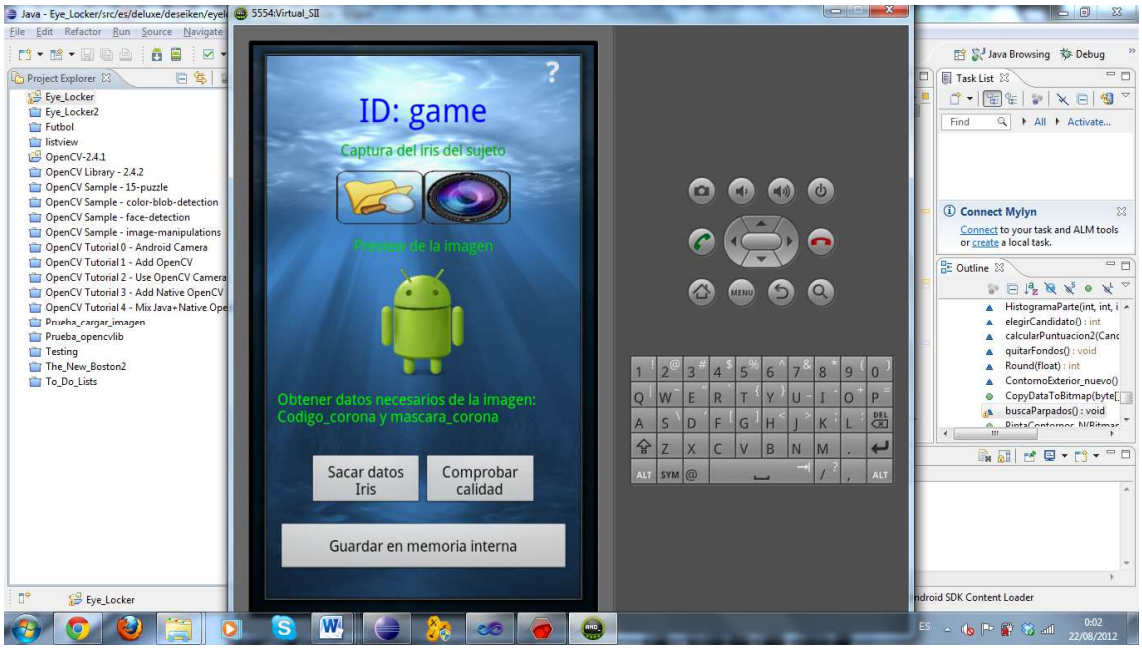

Figura 25: Simulador del Eclipse

6.1. Desarrollo de la prueba

Para poder comprobar el correcto funcionamiento de la aplicación ha sido necesario efectuar distintas pruebas en diferentes dispositivos, estas pruebas han sido efectuadas para poder:

- ✓ Comprobar el correcto funcionamiento de la aplicación.
- ✓ Comprobar el correcto acceso a memoria.
- ✓ Comprobar si es posible la identificación por la cámara del dispositivo.
- ✓ Comprobar el correcto diseño en diferentes pantallas y resoluciones.
- ✓ Comprobar el tiempo de procesado en verificación por pasos y directa con dispositivos de diferentes prestaciones.

6.2. Pruebas en distintos dispositivos móviles

Para el desarrollo de las pruebas de la aplicación en distinto dispositivos, ha sido necesaria la ayuda de personas cercanas dispuestas a probar la aplicación. La mayoría de estas pruebas han sido únicamente probadas para ver si era posible abrir la aplicación y ver si se ajustaba la disposición de los elementos a mostrar en el espacio designado por el dispositivo, evitando que salgan elementos fuera de la pantalla visible.

Se ha podido probar en un Samsung Galaxy Note con Android 4.0.1., un Sony Ericsson Xperia y un LG Optimus Black P970. Ninguno de ellos ha presentado anomalías de presentación, de funcionamiento de los botones o de guardado. El único inconveniente que se ha observado es en el dispositivo Samsung Galaxy Note, y es que la disposición de los elementos es la misma, pero dada a su mayor resolución y tamaño de la pantalla, los elementos quedan un poco reducidos en medio de la pantalla, pero son perfectamente visibles y accesibles. Para solucionar este pequeño problema bastaría con rediseñar el tamaño de los elementos a mostrar y usar resoluciones distintas para cada dispositivo.

A su vez se ha comprobado si los diferentes dispositivos han presentado algún fallo a la hora de realizar algún acceso a memoria y ninguno de ellos ha presentado anomalía alguna en el proceso.

El uso de la cámara se ha podido comprobar en algunos de estos dispositivos, pero en todos ellos, ha resultado que la imagen no ha pasado los controles de calidad. Esto es debido a lo comentado en capítulos anteriores, y es que la cámara disponible hoy en día para los Smartphone no dispone de luz infrarroja, por lo que el más mínimo reflejo o fallo de calidad es suficiente para no poder captar la imagen correctamente. Además se ha podido comprobar que estos dispositivos no enfocan correctamente desde cerca, por lo que si se intenta efectuar una captura desde una distancia aceptable para capturar el ojo completo con iris y todas sus minucias, esta captura no supera los controles de calidad al no estar correctamente enfocada.

El motivo de que no se hicieran unas pruebas más exhaustivas es la pesadez de estos procesos para otras personas, sobretodo para las personas que no encuentran fascinante este mundo.

Pero se han podido realizar pruebas minuciosas en el dispositivo mencionado anteriormente y otro de menor gama, el Samsung Galaxy S GT-I9000 con la versión Android 2.3.4 (Ver **figura 26** y **tabla 5**). Lo bueno de la disposición de este dispositivo para las pruebas, es que es del mismo fabricante (por tanto no afectan las diferencias de prestaciones entre fabricantes) pero una gama inferior, con la mitad de procesador, y con una versión anterior. Por tanto se podrá estudiar el tiempo de media requerido en cada uno de ellos.

En cuanto a diferencia de prestaciones de los dispositivos se observa en la siguiente tabla:

Samsung Galaxy S

Samsung Galaxy S2

Figura 26: Imagen de los dos dispositivos

	Galaxy S i9000	Galaxy S2 i9100
Procesador	**ARM Cortex A8 Hummingbird 1GHZ**	**Exynos Cortex A9 dual-core 1.2 GHZ**
Gráfica	**GPU PowerVR SGX540**	**GPU Mali-400MP**
Memoria RAM	**512MB**	**1GB**
Cámara	**5MPixels**	**8MPixels**

Tabla 5: Diferencia de prestaciones

En la primera prueba de espacio ocupado en el dispositivo, se ha podido observar que recién instalada la aplicación con exactamente la misma versión, ocupa más en el dispositivo S2 que en el S (Ver **figura 27**). También se ha observado que el arranque de la aplicación desde el S2 es más rápido que desde el S, esto es debido a la diferencia de prestaciones.

Samsung Galaxy S

Samsung Galaxy S2

Figura 27: Comparación de espacio ocupado

En los siguientes puntos se comentarán las pruebas realizadas con los dispositivos Galaxy S y Galaxy S2.

6.3. Controles de calidad

Para poder comprobar los distintos test de calidad, se ha hecho una selección de las capturas que presenten fallos en todos los pasos del test. Para saber que capturas presentan estos fallos buscados, se usa el proyecto "Test" de las bibliotecas en C# puesto que al ser en el ordenador se realiza con más rapidez la búsqueda. Se han revisado más de mil capturas, y se han encontrado todos los errores en varias menos dos, el de calidad de escala de grises esto debe ser debido a que al capturar los ojos con cámaras infrarrojas estas imágenes ya están en escala de grises por lo que no hay cabida a fallos en ello y el de comprobación contra fraudes. A continuación se presenta una imagen de la captura que presenta el fallo y su respectivo registro de error para hacerse una idea de las capturas con fallos:

Calidad de sharpness (enfoque):

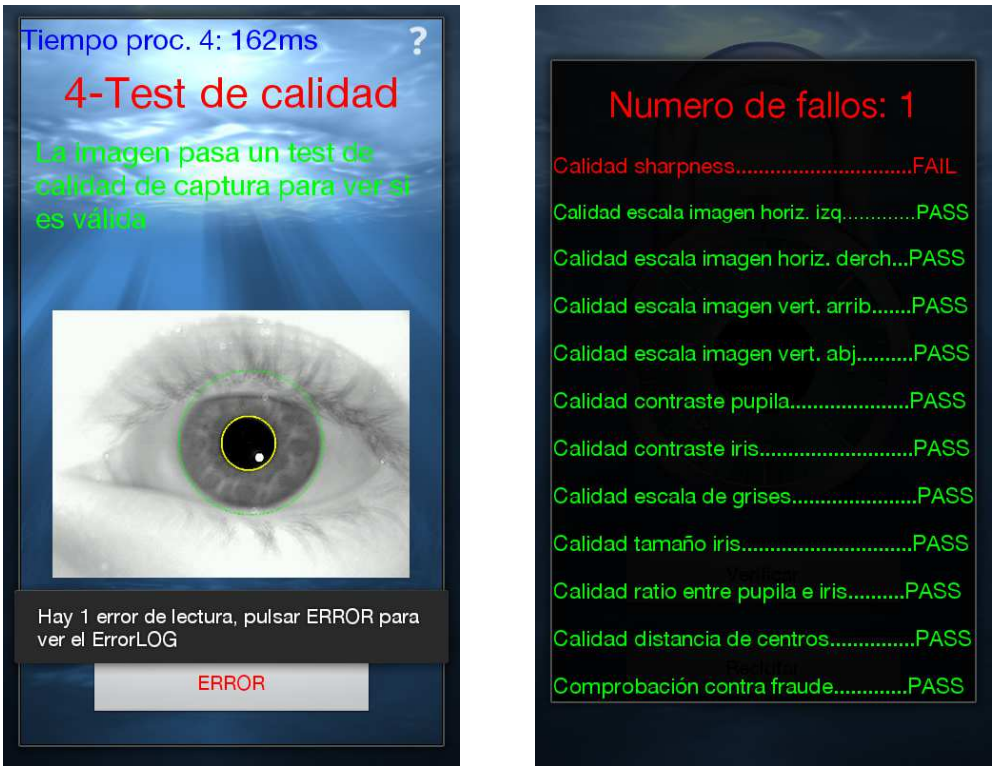

Figura 28: Test de calidad sharpness (enfoque)

Calidad escala de imagen:

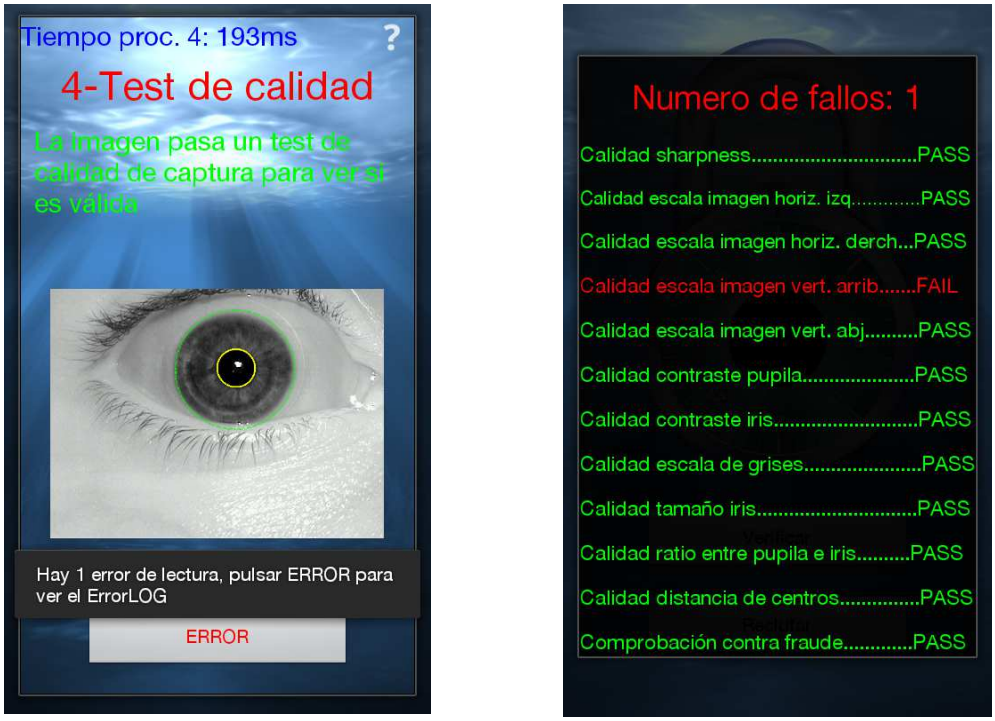

Figura 29: Test de calidad escala vert. arriba

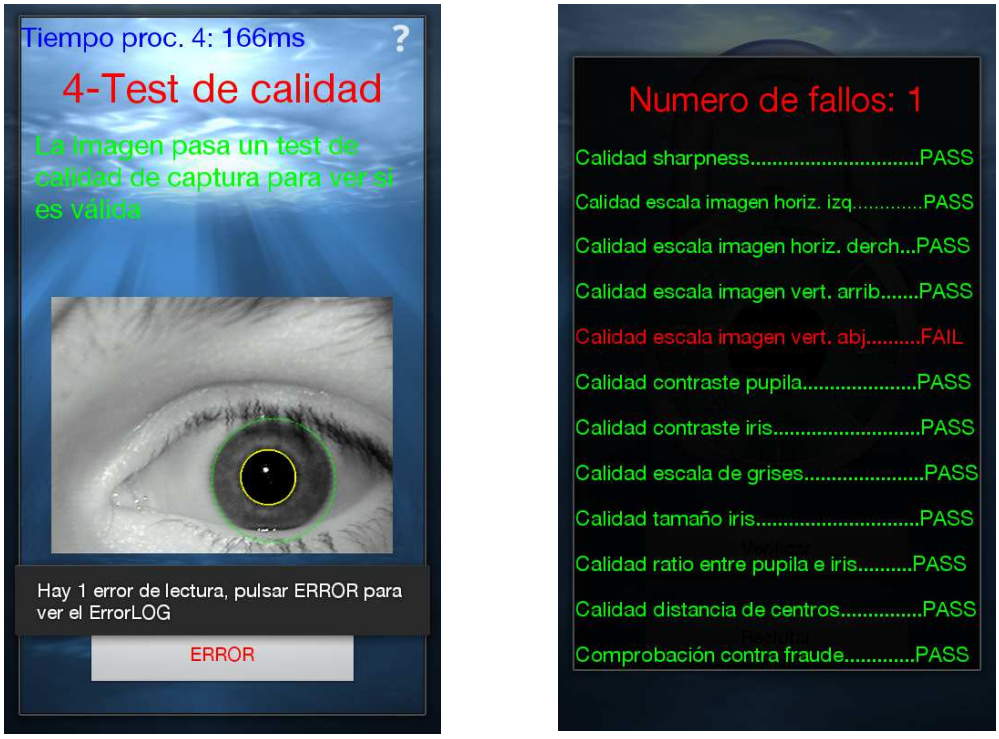

Figura 30: Test de calidad con varios fallos

Calidad contraste iris:

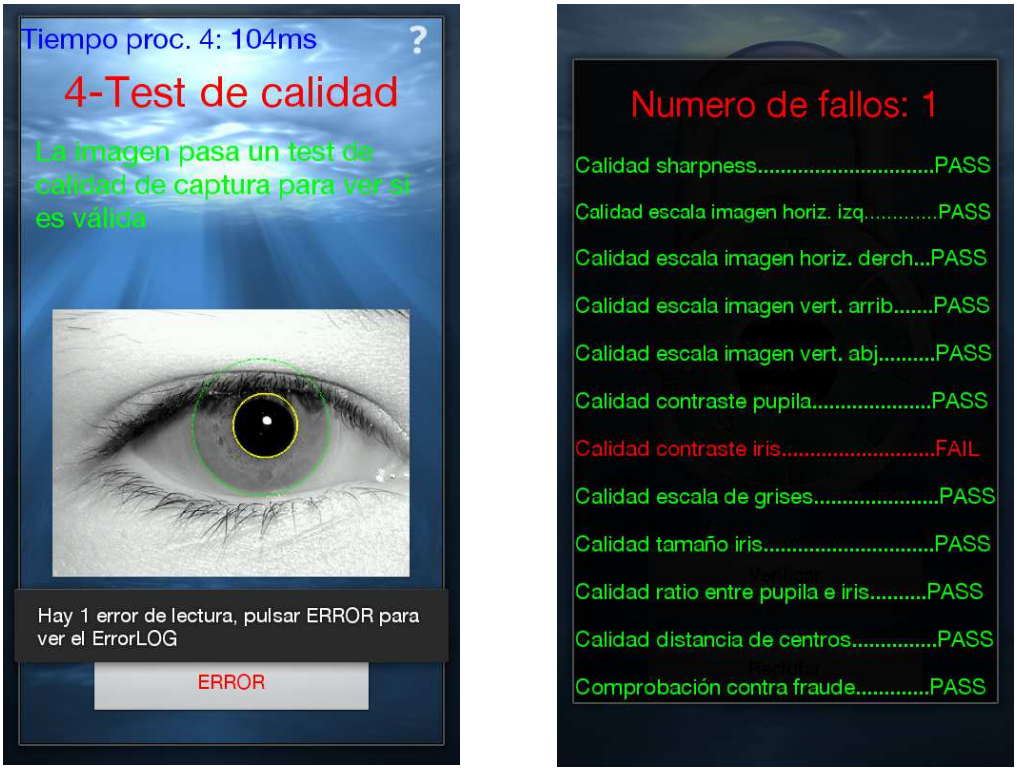

Figura 31: Test de calidad distancia de centros

Calidad tamaño de iris:

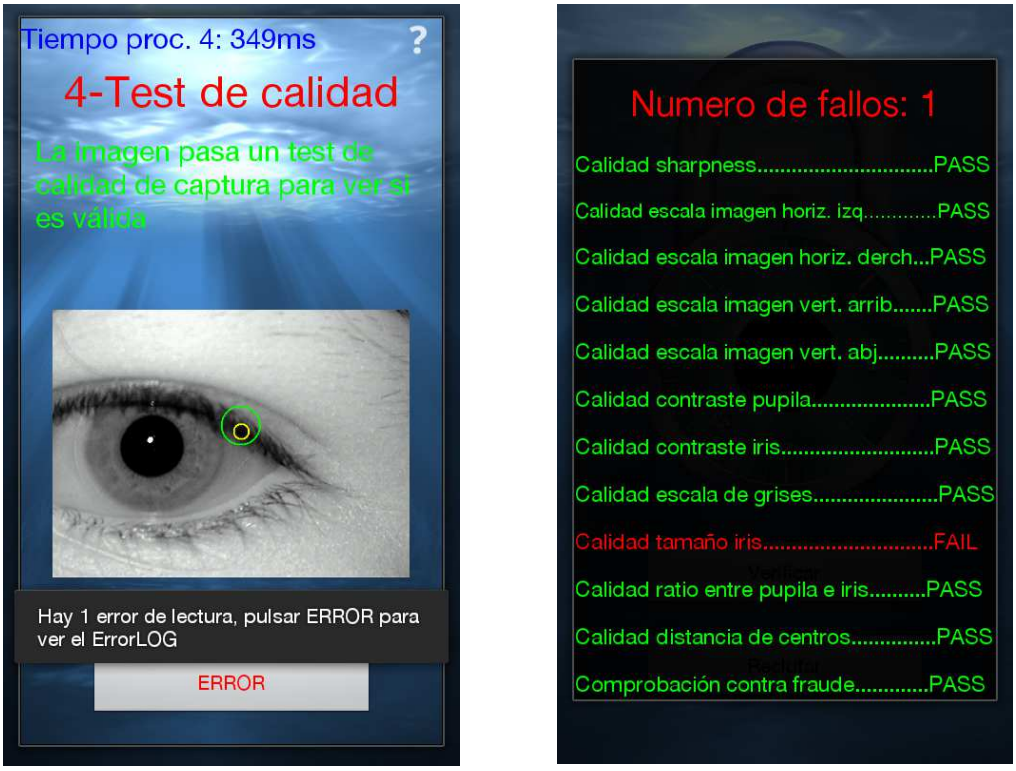

Figura 32: Test de calidad distancia de tamaño del iris

Calidad ratio entre pupila e iris:

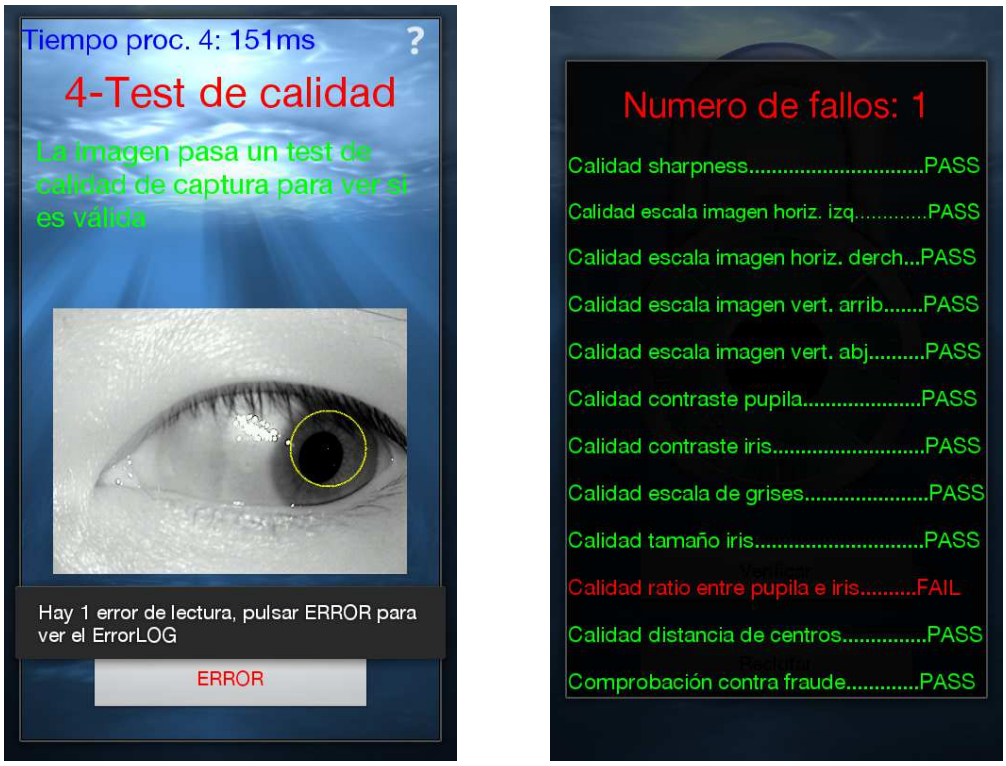

Figura 33: Test de calidad ratio entre pupila e iris

Calidad distancia de centros:

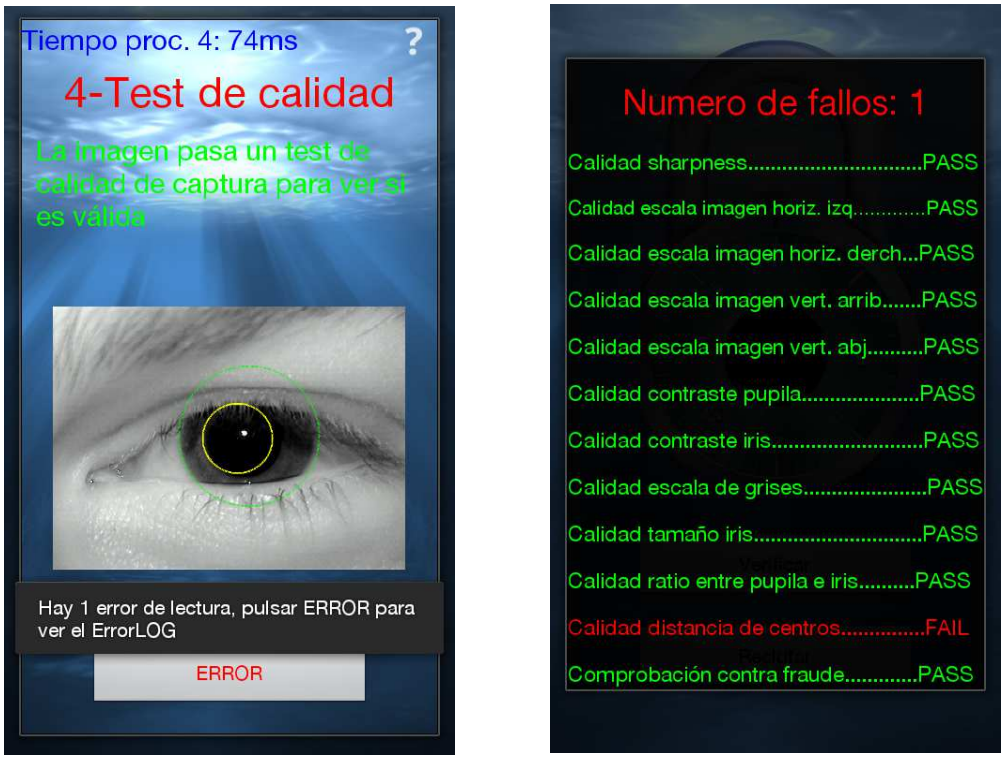

Figura 34: Test de calidad distancia de centros

El fallo más común ha sido el de sharpness, esto es lógico puesto que ha corta distancia es muy fácil obtener una imagen desenfocada.

Lo bueno de la aplicación desarrollada frente al proyecto de las bibliotecas en C#, es que cuando hay un error, esta no solo muestra el primer error registrado como hace el de C# sino que muestra todos los errores que se hayan producido en la captura de la imagen (Ver **figura 35**).

Figura 35: Test de calidad con varios fallos

Además se ha podido comprobar que el desarrollo de la identificación en Android se realiza de forma mucho más eficaz y fiable. El motivo de esto se puede ver fácilmente como ejemplo en la siguiente **figura 36**, en la cual en el proyecto "Test" de C# claramente no identifica correctamente los bordes, mientras que la aplicación desarrollada en Android los identifica sin problemas (Esto ha ocurrido en 6/1000 capturas).

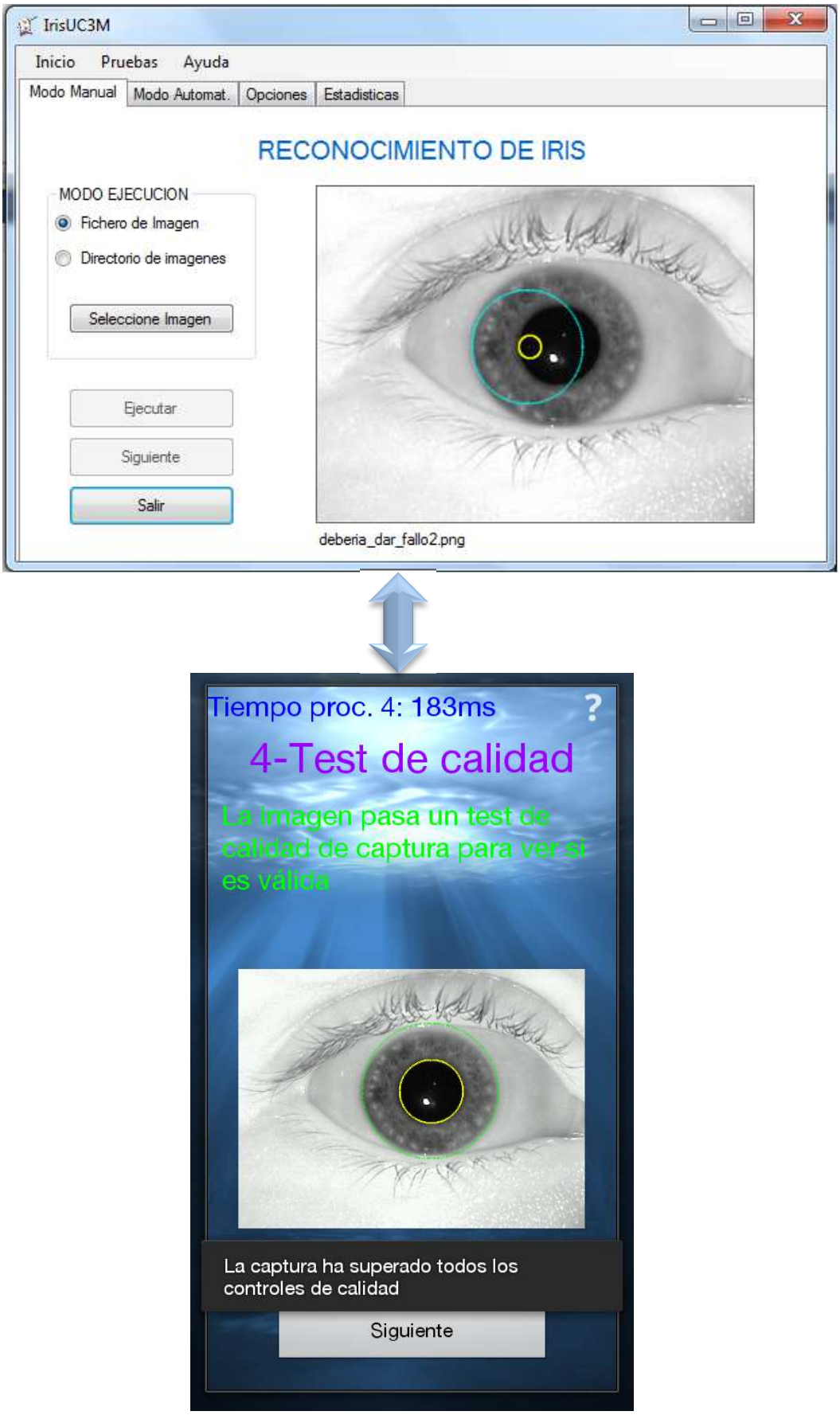

Figura 36: Fallido en C# / Correcto en Android 1

Incluso hay ciertas capturas que el proyecto “Test” identifica como nulas, debida al paso de calidad, pero en Android no detecta ningún error en la imagen (Ver **figura 37**) y como se puede observar a simple vista la identificación en correcta y la imagen no muestra fallos de calidad como desenfoque (Esto ha ocurrido en 1 de cada 3 imágenes con fallos).

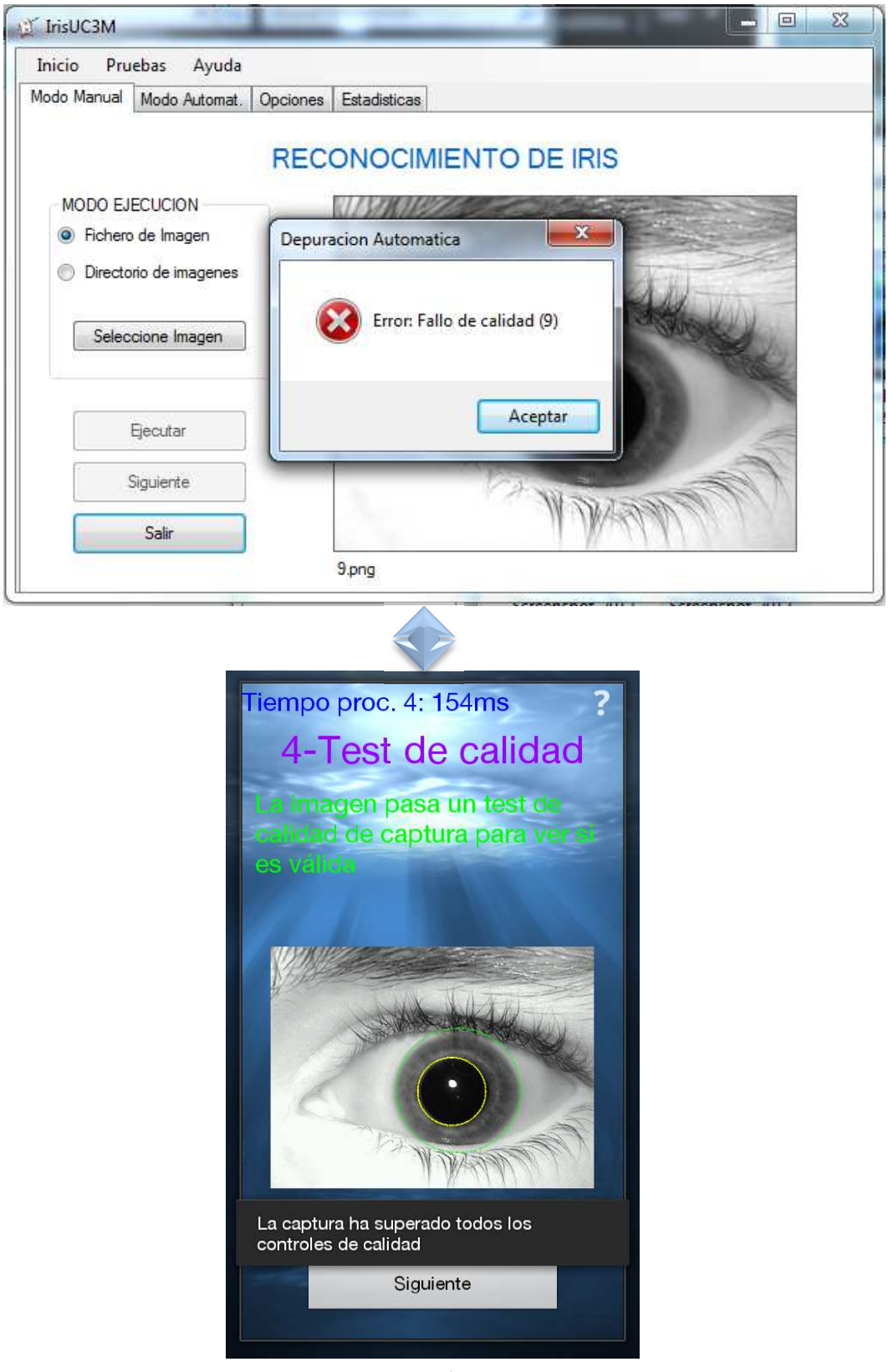

Figura 37: Fallido en C# / Correcto en Android 2

6.4. Media de tiempos de procesado

Se han realizado numerosas pruebas de tiempos de procesado en los dispositivos pero el tiempo de procesado de una imagen no es siempre la misma, aunque se realice en el mismo dispositivo, con el mismo sujeto y con la misma imagen de captura.

Diferencia de tiempos efectuados entre el Dispositivo Galaxy S y Galaxy S2. Estas pruebas determinarán lo importante que resultan las prestaciones del dispositivo para la ejecución de los procesos de identificación.

Se ha comprobado el tiempo de acceso a cada una de las actividades de la aplicación con los dos dispositivos. Y en algunas de arranque pesado debido al envío de información entre actividades, se ha observado como el dispositivo Galaxy S presenta más dificultades al acceder a ellas que el S2, esto es debido a la diferencia de prestaciones.

Para el cálculo del tiempo de procesado en los dos dispositivos se ha creado el mismo sujeto con exactamente la misma imagen de captura, de hecho se ha optado por realizar 4 pruebas generales:

- Elección de la misma imagen que la cargada en la extracción de características (es decir, 100% de acierto).
- Elección de una captura del mismo sujeto y mismo ojo, pero distinta imagen.
- Elección de una captura del mismo sujeto, pero distinto ojo.
- Elección de una captura de distinta persona.

La primera vez que se ejecuta esta operación de identificación con cada sujeto tarda más que las siguientes, esto debe de ser debido a que las direcciones de memoria del procesador DMA están establecidas ya del anterior proceso y al ser la misma dirección ese es el tiempo que se ahorra. Es por esto que se le aplicará más fiabilidad a los primeros tiempos marcados.

En la extracción de características del sujeto creado se ha observado que el tiempo invertido en cada uno de los procesos por parte del dispositivo S es el doble aproximadamente que el invertido por el S2. Incluso en el acceso a memoria para guardar o cargar se observa la diferencia de prestaciones. Pero todos los procesos en ambos dispositivos han sido en todas las pruebas satisfactorios. A continuación se exponen las tablas de comparación de tiempos de la identificación por pasos y directa, con cada uno de los métodos expuestos de elección de imagen de captura:

Elección de la misma imagen que la cargada en la extracción de características.

Verificación por paso:

Procesos	Galaxy S	Galaxy S2
2-Conversión	1638ms	1328ms
3.1-Preprocesado	1831ms	747ms
4-Test de calidad	539ms	314ms
5-Normalizar	145ms	97ms
6-Estraer características	1830ms	968ms
7-Comparación	40ms	5ms
Tiempo TOTAL	6023ms	3459ms
%Acierto	100%	97.57883%

Tabla 6: Prueba de tiempos 1 - Verificación por pasos

Verificación directa:

	Galaxy S	Galaxy S2
Tiempo TOTAL	5629ms	2956ms
%Acierto	100%	97.57883%

Tabla 7: Prueba de tiempos 1 - Verificación directa

RESULTADO FINAL: **ACCESO ACEPTADO**

Elección de una captura del mismo sujeto y mismo ojo, pero distinta imagen.

Verificación por pasos:

Procesos	Galaxy S	Galaxy S2
2-Conversión	1634ms	1356ms
3.1-Preprocesado	1307ms	578ms
4-Test de calidad	116ms	154ms
5-Normalizar	123ms	113ms
6-Estraer características	1823ms	1005ms
7-Comparación	13ms	24ms
Tiempo TOTAL	5016ms	3230ms
%Acierto	78.77358%	79.05093%

Tabla 8: Prueba de tiempos 2 - Verificación por pasos

Verificación directa:

	Galaxy S	Galaxy S2
Tiempo TOTAL	5553ms	2802ms
%Acierto	78.77358%	79.05093%

Tabla 9: Prueba de tiempos 2 - Verificación directa

RESULTADO FINAL: **ACCESO ACEPTADO**

Elección de una captura del mismo sujeto, pero distinto ojo.

