Universidad Carlos III de Madrid
www.uc3m.es

Trabajo Fin de Grado:

INTERFAZ DE PROGRAMACIÓN DE SOLUCIONES BIOMÉTRICAS BAJO BIOAPI EN PLATAFORMAS ANDROID

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA.

AUTOR: Jaime Checa Rodríguez

TUTOR: Raúl Sánchez Reíllo

DIRECTOR: Aitor Mendaza Ormaza

Leganés, 5 de Septiembre de 201

AGRADECIMIENTOS

La finalización del trabajo fin de grado pone el punto final al trabajo duro a lo largo de cuatro largos años y supone la consecución de un objetivo marcado hace muchos tiempo.

Llegado este momento puedo afirmar que estoy muy orgulloso con los resultados obtenidos y satisfecho con la educación recibida tanto personal como académica.

He de agradecer todo el apoyo recibido por parte de mis padres, Baldomero y Gloria, que mediante sus consejos me han enseñado el significado del sacrificio y el trabajo duro.

También agradecer a mi hermana Almudena y a mi hermano Javier, por ser un ejemplo para mí, su apoyo incondicional y sus consejos.

Mención especial para todos los compañeros que me han acompañado a lo largo de este viaje.

Y, por supuesto, gracias a mi novia Lidia, por aguantarme todos estos duros años con su apoyo constante, espero haberte aportado tanto como tú lo has hecho conmigo

RESUMEN

La Identificación Biométrica es cada vez más atractiva para los desarrolladores de sistemas de seguridad debido a las limitaciones existentes en los sistemas actuales de identificación de personas mediante contraseñas y/o tarjetas. Si a este dato le unimos el importante incremento de Smartphones en el mercado actual, los cuales nos proporcionan una gran cantidad de nuevas funcionalidades, obtenemos una mezcla muy interesante en un campo que aporta una gran facilidad de desarrollo. A este incremento de Smartphones ha aportado mucho el sistema operativo Android que se ha impuesto como el principal, en cuanto a cuota de mercado, de la presente generación de teléfonos móviles. En este trabajo se ha desarrollado una aplicación que permite al usuario almacenar una serie de contraseñas en una base de datos, solicitando al usuario la introducción de un patrón de desbloqueo para consultarlas. El propósito de este patrón de acceso es sustitutivo a la colocación de un sensor biométrico para desbloquear el acceso y habilitar la posibilidad futura de que el fabricante incorpore esta utilidad en sus nuevos dispositivos. En este documento se dará a conocer una breve historia de la telefonía móvil y se describirá el proceso de diseño y desarrollo de la aplicación mencionada. El desarrollo comprende, no sólo la aplicación que se instalará y ejecutará en el teléfono móvil, sino también la adaptación del futuro estándar BioAPI 3.0 en lenguaje Java y para plataforma Android y como éste funciona.

ABSTRACT

Biometric Identification is becoming more interesting to systems security developers due to the existing limitations of current identification systems which use passwords and/or cards. If we add to this fact the significant increase of smartphones on the current market, which provide us many new features, we get an interesting mix in a field that provides ease of development. To this smartphones usage increase has contributed the operating system Android which has emerged as the leading one, in terms of market share, of the current generation of mobile phones. In this work has been developed an application which allows the user to store a bunch of passwords in a database safely, requesting the user to enter an unlock pattern for check them. The main purpose of this unlock pattern is a substitution to the use of any biometric sensor for unlock access and bring mobile phone manufacturers the possibility to incorporate this utility in new devices. This document presents brief history of mobile phones and describes the design process and application development mentioned. This development includes not only the application that runs on the mobile phone, but also the adaptation of the next version of the BioAPI 3.0 standard in its Java version and for the Android platform and how it works.

Índice de contenido

Índice de figuras

Índice de tablas

Índice de acrónimos

AMPS Advanced Mobile Phone System (Sistema de telefonía móvil avanzado)

API Application Programming Interface (Interfaz de programación de aplicaciones)

BFP BioAPI Function Provider (proveedor de funciones BioAPI)

BSP Biometric Service Provider (proveedor de servicios biométricos)

CDMA Code Division Multiple Access (Acceso múltiple por división de código)

EDGE Enhanced Data rates for GSM Evolution

EMS Enhanced Messaging System (Sistema mejorado de mensajería)

FMR False Match Rate (Relación de autenticaciones falsas)

FPI Function Provider Interface (Interfaz proveedora de funciones)

GPRS General Packet Radio Service

GSM Global System for Mobile communications

GUI Graphical User Interface (Interfaz gráfica de usuario)

LTE Long Term Evolution (Evolución a largo plazo)

MMS Multimedia Messaging System (Sistema de mensajería multimedia)

NMT Nordic Mobile Telephone (Telefono móvil nórdico)

SPI Service Provider Interface (Interfaz proveedora de servicios)

TACS Total Access Comunication System (Sistema de comunicación de acceso total)

TDMA Time Division Multiple Acces (Acceso múltiple por división de tiempo)

UMTS Universal Mobile Telecommunications System

1. Introducción

A lo largo de esta memoria se va a presentar un Trabajo Fin de Grado (TFG) en el cuál se desarrolla una aplicación cuyo objetivo es el almacenamiento de datos y contraseñas y consulta de los mismos de una manera segura mediante la adaptación de la normativa BioAPI Java 3.0 a través de la plataforma para móviles Android.

El funcionamiento de la aplicación móvil se basa en el almacenamiento en una base de datos de contraseñas de un usuario, para consultarlas mediante la introducción previa de un patrón de desbloqueo solo conocido por el usuario del dispositivo. Esta funcionalidad es un sustituto de la identificación biométrica, pues no existe en el mercado un Smartphone que incorpore o adapte un sensor biométrico (salvo la cámara de fotos que puede utilizarse para realizar un reconocimiento facial), pero recoge las bases para que en un futuro próximo sea posible esta posibilidad.

1.1. Motivación

La motivación principal es el aprendizaje de una plataforma con muchas posibilidades de desarrollo, de código abierto y que posee un gran crecimiento en su sector, lo que significa que ha logrado una gran aceptación entre programadores tanto expertos como principiantes. El aprendizaje de esta nueva plataforma aportará al alumno unos conocimientos que podrán servirle de gran utilidad en su carrera profesional. La elección de desarrollar este tipo de aplicación para un dispositivo móvil viene dado por el gran protagonismo que poseen los Smartphones y Tablets en la actualidad y la necesidad de adaptar a estos dispositivos las posibilidades y mejoras que aporta un campo como es el de la biometría.

Esto ha sido posible gracias a los grandes avances tecnológicos que se han dado en el último lustro, en los que se ha conseguido que los móviles actuales superen en capacidades a los ordenadores de principio del siglo XXI. Gracias a la incorporación de funcionalidades como la conectividad a Internet en movilidad, el GPS, Bluetooth entre otras y su reducido tamaño, estos dispositivos se están convirtiendo en un sustituto de los ordenadores portátiles.

Si a esto le añadimos la decisión de los fabricantes de publicar las APIs de programación para sus respectivos sistemas operativos, se pone en manos del público todas las funcionalidades de los terminales. De este modo los desarrolladores se aprovechan de un nuevo y creciente ámbito de trabajo, y los fabricantes consiguen un mayor número de ventas al contar cada vez con mejores aplicaciones.

Es importante diferenciar entre las aplicaciones para móviles y Tablets y las creadas para ordenador. Aquellas desarrolladas para dispositivos móviles poseen una

menor potencia debido a las limitaciones impuestas por los móviles al poseer un procesador menos potente, una batería con menor capacidad y una pantalla de tamaño reducido. En cuanto a las aplicaciones biométricas, en la actualidad existen pocos desarrollos para dispositivos móviles debido a la necesidad de incorporar un sensor biométrico, el cual no existe para Smartphones o Tablets en el mercado actual.

Hasta ahora ha habido algunos desarrollos que incorporan conceptos de la biometría, como es el ejemplo del desbloqueo por reconocimiento facial mediante la cámara frontal del teléfono, pero no aporta una gran fiabilidad.

Por ello mismo aparece la motivación principal, el cual es desarrollar una aplicación la cual funcione bajo la norma BioAPI, y en concreto su implementación en lenguaje Java (la cual está todavía en desarrollo) que facilita la programación orientada a objetos, y por tanto su adaptación a las nuevas tecnologías. Todo esto se hará adaptando la solución a los dispositivos portátiles para que en un futuro próximo pueda existir un Smartphone o Tablet con un sensor biométrico incorporado.

Con este trabajo se podrán ver las capacidades de uno de los sistemas operativos para móviles con más futuro, ya que cuenta con el respaldo de una empresa líder en nuevas tecnologías, como es Google, con multitud de proyectos en diferentes campos obteniendo gran éxito en todos ellos.