Verificación por pasos:

Procesos	Galaxy S	Galaxy S2
2-Conversión	1575ms	1365ms
3.1-Preprocesado	1529ms	580ms
4-Test de calidad	145ms	144ms
5-Normalizar	181ms	113ms
6-Estraer características	1821ms	1059ms
7-Comparación	15ms	5ms
Tiempo TOTAL	5266ms	3266ms
%Acierto	72.46622 %	72.79412%

Tabla 10: Prueba de tiempos 3 - Verificación por pasos

Verificación directa:

	Galaxy S	Galaxy S2
Tiempo TOTAL	5073ms	2853ms
%Acierto	72.46622 %	72.79412%

Tabla 11: Prueba de tiempos 3 - Verificación directa

RESULTADO FINAL: **ACCESO DENEGADO**

Elección de una captura de distinta persona.

Verificación por pasos:

Procesos	Galaxy S	Galaxy S2
2-Conversión	1617ms	1468ms
3.1-Preprocesado	1461ms	530ms
4-Test de calidad	110ms	154ms
5-Normalizar	146ms	111ms
6-Estraer características	1869ms	1091ms
7-Comparación	13ms	23ms
Tiempo TOTAL	5216ms	3377ms
%Acierto	69.84305%	70.06726%

Tabla 12: Prueba de tiempos 4 - Verificación por pasos

Verificación directa:

	Galaxy S	Galaxy S2
Tiempo TOTAL	4994ms	3009ms
%Acierto	69.84305%	70.06726%

Tabla 13: Prueba de tiempos 4 - Verificación directa

RESULTADO FINAL: **ACCESO DENEGADO**

Se ha comprobado los tiempos efectuados en Android frente a los obtenidos en C#. Ha de tenerse en cuenta que un Smartphone, no dispone de las mismas prestaciones que un ordenador. Incluso si las prestaciones del dispositivo móvil son iguales a las del ordenador (GHz del procesador, memoria RAM, etc…), los resultados no tienen por qué ser los mismos puesto que hay numerosos factores que hay que tener en cuenta como refrigeración de la CPU o el sistema operativo por ejemplo. Los tiempos expuestos a continuación son los obtenidos es la ejecución del algoritmo en C#:

Procesos	Ordenador ejecutando C#
Conversión	224ms
Preprocesado	394ms
Normalizar	69ms
Calidad	21ms
Extraer características	24ms
Comparar	2ms

Tabla 14: Tabla de tiempos en C#

Estos tiempos están sacados con la siguiente plataforma de desarrollo:

- Visual Studio 2008, en C# sobre Windows 7.
- Ordenador Intel Core 2 Duo, de 2.8GHz (E7400), 4GB RAM.

Se puede ver en la **tabla 15** el aumento de tiempos de Android frente a C#. El proceso de prueba de Android seleccionado ha sido el de "Elección de una captura del mismo sujeto y mismo ojo, pero distinta imagen", en verificación por pasos y con el dispositivo Galaxy S2.

Procesos	Tiempos C#	Tiempos Android	Aumento %
Conversión	224ms	1356ms	505.357
Preprocesado	394ms	578ms	46.7
Normalizar	69ms	113ms	63.768
Calidad	21ms	154ms	633.33
Extraer características	24ms	1005ms	4087.5
Comparar	2ms	24ms	1100
Media de aumento			**1072.77%**

7. Conclusiones

Como se ha podido comprobar en las pruebas realizadas, el objetivo primordial de este proyecto se ha cumplido satisfactoriamente, incluso mejor de lo esperado. La identificación de sujetos se realiza de forma precisa y en todas las pruebas realizadas no se ha observado una falsa aceptación o falso rechazo.

El desarrollo de esta identificación es bastante más pesada en Android haciendo que tarde mucho más, pero estamos disponiendo en ellos un sistema de seguridad propia de una empresa con un alto procesado computarizado y disponible en una forma portátil.

Además según se ha podido observar en las pruebas realizadas, la detección de bordes del iris es más precisa en Android. Incluso en capturas que el método de C# identificaba con errores de la calidad suficiente como para no poder continuar el análisis y la correcta detección de los borde, el método aplicado en Android acepta esta captura y localiza correctamente los borde para a continuación hacer un correcto análisis de la imagen y llegar a la conclusión correcta de si es o no el sujeto. Es por esto que el proyecto resultante ha llegado a ser de incluso mayor precisión que el portado para ordenador. De esta manera se podría resolver que esta aplicación es una mejor implementación para una correcta identificación y por tanto es perfectamente apta para otros ámbitos de la telefonía en proyectos futuros. Pero hay que realizar más pruebas para poder determinar tal afirmación, puesto que aun pensando que la solución es una mejor adaptación de las librerías estamos comparando las prestaciones de un ordenador con las de un dispositivo móvil (aunque esto solo afecte al tiempo de procesado).

Se ha determinado que el dispositivo usado para la aplicación llega a afectar enormemente en el tiempo de procesado del sujeto, llegando a poder tardar el doble, como se puede observar en la tabla de tiempo de los dos dispositivos usados. Pero aun así el tiempo de toma de decisión del sistema para identificar al sujeto es aceptable, y no resulta enormemente pesado por parte del usuario tener que esperar de 3 a 5 segundos para obtener la acreditación y poder acceder al lugar protegido.

También se ha podido observar que la versión de Android puede afectar al proceso, como se puede observar en la distinción de porcentajes aportados por cada dispositivo. Se ha realizado cada prueba de tiempos con varios candidatos, y el porcentaje dentro de las mismas capturas y de los mismos dispositivos tienen el mismo porcentaje, pero son distintos de un dispositivo a otro. Esto se puede ver claramente en la **tabla 6**, donde ambos dispositivos deberían de dar 100%, sin embargo, el Galaxy S2 presenta menor porcentaje que el Galaxy S. No se puede saber a simple vista cual es el motivo, pero lo más sensato es pensar que se trata de una reducción de procesos por parte de las nuevas versiones para agilizar los cálculos del dispositivo mediante un redondeo en cada paso. Esto incrementaría la velocidad de procesado pero reduciría la precisión.

Durante el desarrollo se ha podido observar como una buena estructuración hace que el sistema funcione de una forma mucho más fluida y rápida, por eso al terminar y comprobar su funcionamiento, se restructuró algunas funciones y algunos procesos, haciendo de esta manera una mejora en los resultados obtenidos que posteriormente fueron incluidos en las anteriores tablas.

A título personal, este proyecto ha resultado apasionante pero bastante agotador debido a la dificultad añadida de tener que trabajar con dos lenguajes de los cuales nunca había trabajado antes. Aun así al realizarlo se presenta una gran satisfacción, además de un gran conocimiento adquirido sobre el tema. La decisión de adentrar en este proyecto fue motivada por las ganas de entender y manejar a la perfección el entorno de desarrollo para Android así como el lenguaje usado, Java, y viendo el resultado final hace que no aparezca ningún arrepentimiento aun con todas las horas invertidas.

7.1. Líneas futuras

Esta aplicación puede ser el comienzo de innumerables futuros proyectos, por ello se dispuso el código de una forma que sea fácil su compresión, con numerosos comentarios en zonas de posibles dudas.

El motivo de esta afirmación en cuanto a proyectos futuros, es debido al movimiento aportado por las grandes empresas del sector, como se puede ver por ejemplo con Google. La cual ya en su última versión lanzada, la Ice Cream Sandwich, incluyó un identificador facial, el cual mediante captura de numerosas imágenes procesa en un intervalo de 3 a 4 segundos la respuesta de si es o no el usuario del teléfono. Este sistema es muy parecido y de tiempos muy similares al desarrollado en este proyecto, pero no precisa de tanta seguridad ni tanta protección contra fraude como puede portar la identificación por iris. Es por esto que el desarrollo de la protección del móvil a través del iris, puede ser un sistema muy atractivo para dispositivos de una empresa, la cual puede albergar información valiosa y que debe ser bien protegida.

Además también se puede ver cómo el mundo de la telefonía móvil va avanzando de tal manera que cada vez dispongamos de menos elementos encima y nos baste con nuestro dispositivo móvil. Es el ejemplo de Google Wallet, un sistema impartido en Estados Unidos, el cual puedes efectuar pagos mediante el móvil sin necesidad de tarjetas de crédito, todo es cargado a la cuenta de Google en la cual tienes registrada tu tarjeta de crédito. Es decir, es como comprar una aplicación en Google Play.

Esto sin contar con información de tus cuentas de Facebook, Twitter, contactos, cuentas de correos, etc….

Toda esta información cada vez más significativa y valiosa en tu dispositivo, hace que sea una necesidad casi imperiosa el disponer de un sistema de seguridad mayor y con mejores prestaciones.

Por lo que se puede desarrollar aplicaciones como:

- Desbloqueo del dispositivo.
- Bloqueo de información privada como contraseñas, números de tarjetas de crédito, etc.…
- Confirmación de cargos en la tarjeta para pagos por medio del dispositivo o pagos online.

7.2. Conclusión final

La implementación de este sistema de identificación de gran procesado y protección en una plataforma móvil, hace que se llegue a plantear nuevas posibles aplicaciones y futuros proyectos aunque tuviesen que ser desarrollados con tiempo. Además se podría poder llegar a probar el sistema con la implementación por hardware al dispositivo móvil, mediante una cámara de infrarrojos para poder demostrar su correcto funcionamiento, llamando posiblemente la atención a diseñadores de dispositivos.

Este proyecto ha despertado en el autor una tremenda curiosidad por el tema y especialmente por el desarrollo de aplicaciones en Android debido a su posible aplicación en futuros trabajos.

Bibliografía

[1] OpenCV dev team. OpenCV: Using Android binary package with Eclipse. 04-6-2012.http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/android_binary_package.html#android-binary-package[Última consulta: 2-8-2012]

[2] Mattikariluoma. OpenCV: Android Experimental. 19-02-2011. http://opencv.willowgarage.com/wiki/AndroidExperimental[Última consulta: 10-2-2012]

[3] Stanford university. Tutorial on Using OpenCV for Android Projects. EE368 Digital Imagne Processing, Verano 2012. http://www.stanford.edu/class/ee368/Android/Tutorial-2-OpenCV-for-Android-Setup-Linux.pdf[Última consulta: 3-5-2012]

[4] OpenOffice: Exception Description. The Apache Software Foundation. http://www.openoffice.org/api/docs/common/ref/com/sun/star/lang/ArrayIndexOutOfBoundsException.html[Última consulta: 15-6-2012]

[5] Google. Android development: General Search. http://developer.android.com/reference/java/lang/System.html#q=mat[Última consulta: 30-6-2012]

[6] Wikipedia. Complemento a dos. 30-8-2012. http://es.wikipedia.org/wiki/Complemento_a_dos[Última consulta: 15-7-2012]

[7] StackOverflow. Foro de consultas de implementación de lenguajes. http://stackoverflow.com/ [Última consulta: 17-7-2012]

[8] Aforge: Explicación de las funciones usadas en C#. http://www.aforgenet.com/framework/docs/html/d7196dc6-8176-4344-a505-e7ade35c1741.htm[Última consulta: 20-7-2012]

[9] OpenCV dev team. OpenCV: Index for libraries. 4-7-2012. http://docs.opencv.org/android/service/doc/index.html#android-opencv-manager[Última consulta: 25-7-2012]

[10] OpenCV dev team. OpenCV: Descarga de la biblioteca fuente 1. 4-7-2012. http://sourceforge.net/projects/opencvlibrary/files/opencv-android/2.4.2/OpenCV-2.4.2-android-sdk.zip/download[Última consulta: 27-7-2012]

[11] OpenCV dev team. OpenCV Tutorial: Filtro de Canny. 4-7-2012. http://dasl.mem.drexel.edu/~noahKuntz/openCVTut5.html[Última consulta: 28-7-2012]

[12] OpenCV dev team. OpenCV: Descarga de la biblioteca fuente 2. 4-7-2012. http://teyvoniathomas.com/index.php/projects/56-opencv-on-android.html[Última consulta: 30-5-2012]

[13] Tutorial de aplicación biblioteca OpenCV: Filtro de Canny. 2-4-2012. http://es.scribd.com/doc/88285388/38/Canny-Edge-Detector[Última consulta: 9-7-2012]

[14] Introduction to programming with OpenCV. http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html[Última consulta: 12-7-2012]

[15] Gady Agam. OpenCV Tutorial: Basic operations for images Canny edge detection. 31-03-2006. http://programing-tutorial.blogspot.com.es/2010/06/open-cv-tutorial-basic-operations-for.html[Última consulta: 15-7-2012]

[16] Edgewall Software. OpenCV: Ejemplo de aplicación de funciones de manipulación de imagen. https://code.ros.org/trac/opencv/browser/trunk/opencv/samples/android/image-manipulations/src/org/opencv/samples/imagemanipulations/ImageManipulationsView.java?rev=6386[Última consulta: 17-8-2012]

[17] SugarRush Creative Group. Información y guías de Jailbreak y root. http://www.esmandau.com[Última consulta: 20-8-2012]

[18] Dienstag. Información general sobre el melanoma intraocular (relativo al ojo). 31-8-2012. http://www.meb.uni-bonn.de/Cancernet/CDR0000537037.html[Última consulta: 23-8-2012]

[19] Google. Android development: Versiones de Android. 15-8-2012. http://developer.android.com/about/dashboards/index.html[Última consulta: 25-8-2012]

[20] Google. Android development: Activity. http://developer.android.com/reference/android/app/Activity.html[Última consulta: 25-8-2012]

[21] Canal Youtube de Cornboyz Android. Tutorial Android. http://www.youtube.com/user/cornboyzandroid?feature=results_main[Última consulta: 3-23-2012]

[22] Canal Youtube de brainyideas. Tutorial Android. http://www.youtube.com/user/brainyideas?feature=results_main[Última consulta: 23-3-2012]

[23] Canal Youtube de Android Tutorials. Tutorial Android. http://www.youtube.com/user/androidtutorials499?feature=results_main [Última consulta: 30-3-2012]

[24] Canal Youtube de thenewboston. Tutorial Android. http://www.youtube.com/user/thenewboston?feature=results_main[Última consulta: 5-4-2012]

[25] Marino Tapiador Mateos, Juan A.Sigüenza Pizarro - "Tecnologías biométricas aplicadas a la seguridad" - editorial Ra-Ma, 2005.

[26] Alfonso Jiménez Marín – "Aprende a programar con Java: un enfoque práctico partiendo de cero" – editorial Paraninfo, 2012.

[27] Aitor Mendaza Ormaza – "Presentación: Introducción SO Android" – Grupo Universitario de Tecnologías de identificación, Carlos III de Madrid.

[28] Aitor Mendaza Ormaza – "Presentación: Introducción Programación Android" – Grupo Universitario de Tecnologías de identificación, Carlos III de Madrid.

[29] Raúl Sánchez Reíllo – "El Iris Ocular como parámetro para la identificación Biométrica" – Ágora SIC (Volumen 21), Grupo Universitario de Tarjetas Inteligentes, Dpto. de Tecnología Fotónica, 2000.

[30] Raúl Sánchez Reíllo – "Identificación Biométrica y su unión con las Tarjetas Inteligentes" – Ágora SIC (Volumen 19), Grupo Universitario de Tarjetas Inteligentes, Dpto. de Tecnología Fotónica, 2000.

ANEXO 1. Presupuesto:

Costes materiales

Puesto que los materiales para el desarrollo del proyecto ya se disponían de ellos, se expone como concepto la amortización del tiempo invertido en su realización (6 meses) en base al tiempo estimado de vida útil de cada uno.

Concepto	Precio
Samsung Galaxy S2	33.33 €
Asus A53S	93.75 €
Cable de conexión	2 €
TOTAL	**129.08 €**

Tabla 15: Coste material

Costes de personal

Ocupación	Horas	Precio/Hora	Importe
Ingeniero	616h	30 €/h	18480 €
Jefe de proyecto	30h	50 €/h	1500 €
TOTAL			**19980 €**

Tabla 16: Coste de personal

*Para saber el desglose de tiempo invertido en cada fase del proyecto diríjase a al anexo 2.

Costes totales

Concepto	Precio (€)
Costes de material	129.08 €
Costes de personal	19980 €
Costes indirectos y gastos generales (15%)	3016.362 €
Subtotal	**23125.442 €**
IVA (21%)	4856.342 €
TOTAL	**27981.78 €**

Tabla 17: Coste total

El coste total del proyecto es de veinte siete mil novecientos ochenta y un euros con setenta y ocho céntimos.

Leganés, 5 de Septiembre de 2012

ANEXO 2. Planificación efectuada:

Para poder efectuar el presupuesto de la realización de este proyecto, se realizará un desglose de las horas invertidas en cada una de las fases. Las horas expuestas a continuación son meramente orientativas ya que son aproximadas.

Fase 1: Documentación

- Estudio de la plataforma Android y del entorno de desarrollo (15 horas)
- Asistencia a charlas y presentaciones sobre Android (16 horas)
- Preparación de las herramientas de trabajo (5 horas)
- Búsqueda y realización de tutoriales con aplicaciones de prueba (50 horas)

Fase 2: Desarrollo

- Actividad principal (215 horas)
- Pantalla de configuración (25 horas)
- Actividades secundarias (125 horas)
- Interconexión de todas las actividades (25 horas)

Fase 3: Pruebas

- Pruebas (15 horas)
- Corrección y depuración (20 horas)
- Cambios de la presencia final de la aplicación (15 horas)

Fase 4: Memoria

- Redacción de la memoria (80 horas)
- Corrección (10 horas)

Fases	Horas
Documentación	86
Desarrollo	390
Pruebas	50
Memoria	90
TOTAL	616

Tabla 18: Recuento de horas

ANEXO 3. Código de la solución:

En el presente anexo se expone el código de la solución, mostrando la actividad (con la extensión .java) y su respectiva vista (View en Inglés con la extensión .xml) a continuación. Para finalizar se expondrá el AndroidManifest. En el código adjunto se ha prescindido de elementos repetitivos como las importaciones de bibliotecas (llamados import) que sólo se mostrarán en la primera actividad. Asímismo no se han incluido los elementos que se encuentran bajo Protección Industrial por la propia Universidad Carlos III de Madrid, al preferir que el documento sea un documento accesible al público general a través de los servicios de Biblioteca.