A través de este trabajo habrá que hacer frente a problemas reales similares a los que puede tener un ingeniero en cualquier proyecto a lo largo de su carrera profesional. Además permitirá aplicar los conocimientos y habilidades obtenidos durante los años de formación universitaria.

1.2. Objetivos

Los objetivos que se quieren alcanzar con este TFG se han planificado teniendo en cuenta:

- **Limitaciones de tiempo**. El Trabajo Fin de Grado es una asignatura de 12 créditos ECTS, lo que supone que la aplicación y la documentación de la misma, debe estar presupuestada en unas 300 horas de trabajo. Esto implica una duración de un cuatrimestre con una dedicación parcial de 20h/semana.

- **Las capacidades del alumno**. Se trata de un entorno de desarrollo totalmente nuevo, por lo que el trabajo cuenta con una complejidad adicional debida al aprendizaje de dicho entorno.

- **Requisitos de la aplicación**. Cuantos menores requisitos tenga, será más accesible, consiguiendo por tanto mayor penetración de mercado.

El desarrollo desde el inicio de aplicaciones de este tipo, aporta al alumno nuevos conocimientos y situaciones semejantes a las que podemos encontrar en el mundo profesional. Por lo tanto, el objetivo principal de un trabajo fin de grado debe ser el de aprender y preparar al alumno a la vida laboral.

Centrándonos en la aplicación a desarrollar, el objetivo es el de almacenar datos (contraseñas, números, etc.) de una forma segura y accesible solo por el propietario del dispositivo a través de la adaptación de la normativa bioAPI Java de la que se hablará más adelante. Para ello será necesario el desarrollo de una aplicación que solicite la introducción de un patrón de desbloqueo en lugar de una contraseña o similar, y la posterior programación de una base de datos donde se almacenen aquellos introducidos por el usuario en el dispositivo, para su posterior consulta. El usuario podrá consultar aquellos datos almacenados en la base, tras haber introducido correctamente el patrón de desbloqueo (también almacenado en otra base de datos) simulando de este modo como si se tratara de un desbloqueo biométrico (mediante huella dactilar, iris, etc.). Se quiere conseguir una aplicación con un funcionamiento robusto y sin errores y que tenga un alto grado de simplicidad.

1.3. Estructura

El siguiente documento está estructurado siguiendo un formato por el cual se introducirán las tecnologías abordadas en el TFG. Seguido de este punto se introducirá la plataforma de desarrollo elegida para la aplicación, donde se defenderá porque ha sido esta la elección frente a otras.

Una vez terminado de introducir, se procede a describir el diseño y el desarrollo de la solución para este TFG razonando las elecciones tomadas y explicando las dificultades encontradas a lo largo del mismo

Para terminar se demostrará y describirá el funcionamiento de la aplicación en un dispositivo real con imágenes tomadas al mismo en pleno funcionamiento y se concluirá con el apartado de conclusiones y líneas futuras.

Se espera que esta estructura ayude a comprender al lector el proceso que se ha llevado a cabo para realizar el trabajo.

2 Análisis del estado del arte

En este capítulo se lleva a cabo un breve resumen sobre la evolución de la telefonía móvil, una comparativa de las distintas plataformas móviles, un repaso a la historia de la biometría y una descripción de las distintas formas de identificación.

2.1 Evolución de la telefonía móvil

Los primeros sistemas de telefonía móvil civil empiezan a desarrollarse a principios de los años 80. Se popularizó el uso de sistemas FM gracias a su superior calidad de audio y resistencia a las interferencias. Los primeros equipos eran enormes y pesados, por lo que estaban destinados casi exclusivamente a su uso a bordo de vehículos. Generalmente se instalaba el equipo de radio en el maletero y se pasaba un cable con el teléfono hasta el salpicadero del coche.

La compañía Motorola lanzó al mercado el primer teléfono móvil comercial de la historia, el Motorola DynaTAC 8000X. Aquel teléfono pesaba 800 gramos, con unas dimensiones de 33 x 9 x 4.5 centímetros y gracias a su batería podíamos mantener una conversación de unos 30 minutos y de 8 horas en espera, previa carga de 10 horas. Por esa época, estos terminales estaban considerados como un lujo y estaban orientados principalmente al sector empresarial.

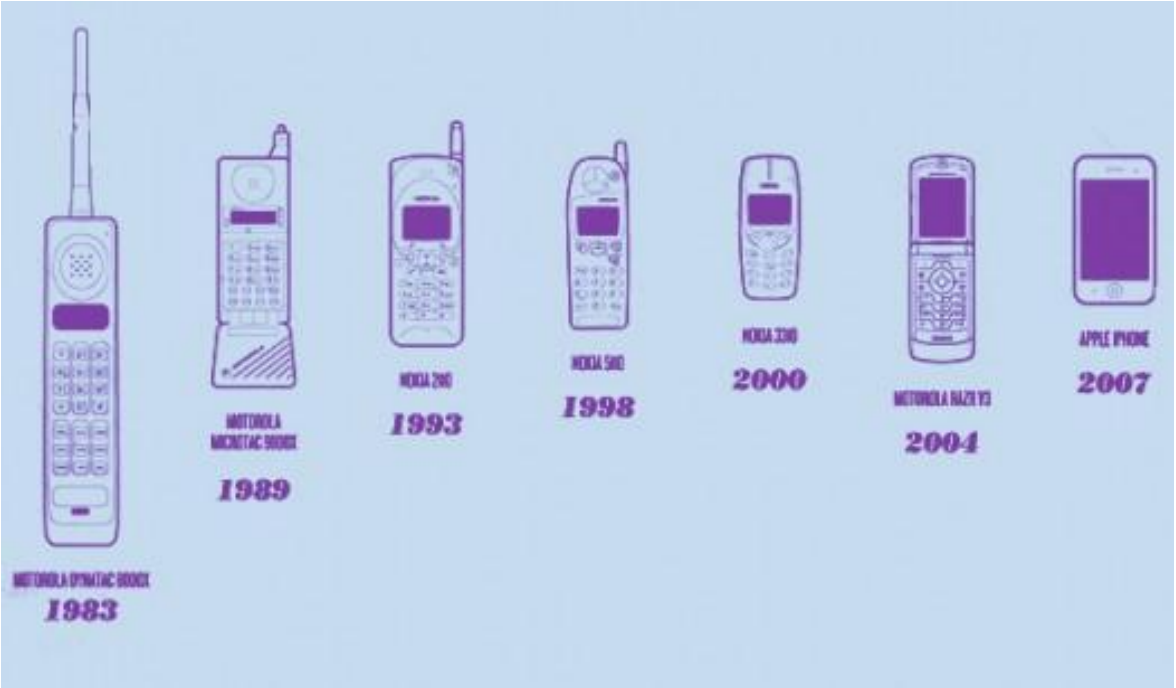

Figura 1: Evolución de los terminales móviles [1]

En 1986, Ericsson modernizó el sistema, llevándolo hasta el nivel NMT 900. Esta nueva versión funcionaba a frecuencias del orden de 900 MHz. Esto posibilitó dar servicio a un mayor número de usuarios y avanzar en la portabilidad de los terminales.

Además del sistema NMT, en los 80 se desarrollaron otros sistemas de telefonía móvil tales como: AMPS (Advanced Mobile Phone System) en EEUU y TACS (Total Access Comunication System).

El sistema TACS se utilizó en España con el nombre comercial de MoviLine. Estuvo en servicio hasta su extinción en 2003.

En la década de 1990 nace la segunda generación, que utiliza sistemas como GSM, IS-136, iDEN e IS-95. Las frecuencias utilizadas en Europa fueron de 900 y 1800 MHz.

El desarrollo de esta generación tiene como piedra angular la digitalización de las comunicaciones. Las comunicaciones digitales ofrecen una mejor calidad de voz que las analógicas, además se aumenta el nivel de seguridad y se simplifica la fabricación del Terminal con la reducción de costos que ello conlleva. Muchas operadoras telefónicas móviles implementaron Acceso múltiple por división de tiempo (TDMA) y Acceso múltiple por división de código (CDMA) sobre las redes AMPS existentes convirtiéndolas así en redes D-AMPS. Esto trajo como ventaja para estas empresas poder lograr una migración de señal analógica a señal digital sin tener que cambiar elementos como antenas, torres, cableado, etc.

El estándar que ha universalizado la telefonía móvil ha sido el archiconocido GSM (Global System for Mobile communications o Groupe Spécial Mobile). Realmente, GSM ha cumplido con todos sus objetivos pero al cabo de un tiempo empezó a acercarse a la obsolescencia porque sólo ofrecía un servicio de datos a baja velocidad (9.6 Kbps) y el mercado empezaba a requerir servicios multimedia que hacían necesario un aumento de la capacidad de transferencia de datos del sistema. Es en este momento cuando se empieza a gestar la idea de 3G, pero como la tecnología CDMA no estaba lo suficientemente madura en aquel momento se optó por dar un paso intermedio: 2.5G.

Dado que la tecnología de 2G fue incrementada a 2.5G, en la cual se incluyen nuevos servicios como EMS y MMS:

- EMS es el servicio de mensajería mejorado, permite la inclusión de melodías e iconos dentro del mensaje basándose en los sms; un EMS equivale a 3 o 4 sms.
- MMS (Sistema de Mensajería Multimedia) Este tipo de mensajes se envían mediante GPRS y permite la inserción de imágenes, sonidos, videos y texto. Un MMS se envía en forma de diapositiva, en la cual cada plantilla solo puede contener un archivo de cada tipo aceptado, es decir, solo puede contener una imagen, un sonido y un texto en cada plantilla, si se desea agregar más de estos tendría que agregarse otra plantilla. Cabe mencionar que no es posible enviar un vídeo de más de 15 segundos de duración.

Para poder prestar estos nuevos servicios se hizo necesaria una mayor velocidad de transferencia de datos, que se hizo realidad con las tecnologías GPRS y EDGE.

- GPRS (General Packet Radio Service) permite velocidades de datos desde 56Kbps hasta 114 Kbps.
- EDGE (Enhanced Data rates for GSM Evolution) permite velocidades de datos hasta 384 Kbps.

3G nace de la necesidad de aumentar la capacidad de transmisión de datos para poder ofrecer servicios como la conexión a Internet desde el móvil, la videoconferencia, la televisión y la descarga de archivos. En este momento el desarrollo tecnológico ya posibilita un sistema totalmente nuevo: UMTS (Universal Mobile Telecommunications System).

UMTS utiliza la tecnología CDMA, lo cual le hace alcanzar velocidades realmente elevadas (de 144 Kbps hasta 7.2 Mbps, según las condiciones del terreno).

Por último, la generación 4, o 4G es la evolución tecnológica que ofrece al usuario de telefonía móvil un mayor ancho de banda que permite, entre muchas otras cosas, la recepción de televisión en Alta Definición. Hoy en día existe un sistema de este nivel operando con efectividad solo con algunas compañías de EEUU, llamado LTE.

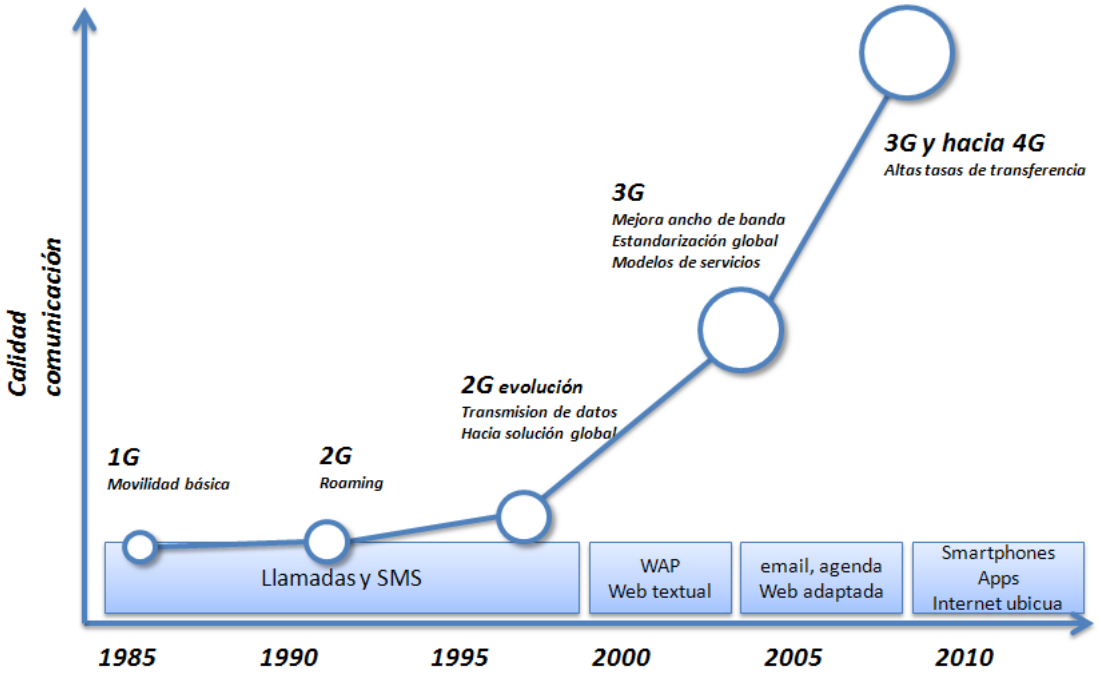

Figura 2: Evolución de las comunicaciones móviles. [2]

2.2 Principales plataformas móviles

Las distintas plataformas móviles se pueden ver en [19] siendo las principales:

2.2.1 Android

Sistema operativo desarrollado inicialmente por Android Inc. y posteriormente adquirida por la OHA (Open Handset Alliance). Este sistema operativo está basado en el kernel de Linux.

Se trata de un sistema de código abierto y se puede acceder tanto al código fuente como al listado de incidencias donde podemos ver problemas aún no resueltos y reportar problemas nuevos.

Fue estreno tuvo lugar en Octubre de 2008 y su principal objetivo fue plantar cara al sistema operativo móvil lanzado por Apple. Su cuota de mercado ha sufrido un gran crecimiento en los últimos años hasta alcanzar un 59% actualmente.

- **Ventajas:** Al ser un sistema operativo desarrollado en código abierto aporta una alta capacidad de comunicación hacia los desarrolladores, ya que los errores pueden ser reportados por parte de los usuarios. Otro punto a favor de Android es la increíble confianza que está recibiendo de los fabricantes. Gracias a ello, la oferta de teléfonos con Android es amplia y la oferta es variada tanto en marcas como en precios.
- **Inconvenientes:** Hoy en día uno de los aspectos negativos de Android es su fragmentación: aunque va mejorando, actualizar el sistema operativo a nuevas versiones no es tan fácil, debido a que la responsabilidad de proveer a los usuarios de las nuevas versiones recae sobre las compañías telefónicas, las cuales a veces no están interesadas en ofrecer esa posibilidad a sus usuarios. Debido a este contratiempo, Android ha recibido muchas críticas por parte del mercado.

2.2.2 IOS

Es el sistema implementado en el iPhone (el famoso Smartphone fabricado por Apple) el cual está basado en el Mac kernel que se encuentra en Mac OS X. El diseño de este sistema operativo fue una gran innovación y revolución en el mercado de la telefonía móvil.

El sistema iOS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de Servicios Principales, la capa de Medios de comunicación y la capa de Cocoa Touch. El sistema operativo ocupa bastante menos de medio gigabyte

del total del dispositivo. Esto se realizó para poder soportar futuras aplicaciones de Apple, así como aplicaciones de terceros.

- **Ventajas:** Algunos aspectos positivos como el buen diseño, la funcionalidad, su facilidad de uso y una variedad de aplicaciones y juegos enorme lo convierten en un referente en el mercado. También su perfecta integración con servicios en la nube y equipos de sobremesa, especialmente Mac, es otro de sus puntos fuertes.
- **Inconvenientes:** El sistema de Apple es cerrado, por lo que la posibilidad de realizar cambios son escasas. Otro inconveniente a destacar es que si quieres disfrutar de este sistema operativo, tendrás que desembolsar una cantidad alta de dinero para adquirir el dispositivo, pues solo hay un fabricante que lo incorpore. Además, las licencias de desarrollo no son gratuitas.

2.2.3 Windows Phone

Es un sistema operativo móvil compacto desarrollado por Microsoft, y diseñado para su uso en Smartphones y otros dispositivos móviles. Windows Phone hace parte de los sistemas operativos con interfaz natural de usuario.

Se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows. Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente.

- **Ventajas:** Un diseño moderno, práctico, atractivo y con características innovadoras han sorprendido ya a más de uno. Windows Phone cuenta con una gran inversión y se ha diseñado para competir con los más grandes: el resultado es un sistema moderno y capaz de hacer frente a la competencia.
- **Inconvenientes:** La variedad de móviles con Windows Phone no es tan amplia como la que ofrecen Android o Symbian, aunque está en crecimiento. Por otra parte, al llegar algo más tarde a la primera división del firmamento móvil, la cantidad de aplicaciones disponibles en estos momentos es baja. Su demanda se ha ralentizado actualmente, al estar a la espera de que salga la nueva versión, basada en Windows 8.

2.2.4 Symbian OS

Symbian es un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, Siemens, Fujitsu, LG, Motorola, Panasonic, etc.

El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile de Microsoft. Cuando se convirtió en el sistema operativo con más cuota de mercado, aparecieron y ahora Android., iOS y BlackBerry 6 RIM.