Presentación incial

Splash.java

```
public class Splash extends Activity {

	MediaPlayer ourSong; // Variable global para que onPause lo pueda usar

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.splash);
		ourSong = MediaPlayer.create(Splash.this, R.raw.splash_music);

		// Para poder obtener los datos de Preds
		SharedPreferences getPrefs = PreferenceManager
				.getDefaultSharedPreferences(getBaseContext());
		boolean music = getPrefs.getBoolean("checkbox", true);
		if (music == true) {
			ourSong.start();
		}
		Thread timer = new Thread() {
			public void run() {
				try {
					sleep(1000); // 1 segundos
				} catch (InterruptedException e) {
					e.printStackTrace();
				} finally {
	Intent startActivity = new
Intent("es.deluxe.deseiken.eyelocker.MENU");
	startActivity(startActivity);
				}
			}
		};
		timer.start();
	}
```

```
	@Override
	protected void onPause() {
		super.onPause();
		ourSong.release();
		finish();
	}
	}
```

splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/splash_background" >


</LinearLayout>
```

Menú principal

Menu.java

```
package es.deluxe.deseiken.eyelocker;

import es.deluxe.deseiken.eyelocker.R.id;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;

public class Menu extends Activity implements OnClickListener {

	Button bVerifMenu, bReclutMenu, bHelp;
	Intent iSelectUserVerif, iMenuReclut;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.menu);
		initialize();
	}
```

```
	private void initialize() {
		// TODO Auto-generated method stub
		bVerifMenu = (Button) findViewById(id.bVerifMenu);
		bReclutMenu = (Button) findViewById(id.bReclutMenu);
		bHelp = (Button) findViewById(id.bHelp);
		bHelp.setOnClickListener(this);
		bReclutMenu.setOnClickListener(this);
		bVerifMenu.setOnClickListener(this);
	}

	// Para llamar a un menú inflable de pref. hay que crear el método
	@Override
	public boolean onCreateOptionsMenu(android.view.Menu menu) {
		// Para que aparezca el menú inflable
		MenuInflater blowUp = getMenuInflater();
		blowUp.inflate(R.menu.prefmenu, menu);
		return true;
	}

	// Cada vez que es pulsado un botón del menú inflable se llama a este
método
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		switch (item.getItemId()) {
		case R.id.aboutUs:
			Intent i = new
Intent("es.deluxe.deseiken.eyelocker.ABOUT");
			startActivity(i);
			break;
		case R.id.preferences:
			Intent p = new
Intent("es.deluxe.deseiken.eyelocker.PREFS");
			startActivity(p);
			break;
		case R.id.exit:
			finish();
			break;
		}
		return false;
	}

	public void onClick(View pressed) {
		// TODO Auto-generated method stub
		switch (pressed.getId()) {
		case R.id.bVerifMenu:
			//iMenuVerific = new Intent(this, MenuVerific.class);
			iSelectUserVerif = new
Intent("es.deluxe.deseiken.eyelocker.SELECTUSERVERIF");
			startActivity(iSelectUserVerif);
			break;

		case R.id.bReclutMenu:
			iMenuReclut = new
Intent("es.deluxe.deseiken.eyelocker.MENURECLUT");
			startActivity(iMenuReclut);
			break;

		case R.id.bHelp:
```

```
                    Intent help = new Intent(this, Help.class);
                    String Actividad = "Menu";
                    help.putExtra("ActivityFrom", Actividad);
                    startActivity(help);
                    break;
                }

        }

}
```

menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical" >

    <Button
        android:id="@+id/bHelp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="25dp"
        android:background="@drawable/ic_launcher" />

    <Button
        android:id="@+id/bVerifMenu"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="17dp"
        android:text="Verificar" />

    <Button
        android:id="@+id/bReclutMenu"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="17dp"
        android:text="Reclutar" />

</LinearLayout>
```

Menú Reclutación

MenuReclut.java

```
public class MenuReclut extends Activity implements OnClickListener,
            OnKeyListener {

       ArrayList<String> Lista_ID_Sujetos, Lista_ID_Checked;
       ArrayAdapter<String> aaList;
```

```
	byte[] datos_list = null;
	byte[] datos_check = null;
	// Direccion del fichero para guardar todo el ArrayList
	String fileNameID = "path_list";
	String checkPath = "path_check";

	//boolean[] Checked_ID;
	int Pos_ID;

	/*** ---VARIABLES DE PRESENTACIÓN Y BOTONES--- ***/
	CheckBox chckboxDeleteData;
	ListView listUsers;
	Button bAñadirSuj, bHelp;
	EditText etSujetoNuevo;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.menureclut);

		Toast.makeText(getApplicationContext(),"Cuando salga de esta
actividad, todos los cambios guardados NO SE PODRAN MODIFICAR por seguridad",
Toast.LENGTH_LONG).show();
		initialize_android();
	}

	private void initialize_android() {
		chckboxDeleteData = (CheckBox)
findViewById(R.id.chcbxBorrarUsuario);
		listUsers = (ListView) findViewById(R.id.lstUsers);
		bAñadirSuj = (Button) findViewById(R.id.bAdrSuj);
		etSujetoNuevo = (EditText) findViewById(R.id.etSujNu);
		bHelp = (Button) findViewById(R.id.bHelp);

		Lista_ID_Sujetos = new ArrayList<String>();
		Lista_ID_Checked = new ArrayList<String>();

// simple_list lo puedo cambiar a los diferentes tipos que hay de lista
		aaList = new ArrayAdapter<String>(this,
				android.R.layout.simple_list_item_checked,
Lista_ID_Sujetos);
		listUsers.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
		listUsers.setAdapter(aaList);

// Para CARGAR la lista de sujetos guardada nada mas empezar la actividad
		String dataString = readFile(fileNameID);
		if (dataString != null){
			byte[] dataArray = dataString.getBytes();

	Lista_ID_Sujetos.addAll(readFromArrayDataToListArray(dataArray));
			aaList.notifyDataSetChanged();
		}
		// Lo mismo, para CARGAR la lista pero de checked
		String datacheckString = readFile(checkPath);
```

```
            if (datacheckString != null){
                  byte[] dataArraycheck = datacheckString.getBytes();

      Lista_ID_Checked.addAll(readFromArrayDataToListArray(dataArraycheck));
            }

// Recorro toda la lista y cambio su estado a true o false según corresponda
            for (int i = 0; i < Lista_ID_Sujetos.size(); i++){
                  String result = Lista_ID_Checked.get(i);
                  if (result.equals("true") == true){
                        listUsers.setItemChecked(i, true);
                  } else {
                        listUsers.setItemChecked(i, false);
                  }
            }

            bHelp.setOnClickListener(this);
            bAñadirSuj.setOnClickListener(this);
            etSujetoNuevo.setOnKeyListener(this);
            listUsers.setOnItemClickListener(new OnItemClickListener() {

                  public void onItemClick(AdapterView<?> arg0, View arg1,
int position,
                              long id) {

                  //CheckedTextView textView = (CheckedTextView) arg1;
                        final String filePath = ((TextView)
arg1).getText().toString();
                        String result = Lista_ID_Checked.get(position);
                  //listUsers.setItemChecked(position, true);

                        if (chckboxDeleteData.isChecked() == false) {

                  // Para evitar que se quite el checked una vez dado
                              if (result.equals("false") == true) {
                  // Envia a la actividad UsuariosData el nombre del ID.
      String NombreID = ((TextView)arg1).getText().toString();
      Intent UserData = new Intent(MenuReclut.this, UsuariosData.class);
                                    UserData.putExtra("UserID", NombreID);
                                    Pos_ID = position;
                                    UserData.putExtra("Pos_ID", Pos_ID);
                                    startActivityForResult(UserData, 0);
                              } else {
      Toast.makeText(getApplicationContext(), ((TextView) arg1).getText()
+ " ya ha sido registrado y guardado en otro momento, por seguridad no se
permite modificarlo",Toast.LENGTH_LONG).show();
                              }
                        }else{

                              // Se elimina ese Item
      AlertDialog.Builder adb = new AlertDialog.Builder(MenuReclut.this);
      final CharSequence elim = ((TextView) arg1).getText();
      adb.setTitle("¿Eliminar sujeto?");
      adb.setMessage("Está seguro que desea eliminar " + ((TextView)
arg1).getText());
            final int positionToRemove = position;
            adb.setNegativeButton("Cancel", null);
            adb.setPositiveButton("Ok", new AlertDialog.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
```

```
                        Lista_ID_Sujetos.remove(positionToRemove);
                Toast.makeText(getApplicationContext(), elim + " eliminado",
Toast.LENGTH_SHORT).show();
                aaList.notifyDataSetChanged();
// Y eliminar toda la información que hay en ese fichero en esa dirección
                DeleteFile(filePath);
// Borro la posicion de true o false de la list checked
                Lista_ID_Checked.remove(positionToRemove);

             // Se guarda la lista cada vez que se borre.
                datos_list = writeArraylistToByteArray(Lista_ID_Sujetos);
                saveToFile(datos_list, fileNameID);
             // Se guarda los elementos checkeados
                datos_check = writeArraylistToByteArray(Lista_ID_Checked);
                saveToFile(datos_check, checkPath);
                        }});

                    adb.show();
                         }

// Recorro toda la lista y cambio su estado a true o false según corresponda
                         for (int i = 0; i < Lista_ID_Sujetos.size(); i++){
                               String resultad = Lista_ID_Checked.get(i);
                               if (resultad.equals("true") == true){
                                     listUsers.setItemChecked(i, true);
                               } else {
                                     listUsers.setItemChecked(i, false);
                               }
                         }

                }});
      }

      public void DeleteFile(String fileNames) {
            try {
                  FileOutputStream fOut = openFileOutput(fileNames,
Context.MODE_PRIVATE);
                  fOut.write(null);
                  fOut.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      private ArrayList<String> readFromArrayDataToListArray(byte[]
dataArray) {
            ArrayList<String> Lista_IDS = new ArrayList<String>();
            ByteArrayInputStream bais = new ByteArrayInputStream(dataArray);
            DataInputStream in = new DataInputStream(bais);
            try {
                  while (in.available() > 0) {
                      String element = in.readUTF();
                      Lista_IDS.add(element);
                      }
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return Lista_IDS;
      }
```

```
	private void addUser(String User) {
		if (User.length() > 0) {
			Lista_ID_Sujetos.add(User);
			aaList.notifyDataSetChanged();
			etSujetoNuevo.setText("");
// Se añade a la lista de checked un false para ocupar el espacio
			Lista_ID_Checked.add("false");
		}
	}

	public void onClick(View v) {
		if (v == bAñadirSuj) {
			addUser(etSujetoNuevo.getText().toString());
// Cada vez que se pulse se ha de guardar los datos de toda la lista
			datos_list = writeArraylistToByteArray(Lista_ID_Sujetos);
			datos_check = writeArraylistToByteArray(Lista_ID_Checked);
			saveToFile(datos_list, fileNameID);
			saveToFile(datos_check, checkPath);
		} else if (v == bHelp) {
			Intent help = new Intent(this, Help.class);
			String Actividad = "MenuReclut";
			help.putExtra("ActivityFrom", Actividad);
			startActivity(help);
		}
	}

	private byte[] writeArraylistToByteArray(ArrayList<String> lista_ID) {
		byte[] list_array = null;
		ByteArrayOutputStream baos = new ByteArrayOutputStream();
		DataOutputStream out = new DataOutputStream(baos);
		for (String element : lista_ID) {
		    try {
				out.writeUTF(element);
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
		list_array = baos.toByteArray();
		return list_array;
	}

	private void saveToFile(byte[] datos, String FilePath) {
		try {
			FileOutputStream fOut = openFileOutput(FilePath,
Context.MODE_PRIVATE);
			fOut.write(datos);
			fOut.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public String readFile(String fileName) {
		String dataget = null;
		try {
	BufferedReader input = new BufferedReader(new InputStreamReader(
				openFileInput(fileName)));
```

```
                dataget = input.readLine().toString();
                input.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            return dataget;
        }

        public boolean onKey(View v, int keyCode, KeyEvent event) {

            return false;
        }
}
```

menureclut.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical"
    android:paddingBottom="15dp"
    android:paddingLeft="15dp"
    android:paddingRight="15dp"
    android:paddingTop="7dp" >

    <Button
        android:id="@+id/bHelp"
        android:layout_width="wrap_content"
        android:layout_height="30dp"
        android:layout_gravity="right"
        android:background="@drawable/actionhelpwhite"
        android:gravity="right" />

    <TextView
        android:id="@+id/tvTitulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Sujetos"
        android:textColor="#0000FF"
        android:textSize="30dp" />

    <EditText
        android:id="@+id/etSujNu"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="10"
        android:hint="ID del Sujeto sin espacios" />

    <Button
        android:id="@+id/bAdrSuj"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
        android:text="Añadir Sujeto" />

    <ListView
        android:id="@+id/lstUsers"
        android:layout_width="fill_parent"
        android:layout_height="164dp"
        android:layout_weight="0.15" />

    <CheckBox
        android:id="@+id/chcbxBorrarUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:checked="false"
        android:text="Borrar Sujeto"
        android:textColor="#009F00" />

</LinearLayout>
```

Extracción Características Usuario

Por confidencialidad no se permite incluir todo el código de esta actividad. Esto es debido a que está basado en un código que se encuentra bajo protección industrial de la Universidad Carlos III. En los campos correspondientes se a procedido ha su entera eliminación y en las declaraciones de las mismas han sido sustituidas por el indicativo "****Función Protegida****". Se han eliminado las funciones traducidas he implementadas del proyecto IrisBiometrics, variables globales que usa dicha biblioteca y sus declaraciones. Todo ellos para la protección del concepto de estos datos.

UsuariosData.java

```
public class UsuariosData extends Activity implements OnClickListener {

	/*** ---Inicio variables IrisBiometrics--- ***/
	/*** ---CONSTANES DE CÓDIGO--- ***/

	int error_Sharpness, error_Image_Scale_H_Left,
error_Image_Scale_H_Right, error_Image_Scale_V_Up, error_Image_Scale_V_Down,
error_Contrast_Pupila, error_Contrast_Iris, error_Gray_Scale_Density,
error_Pupil_Iris_Ratio, error_Iris_Size, error_Eccentric, error_Fraude = 0;

	/*** ---Fin variables IrisBiometrics--- ***/

	// **Establecer las Variables como final y con valor estático hasta
nuevo
	// cambio**
	private static final int SELECT_IMAGE = 0;
	private static final int CAMERA_IMAGE = 1;

// Se le asigna un valor distinto de 0 ó 1 para que solo se realice una vez
// la comprobación
	int ComprobError = 5;
```

```
        int ComprobGetIris = 5;
        int contError = 0;
        byte[] ArrayErrores = null;

        Intent GalleryUser, TakePhotoUser;
        Uri SelectImagUser;
        String FilePathUser, UserID, line;
        String checkPath = "path_check";
        String FilePath_Code_UserID;
        String FilePath_Mask_UserID;
        // eol = end of line => "\n"
        String eol = System.getProperty("line.separator");

        // boolean Checked_ID;
        int All_OK = 0;
        int GetIrisData_OK = 5;
        int Pos_ID;

// Para la obtención de caracteríticas de la imagen del User en memoria
        byte[] Caract_Code_UserID, Caract_Mask_UserID;

        /*** ---VARIABLES DE PRESENTACIÓN Y BOTONES--- ***/
        Button bSave, bGetIrisData, bFolderUser, bCamaraUser, bCalidad, bHelp;
        ImageView ivImagePreview;
        Bitmap bmOjo = null, bmOjoContorno = null;
        TextView tvIDName;
        ArrayList<String> Lista_ID_Checked;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                requestWindowFeature(Window.FEATURE_NO_TITLE);
                getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                                WindowManager.LayoutParams.FLAG_FULLSCREEN);

                setContentView(R.layout.usuariosdata);

                initialize_android();
                getdataId();
        }

        private void initialize_android() {
                bSave = (Button) findViewById(R.id.bSave);
                bGetIrisData = (Button) findViewById(R.id.bGetIrisData);
                bFolderUser = (Button) findViewById(R.id.bFolderUser);
                bCamaraUser = (Button) findViewById(R.id.bCamaraUser);
                bCalidad = (Button) findViewById(R.id.bCalidad);
                ivImagePreview = (ImageView) findViewById(R.id.ivPrevioUser);
                tvIDName = (TextView) findViewById(R.id.tvName);
                bHelp = (Button) findViewById(R.id.bHelp);

                Lista_ID_Checked = new ArrayList<String>();

                bHelp.setOnClickListener(this);
                bSave.setOnClickListener(this);
                bGetIrisData.setOnClickListener(this);
                bFolderUser.setOnClickListener(this);
                bCamaraUser.setOnClickListener(this);
```

```
            bCalidad.setOnClickListener(this);
      }

      public void onClick(View pressed) {
            switch (pressed.getId()) {
            case R.id.bFolderUser:
                  GalleryUser = new Intent(Intent.ACTION_GET_CONTENT);
                  GalleryUser.setType("image/*");
                  startActivityForResult(GalleryUser, SELECT_IMAGE);
                  break;
            case R.id.bCamaraUser:
                  TakePhotoUser = new Intent(

      android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
                  startActivityForResult(TakePhotoUser, CAMERA_IMAGE);
                  break;
            case R.id.bCalidad:
      // Carga los datos en IrisBiometrics y le pasa todos los parametros
      // de calculo de errors
            if (bmOjo != null) {
                  if (ComprobError == 5) {
            // Si comprobCalidad detecta un error tiene que retornar un
            // valor 1 sino un 0
                  ComprobError = ComprobCalidad();
                  // Si hay fallo entonces...
                  if (ComprobError == 1) {
      bCalidad.setBackgroundResource(R.drawable.negativo_aspa);
                        if (contError == 1) {
            Toast.makeText(getApplicationContext(),"La imagen tomada de "
+ UserID + " contiene 1 error de captura, para ver el Log de errores pulse el
aspa", Toast.LENGTH_LONG).show();
            bCalidad.setText("");
            } else {
            Toast.makeText(getApplicationContext(),"La imagen tomada de " +
UserID + " contiene " + contError + " errores de captura, para ver el Log de
errores pulse el aspa", Toast.LENGTH_LONG).show();
            bCalidad.setText("");
            }
                } else { // Si esta todo correcto
      bCalidad.setBackgroundResource(R.drawable.afirmativo_tick);
      bCalidad.setText("");
      Toast.makeText(getApplicationContext(), "La imagen tomada de " +
UserID + " ha pasado todos los controles de calidad",
Toast.LENGTH_LONG).show();
                              }
            } else {
                  if (ComprobError == 1) {
                  // hay fallo y se quiere acceder a la actividad que se
                  // vean los fallos (LogError)
                  Intent logError = new Intent(this, LogError.class);
                  logError.putExtra("contError", contError);
                  // Se pasa el array de errores que cada posición
                  // determina un error
                  ArrayErrores = setArrayErrores();
                  logError.putExtra("ArrayErrores", ArrayErrores);
                  startActivity(logError);
                  } else {
```

```
                    Toast.makeText(getApplicationContext(), "La imagen tomada
de " + UserID + " ya ha sido comprobada, seleccione otra imagen para un nuevo
análisis", Toast.LENGTH_LONG).show();
                                }
                            }
                    } else {
              Toast.makeText(getApplicationContext(), "No se ha tomado ninguna
imagen de " + UserID, Toast.LENGTH_SHORT).show();
                    }

                    break;

              case R.id.bGetIrisData:
        // Se carga la libreria y se implementan los procesos que use para
        // sacar
        // la mascara y el codigo de la corona
                    if (ComprobGetIris == 5) {

                          if (bmOjo != null) {
        // Se imlementa todas las funciones para sacar los datos de imagen
        // Si todo va bien retonra 0 si no se ha conseguido 1
        // HAY QUE IMPLEMENTAR GetIrisData_OK EN LA SALIDA DE DATOS DE MASK...
              if (ComprobError == 0) {
                    int p = ****Función Protegida****;
                       if (p == 0) {
                          Caract_Mask_UserID = ****Función Protegida****;
                          Caract_Code_UserID = ****Función Protegida****;
                                    } else {
                          Toast.makeText(getApplicationContext(),"No se ha
podido normalizar el iris de la captura de " + UserID,
Toast.LENGTH_LONG).show();
                                }
                    if ((Caract_Code_UserID == null) || (Caract_Mask_UserID ==
null)) {
                          GetIrisData_OK = 1;
                          ComprobGetIris = 0;
                          Toast.makeText( getApplicationContext(), "A
ocurrido un error al intentar obtener los datos de la captura de " + UserID +
", por favor, introduzca otra captura", Toast.LENGTH_LONG).show();
bGetIrisData.setBackgroundResource(R.drawable.negativo_aspa);
                          bGetIrisData.setText("");
                    } else {
                          GetIrisData_OK = 0;
                          ComprobGetIris = 0;
                    Toast.makeText(getApplicationContext(), "Se han podido
obtener las caracteríticas de la captura de " + UserID,
Toast.LENGTH_LONG).show();

        bGetIrisData.setBackgroundResource(R.drawable.afirmativo_tick);
        bGetIrisData.setText("");
                                }
                              } else {
                                    if (ComprobError == 5) {
                          Toast.makeText(getApplicationContext(),"Ha de pasar
primero el test de calidad de la captura de " + UserID,
Toast.LENGTH_LONG).show();
                                } else {
```

```
                        Toast.makeText(getApplicationContext(), "La captura
de " + UserID + " contiene errores de calidad, carge otra imagen",
Toast.LENGTH_LONG).show();
                                    }
                              }

                        } else {
                        Toast.makeText(getApplicationContext(), "No se ha
tomado ninguna imagen de " + UserID, Toast.LENGTH_SHORT).show();
                        }
                  } else if (ComprobGetIris == 0 && GetIrisData_OK == 0) {
                        Toast.makeText(getApplicationContext(), "Los datos
necesarios de " + UserID + " ya han sido tomados",
Toast.LENGTH_SHORT).show();
                  } else if (ComprobGetIris == 0 && GetIrisData_OK == 1) {
                        Toast.makeText(getApplicationContext(), "No se
puede sacar los datos de " + UserID + " cargue otra imagen",
Toast.LENGTH_SHORT).show();
                  }
                  break;

            case R.id.bSave:
      // Poner que se guarde si todo ha ido bien en la carga de variables...
                  All_OK = Comprob_All_OK();
                  if (All_OK == 0) {
                        // Guarda el code y el mask en archivos diferentes
                        saveToFile_code(Caract_Code_UserID);
                        saveToFile_mask(Caract_Mask_UserID);
                        Toast.makeText(getApplicationContext(),
                              "Datos de " + UserID + " guardado
correctamente", Toast.LENGTH_LONG).show();
                        // Se guarda en la lista de checked el true
                        String datacheckString = readFile(checkPath);
                        if (datacheckString != null) {
                              byte[] dataArraycheck =
datacheckString.getBytes();
                              Lista_ID_Checked
      .addAll(readFromArrayDataToListArray(dataArraycheck));
                        }
                        Lista_ID_Checked.set(Pos_ID, "true");
                        // Se guarda de nuevo todo el list checkeado
                        byte[] datos_check =
writeArraylistToByteArray(Lista_ID_Checked);
                        saveToFile(datos_check, checkPath);

                        // Retorno_Checked();
                        finish();
                  } else {
                        if (GetIrisData_OK == 1) {
            Toast.makeText(getApplicationContext(), "No se han podido
extraer las características correctamente de " + UserID,
Toast.LENGTH_LONG).show();
                        } else if (bmOjo == null) {
            Toast.makeText(getApplicationContext(), "No ha cargado ninguna
imagen para " + UserID, Toast.LENGTH_LONG).show();
                        } else if (ComprobError == 1) {
            Toast.makeText(getApplicationContext(), "No se puede guardar los
datos de  " + UserID + " mientras la captura contenga errores",
Toast.LENGTH_LONG).show();
```

```
                        } else {
            Toast.makeText(getApplicationContext(),"No hay datos de " +
UserID + " para guardar", Toast.LENGTH_LONG).show();
                        }

                  }
                  break;

            case R.id.bHelp:
                  Intent help = new Intent(this, Help.class);
                  String Actividad = "UsuarioData";
                  help.putExtra("ActivityFrom", Actividad);
                  startActivity(help);
                  break;
            }
      }

      private void saveToFile(byte[] datos, String FilePath) {
            try {
      FileOutputStream fOut = openFileOutput(FilePath,
Context.MODE_PRIVATE);
                  fOut.write(datos);
                  fOut.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }


      private int Comprob_All_OK() {
            if ((ComprobError == 0) && (bmOjo != null) && (GetIrisData_OK ==
0)) {
                  return 0;
            }
            return 1;
      }



// Función para añadir los errores y enviarlos a la actividad de registro de
errores
      private byte[] setArrayErrores() {
            byte[] Array = new byte[12];
            Array[0] = (byte) error_Sharpness;
            Array[1] = (byte) error_Image_Scale_H_Left;
            Array[2] = (byte) error_Image_Scale_H_Right;
            Array[3] = (byte) error_Image_Scale_V_Up;
            Array[4] = (byte) error_Image_Scale_V_Down;
            Array[5] = (byte) error_Contrast_Pupila;
            Array[6] = (byte) error_Contrast_Iris;
            Array[7] = (byte) error_Gray_Scale_Density;
            Array[8] = (byte) error_Pupil_Iris_Ratio;
            Array[9] = (byte) error_Iris_Size;
            Array[10] = (byte) error_Eccentric;
            Array[11] = (byte) error_Fraude;
            return Array;
      }

      private byte[] writeArraylistToByteArray(ArrayList<String> lista_ID) {
```

```
		byte[] list_array = null;
		ByteArrayOutputStream baos = new ByteArrayOutputStream();
		DataOutputStream out = new DataOutputStream(baos);
		for (String element : lista_ID) {
			try {
				out.writeUTF(element);
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
		list_array = baos.toByteArray();
		return list_array;
	}

	@Override
	protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
		super.onActivityResult(requestCode, resultCode, data);

		switch (requestCode) {
		case CAMERA_IMAGE:
			if (resultCode == RESULT_OK) {

				// **Para mostrar la imagen de la cámara**
				Bundle extras = data.getExtras();
				bmOjo = (Bitmap) extras.get("data");
				ivImagePreview.setImageBitmap(bmOjo);
				ComprobError = 5;
				bGetIrisData

	.setBackgroundResource(android.R.drawable.btn_default);

	bCalidad.setBackgroundResource(android.R.drawable.btn_default);
				bCalidad.setText("Comprobar calidad");
				bGetIrisData.setText("Sacar datos Iris");
				contError = 0;
				ArrayErrores = null;
				ComprobError = 5;
			}
			break;

		case SELECT_IMAGE:
			if (resultCode == RESULT_OK) {

				SelectImagUser = data.getData();
			String[] FilePathArray = { MediaStore.Images.Media.DATA };
			Cursor cursor = getContentResolver().query(SelectImagUser,
		FilePathArray, null, null, null);
				cursor.moveToFirst();
				int ColumnIndex =
cursor.getColumnIndex(FilePathArray[0]);
				FilePathUser = cursor.getString(ColumnIndex);
				cursor.close();
				bmOjo = BitmapFactory.decodeFile(FilePathUser);

				ivImagePreview.setImageBitmap(bmOjo);
				ComprobError = 5;
				// Se establece el boton con la aparencia inicial
```

```
                        bGetIrisData
.setBackgroundResource(android.R.drawable.btn_default);

        bCalidad.setBackgroundResource(android.R.drawable.btn_default);
                        bCalidad.setText("Comprobar calidad");
                        bGetIrisData.setText("Sacar datos Iris");
                        contError = 0;
                        ArrayErrores = null;
                        ComprobError = 5;
                }
                break;
        }
    }

    private void saveToFile_mask(byte[] data) {
        // Se guarda en un file que se llame UserID
        try {
FileOutputStream fOut = openFileOutput(FilePath_Mask_UserID,
                        Context.MODE_PRIVATE);
            fOut.write(data);
            fOut.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void saveToFile_code(byte[] data) {
        // Se guarda en un file que se llame UserID
        try {
FileOutputStream fOut = openFileOutput(FilePath_Code_UserID,
                        Context.MODE_PRIVATE);
            fOut.write(data);
            fOut.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public String readFile(String fileName) {
        String dataget = null;
        try {
            BufferedReader input = new BufferedReader(new
InputStreamReader(
                        openFileInput(fileName)));
            dataget = input.readLine();
            input.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return dataget;
    }

    private ArrayList<String> readFromArrayDataToListArray(byte[]
dataArray) {
        ArrayList<String> Lista_IDS = new ArrayList<String>();
        ByteArrayInputStream bais = new ByteArrayInputStream(dataArray);
        DataInputStream in = new DataInputStream(bais);
        try {
            while (in.available() > 0) {
```

```
                        String element = in.readUTF();
                        Lista_IDS.add(element);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
                return Lista_IDS;
        }

        // Coge el dato de ID de la actividad MenuReclut y la carga en la
actividad
        private void getdataId() {
                // String ItemID; "UserID"
                Bundle gotString = getIntent().getExtras();
                UserID = gotString.getString("UserID");
                tvIDName.setText("ID: " + UserID);

                // Se crean las direcciones para el ID específico
                FilePath_Code_UserID = UserID + "_code";
                FilePath_Mask_UserID = UserID + "_mask";

                // Se recoge el valor de la posicion
                Bundle gotPos = getIntent().getExtras();
                Pos_ID = gotPos.getInt("Pos_ID");
        }

        // Si todo va bien retorna 0 si hay algun error retorna un 1
        public int ComprobCalidad() {
        // TODOS Los procesos necesarios para la comprobacion de que la imagen
        // es la correcta
        ****Función Protegida****
        // Pasar a escala de grises
        bmOjo = ConvertToGrayscale_Preciso(bmOjo);
        // Cargar datos en los Structs
        ****Función Protegida****
// a) Reducir imagen a tamaño fijo (ancho más o menos en 150 pixeles) y
        // quitar fondos
                // Se calcula el valor de reducción
                ****Función Protegida****

                ****Función Protegida****
                ****Función Protegida****
                // b) Histograma
                ****Función Protegida****
                // c) Busqueda Pupila
                ****Función Protegida****
                // e) Ubicar borde externo
                // Si respuesta es 0 entonces se ha hecho con exito sino no.
                int respuesta = ****Función Protegida****;
                if (respuesta == 0) {
                        // f) si todo ha ido bien, buscar los párpados
                        ****Función Protegida****
                }

                bmOjoContorno = Bitmap.createBitmap(bmOjo.getWidth(),
                                bmOjo.getHeight(), Bitmap.Config.RGB_565);
                bmOjoContorno = bmOjo;

                ****Función Protegida****
```

```
            ivImagePreview.setImageBitmap(bmOjoContorno);

            CalidadPost();
contError = error_Sharpness + error_Image_Scale_H_Left +
error_Image_Scale_H_Right + error_Image_Scale_V_Up + error_Image_Scale_V_Down
+ error_Contrast_Pupila + error_Contrast_Iris + error_Gray_Scale_Density +
error_Pupil_Iris_Ratio + error_Iris_Size + error_Eccentric;

            if (contError == 0) {
                  return 0;
            } else {
                  return 1;
            }

      }


      public Bitmap Array_To_Bitmap_GettingOneByOne(byte[] data, int width,
                  int height) {
      // Se crea un bitmap mutable y se introduce los datos pixel a pixel.
            Bitmap bmp = Bitmap.createBitmap(width, height,
Bitmap.Config.RGB_565);
            int R, pixel;
            for (int y = 0; y < height; y++)
                  for (int x = 0; x < width; x++) {
                        pixel = data[x + width * y];
                        R = pixel & 0xFF;
                        bmp.setPixel(x, y, Color.rgb(R, R, R));
                  }
            return bmp;
      }


      public Bitmap ConvertToGrayscale_Preciso(Bitmap bmSource) {
            // Dos metodos de paso a escala de grises:

            // PRIMER MÉTODO, lento de alta calidad.

            Bitmap bmAux = Bitmap.createBitmap(bmSource.getWidth(),
                        bmSource.getHeight(), Bitmap.Config.RGB_565);
            int c, R, G, B, luma;
            for (int y = 0; y < bmAux.getHeight(); y++) {
                  for (int x = 0; x < bmAux.getWidth(); x++) {

                        // Se recoje el color de la dirección

                        c = bmSource.getPixel(x, y);
                        R = Color.red(c);
                        G = Color.green(c);
                        B = Color.blue(c);
                        luma = (int) (R * 0.3 + G * 0.59 + B * 0.11);
                        bmAux.setPixel(x, y, Color.rgb(luma, luma, luma));
                  }
            }
            return bmAux;
      }

      public Bitmap ConvertToGrayscale_Rapido(Bitmap bmSource) {
      // SEGUNDO MÉTODO, rapido pero de menor calidad.
```

```
        int height = bmSource.getHeight();
        int width = bmSource.getWidth();

        Bitmap bmpGrayscale = Bitmap.createBitmap(width, height,
                    Bitmap.Config.RGB_565);
        Canvas c = new Canvas(bmpGrayscale);
        Paint paint = new Paint();
        ColorMatrix cm = new ColorMatrix();
        cm.setSaturation(0);
        ColorMatrixColorFilter f = new ColorMatrixColorFilter(cm);
        paint.setColorFilter(f);
        c.drawBitmap(bmSource, 0, 0, paint);
        return bmpGrayscale;
    }


    void CalidadPost() {
        // Si error = 0 todo OK
        // Si error_... = 1 hay error
        if (Sharpness() == 1) {
            error_Sharpness = 1;
        } else {
            error_Sharpness = 0;
        }
        if (Image_Scale() == 2) {
            error_Image_Scale_H_Left = 1;
        } else {
            error_Image_Scale_H_Left = 0;
        }
        if (Image_Scale() == 3) {
            error_Image_Scale_H_Right = 1;
        } else {
            error_Image_Scale_H_Right = 0;
        }
        if (Image_Scale() == 4) {
            error_Image_Scale_V_Up = 1;
        } else {
            error_Image_Scale_V_Up = 0;
        }
        if (Image_Scale() == 5) {
            error_Image_Scale_V_Down = 1;
        } else {
            error_Image_Scale_V_Down = 0;
        }
        if (Contrast() == 6) {
            error_Contrast_Pupila = 1;
        } else {
            error_Contrast_Pupila = 0;
        }
        if (Contrast() == 7) {
            error_Contrast_Iris = 1;
        } else {
            error_Contrast_Iris = 0;
        }
        if (Gray_Scale_Density() == 8) {
            error_Gray_Scale_Density = 1;
        } else {
            error_Gray_Scale_Density = 0;
```

```
            }
            if (Pupil_Iris_Ratio() == 10) {
                  error_Pupil_Iris_Ratio = 1;
            } else {
                  error_Pupil_Iris_Ratio = 0;
            }
            if (Iris_Size() != 0) {
                  error_Iris_Size = 1;
            } else {
                  error_Iris_Size = 0;
            }
            if (Eccentric() != 0) {
                  error_Eccentric = 1;
            } else {
                  error_Eccentric = 0;
            }
            }
```

UsuariosData.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingBottom="10dp" >

    <Button
        android:id="@+id/bHelp"
        android:layout_width="wrap_content"
        android:layout_height="30dp"
        android:background="@drawable/actionhelpwhite"
        android:layout_gravity="right" />

    <TextView
        android:id="@+id/tvName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="ID: "
        android:textColor="#0000FF"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:paddingTop="7dp"
        android:text="Captura del iris del sujeto"
        android:textColor="#00AF00"
        android:textSize="14dp"
        android:paddingBottom="10dp" />
```

```
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal"
        android:paddingLeft="55dp"
            android:paddingRight="55dp"
        android:weightSum="100" >

        <Button
            android:id="@+id/bFolderUser"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_weight="50"
            android:background="@drawable/icon_folder" />

        <Button
            android:id="@+id/bCamaraUser"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_weight="50"
            android:background="@drawable/icon_camara" />
    </LinearLayout>

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Preview de la imagen"
        android:textColor="#00DF00"
        android:textSize="14dp"
        android:paddingTop="10dp" />

    <ImageView
        android:id="@+id/ivPrevioUser"
        android:layout_width="fill_parent"
        android:layout_height="125dp"
        android:layout_gravity="center"
        android:padding="10dp"
        android:src="@drawable/android512" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:paddingBottom="12dp"
        android:text="Obtener datos necesarios de la imagen: Codigo_corona y
mascara_corona"
        android:textColor="#00FF00"
        android:textSize="14dp" />

    <LinearLayout
        android:layout_width="243dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal"
        android:padding="14dp"
```

```
        android:weightSum="100" >

        <Button
            android:id="@+id/bGetIrisData"
            android:layout_width="75dp"
            android:layout_height="50dp"
            android:layout_weight="50"
            android:text="Sacar datos Iris" />

        <Button
            android:id="@+id/bCalidad"
            android:layout_width="75dp"
            android:layout_height="50dp"
            android:layout_weight="50"
            android:text="Comprobar calidad" />
    </LinearLayout>

    <Button
        android:id="@+id/bSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Guardar en memoria interna" />

      </LinearLayout>
```