- **Ventajas:** Symbian ha sido siempre fiable e innovador. Con fuerte énfasis en las funciones básicas de telefonía y multimedia de sus dispositivos, cuenta con una tremenda variedad de dispositivos disponibles. Se trata de una excelente opción para conseguir terminales de gama media y baja, debido a su fiabilidad y al precio asequible de muchos de sus modelos.
- **Inconvenientes:** Symbian ha perdido protagonismo con la llegada de iPhone y Android, sobre todo en los Smartphones punteros. Este sistema parece que ha quedado obsoleto en cuanto a las novedades del mercado y sus cifras están disminuyendo a gran velocidad.

2.2.5 Blackberry

Es un sistema operativo multitarea (OS) para la gama de productos BlackBerry. Es desarrollado por la empresa RIM, y su capacidad multitarea le permite un uso intensivo de los dispositivos de entrada disponibles en los teléfonos, en particular la rueda de desplazamiento y el teclado físico.

Las versiones anteriores permitían la sincronización inalámbrica con Microsoft Exchange Server para el correo electrónico y calendario, al igual como con Lotus Domino e-mail. Fue un dispositivo con una gran fama entre usuarios que necesitaban estar conectados continuamente al correo electrónico.

- **Ventajas:** Perfecto para el uso de correo electrónico, Blackberry destaca también por los aspectos de seguridad y por sus teclados QWERTY que, al estilo de un teclado de PC, permiten una escritura muy rápida.

- **Inconvenientes:** El potencial multimedia no es su fuerte principal. Tampoco existen una gran cantidad de aplicaciones disponibles para este dispositivo, por lo que no se puede comparar con las propias de Android y Apple.

En la figura 3 se puede observar una comparación esquemática de las diferentes plataformas anteriormente expuestas y que resume los datos aportados de una forma más compacta:

Detalles básicos					Inicio
	 Android Cupcake	 BlackBerry OS 4.7	 iPhone OS 3.0	 S60 5th Edition	 Windows Mobile 6.5
Tipo de núcleo	Linux	Propietario	OS X	Symbian	Windows CE
Adaptabilidad	Excelente	Buena	Mala	Excelente	Excelente
Edad de la plataforma	Joven	Madura	Adolescente	Madura	Madura
Soporte para empresas	Nada	BlackBerry	Exchange	Exchange, Domino, BlackBerry	Exchange, Domino, BlackBerry
Tecnologías inalámbricas	GSM, WiFi	GSM, CDMA, WiFi	GSM, WiFi	GSM, WiFi	GSM, CDMA, WiFi

Figura 3: Comparativa Sistemas Operativos móviles [3]

2.3 Historia de la biometría

La Identificación Biométrica, es decir, el reconocer a una persona por alguna característica biofísica o de comportamiento, está tomando cada vez más importancia. Esta importancia nace de las limitaciones existentes en los sistemas actuales de identificación de una persona (por contraseñas y/o tarjetas). En este artículo se presenta una introducción a la Biometría, comentando las diversas etapas de que se compone un Sistema de Identificación Biométrica [4].

La expansión de las redes telemáticas y la proliferación de distintas soluciones en las que nunca se encuentran cara a cara dos personas, para que una de ellas identifique a la otra, complican de gran manera el proceso de identificación.

Estas necesidades se plantean cada vez más en los nuevos sistemas que aparecen. Los sectores en los que se requiere una identificación electrónica (es decir, una identificación que debe ser realizada por una máquina), son muy variados. Desde sistemas de mínima seguridad, como puede ser la identificación del socio de un club de campo para reservar una pista de tenis, hasta la consulta de información sanitaria de un paciente. Sin embargo es el movimiento de dinero, y por tanto las aplicaciones bancarias y comerciales, las que suelen tomar mucho más protagonismo a la hora de plantear los esquemas de identificación a utilizar.

A lo largo de las últimas décadas, diversos sistemas se han propuesto para solucionar la identificación de forma electrónica, siendo los más representativos las contraseñas, los elementos de identificación (como el pasaporte o el DNI) y las características biológicas o de comportamiento. Éstas últimas, como se verá más adelante, son la única solución que permite una verdadera identificación de la persona.

Según el Diccionario de la Real Academia Española, se define BIOMETRÍA como «Estudio mensurable o estadístico de los fenómenos o procesos biológicos». Esta definición se hace más específica cuando se utiliza el término de Biometría dentro del campo de la Identificación de Personas. Se podría decir en este caso, que Biometría es la ciencia por la que se puede identificar a una persona basándose en sus características biofísicas o de comportamiento. Expuesto en forma de ejemplos, es la ciencia que consigue reconocer a una persona mediante una imagen de su rostro o mediante la impresión de su huella dactilar.

Como es lógico, la capacidad de identificación biométrica es algo innato en los seres vivos, ya que poseen la característica de reconocer a sus semejantes. Pero la Biometría como ciencia de estudio de la individualidad de las personas, nace seriamente a finales del siglo XIX. Es entonces cuando en Europa se extendió con gran éxito el sistema francés de Identificación Antropométrica de Bertillon, en el que se realizaban numerosas medidas del cuerpo de una persona. Fue precisamente un experto en este sistema, Sir Francis Galton, quien realizó a finales del siglo XIX estudios muy detallados sobre la huella dactilar, estudiando su estabilidad, unicidad y morfología. Sus trabajos, complementados por los de Vucetich, Henry, Hershel y Faulds (cada uno de forma independiente), consiguieron que la identificación por huella fuera aceptada y se convirtiera en el método de identificación biométrica más utilizado por la policía mundial. La evolución de la tecnología, así como la dificultad, en muchas ocasiones, de captar la huella de una persona y, por supuesto, el progreso por

parte de los supuestos criminales de evitar su posible identificación mediante esos métodos, han empujado a pensar en nuevas vías de realizar la identificación biométrica, desarrollándose diversas soluciones alternativas, como las basadas en voz, rostro, etc.

2.4 Etapas en la identificación biométrica

Las técnicas de identificación biométrica son muy diversas, ya que cualquier elemento significativo de una persona es potencialmente utilizable como elemento de identificación biométrica. Las distintas técnicas que existen serán tratadas en el próximo apartado. Sin embargo, incluso con la diversidad de técnicas existentes, a la hora de desarrollar un sistema de identificación biométrica, se mantiene un esquema totalmente independiente de la técnica empleada. Los sistemas, tal y como se puede ver en la Figura 3, se basan en dos fases totalmente diferenciadas:

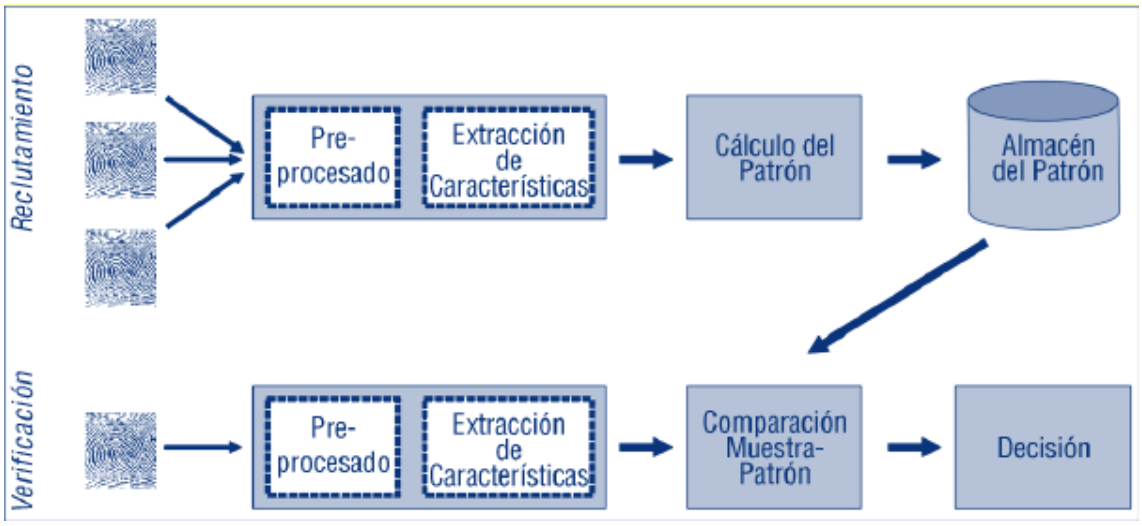

Figura 4: Etapas en un sistema de identificación biométrica [4]

1. **Reclutamiento**: en esta fase, se toma una serie de muestras del usuario, y se procesan, para posteriormente extraer un patrón, el cual se almacenará y será el conjunto de datos que caracterizará a ese usuario. Si se captura más de una muestra, el patrón suele ser el resultado de una media de las características obtenidas. Este proceso se hace de forma supervisada, es decir, existe una persona encargada de controlar cómo se produce la captura de los datos, así como de asegurar la identidad de la persona que se está reclutando en el sistema. Además, se aprovecha esta fase para enseñar al usuario cómo funciona el sistema y aclararle todas las dudas que pudiera tener.

2. **Utilización**: una vez que se tiene almacenado el patrón del usuario, éste puede utilizar el sistema con normalidad, y sus características son comparadas con el patrón almacenado, determinando el éxito o fracaso de esa comparación. Pero como se observa en la Figura 1, cada una de las fases mencionadas, está basada

en una serie de bloques que hacen que las características biológicas o de comportamiento del individuo acaben siendo un elemento que lo identifique.

Estas fases son:

- **Captura**: se toman los datos biofísicos o de comportamiento del sujeto. La toma de los datos depende, evidentemente, de la técnica biométrica empleada, pero también se pueden encontrar muchas variaciones para la misma técnica biométrica. Por ejemplo, la huella dactilar puede ser obtenida por cámara de vídeo, ultrasonidos, efecto capacitivo sobre un semiconductor o exploración por láser.
- **Pre-procesado**: en este bloque se adecuan los datos capturados para facilitar el tratamiento que tiene que realizar el siguiente bloque. Este bloque se encarga, dependiendo de la técnica, de tareas como: reconocer el inicio de una frase y medir el ruido de fondo, hacer una extracción de bordes de la imagen capturada, localizar la muestra, rotarla y ampliarla (o reducirla), para que se encuentre entre los márgenes que reconoce el algoritmo siguiente, etc.
- **Extracción de Características**: se puede considerar el bloque más significativo de la técnica a utilizar. Es el bloque en el que se fundamenta la capacidad del sistema de distinguir entre sujetos. Sin embargo, debido a distintas aproximaciones al problema, este bloque puede seguir orientaciones muy diversas, e incluso contradictorias, para la misma técnica, creándose distintos métodos dentro de una misma técnica. Por otro lado, en algunas ocasiones, el desconocimiento sobre las características que se deben extraer, lleva a utilizar técnicas basadas en Redes Neuronales, que mediante entrenamiento de las mismas, se intentan adecuar a los resultados esperados.
- **Comparación**: una vez extraídas las características de la muestra capturada, se han de comparar éstas con las previamente almacenadas, es decir, el patrón. Lo más importante que hay que dejar claro cuando se habla de este bloque, es que no se trata de una comparación binaria (o de igualdad), sino que la variación de las muestras, por variaciones en la captura o leve variación de las características de sujeto, hacen que la comparación dé como resultado una probabilidad de semejanza. Por tanto, para determinar el éxito o fracaso de la comparación, habrá que determinar un umbral en esa probabilidad.

Por otro lado, el modo en el que se hace el reclutamiento no es tampoco trivial. En algunas técnicas basta una única toma de los datos, mientras que en otras puede ser necesario tomar varias muestras y en distintas sesiones (días o semanas), tal y como ocurre, por ejemplo, en los sistemas basados en voz.

Hasta ahora se ha estado hablando siempre de Identificación Biométrica; sin embargo, la Identificación se puede realizar basándose en dos esquemas de funcionamiento del Sistema de Identificación Biométrica:

- **Reconocimiento**: también llamado, en algunos textos, simplemente Identificación (lo cual llega a causar cierta confusión). Se basa en identificar a un usuario dentro de todos los usuarios que ya se encuentran en el sistema. Por lo tanto, se comparan las características extraídas con los patrones de todos los usuarios reclutados por el sistema. Este esquema de funcionamiento, necesario para muchas aplicaciones, tiene como inconvenientes la necesidad de una Base de Datos de patrones (con los requisitos oportunos de capacidad de almacenamiento y seguridad de los datos) y la existencia de una red de comunicaciones, siempre on-line, que comunique los puestos de identificación con la Base de Datos.
- **Autenticación**: también llamado sencillamente Verificación. Trata de responder a la pregunta: ¿es este sujeto la persona que dice ser? En este esquema de funcionamiento, el usuario, al que se le toman sus características biométricas, también comunica su identidad. El sistema se encarga, entonces, de comparar las características extraídas, con el patrón del usuario indicado. El patrón del usuario puede estar almacenado en una Base de Datos, tal y como se hace en los sistemas de Reconocimiento, o, si el patrón es suficientemente pequeño, en un sistema portátil de información como puede ser una tarjeta.

2.5 Modalidades biométricas

Aunque las características de la huella dactilar son, sin lugar a duda, las más ampliamente utilizadas para realizar una identificación biométrica, cualquier otra característica biológica o del comportamiento de una persona puede ser usada para realizar la identificación, siempre que dichas características se demuestren propias y únicas de la persona a identificar. Las distintas modalidades que se están estudiando actualmente se pueden ver descritas a continuación:

- **Voz**: es una técnica con uno de los mayores potenciales comerciales: los servicios de atención telefónica personal, como la Banca Telefónica. Es una técnica que se lleva estudiando durante varias décadas, existiendo innumerables métodos para realizar, tanto la extracción de características, como la comparación. Algunos métodos son dependientes del texto pronunciado (es decir, todo o parte del texto que se recita debe ser idéntico en todas las ocasiones), mientras otros son independientes del mismo (pudiéndose recitar cualquier locución para realizar la identificación). Desgraciadamente no están todavía determinados todos los factores que

influyen en las locuciones, tales como la edad, las enfermedades, el comportamiento, el estado de ánimo, etc.

- **Huella Dactilar**: tal y como ya se ha comentado, es, sin lugar a duda, la más estudiada y probada. Existen numerosos estudios científicos que avalan la unicidad de la huella de una persona y, lo que es más importante, la estabilidad con el tiempo, la edad,etc. En estos aspectos es una técnica que les lleva mucha ventaja a las demás, debido a su siglo de existencia. Su captura recibe diversas formas, sobre todo últimamente, debido a la innovación tecnológica.
- **Rostro**: el método de identificación que nuestro cerebro usa más a menudo y de una forma más sencilla. En la actualidad existen muchos grupos de investigación trabajando en esta técnica con diversos métodos (estudios morfológicos, transformadas multiresolución, etc.). Los resultados que se están consiguiendo son bastante prometedores, aunque le falta todavía bastante hasta llegar al nivel de otras técnicas. El gran inconveniente encontrado es la variabilidad del rostro del sujeto a lo largo del tiempo: gafas, barba, longitud del pelo, peinado, expresiones, etc.
- **Iris**: esta técnica fue impulsada por John G. Daugman en 1993. Los resultados obtenidos son, sin lugar a dudas, unos de los mejores de la actualidad teniendo en cuenta que las características en las que está basada, el patrón de la textura del iris ocular, permanece inalterable durante la vida del sujeto debido a la protección que le proporciona la córnea. Por otro lado, los estudios sobre la unicidad de sus características, la han colocado muy por encima de la huella dactilar. Su gran inconveniente es el coste de los equipos, aunque teniendo en cuenta el grado de fiabilidad alcanzado, existen numerosas aplicaciones de alta seguridad que podrían usar esta técnica.
- **Geometría del Contorno de la Mano y/o del Dedo**: se trata de una técnica en la que se estudian diversos parámetros morfológicos de la mano (o el dedo) del usuario, tales como anchuras, alturas, etc. La técnica basada en geometría del dedo se puede considerar como una simplificación de la basada en contorno de la Mano. El gran atractivo de esta técnica, debido a su simplicidad, bajo coste y mínimo tamaño del patrón, la han convertido en la técnica con mayor éxito comercial en el último par de años.

Técnica	Ventajas	Inconvenientes
Voz 	- Muy bajo coste - En algunas aplicaciones puede resultar inapreciable al usuario (p.e., servicios telefónicos)	- Rendimiento bajo - Se está estudiando el aumentar la unicidad y estabilidad
Huella 	- Muy estudiado/desarrollado - Unicidad, estabilidad y rendimiento altos - Reconocimiento legal - Medio coste	- Connotaciones «policiales» para el usuario - Detección de dedo vivo depende de pruebas colaterales a la captura
Iris 	- Unicidad mayor que huella - Gran estabilidad por protección de la córnea - FAR prácticamente nula - Fácil detección de ojo vivo	- Alto coste - Inicialmente incómodo para el usuario
Mano 	- Fácil uso y gran aceptación por el usuario - Medio coste - Bajo coste computacional - Sin connotación «policial»	- Unicidad y estabilidad no probadas en grandes poblaciones - Detección de mano viva depende de pruebas colaterales
Rostro 	- Cómodo, e incluso inapreciable, para el usuario - Medio coste	- Sensible a cambios del sujeto (barba, gafas, pelo, ...) - Todavía en investigación y desarrollo

Figura 5: Comparativa entre las técnicas más importante [4]

A la hora de juzgar una técnica biométrica, son muchos los parámetros que hay que considerar, de los que se pueden destacar los siguientes:

- **Universalidad**: si las características se pueden extraer de cualquier usuario o no.
- **Unicidad**: la probabilidad de que no existan dos sujetos con las mismas características.
- **Estabilidad**: si las características que se extraen permanecen inalterables en relación con diversos parámetros (tiempo, edad, enfermedades, etc.).
- **Facilidad** de captura: si existen mecanismos sencillos de captura de los datos biológicos o de comportamiento del sujeto.
- **Rendimiento**: o tasas de acierto y error.
- **Aceptación por los usuarios**
- **Robustez frente a la burla del sistema**: si la técnica puede reconocer el falseamiento de los datos capturados (uso de fotos, dedos de látex, etc.).
- **Coste**

3 Plataforma de desarrollo: Android

3.1 ¿Qué es Android?