Seleccionar usuario para verificación

SelectUserVerif.java

```
public class SelectUserVerif extends Activity {

      ArrayList<String> Lista_IDs;
      ArrayAdapter<String> aaListIDs;
      byte[] datos_listID = null;
      // Tiene que ser la misma que en la actividad MenuReclut
      String fileName = "path_list";
      String checkPath = "path_check";

      ArrayList<String> Lista_ID_Checked;

      /*** ---VARIABLES DE PRESENTACIÓN Y BOTONES--- ***/
      ListView listUser;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            requestWindowFeature(Window.FEATURE_NO_TITLE);
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN);

            setContentView(R.layout.selectuserverif);

            initialize_android();
```

```
		Toast.makeText(getApplicationContext(),"Solo se mostrarán los sujetos
con datos guardados", Toast.LENGTH_SHORT).show();
		}

		private void initialize_android() {
			listUser = (ListView) findViewById(R.id.lstUser);

			Lista_ID_Checked = new ArrayList<String>();
			Lista_IDs = new ArrayList<String>();
			aaListIDs = new ArrayAdapter<String>(this,
					android.R.layout.simple_list_item_1, Lista_IDs);
			listUser.setAdapter(aaListIDs);

			// Lo mismo, para CARGAR la lista pero de checked
			String datacheckString = readFile(checkPath);
			if (datacheckString != null){
				byte[] dataArraycheck = datacheckString.getBytes();

		Lista_ID_Checked.addAll(readFromArrayDataToListArray(dataArraycheck));
			}
			// Para CARGAR la lista de sujetos guardada nada mas empezar la
			// actividad
			String dataString = readFile(fileName);
			if (dataString != null) {
				byte[] dataArray = dataString.getBytes();
				Lista_IDs.addAll(readFromArrayDataToListArray(dataArray));
				aaListIDs.notifyDataSetChanged();
			} else {
				Toast.makeText(getApplicationContext(),
						"No hay ningun sujetos reclutados",
						Toast.LENGTH_SHORT).show();
				finish();
			}

// Recorro el array de checked y elimino los que no estan con datos solo en
el View
			for (int i=0; i<Lista_IDs.size(); i++) {
				String result = Lista_ID_Checked.get(i);
				if (result.equals("true") == false){
					Lista_IDs.remove(i);
				}
			}

			listUser.setOnItemClickListener(new OnItemClickListener() {

				public void onItemClick(AdapterView<?> arg0, View arg1,
						int position, long id) {

					String result = Lista_ID_Checked.get(position);

					if (result.equals("true") == true){
					Toast.makeText(getApplicationContext(),
						Ha seleccionado el ID: " + ((TextView)
arg1).getText(),Toast.LENGTH_SHORT).show();

// Envia a la actividad UsuariosData el nombre del ID.
				String NombreID = ((TextView) arg1).getText().toString();
Intent MenuVerif = new Intent(SelectUserVerif.this, MenuVerific.class);
					MenuVerif.putExtra("UserID", NombreID);
```

```
                        startActivity(MenuVerif);
                        finish();
                        } else {
            Toast.makeText(getApplicationContext(),"El sujeto " +
((TextView) arg1).getText() + " no contiene datos, seleccione otro sujeto",
Toast.LENGTH_SHORT).show();
                        }
                  }
            });
      }

      private ArrayList<String> readFromArrayDataToListArray(byte[]
dataArray) {
            ArrayList<String> Lista_IDS = new ArrayList<String>();
            ByteArrayInputStream bais = new ByteArrayInputStream(dataArray);
            DataInputStream in = new DataInputStream(bais);
            try {
                  while (in.available() > 0) {
                        String element = in.readUTF();
                        Lista_IDS.add(element);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return Lista_IDS;
      }

      private String readFile(String fileName2) {
            String dataget = null;
            try {
                  BufferedReader input = new BufferedReader(new
InputStreamReader(
                  openFileInput(fileName2)));
                  dataget = input.readLine().toString();
                  input.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return dataget;
      }

}
```

selectuserverif.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical"
    android:padding="15dp" >

    <TextView
        android:id="@+id/tvTitulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_gravity="center"
        android:text="Sujetos"
        android:textColor="#0000FF"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Seleccione el ID del sujeto a analizar"
        android:textColor="#009F00"
        android:textSize="12dp" />

    <ListView
        android:id="@+id/lstUser"
        android:layout_width="fill_parent"
        android:layout_height="164dp"
        android:layout_weight="0.15" />

      </LinearLayout>
```

Menú de verificación

MenuVerific.java

```
public class MenuVerific extends Activity implements OnClickListener {

      Button bSFolder, bVerificar, bSCamara;
      ImageView ivPrevio;
      TextView tvNomFich;
      CheckBox cbPasos;
      Intent VerifDir, VerifPasos;
      Intent TakePhoto, Gallery;

      // **Establecer las Variables como final y con valor estático hasta
nuevo
      // cambio**
      private static final int SELECT_IMAGE = 0;
      private static final int CAMERA_IMAGE = 1;

      String UserID;
      Uri SelectImag;
      String FilePath;
      Bundle fieldresults;

      // Para seleccionar una imagen del terminal
      // String nameFich = "Nombre Fichero";
      // Intent PickImg, captura;
      // int code;

      Bitmap bmOjo = null;
      byte[] arrayOjo;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
```

```
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.menuverific);
		initialize();
		getUserID();

		// nameFich = Environment.getExternalStorageDirectory() +
		// "/EyeLocker/Ojo_Seleccionado.jpg";
	}

	private void getUserID() {
		// String ItemID; "UserID"
		Bundle gotString = getIntent().getExtras();
		UserID = gotString.getString("UserID");
		tvNomFich.setText("ID: " + UserID);
	}

	private void initialize() {
		// TODO Auto-generated method stub
		bSFolder = (Button) findViewById(id.bFolder);
		bVerificar = (Button) findViewById(id.bVerif);
		ivPrevio = (ImageView) findViewById(id.ivMenuOjo);
		tvNomFich = (TextView) findViewById(id.tvNomFich);
		cbPasos = (CheckBox) findViewById(id.cbPasoProc);
		bSCamara = (Button) findViewById(id.bCamara);

		bVerificar.setOnClickListener(this);
		bSFolder.setOnClickListener(this);
		bSCamara.setOnClickListener(this);
	}

	// Para llamar a un menu inflable de pref. hay que crear el metodo
	@Override
	public boolean onCreateOptionsMenu(android.view.Menu menu) {
		// TODO Auto-generated method stub
		// Para que aparezca el menu inflable
		MenuInflater blowUp = getMenuInflater();
		blowUp.inflate(R.menu.prefmenu, menu);
		return true;
	}

// Cada vez que es pulsado un boton del menu inflable se llama a este método
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// TODO Auto-generated method stub
		switch (item.getItemId()) {
		case R.id.aboutUs:
			Intent i = new
Intent("es.deluxe.deseiken.eyelocker.ABOUT");
			startActivity(i);
			break;
		case R.id.preferences:
			Intent p = new
Intent("es.deluxe.deseiken.eyelocker.PREFS");
```

```
                startActivity(p);
                break;
            case R.id.exit:
                finish();
                break;
            }
            return false;
        }

        public void onClick(View pressed) {

            switch (pressed.getId()) {
            case R.id.bFolder:
                Gallery = new Intent(Intent.ACTION_GET_CONTENT);
                // Establece el modo de visión de busqueda de imagen
                Gallery.setType("image/*");
                startActivityForResult(Gallery, SELECT_IMAGE);
                break;

            case R.id.bCamara:
                TakePhoto = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(TakePhoto, CAMERA_IMAGE);
                break;

            case R.id.bVerif:

                if (bmOjo != null) {
                    // Se carga el valor de la imagen para la actividad
                    arrayOjo = Bitmap_To_Array(bmOjo);
                    VerifDir = new Intent(this, VerificDirecta.class);
                    VerifPasos = new Intent(this, VerificPasos.class);

                    // Se usa putExtra para incluir los datos
                    VerifDir.putExtra("ArrayOjo", arrayOjo);
                    VerifDir.putExtra("UserID", UserID);
                    VerifPasos.putExtra("ArrayOjo", arrayOjo);
                    VerifPasos.putExtra("UserID", UserID);

                    if (cbPasos.isChecked() == false) {
            // **Se activa la actividad Verificacion Directa**
                        startActivity(VerifDir);
                        finish();
                    } else {
            // **Se activa la actividad Verificacion por Pasos**
                        startActivity(VerifPasos);
                        finish();
                    }
                } else {
            Toast.makeText(getApplicationContext(),"No hay captura de " +
UserID, Toast.LENGTH_LONG).show();
                }

                break;
            }

        }

        @Override
```

```
	protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
		// TODO Auto-generated method stub
		super.onActivityResult(requestCode, resultCode, data);

		switch (requestCode) {
		case CAMERA_IMAGE:
			if (resultCode == RESULT_OK) {

				// **Para mostrar la imagen de la cámara**
				Bundle extras = data.getExtras();
				bmOjo = (Bitmap) extras.get("data");
				ivPrevio.setImageBitmap(bmOjo);
			}
			break;

		case SELECT_IMAGE:
			if (resultCode == RESULT_OK) {

		SelectImag = data.getData();
		String[] FilePathArray = { MediaStore.Images.Media.DATA };
		Cursor cursor = getContentResolver().query(SelectImag,
					FilePathArray, null, null, null);
		cursor.moveToFirst();
		int ColumnIndex = cursor.getColumnIndex(FilePathArray[0]);
				FilePath = cursor.getString(ColumnIndex);
				cursor.close();
				bmOjo = BitmapFactory.decodeFile(FilePath);

				ivPrevio.setImageBitmap(bmOjo);
			}
			break;
		}
	}

	public byte[] Bitmap_To_Array(Bitmap Source) {
		Bitmap bmOjo_aux = Source;
		// Saco el tamaño del bitmap para el array
		int size = Source.getWidth() * Source.getHeight();
		ByteArrayOutputStream out = new ByteArrayOutputStream(size);
		// Pasa el bitmap a PNG y no se realiza ninguna compresion
		bmOjo_aux.compress(Bitmap.CompressFormat.PNG, 100, out);
		// El valor de bmArrayOjo es el resultado de array de la imagen
		byte[] ArrayOjo = out.toByteArray();

		return ArrayOjo;
	}
}
```

menuverific.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="15dp"
        android:orientation="vertical"
        android:paddingBottom="5dp" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:orientation="horizontal"
            android:paddingLeft="55dp"
            android:paddingRight="55dp"
            android:weightSum="100" >

            <Button
                android:id="@+id/bFolder"
                android:layout_width="75dp"
                android:layout_height="75dp"
                android:layout_weight="50"
                android:background="@drawable/icon_folder" />

            <Button
                android:id="@+id/bCamara"
                android:layout_width="75dp"
                android:layout_height="75dp"
                android:layout_weight="50"
                android:background="@drawable/icon_camara" />
        </LinearLayout>

        <TextView
            android:id="@+id/tvNomFich"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Preview Imagen"
            android:textSize="20dp"
            android:textColor="#00DF00" />

        <ImageView
            android:id="@+id/ivMenuOjo"
            android:layout_width="250dp"
            android:layout_height="250dp"
            android:layout_gravity="center"
            android:layout_marginBottom="17dp"
            android:src="@drawable/android512" />

        <Button
            android:id="@+id/bVerif"
```

```
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="17dp"
            android:text="Verificar" />

        <CheckBox
            android:id="@+id/cbPasoProc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:checked="true"
            android:text="Ver Proceso por pasos" />
    </LinearLayout>

</LinearLayout>
```

Verificación por pasos

Por confidencialidad no se permite incluir todo el código de esta actividad. Esto es debido a que está basado en un código que se encuentra bajo protección industrial de la Universidad Carlos III. En los campos correspondientes se a procedido ha su entera eliminación, y en las declaraciones de las mismas han sido sustituidas por el indicativo "****Función Protegida****". Se han eliminado las funciones traducidas he implementadas del proyecto IrisBiometrics, variables globales que usa dicha biblioteca y sus declaraciones. Todo ellos para la protección del concepto de estos datos.