Android es un sistema operativo móvil basado en Linux que está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos, al igual que iOS, Symbian, BlackBerry OS y Windows Mobile [5]. Pero, ¿qué hace a Android tan especial frente a los diferentes sistemas operativos?:

- Android es una plataforma de desarrollo completamente abierta y gratuita basada en Linux y en código abierto. Este hecho hace a Android una plataforma muy atractiva a los desarrolladores debido a que pueden utilizarla y personalizarla a su gusto porque no está limitada a ningún vendedor.
- Android posee una arquitectura tipo componentes, es decir, las partes de una aplicación pueden utilizarse en otra distinta por otro desarrollador. Este hecho creará una nueva corriente creativa en el espacio móvil.
- Gestión automática del ciclo de vida de una aplicación. Los programas están aislados los unos de los otros, mediante la inclusión de varias capas de seguridad. El propio sistema Android está optimizado para ser implementado en dispositivos de poca memoria y poca batería.
- Android consta de una arquitectura que consta de cinco módulos, lo que se puede observar en la siguiente imagen:

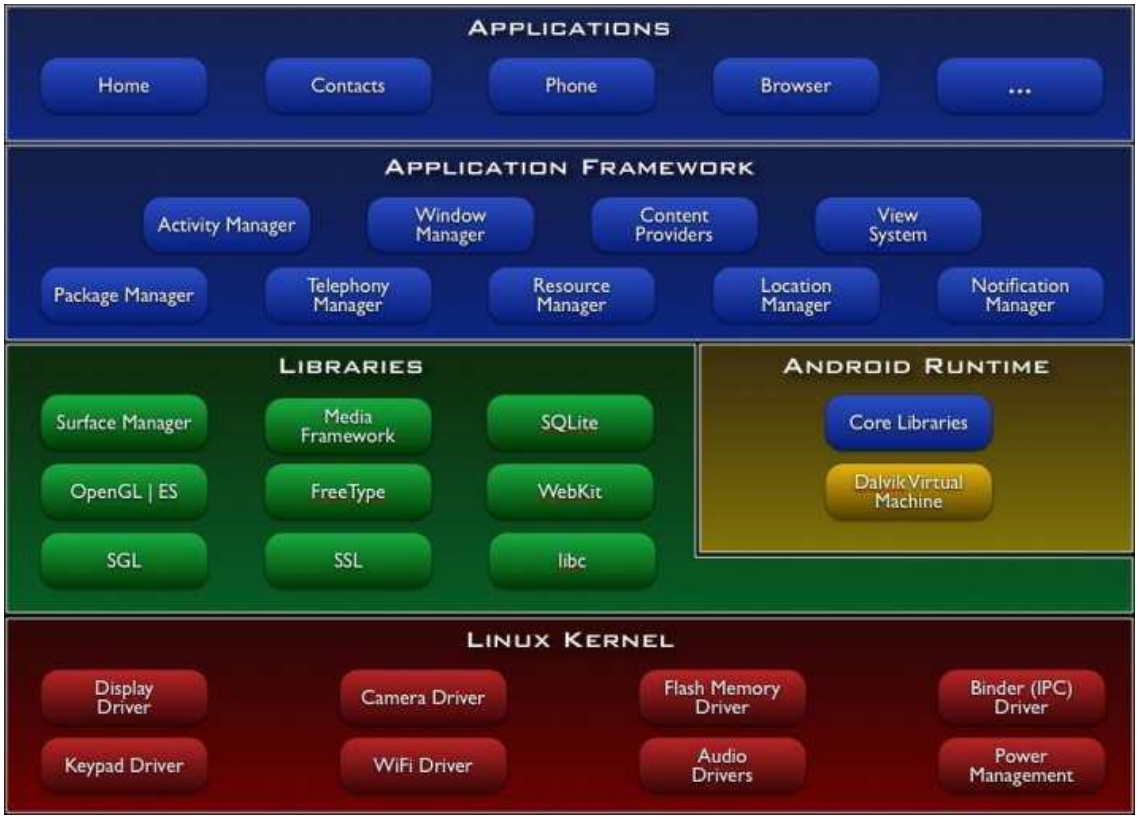

Figura 6: Arquitectura de Android [16]

- Portabilidad a través de un amplio rango de hardware actual y futuro. Todos sus programas están escritos en lenguaje Java y ejecutados por la máquina virtual Dalvik de Android, de modo que su código se podrá portar a arquitecturas como ARM, x86 y otras.

3.2. Historia de Android.

Allá por octubre del año 2003, Andy Rubin, Rich Miner, Nick Sears y Chris White daban forma a Android Inc., la compañía con sede en Palo Alto (California). En sus inicios, únicamente trascendió que la actividad de la empresa se centraba en "el desarrollo de software para teléfonos móviles". Android Inc. pasó casi dos años trabajando "en la sombra", hasta que Google comenzó a adquirir algunas empresas "startup" del sector móvil, con la clara intención de replicar su éxito de la Web en el futuro de las telecomunicaciones inalámbricas. En el mes de agosto Android Inc. fue adquirida por Google [6].

El 5 de noviembre de 2007 la Open Handset Alliance, un consorcio de varias compañías entre las que están Texas Instruments, Broadcom Corporation, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, Intel, LG, Marvell Technology Group, Motorola, y T-Mobile; se estrenó con el fin de desarrollar estándares abiertos para dispositivos móviles. Junto con la formación de la Open Handset Alliance, la OHA estrenó su primer producto, Android, una plataforma para dispositivos móviles construida sobre la versión 2.6 del kernel de Linux.

El 9 de diciembre de 2008, se anunció que 14 nuevos miembros se unirían al proyecto Android, incluyendo PacketVideo, ARM Holdings, Atheros Communications, Asustek, Garmin, Softbank, Sony Ericsson, Toshiba, Vodafone y ZTE.

El mismo día se dio a conocer por vez primera lo que hoy conocemos como Android, una plataforma de código abierto para móviles que se presentaba con la garantía de estar basada en el sistema operativo Linux. Pocos días después, en concreto el 12 del mismo mes de noviembre, se "liberaba" el primer "kit" de desarrollo de aplicaciones para que los programadores comenzasen a "hacer de las suyas": nacían las apps del sistema.

3.2.1 La primera versión

El 23 de septiembre de 2008 veía la luz en los EEUU un móvil con la versión 1.0 de Android. El modelo **G1 de HTC** fue el primer dispositivo disponible en el mercado que incorporaba esta nueva plataforma

Figura 7: Primer teléfono que incorpora Android [7]

En esta primera versión de Android destaca la **plena integración de los servicios de Google**, el navegador Web compatible con HTML y XHTML y la inclusión por vez primera del Android Market, el almacén de aplicaciones más popular.

3.2.2 Las actualizaciones

Android 1.1

En febrero de 2009 llegó la primera actualización para Android, unos tres meses después del lanzamiento del G1. La versión 1.1 fue dedicada básicamente a reparar errores y a implementar las actualizaciones que hasta ese momento ninguna plataforma estaba haciendo [8].

Esta versión fue bautizada como Bender, pero nunca pudo utilizarse por motivos de marcas registradas [9].

Android 1.5 Cupcake

Android 1.5 es más conocido por su nombre en clave, **Cupcake**, fue la primera versión en utilizar nombre de postres. Cada versión después de Cupcake ha sido nombrada con un nombre de postre continuando el orden alfabético.

En esta versión se aprecian algunos cambios en la interfaz de usuario y se introdujo el teclado virtual coincidiendo con la salida del primer Android con pantalla táctil y sin teclado físico, el HTC Magic.