VerificPasos.java

```
public class VerificPasos extends Activity implements OnClickListener {

	/*** ---CONSTANES DE CÓDIGO--- ***/
	int error_Sharpness, error_Image_Scale_H_Left,
error_Image_Scale_H_Right, error_Image_Scale_V_Up, error_Image_Scale_V_Down,
error_Contrast_Pupila, error_Contrast_Iris, error_Gray_Scale_Density,
error_Pupil_Iris_Ratio, error_Iris_Size, error_Eccentric, error_Fraude = 0;

	int contError = 0;
	int ComprobError = 0;
	byte[] ArrayErrores = null;

	// Las direcciones donde se almacenan los datos de caract. del sujeto
	String FilePath_Code_UserID;
	String FilePath_Mask_UserID;

	long startTime, finishTime;
	long memocronometro;

	/*** ---VARIABLES DE PRESENTACIÓN Y BOTONES--- ***/
	Bitmap foto = null, bmOjo = null, bmOjoContorno = null;
	ImageView ivPasos;
```

```
	Button bHelp, bNext;
	TextView tvPason, tvDescrip, tvChrono;

	// Variables de pasos
	String tvPason_paso_n;
	String tvDescrip_paso_n;
	int cont = 1;
	int finalizar = 0;
	byte[] arrayOjo, arrayOjoaux;
	byte[] bmArrayOjo = null;
	String UserID;
	Bitmap bmCannyView = null;
	Bitmap dibuCartes = null;
// Para la extracción de características de la imagen a verificar
	byte[] Caract_Code_Verif, Caract_Mask_Verif;
// Para la obtención de caracteríticas de la imagen del User en memoria
	byte[] Caract_Code_UserID, Caract_Mask_UserID;
	// Comparación por Hamming
	float Dist_Hamming;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.verificpasos);
		****Función Protegida****
		initialize_android();
		Load_image();
		getdataId();
		textos_pasos();
	Caract_Code_UserID = get_Caract_Code_UserID(FilePath_Code_UserID);
	Caract_Mask_UserID = get_Caract_Mask_UserID(FilePath_Mask_UserID);

		ivPasos.setImageBitmap(bmOjo);

		// Set de colores en los textos y tamaños
		tvPason.setTextColor(Color.rgb(155, 0, 255));
		tvDescrip.setTextColor(Color.rgb(0, 255, 0));
		tvDescrip_paso_n = "Cargar la imagen en la base de datos y
pasarlo a array. Se inicializa las variables";
		tvDescrip.setText(tvDescrip_paso_n);
	}

	private byte[] get_Caract_Mask_UserID(String filePath_Mask_UserID2) {
		// Read y se guarda en Caract_Mask_UserID;
		byte[] dataget = null;
		try {
			BufferedReader input = new BufferedReader(new
InputStreamReader(
					openFileInput(filePath_Mask_UserID2)));
			dataget = input.readLine().getBytes();
			input.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
```

```
            return dataget;
      }

      private byte[] get_Caract_Code_UserID(String filePath_Code_UserID2) {
            // Read y se guarda en Caract_Code_UserID;
            byte[] dataget = null;
            try {
BufferedReader input = new BufferedReader(new
InputStreamReader(openFileInput(filePath_Code_UserID2)));
                  dataget = input.readLine().getBytes();
                  input.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return dataget;
      }

      private void Load_image() {
            Bundle gotBasket = getIntent().getExtras();
            arrayOjo = gotBasket.getByteArray("ArrayOjo");
            bmOjo = Array_To_Bitmap(arrayOjo);
            bmOjoContorno = bmOjo;
      }

      private void getdataId() {
            // String ItemID; "UserID"
            Bundle gotString = getIntent().getExtras();
            UserID = gotString.getString("UserID");

            // Saco las direcciones de caract. de ese sujeto
            FilePath_Code_UserID = UserID + "_code";
            FilePath_Mask_UserID = UserID + "_mask";
      }

      private Bitmap Array_To_Bitmap(byte[] arraySource) {
            Bitmap bmSource = BitmapFactory.decodeByteArray(arraySource, 0,
                        arraySource.length);

            // Paso el Bitmap creado (Imutable) a mutable
            Bitmap bmSourceMutable = bmSource.copy(Bitmap.Config.RGB_565,
true);

            return bmSourceMutable;
      }

      private void initialize_android() {

            ivPasos = (ImageView) findViewById(id.ivOjoPasos);
            bHelp = (Button) findViewById(id.bHelp);
            tvPason = (TextView) findViewById(id.tvPason);
            tvDescrip = (TextView) findViewById(id.tvDescrip);
            bNext = (Button) findViewById(id.bNext);
            tvChrono = (TextView) findViewById(id.tvChrono);
            bHelp.setOnClickListener(this);
            bNext.setOnClickListener(this);
      }


      public void onClick(View pressed) {
```

```
		switch (pressed.getId()) {
		case R.id.bHelp:
			Intent help = new Intent(this, Help.class);
			String Actividad = "VerificPasos";
			help.putExtra("contPaso", cont);
			help.putExtra("ActivityFrom", Actividad);
			startActivity(help);
			break;

		case R.id.bNext:
			if (ComprobError == 0) {
				cont++;
				textos_pasos();
			} else {
				// Hay errores y no se puede continuar
				// Se abre el LogError
				Intent logError = new Intent(this, LogError.class);
				logError.putExtra("contError", contError);

				ArrayErrores = setArrayErrores();
				logError.putExtra("ArrayErrores", ArrayErrores);
				startActivity(logError);
				finish();
			}
			break;
		}
	}

	private void textos_pasos() {
		switch (cont) {
		case 1:
			tvPason_paso_n = "1-Inicializar";
			tvDescrip_paso_n = "Adquisición de la imagen del supuesto
sujeto e Inicialización";
			tvPason.setText(tvPason_paso_n);
			tvDescrip.setText(tvDescrip_paso_n);
			break;

		case 2:
			// Empieza a contar el reloj
			startTime = System.nanoTime();

			// ** COMIENZAN LOS PROCESOS **
			// Pasa a escala de grises y se muestra por pantalla
			bmOjo = ConvertToGrayscale_Preciso(bmOjo);
			// Cargar datos en los Structs. Iris_Open
			****Función Protegida****
			// ** TERMINAN LOS PROCESOS **
			finishTime = System.nanoTime() - startTime;
			// Lo paso a milisegundos ya que es muy grande el valor
			finishTime = (long) (finishTime * 0.000001);
			memocronometro += finishTime;
			tvChrono.setText("Tiempo proc. 2: " + finishTime + "ms");

			ivPasos.setImageBitmap(bmOjo);
			tvPason_paso_n = "2-Conversión";
			tvDescrip_paso_n = "Prepara el Pre-Procesado de la imagen,
pasar a escala de grises y carga los datos";
			tvPason.setText(tvPason_paso_n);
```

```
                    tvDescrip.setText(tvDescrip_paso_n);
                    break;
              case 3:
                    // Empieza a contar el reloj
                    startTime = System.nanoTime();

                    // ** COMIENZAN LOS PROCESOS **
                    Preprocesado();
                    // Se muestra el contorno de Iris y Pupila localizado
                    ****Función Protegida****
                    // ** TERMINAN LOS PROCESOS **
                    finishTime = System.nanoTime() - startTime;
                    // Lo paso a milisegundos ya que es muy grande el valor
                    finishTime = (long) (finishTime * 0.000001);
                    memocronometro += finishTime;
              tvChrono.setText("Tiempo proc. 3.1: " + finishTime + "ms");

                    tvPason_paso_n = "3.1-Preprocesado";
                    tvDescrip_paso_n = "Se calcula el histograma, se quita los
fondos y se busca el radio y centro de pupila e iris";
                    tvPason.setText(tvPason_paso_n);
                    tvDescrip.setText(tvDescrip_paso_n);
                    ivPasos.setImageBitmap(bmOjoContorno);
                    break;
              case 4:
                    tvPason_paso_n = "3.2-Preprocesado";
                    tvDescrip_paso_n = "Busqueda de la frontera de los
párpados. Creación de la máscara apartir del iris";
                    tvPason.setText(tvPason_paso_n);
                    tvDescrip.setText(tvDescrip_paso_n);
                    // Se pone el bitmap de mascara
                    ivPasos.setImageBitmap(dibuCartes);

                    tvChrono.setText("No hay procesos");
                    break;
              case 5:
                    tvPason_paso_n = "3.3-Preprocesado";
                    tvDescrip_paso_n = "Aplicación del filtrado de Canny para
delimitar el borde y así crear la máscara";
                    tvPason.setText(tvPason_paso_n);
                    tvDescrip.setText(tvDescrip_paso_n);
                    // Se pone el bitmap de la máscara con la aplicacion
                    // de Canny
                    ivPasos.setImageBitmap(bmCannyView);
                    tvChrono.setText("No hay procesos");
                    break;
              case 6:
                    // Empieza a contar el reloj
                    startTime = System.nanoTime();

                    // ** COMIENZAN LOS PROCESOS **
                    // Comprobación de errores en la captura de la
                    // imagen
                    ComprobError = CalidadPost();
                    // Cuando ComprobError == 1 hay errores
                    if (ComprobError == 1) {
                          bNext.setText("ERROR");
                          bNext.setTextColor(Color.rgb(255, 0, 0));
                          tvPason.setText("ERROR");
```

```
                        tvPason.setTextColor(Color.rgb(255, 0, 0));
                        tvDescrip.setText("No se puede continuar con el
analisis si la captura contiene errores");
                        if (contError == 1) {
            Toast.makeText(getApplicationContext(), "Hay " + contError
+ " error de lectura, pulsar ERROR para ver el ErrorLOG",
Toast.LENGTH_LONG).show();
                        } else {
            Toast.makeText(getApplicationContext(),Hay un total de "
+ contError + " errores de lectura, pulsar ERROR para ver el ErrorLOG",
Toast.LENGTH_LONG).show();
                        }
                  } else {
            Toast.makeText(getApplicationContext(),"La captura ha superado
todos los controles de calidad",Toast.LENGTH_LONG).show();
                  }
                  // ** TERMINAN LOS PROCESOS **
                  finishTime = System.nanoTime() - startTime;
                  // Lo paso a milisegundos ya que es muy grande el valor
                  finishTime = (long) (finishTime * 0.000001);
                  memocronometro += finishTime;
                  tvChrono.setText("Tiempo proc. 4: " + finishTime + "ms");

                  tvPason_paso_n = "4-Test de calidad";
                  tvDescrip_paso_n = "La imagen pasa un test de calidad de
captura para ver si es válida";
                  tvPason.setText(tvPason_paso_n);
                  tvDescrip.setText(tvDescrip_paso_n);
                  ivPasos.setImageBitmap(bmOjoContorno);
                  break;
            case 7:
                  startTime = System.nanoTime();

                  // ** COMIENZAN LOS PROCESOS **
                  ****Función Protegida****
                  // ** TERMINAN LOS PROCESOS **

                  finishTime = System.nanoTime() - startTime;
                  // Lo paso a milisegundos ya que es muy grande el valor
                  finishTime = (long) (finishTime * 0.000001);
                  memocronometro += finishTime;
                  tvChrono.setText("Tiempo proc. 5: " + finishTime + "ms");

                  tvPason_paso_n = "5-Normalizar";
                  tvDescrip_paso_n = "Transf. a coordenadas cartesianas del
iris y su máscara";
                  tvPason.setText(tvPason_paso_n);
                  tvDescrip.setText(tvDescrip_paso_n);
                  break;

            case 8:
                  startTime = System.nanoTime();

                  // ** COMIENZAN LOS PROCESOS **
                  Caract_Mask_Verif = ****Función Protegida****;
                  Caract_Code_Verif = ****Función Protegida****;
                  // ** TERMINAN LOS PROCESOS **

                  finishTime = System.nanoTime() - startTime;
```

```
                    // Lo paso a milisegundos ya que es muy grande el valor
                    finishTime = (long) (finishTime * 0.000001);
                    memocronometro += finishTime;
                    tvChrono.setText("Tiempo proc. 6: " + finishTime + "ms");

                    tvPason_paso_n = "6-Extraer caract.";
                    tvDescrip_paso_n = "Se extraen las características
necesarias para la comparación";
                    tvPason.setText(tvPason_paso_n);
                    tvDescrip.setText(tvDescrip_paso_n);
                    break;

             case 9:
                    startTime = System.nanoTime();

                    // ** COMIENZAN LOS PROCESOS **
                    Dist_Hamming = ****Función Protegida****;
                    // ** TERMINAN LOS PROCESOS **

                    finishTime = System.nanoTime() - startTime;
                    // Lo paso a milisegundos ya que es muy grande el valor
                    finishTime = (long) (finishTime * 0.000001);
                    memocronometro += finishTime;
                    tvChrono.setText("Tiempo proc. 7: " + finishTime + "ms");

                    tvPason_paso_n = "7-Comparación";
                    tvDescrip_paso_n = "Se compara la imagen obtenida con la
guardada en el sujeto";
                    tvPason.setText(tvPason_paso_n);
                    tvDescrip.setText(tvDescrip_paso_n);

                    break;

             case 10:
                    tvPason_paso_n = "8-Resultado";
                    tvPason.setText(tvPason_paso_n);
             tvChrono.setText("Tiempo TOTAL: " + memocronometro + "ms");

                    float Porc_acierto = 100 - Dist_Hamming * 100;
                    // Limito el numero de decimales a mostrar
                    String Porc = String.format("%.5f", Porc_acierto);
                    if (Dist_Hamming < 0.25) {
                           tvDescrip_paso_n = "La persona ha sido identificada
como la correcta en un "
                                         + Porc + "% de acierto";
                           tvDescrip.setText(tvDescrip_paso_n);
                           bNext.setTextColor(Color.rgb(0, 255, 0));

                           InputStream is = getResources().openRawResource(
                                         R.drawable.acceso_concedido);
                           Bitmap auxSource = BitmapFactory.decodeStream(is);
                           ivPasos.setImageBitmap(auxSource);

                    Toast.makeText(getApplicationContext(),"VERIFICACIÓN DE "
+ UserID + " ACEPTADA", Toast.LENGTH_LONG).show();
                    } else {
                           tvDescrip_paso_n = "La persona no ha sido
identificada como la correcta en un "
                                         + Porc + "% de acierto";
```

```
                        tvDescrip.setText(tvDescrip_paso_n);
                        bNext.setTextColor(Color.rgb(255, 0, 0));

                        InputStream is = getResources().openRawResource(
                                    R.drawable.acceso_denegado);
                        Bitmap auxSource = BitmapFactory.decodeStream(is);
                        ivPasos.setImageBitmap(auxSource);

                  Toast.makeText(getApplicationContext(),"VERIFICACIÓN DE "
+ UserID + " DENEGADA", Toast.LENGTH_LONG).show();
                  }

                  bNext.setText("Finalizar");
                  finalizar = 1;

                  break;

            default:

                  finalizar = 0;
                  finish();
                  break;
            }

      }

      public void Preprocesado() {
            // a) Reducir imagen a tamaño fijo (ancho más o menos en 150
pixeles) y
            // quitar fondos
            // Se calcula el valor de reducción
            ****Función Protegida****
            // b) Histograma
            ****Función Protegida****

            // c) Busqueda Pupila
            ****Función Protegida****

            // e) Ubicar borde externo
            // Si respuesta es 0 entonces se ha hecho con exito sino no.
            int respuesta = ****Función Protegida****;
            // Si respuesta == 0 todo OK
            if (respuesta == 0) {
                  // f) si todo ha ido bien, buscar los párpados
                  ****Función Protegida****
            }

      }

      private byte[] setArrayErrores() {
            byte[] Array = new byte[12];
            Array[0] = (byte) error_Sharpness;
            Array[1] = (byte) error_Image_Scale_H_Left;
            Array[2] = (byte) error_Image_Scale_H_Right;
            Array[3] = (byte) error_Image_Scale_V_Up;
            Array[4] = (byte) error_Image_Scale_V_Down;
            Array[5] = (byte) error_Contrast_Pupila;
            Array[6] = (byte) error_Contrast_Iris;
            Array[7] = (byte) error_Gray_Scale_Density;
```

```
		Array[8] = (byte) error_Pupil_Iris_Ratio;
		Array[9] = (byte) error_Iris_Size;
		Array[10] = (byte) error_Eccentric;
		Array[11] = (byte) error_Fraude;
		return Array;
	}

	public Bitmap Array_To_Bitmap_GettingOneByOne(byte[] data, int width,
			int height) {
// Se crea un bitmap mutable y se introduce los datos pixel a pixel.
		Bitmap bmp = Bitmap.createBitmap(width, height,
Bitmap.Config.RGB_565);
		int R, pixel;
		for (int y = 0; y < height; y++)
			for (int x = 0; x < width; x++) {
				pixel = data[x + width * y];
				R = pixel & 0xFF;
				bmp.setPixel(x, y, Color.rgb(R, R, R));
			}
		return bmp;
	}

	public Bitmap ConvertToGrayscale_Preciso(Bitmap bmSource) {
		// Dos metodos de paso a escala de grises:

		// PRIMER MÉTODO, lento de alta calidad.

		Bitmap bmAux = Bitmap.createBitmap(bmSource.getWidth(),
				bmSource.getHeight(), Bitmap.Config.RGB_565);
		int c, R, G, B, luma;
		for (int y = 0; y < bmAux.getHeight(); y++) {
			for (int x = 0; x < bmAux.getWidth(); x++) {
				// Se recoje el color de la dirección
				c = bmSource.getPixel(x, y);
				R = Color.red(c);
				G = Color.green(c);
				B = Color.blue(c);
				luma = (int) (R * 0.3 + G * 0.59 + B * 0.11);
				bmAux.setPixel(x, y, Color.rgb(luma, luma, luma));
			}
		}
		return bmAux;
	}

	int CalidadPost() {
		// Si error = 0 todo OK
		// Si error_... = 1 hay error

		if (Sharpness() == 1) {
			error_Sharpness = 1;
		} else {
			error_Sharpness = 0;
		}
		if (Image_Scale() == 2) {
			error_Image_Scale_H_Left = 1;
		} else {
			error_Image_Scale_H_Left = 0;
		}
		if (Image_Scale() == 3) {
```

```
                error_Image_Scale_H_Right = 1;
        } else {
                error_Image_Scale_H_Right = 0;
        }
        if (Image_Scale() == 4) {
                error_Image_Scale_V_Up = 1;
        } else {
                error_Image_Scale_V_Up = 0;
        }
        if (Image_Scale() == 5) {
                error_Image_Scale_V_Down = 1;
        } else {
                error_Image_Scale_V_Down = 0;
        }
        if (Contrast() == 6) {
                error_Contrast_Pupila = 1;
        } else {
                error_Contrast_Pupila = 0;
        }
        if (Contrast() == 7) {
                error_Contrast_Iris = 1;
        } else {
                error_Contrast_Iris = 0;
        }
        if (Gray_Scale_Density() == 8) {
                error_Gray_Scale_Density = 1;
        } else {
                error_Gray_Scale_Density = 0;
        }
        if (Pupil_Iris_Ratio() == 10) {
                error_Pupil_Iris_Ratio = 1;
        } else {
                error_Pupil_Iris_Ratio = 0;
        }
        if (Iris_Size() != 0) {
                error_Iris_Size = 1;
        } else {
                error_Iris_Size = 0;
        }
        if (Eccentric() != 0) {
                error_Eccentric = 1;
        } else {
                error_Eccentric = 0;
        }
        contError = error_Sharpness + error_Image_Scale_H_Left
+ error_Image_Scale_H_Right + error_Image_Scale_V_Up
+ error_Image_Scale_V_Down + error_Contrast_Pupila
+ error_Contrast_Iris + error_Gray_Scale_Density
+ error_Pupil_Iris_Ratio + error_Iris_Size + error_Eccentric;
        if (contError == 0) {
                return 0;
        } else {
                return 1;
        }
}
```

verificpasos.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="100" >

        <TextView
            android:id="@+id/tvChrono"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="left"
            android:layout_weight="99.26"
            android:text="Tiempo de procesado"
            android:textSize="20dp"
             android:textColor="#0000FF"/>


        <Button
            android:id="@+id/bHelp"
            android:layout_width="wrap_content"
            android:layout_height="30dp"
            android:background="@drawable/actionhelpwhite"
            android:gravity="right" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="513dp"
        android:orientation="vertical"
        android:paddingBottom="5dp" >

        <TextView
            android:id="@+id/tvPason"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:paddingBottom="8dp"
            android:text="Paso 1"
            android:textSize="30dp" />

        <TextView
            android:id="@+id/tvDescrip"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:paddingLeft="10dp"
            android:paddingRight="10dp"
            android:text="Descripción proceso"
            android:textSize="20dp" />
```

```
        <ImageView
            android:id="@+id/ivOjoPasos"
            android:layout_width="250dp"
            android:layout_height="250dp"
            android:layout_gravity="center"
            android:layout_marginBottom="20dp"
            android:layout_marginTop="20dp"
            android:src="@drawable/android512" />

        <Button
            android:id="@+id/bNext"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="17dp"
            android:text="Siguiente" />
    </LinearLayout>

</LinearLayout>
```

Verificación directa

Por confidencialidad no se permite incluir todo el código de esta actividad. Esto es debido a que está basado en un código que se encuentra bajo protección industrial de la universidad. En los campos correspondientes se ha procedido ha su entera eliminación, y en las declaraciones de las mismas han sido sustituidas por el indicativo "****Función Protegida****". Se han eliminado las funciones traducidas he implementadas del proyecto IrisBiometrics, variables globales que usa dicha biblioteca y sus declaraciones. Todo ellos para la protección del concepto de estos datos.

VerificDirecta.java

```
public class VerificDirecta extends Activity implements OnClickListener {

      /*** ---CONSTANES DE CÓDIGO--- ***/
      int error_Sharpness, error_Image_Scale_H_Left,
error_Image_Scale_H_Right, error_Image_Scale_V_Up, error_Image_Scale_V_Down,
error_Contrast_Pupila, error_Contrast_Iris, error_Gray_Scale_Density,
error_Pupil_Iris_Ratio, error_Iris_Size, error_Eccentric, error_Fraude = 0;

      int contError = 0;
      int ComprobError = 0;
      byte[] ArrayErrores = null;

      // Las direcciones donde se almacenan los datos de caract. del sujeto
      String FilePath_Code_UserID;
      String FilePath_Mask_UserID;

      long startTime, finishTime;
      long memocronometro;
```

```
	/*** ---VARIABLES DE PRESENTACIÓN Y BOTONES--- ***/
	Bitmap foto = null, bmOjo = null, bmOjoContorno = null;
	ImageView iv;
	Button bHelp, bComenzar;
	TextView tvAcceso, tvDescrip, tvChrono, tvPorcent;

	// Variables de pasos
	String tvPason_paso_n;
	String tvDescrip_paso_n;
	int cont = 1;
	int finalizar = 0;
	byte[] arrayOjo, arrayOjoaux;
	byte[] bmArrayOjo = null;
	String UserID;
	Bitmap bmCannyView = null;
	Bitmap dibuCartes = null;
	int ERROR = 0;
	// Para la extracción de características de la imagen a verificar
	byte[] Caract_Code_Verif, Caract_Mask_Verif;
	// Para la obtención de caracteríticas de la imagen del User en
memoria
	byte[] Caract_Code_UserID, Caract_Mask_UserID;
	// Comparación por Hamming
	float Dist_Hamming;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.verificdirecta);
		****Función Protegida****
		initialize_android();
		Load_image();
		getdataId();
		Caract_Code_UserID =
get_Caract_Code_UserID(FilePath_Code_UserID);
		Caract_Mask_UserID =
get_Caract_Mask_UserID(FilePath_Mask_UserID);

		iv.setImageBitmap(bmOjo);

	}

	private byte[] get_Caract_Mask_UserID(String filePath_Mask_UserID2) {
		// Read y se guarda en Caract_Mask_UserID;
		byte[] dataget = null;
		try {
		BufferedReader input = new BufferedReader(new InputStreamReader(
openFileInput(filePath_Mask_UserID2)));
			dataget = input.readLine().getBytes();
			input.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
		return dataget;
```

```
	}

	private byte[] get_Caract_Code_UserID(String filePath_Code_UserID2) {
		// Read y se guarda en Caract_Code_UserID;
		byte[] dataget = null;
		try {
			BufferedReader input = new BufferedReader(new
InputStreamReader(
					openFileInput(filePath_Code_UserID2)));
			dataget = input.readLine().getBytes();
			input.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
		return dataget;
	}

	private void Load_image() {
		Bundle gotBasket = getIntent().getExtras();
		arrayOjo = gotBasket.getByteArray("ArrayOjo");
		bmOjo = Array_To_Bitmap(arrayOjo);
		bmOjoContorno = bmOjo;
	}

	private void getdataId() {
		// String ItemID; "UserID"
		Bundle gotString = getIntent().getExtras();
		UserID = gotString.getString("UserID");

		// Saco las direcciones de caract. de ese sujeto
		FilePath_Code_UserID = UserID + "_code";
		FilePath_Mask_UserID = UserID + "_mask";
	}

	private Bitmap Array_To_Bitmap(byte[] arraySource) {
		Bitmap bmSource = BitmapFactory.decodeByteArray(arraySource, 0,
				arraySource.length);

		// Paso el Bitmap creado (Imutable) a mutable
		Bitmap bmSourceMutable = bmSource.copy(Bitmap.Config.RGB_565,
true);

		return bmSourceMutable;
	}

	private void initialize_android() {

		iv = (ImageView) findViewById(R.id.iv);
		bHelp = (Button) findViewById(R.id.bHelp);
		tvAcceso = (TextView) findViewById(R.id.tvAcceso);
		bComenzar = (Button) findViewById(R.id.bComenzar);
		tvChrono = (TextView) findViewById(R.id.tvChrono);
		tvPorcent = (TextView) findViewById(R.id.tvPorcent);
		bHelp.setOnClickListener(this);
		bComenzar.setOnClickListener(this);
	}

	public void onClick(View pressed) {
		switch (pressed.getId()) {
```

```
		case R.id.bHelp:
			Intent help = new Intent(this, Help.class);
			String Actividad = "VerificDirecta";
			help.putExtra("ActivityFrom", Actividad);
			startActivity(help);
			break;

		case R.id.bComenzar:
			if (ERROR == 0){
			if (finalizar == 0) {
				MegaProceso();
			} else {
				finish();
			}
			} else {
				Intent logError = new Intent(this, LogError.class);
				logError.putExtra("contError", contError);

				ArrayErrores = setArrayErrores();
				logError.putExtra("ArrayErrores", ArrayErrores);
				startActivity(logError);
				finish();
			}
			break;
		}
	}

	// Se hace todos los procesos de golpe y sin imagenes ni nada, se
calcula
	// tambien el tiempo total
	private void MegaProceso() {
		// Empieza a contar el reloj
		startTime = System.nanoTime();

		// ** COMIENZAN LOS PROCESOS **
		// Pasa a escala de grises y se muestra por pantalla
		bmOjo = ****Función Protegida****;

		// Cargar datos en los Structs. Iris_Open
		****Función Protegida****

		****Función Protegida****

		// Comprobación de errores en la captura de la
		// imagen
		ComprobError = CalidadPost();
		// Cuando ComprobError == 1 hay errores
		if (ComprobError == 1) {
			// Hay errores y no se puede continuar
			// Se abre el LogError
			if (contError == 1) {
		Toast.makeText(getApplicationContext(), "Hay un error de
lectura, no se puede continuar el proceso", Toast.LENGTH_LONG).show();
			} else {
		Toast.makeText(getApplicationContext(),"Hay un total de "
+ contError + " errores de lectura, no se puede continuar el proceso",
Toast.LENGTH_LONG).show();
			}
			bComenzar.setText("ERROR");
```