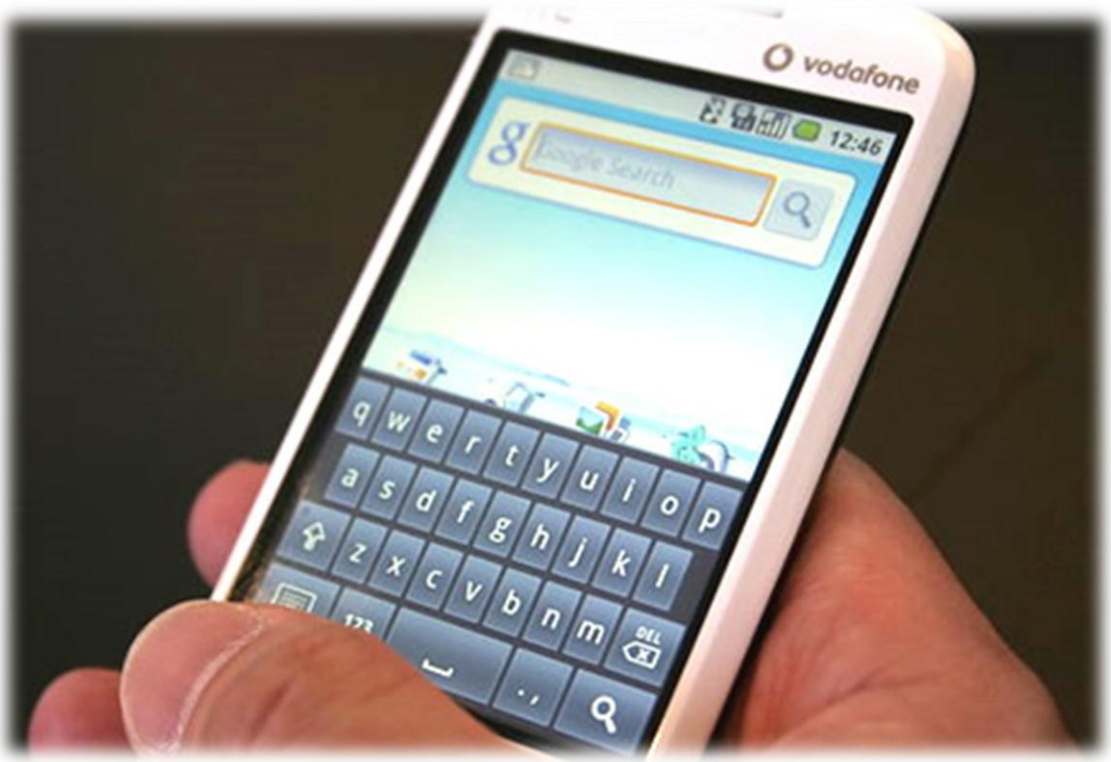

Figura 8: Versión 1.5 Cupcake con teclado virtual [20]

Desde el principio Google introdujo la posibilidad de que desarrolladores terceros pudieran crear sus propios teclados virtuales, algo que a día de hoy sigue diferenciándolo de otras plataformas para dispositivos móviles.

En esta nueva versión se incluyeron por primera vez los widgets, que son pequeñas aplicaciones o programas cuyos sus objetivos son dar fácil acceso a funciones frecuentemente usadas y proveer de información visual, y se permitió que los desarrollados por terceros estuvieran disponibles para los usuarios.

Android 1.6 Donut

Con la llegada de Donut vino el soporte para redes CDMA haciendo que Android llegara a Estados Unidos y Asia aumentando sus miras de mercado. Pero tal vez la mejora más significativa fue la posibilidad de correr el sistema operativo en múltiples resoluciones de pantalla y relaciones de aspecto. A raíz de esta actualización se establecieron las bases para que hoy en día podamos disfrutar de pantallas con resolución QVGA, HVGA, WVGA, FWVGA, QHD y 720p.

Android 2.0/2.1 Eclair

Un año después del estreno del HTC G1, a principios de noviembre de 2009 fue lanzado Android 2.0 Eclair. Fue ofrecido en exclusiva con Verizon y el Motorola Droid, un teléfono que marcó un antes y después para Android y con el que Motorola volvió a ser la gran marca que fue.

El Droid de Motorola fue el teléfono más potente que se había visto en el mercado hasta la fecha, con una pantalla con resolución de 854 x 480. Pero no solo fue el Droid el que impulsó las ventas de Android sino las mejoras que se introdujeron en la versión 2.0.

Por primera vez se podrían añadir varias cuentas en el mismo dispositivo con acceso al correo electrónico y a los contactos de cada una, además también se introdujo soporte para cuentas de Exchange. También Google Maps Navigation fue publicado junto con la versión 2.0 y fue un paso adelante parar integrar un sistema de navegación de automóviles en el móvil. Hoy en día sigue siendo una de las mejores opciones para tu teléfono.

Después del Droid/Milestone prácticamente la mayoría de teléfonos lanzados llegaron con Android 2.1, una corrección de errores y que Google no renombró dejándola con el nombre de Eclair. En esta corrección aparecieron por vez primera los fondos de pantalla animados en Android, la utilidad de transcribir voz a texto a modo que los usuarios podían dictar a su teléfono y éste lo transcribía a texto (TTS).

Y entonces llegó el Nexus One, el primer teléfono de Google fabricado por HTC y que estaba destinado a mostrar el más puro estilo Android 2.1 sin ninguna modificación. Fue uno de los primeros smartphones con procesador de 1GHz Qualcomm Snapdragon del mercado, además contaba con una pantalla AMOLED con resolución WVGA. Hoy en día se sigue utilizando y es aún un buen teléfono.

Android 2.2 Froyo

Lanzado a mediados de 2010 trajo una gran cantidad de cambios. La pantalla de inicio fue rediseñada, se ampliaron los 3 paneles existentes desde el inicio a 5 con un nuevo grupo de accesos directos dedicados y se agregaron unos puntos para saber en cada momento en la pantalla donde nos encontrábamos. El Nexus One fue el primer teléfono en actualizarse a Android 2.2.

Froyo también introducía una galería completamente rediseñada con imágenes en 3D que aparecen al inclinar el teléfono. Además se introdujo soporte para hotspot móvil (compartir la conexión 3G). En esta versión se agregó la posibilidad de poner una

contraseña o PIN en la pantalla de bloqueo para aquellos usuarios que no les gustaba el patrón de desbloqueo.

Figura 9: Desbloqueo por patrón [18]

Android 2.3 Gingerbread

Un año y medio después del lanzamiento de Froyo y el Nexus One, Google volvió con un nuevo móvil de marca propia pero esta vez en colaboración con Samsung, el Nexus S y aprovechó para lanzar la nueva versión del sistema operativo, Android 2.3 Gingerbread. Fue una actualización menor en muchos sentidos pero trajo algunos cambios importantes en la interfaz de usuario.

Se mejoró el control en copiar y pegar, el teclado virtual y Google buscó la maximización de la batería creando herramientas de gestión de desarrollo. También se incluyo soporte para cámara frontal. El Nexus S ya dispondría de cámara frontal, aunque en un principio solo servía para tomar fotos con ella.

Android 3.0 Honeycomb

Esta es la versión de Android adaptada para tablets, que presentó de la mano de Motorola junto con el Xoom. Cambió de color, del verde típico de Android al azul que se utilizó para la batería, el widget del reloj, indicadores de señal y algunas otras características de la interfaz. Se integró una barra en la parte inferior de la pantalla con una serie de botones virtuales que hacen que no se necesiten botones dedicados.

La multitarea es mejorada gracias al diseñador Matías Duarte, ex diseñador de webOS contratado por Google. Aparece un nuevo botón virtual de Aplicaciones recientes en la parte inferior de la pantalla en la que podemos ver una lista de las últimas aplicaciones utilizadas con capturas de pantalla de las mismas.

Se introduce el concepto de barra de acción, una barra permanente situada en la parte superior de cada aplicación que los desarrolladores pueden utilizar para mostrar las opciones de acceso frecuente, menús, etc. Es como una barra de estado dedicada a cada aplicación.

Android 4.0 Ice Cream Sandwich

Llegamos a la última versión del sistema operativo de Google, Android 4.0 Ice Cream Sandwich. Ha sido lanzada junto con el Galaxy Nexus, el nuevo Smartphone Google fabricado por Samsung. Por primera vez se modifica el tipo de letra. Como no, el teclado virtual también ha sido modificado donde esta vez incluye un sistema de corrección mucho más avanzado que subraya en color rojo las palabras mal escritas e incorpora también un diccionario.

La pantalla de notificaciones también ha recibido una pequeña actualización con las notificaciones individuales extraíbles que permiten deslizar cualquier notificación fuera de la pantalla y acceder a ella. El soporte de la tecnología NFC ya había sido promocionado fuertemente con el lanzamiento de Gingerbread y el Nexus S aunque no había prácticamente ninguna aplicación. En Ice Cream Sandwich se busca potenciar el uso de NFC con una nueva característica para transferencia de datos entre dos teléfonos con solo tocarlos.