```
                        bComenzar.setTextColor(Color.RED);
                        ERROR = 1;
                } else {

                        ****Función Protegida****

                        Caract_Mask_Verif = ****Función Protegida****
                        Caract_Code_Verif = ****Función Protegida****

                        Dist_Hamming = ****Función Protegida****;
                        finishTime = System.nanoTime() - startTime;
                        // Lo paso a milisegundos ya que es muy grande el valor
                        finishTime = (long) (finishTime * 0.000001);
                        tvChrono.setText("Tiempo TOTAL: " + finishTime + "ms");

                        float Porc_acierto = 100 - Dist_Hamming * 100;
                        // Limito el numero de decimales a mostrar
                        String Porc = String.format("%.5f", Porc_acierto);
                        if (Dist_Hamming < 0.25) {
                                tvDescrip_paso_n = "La persona ha sido identificada
                        como la correcta en un " + Porc + "% de acierto";
                                tvPorcent.setText(tvDescrip_paso_n);
                                bComenzar.setTextColor(Color.rgb(0, 255, 0));

                                InputStream is = getResources().openRawResource(
                                                R.drawable.acceso_concedido);
                                Bitmap auxSource = BitmapFactory.decodeStream(is);
                                iv.setImageBitmap(auxSource);

                                Toast.makeText(getApplicationContext(),
                                        "VERIFICACIÓN DE " + UserID + " ACEPTADA",
                                                Toast.LENGTH_LONG).show();
                        } else {
                                tvDescrip_paso_n = "La persona no ha sido
identificada como la correcta en un " + Porc + "% de acierto";
                                tvPorcent.setText(tvDescrip_paso_n);
                                bComenzar.setTextColor(Color.rgb(255, 0, 0));

                                InputStream is = getResources().openRawResource(
                                                R.drawable.acceso_denegado);
                                Bitmap auxSource = BitmapFactory.decodeStream(is);
                                iv.setImageBitmap(auxSource);

                                Toast.makeText(getApplicationContext(),
                                                "VERIFICACIÓN DE " + UserID + "
DENEGADA",
                                                Toast.LENGTH_LONG).show();
                        }
                        bComenzar.setText("Finalizar");
                        finalizar = 1;
                }
        }

        public void Preprocesado() {
                // a) Reducir imagen a tamaño fijo (ancho más o menos en 150
pixeles) y
                // quitar fondos
                // Se calcula el valor de reducción
                ****Función Protegida****       }
```

```
			****Función Protegida****
			// b) Histograma
			****Función Protegida****

			// c) Busqueda Pupila
			****Función Protegida****

			// e) Ubicar borde externo
			// Si respuesta es 0 entonces se ha hecho con exito sino no.
			int respuesta = ****Función Protegida****;
			// Si respuesta == 0 todo OK
			if (respuesta == 0) {
				// f) si todo ha ido bien, buscar los párpados
				****Función Protegida****
			}

		}

		private byte[] setArrayErrores() {
			byte[] Array = new byte[12];
			Array[0] = (byte) error_Sharpness;
			Array[1] = (byte) error_Image_Scale_H_Left;
			Array[2] = (byte) error_Image_Scale_H_Right;
			Array[3] = (byte) error_Image_Scale_V_Up;
			Array[4] = (byte) error_Image_Scale_V_Down;
			Array[5] = (byte) error_Contrast_Pupila;
			Array[6] = (byte) error_Contrast_Iris;
			Array[7] = (byte) error_Gray_Scale_Density;
			Array[8] = (byte) error_Pupil_Iris_Ratio;
			Array[9] = (byte) error_Iris_Size;
			Array[10] = (byte) error_Eccentric;
			Array[11] = (byte) error_Fraude;
			return Array;
		}

		public Bitmap Array_To_Bitmap_GettingOneByOne(byte[] data, int width,
				int height) {
			// Se crea un bitmap mutable y se introduce los datos pixel a
pixel.
			Bitmap bmp = Bitmap.createBitmap(width, height,
Bitmap.Config.RGB_565);
			int R, pixel;
			for (int y = 0; y < height; y++)
				for (int x = 0; x < width; x++) {
					pixel = data[x + width * y];
					R = pixel & 0xFF;
					bmp.setPixel(x, y, Color.rgb(R, R, R));
				}
			return bmp;
		}

		public Bitmap ConvertToGrayscale_Preciso(Bitmap bmSource) {
			// Dos metodos de paso a escala de grises:

			// PRIMER MÉTODO, lento de alta calidad.

			Bitmap bmAux = Bitmap.createBitmap(bmSource.getWidth(),
					bmSource.getHeight(), Bitmap.Config.RGB_565);
			int c, R, G, B, luma;
```

```
		for (int y = 0; y < bmAux.getHeight(); y++) {
			for (int x = 0; x < bmAux.getWidth(); x++) {

				// Se recoje el color de la dirección

				c = bmSource.getPixel(x, y);
				R = Color.red(c);
				G = Color.green(c);
				B = Color.blue(c);
				luma = (int) (R * 0.3 + G * 0.59 + B * 0.11);
				bmAux.setPixel(x, y, Color.rgb(luma, luma, luma));
			}
		}
		return bmAux;
	}

	int CalidadPost() {
		// Si error = 0 todo OK
		// Si error_... = 1 hay error

		if (Sharpness() == 1) {
			error_Sharpness = 1;
		} else {
			error_Sharpness = 0;
		}
		if (Image_Scale() == 2) {
			error_Image_Scale_H_Left = 1;
		} else {
			error_Image_Scale_H_Left = 0;
		}
		if (Image_Scale() == 3) {
			error_Image_Scale_H_Right = 1;
		} else {
			error_Image_Scale_H_Right = 0;
		}
		if (Image_Scale() == 4) {
			error_Image_Scale_V_Up = 1;
		} else {
			error_Image_Scale_V_Up = 0;
		}
		if (Image_Scale() == 5) {
			error_Image_Scale_V_Down = 1;
		} else {
			error_Image_Scale_V_Down = 0;
		}
		if (Contrast() == 6) {
			error_Contrast_Pupila = 1;
		} else {
			error_Contrast_Pupila = 0;
		}
		if (Contrast() == 7) {
			error_Contrast_Iris = 1;
		} else {
			error_Contrast_Iris = 0;
		}
		if (Gray_Scale_Density() == 8) {
			error_Gray_Scale_Density = 1;
		} else {
			error_Gray_Scale_Density = 0;
```

```
            }
            if (Pupil_Iris_Ratio() == 10) {
                  error_Pupil_Iris_Ratio = 1;
            } else {
                  error_Pupil_Iris_Ratio = 0;
            }
            if (Iris_Size() != 0) {
                  error_Iris_Size = 1;
            } else {
                  error_Iris_Size = 0;
            }
            if (Eccentric() != 0) {
                  error_Eccentric = 1;
            } else {
                  error_Eccentric = 0;
            }
            // if (Fraude_FFT() == 12) {
            // error_Fraude = 1;
            // } else {
            // error_Fraude = 0;
            // }

      contError = error_Sharpness + error_Image_Scale_H_Left +
error_Image_Scale_H_Right + error_Image_Scale_V_Up + error_Image_Scale_V_Down
+ error_Contrast_Pupila + error_Contrast_Iris + error_Gray_Scale_Density
+ error_Pupil_Iris_Ratio + error_Iris_Size + error_Eccentric;
            if (contError == 0) {
                  return 0;
            } else {
                  return 1;
            }
            }
```

verificdirecta.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_menu"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="100" >

        <TextView
            android:id="@+id/tvChrono"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="left"
            android:layout_weight="99.26"
            android:text="Tiempo TOTAL"
            android:textSize="20dp"
```

```
            android:textColor="#0000FF"/>


        <Button
            android:id="@+id/bHelp"
            android:layout_width="wrap_content"
            android:layout_height="30dp"
            android:background="@drawable/actionhelpwhite"
            android:gravity="right" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="513dp"
        android:orientation="vertical"
        android:paddingBottom="5dp" >

        <TextView
            android:id="@+id/tvAcceso"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:paddingBottom="8dp"
            android:text="Comprob. Directa"
            android:textColor="#00DF00"
            android:textSize="30dp" />

        <TextView
            android:id="@+id/tvPorcent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:paddingLeft="10dp"
            android:paddingRight="10dp"
            android:text="Se desarrollarán todos los procesos en continuidad
y lo más rápido posible"
            android:textSize="20dp"
            android:textColor="#FFDF00" />

        <ImageView
            android:id="@+id/iv"
            android:layout_width="250dp"
            android:layout_height="250dp"
            android:layout_gravity="center"
            android:layout_marginBottom="20dp"
            android:layout_marginTop="20dp"
            android:src="@drawable/android512" />

        <Button
            android:id="@+id/bComenzar"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="17dp"
            android:text="Comenzar" />
    </LinearLayout>

      </LinearLayout>
```

Preferencias

Prefs.java

```
public class Prefs extends PreferenceActivity{

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		//método como el setContentView pero con datos
		addPreferencesFromResource(R.xml.prefs);
	}
	}
```

prefs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <EditTextPreference
        android:key="name"
        android:summary="Enter Your Name"
        android:title="EditText" >
    </EditTextPreference>

    <CheckBoxPreference
        android:defaultValue="true"
        android:key="checkbox"
        android:summary="for the splash screen"
        android:title="Music" >
    </CheckBoxPreference>

    <ListPreference
        android:entries="@array/list"
        android:entryValues="@array/lValues"
        android:key="list"
        android:summary="This is a list to choose from"
        android:title="list" >
    </ListPreference>

      </PreferenceScreen>
```

Registro de error

LogError.java

```
public class LogError extends Activity {

	byte[] ArrayErrores = null;
	int contError = 0;
	int error_Sharpness, error_Image_Scale_H_Left,
error_Image_Scale_H_Right,
	error_Image_Scale_V_Up, error_Image_Scale_V_Down,
	error_Contrast_Pupila, error_Contrast_Iris,
	error_Gray_Scale_Density, error_Pupil_Iris_Ratio, error_Iris_Size,
	error_Eccentric, error_Fraude = 0;

	/*** ---VARIABLES DE PRESENTACIÓN Y BOTONES--- ***/

	TextView tvNumFallos, tvErrSharp, tvErrScalHL, tvErrScalHR,
tvErrScalVU,
			tvErrScalVD, tvErrContrPup, tvErrContrIris,
tvErrGrayScale,
			tvErrIrisSize, tvErrRatio, tvErrDistCentros, tvErrFraude;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);

		setContentView(R.layout.logerror);

		initialize_android();
		getdataError();
		DecodificarArray();
		GestionarView();
	}

	private void DecodificarArray() {
		error_Sharpness = (int) ArrayErrores[0];
		error_Image_Scale_H_Left = (int) ArrayErrores[1];
		error_Image_Scale_H_Right = (int) ArrayErrores[2];
		error_Image_Scale_V_Up = (int) ArrayErrores[3];
		error_Image_Scale_V_Down = (int) ArrayErrores[4];
		error_Contrast_Pupila = (int) ArrayErrores[5];
		error_Contrast_Iris = (int) ArrayErrores[6];
		error_Gray_Scale_Density = (int) ArrayErrores[7];
		error_Pupil_Iris_Ratio = (int) ArrayErrores[8];
		error_Iris_Size = (int) ArrayErrores[9];
		error_Eccentric = (int) ArrayErrores[10];
		error_Fraude = (int) ArrayErrores[11];
	}

	private void GestionarView() {
		tvNumFallos.setText("Numero de fallos: " + contError);
		if (error_Sharpness == 1){
```

```
                    tvErrSharp.setText("Calidad
sharpness.............................FAIL");
                    tvErrSharp.setTextColor(Color.RED);
              } else{

              }
              if (error_Image_Scale_H_Left == 1){
                    tvErrScalHL.setText("Calidad escala imagen horiz.
izq.............FAIL");
                    tvErrScalHL.setTextColor(Color.RED);
              } else{

              }
              if (error_Image_Scale_H_Right == 1){
                    tvErrScalHR.setText("Calidad escala imagen horiz.
derch...FAIL");
                    tvErrScalHR.setTextColor(Color.RED);
              } else{

              }
              if (error_Image_Scale_V_Up == 1){
                    tvErrScalVU.setText("Calidad escala imagen vert.
arrib.......FAIL");
                    tvErrScalVU.setTextColor(Color.RED);
              } else{

              }
              if (error_Image_Scale_V_Down == 1){
                    tvErrScalVD.setText("Calidad escala imagen vert.
abj..........FAIL");
                    tvErrScalVD.setTextColor(Color.RED);
              } else{

              }
              if (error_Contrast_Pupila == 1){
                    tvErrContrPup.setText("Calidad contraste
pupila.....................FAIL");
                    tvErrContrPup.setTextColor(Color.RED);
              } else{

              }
              if (error_Contrast_Iris == 1){
                    tvErrContrIris.setText("Calidad contraste
iris.........................FAIL");
                    tvErrContrIris.setTextColor(Color.RED);
              } else{

              }
              if (error_Gray_Scale_Density == 1){
                    tvErrGrayScale.setText("Calidad escala de
grises.....................FAIL");
                    tvErrGrayScale.setTextColor(Color.RED);
              } else{

              }
              if (error_Pupil_Iris_Ratio == 1){
                    tvErrRatio.setText("Calidad ratio entre pupila e
iris..........FAIL");
                    tvErrRatio.setTextColor(Color.RED);
```

```
            } else{

            }
            if (error_Iris_Size == 1){
                  tvErrIrisSize.setText("Calidad tamaño
iris.............................FAIL");
                  tvErrIrisSize.setTextColor(Color.RED);
            } else{

            }
            if (error_Eccentric == 1){
                  tvErrDistCentros.setText("Calidad distancia de
centros..............FAIL");
                  tvErrDistCentros.setTextColor(Color.RED);
            } else{

            }
            if (error_Fraude == 1){
                  tvErrFraude.setText("Comprobación contra
fraude.............FAIL");
                  tvErrFraude.setTextColor(Color.RED);
            } else{

            }
      }

      private void getdataError() {
            Bundle gotBasket = getIntent().getExtras();
            ArrayErrores = gotBasket.getByteArray("ArrayErrores");
            contError = gotBasket.getInt("contError");
      }

      private void initialize_android() {
            tvNumFallos = (TextView) findViewById(R.id.tvNumFallos);
            tvErrSharp = (TextView) findViewById(R.id.tvErrSharp);
            tvErrScalHL = (TextView) findViewById(R.id.tvErrScalHL);
            tvErrScalHR = (TextView) findViewById(R.id.tvErrScalHR);
            tvErrScalVU = (TextView) findViewById(R.id.tvErrScalVU);
            tvErrScalVD = (TextView) findViewById(R.id.tvErrScalVD);
            tvErrContrPup = (TextView) findViewById(R.id.tvErrContrPup);
            tvErrContrIris = (TextView) findViewById(R.id.tvErrContrIris);
            tvErrGrayScale = (TextView) findViewById(R.id.tvErrGrayScale);
            tvErrIrisSize = (TextView) findViewById(R.id.tvErrIrisSize);
            tvErrRatio = (TextView) findViewById(R.id.tvErrRatio);
            tvErrDistCentros = (TextView)
findViewById(R.id.tvErrDistCentros);
            tvErrFraude = (TextView) findViewById(R.id.tvErrFraude);  } }
```

logerror.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/tvNumFallos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="25dp"
        android:textColor="#FF0000"
        android:text="Numero de fallos: "
        android:layout_gravity="center"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrSharp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15dp"
        android:textColor="#00FF00"
        android:text="Calidad sharpness.............................PASS"
        android:paddingTop="15dp" />

    <TextView
        android:id="@+id/tvErrScalHL"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:text="Calidad escala imagen horiz. izq.............PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrScalHR"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad escala imagen horiz. derch...PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrScalVU"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad escala imagen vert. arrib.......PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrScalVD"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
```

```
        android:text="Calidad escala imagen vert. abj.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrContrPup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad contraste pupila.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrContrIris"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad contraste iris.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrGrayScale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad escala de grises.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrIrisSize"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad tamaño iris.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrRatio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad ratio entre pupila e iris.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrDistCentros"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Calidad distancia de centros.....PASS"
        android:paddingTop="12dp" />

    <TextView
        android:id="@+id/tvErrFraude"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#00FF00"
        android:textSize="15dp"
        android:text="Comprobación contra fraude.............PASS"
        android:paddingTop="12dp" />

    </LinearLayout>
```

Registro de error

Help.java

```
public class Help extends Activity{

    String Activity;
    TextView tvHelp;
    int contPaso = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.help);

        tvHelp = (TextView) findViewById(R.id.tvHelp);
        getActivity();

        SetHelp(Activity);
    }

    private void SetHelp(String activitys) {
        if (activitys.equals("VerificPasos") == true) {

            switch (contPaso) {
            case 1:
                tvHelp.setText("Esta clase es la de VerifPasos 1");
                break;

            case 2:
                tvHelp.setText("Esta clase es la de VerifPasos 2");
                break;

            case 3:
                tvHelp.setText("Esta clase es la de VerifPasos 3");
                break;

            case 4:
                tvHelp.setText("Esta clase es la de VerifPasos 4");
                break;

            case 5:
                tvHelp.setText("Esta clase es la de VerifPasos 5");
                break;

            case 6:
```

```
                        tvHelp.setText("Esta clase es la de VerifPasos 6");
                        break;

                  case 7:
                        tvHelp.setText("Esta clase es la de VerifPasos 7");
                        break;

                  case 8:
                        tvHelp.setText("Esta clase es la de VerifPasos 8");
                        break;

                  case 9:
                        tvHelp.setText("Esta clase es la de VerifPasos 9");
                        break;

                  case 10:
                        tvHelp.setText("Esta clase es la de VerifPasos
10");
                        break;
                  }


            } else if (activitys.equals("VerificDirecta") == true) {

                  tvHelp.setText("Esta clase es la de VerifDirecta");

            } else if (activitys.equals("MenuReclut") == true) {

                  tvHelp.setText("Esta clase es la de MenuReclut");

            } else if (activitys.equals("Menu") == true) {

                  tvHelp.setText("Esta clase es la de Menu");

            } else if (activitys.equals("UsuarioData") == true) {

                  tvHelp.setText("Esta clase es la de UsuarioData");

            }
      }

      private void getActivity() {
            Bundle gotBasket = getIntent().getExtras();
            Activity = gotBasket.getString("ActivityFrom");
            contPaso = gotBasket.getInt("contPaso");
      }
      }
```

help.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/tvHelp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textSize="20dp" />

      </LinearLayout>
```

Comentarios del creador

AboutUs.java

```
public class AboutUs extends Activity{

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.about);
      }

      }
```

about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Proyecto presentado para la asignatura de Trabajo de
Fin de Grado de la universidad Carlos III" />

      </LinearLayout>
```