Además del bloqueo con contraseña y con patrón de desbloqueo se ha agregado la opción del desbloqueo facial.

Se añade un gestor para el uso de los datos en el que se informa de las aplicaciones que consumen más datos, se puede ver el uso total desglosado en un periodo de tiempo configurable por el usuario.

Jelly bean, el futuro.

Por ahora no se ha desvelado nada de cómo será la nueva versión, aunque sí sabemos que su nombre será Jelly Bean [9]. Para su presentación toca esperar como mínimo para el próximo Google I/O de 2012.

A continuación podemos observar la cuota de mercado de las distintas versiones de Android a fecha 14/08/2012:

Version	Codename	API Level	Distribution
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%

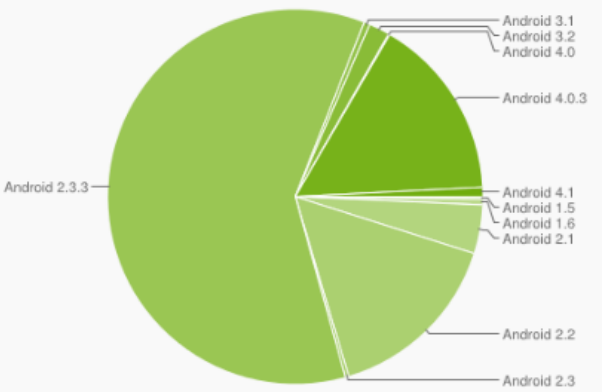

Figura 10: Cuota de mercado de las distintas versiones Android [10]

3.2 Estructura de una aplicación.

3.2.1 Componentes de una aplicación

Los componentes de la aplicación son los pilares esenciales. Cada componente es otro punto a través del cual el sistema puede entrar en ella. No todos los componentes son verdaderos puntos de entrada para el usuario y algunos dependen el uno del otro, pero cada uno existe como una entidad propia y desempeña un papel específico, cada uno es una pieza única que ayuda a definir el comportamiento global de la aplicación.

Como podemos ver en [11] hay cuatro tipos diferentes:

- **Actividades**: Una actividad representa una pantalla única con una interfaz de usuario. Aunque hay actividades que trabajan juntas para formar una experiencia de usuario coherente, cada una es independiente de las otras. Como tal, una aplicación diferente puede iniciar cualquiera de estas actividades (si la aplicación lo permite).
- **Servicios**: Un servicio es un componente que se ejecuta en segundo plano para realizar operaciones de larga duración o para realizar un trabajo para procesos remotos. Un servicio no proporciona una interfaz de usuario. Otro componente, tal como una actividad, puede iniciar el servicio y dejarlo funcionar o unirse a él con el fin de interactuar.
- **Proveedores de contenido (Content providers)**: Un content provider gestiona un conjunto compartido de datos de aplicación. Puede almacenar los datos en el sistema de archivos, una base de datos SQLite, en la web, o cualquier otro

lugar de almacenamiento persistente aplicación pueda acceder a él. A través del content provider, otras aplicaciones pueden consultar o modificar incluso los datos (si el content provider lo permite).

- **Receptores de anuncio de difusión (Broadcast receiver):** Un broadcast receiver es un componente que responde a los anuncios de difusión en todo el sistema. Muchas transmisiones se originan en el sistema, por ejemplo, una emisión de anunciar que la pantalla se apaga, la batería está baja, o una imagen fue capturada. Las aplicaciones también pueden iniciar emisiones. Por ejemplo, para permitir que otras aplicaciones sepan que algunos datos se han descargado en el dispositivo y está disponible para que los utilice. Aunque los broadcast receivers no muestran una interfaz de usuario, se puede crear una barra de estado de notificación para alertar al usuario cuando se produce un evento de difusión.

3.2.2 Ciclo de vida de una actividad.

Cada actividad de una aplicación Android puede estar en alguno de los estados que se muestran en la figura 11. Como desarrollador, no tiene control sobre el estado en el que se encuentra su programa; es un procedimiento controlado por el sistema. Sin embargo, recibe notificaciones cuando el estado va a cambiar gracias a las llamadas al método onXX().

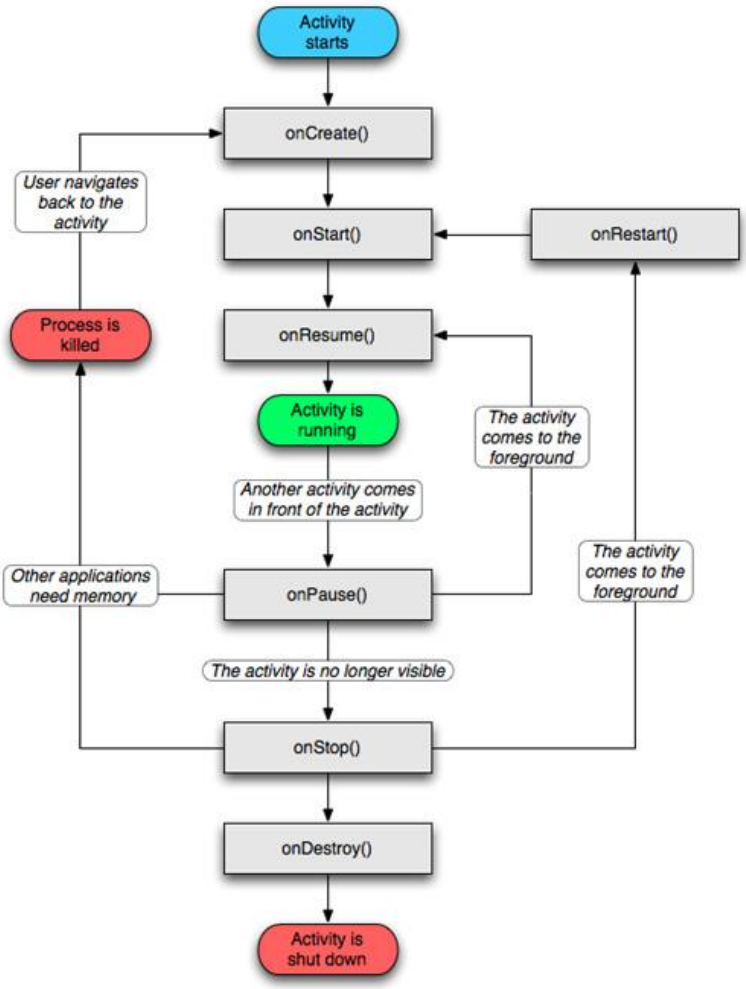

Figura 11: Ciclo de vida de una actividad Android [11]

- **onCreate(Bundle)**: Este método es llamado cuando se inicia la actividad por primera vez.
- **onStart()**: Éste indica que la actividad va a ser mostrada al usuario.
- **onResume()**: A éste se le llama cuando la actividad puede comenzar a interactuar con el usuario.
- **onPause()**: Se ejecuta cuando la actividad se dispone a pasar a segundo plano, normalmente porque otra actividad ha sido ejecutada en primer plano.
- **onStop()**: Se llama cuando la actividad ya no es visible por el usuario y no será necesaria durante un tiempo.
- **onRestart()**: Si se llama a este método, indica que se está volviendo a mostrar la actividad al usuario desde un estado detenido.
- **onDestroy()**: Se llama a este método justo antes de que la actividad sea destruida.

3.3 Herramientas de desarrollo.

Todos los elementos necesarios para el desarrollo de aplicaciones para Android están disponibles gratuitamente en Internet para todas las plataformas. El entorno de desarrollo elegido en este proyecto para el desarrollo de la aplicación es Eclipse, un editor de código con resaltado de sintaxis que nos proporciona compilación en tiempo real, y también se utilizará el entorno de desarrollo NetBeans, el cual utilizaremos para abrir y hacer las modificaciones pertinentes en la norma BioAPI Java 3.0 para posteriormente adaptarla para Android utilizando Eclipse.

En internet está disponible gratuitamente el Android SDK (Android Software Development Kit) que es el entorno de desarrollo que incorpora todo lo necesario para desarrollar aplicaciones Android, incluido un emulador, que nos permitirá probar nuestras aplicaciones en un terminal virtual, y numerosas herramientas para depurar y empaquetar nuestras aplicaciones.

Una vez instalados el Android SDK y el plugin para Eclipse podemos actualizar estos componentes a través del manager incluido, lo que nos permite incorporar las nuevas APIs de las nuevas versiones de Android.

También cabe destacar que existe la posibilidad de instalar el Android SDK en NetBeans mediante su plugin correspondiente, pero se ha decidido utilizar dos entornos para evitar posibles confusiones o complicaciones y, de este modo permitir al alumno conocer ambos entornos de desarrollo

A continuación se muestra unas imágenes de los entornos dónde se va a trabajar durante la realización de este trabajo.

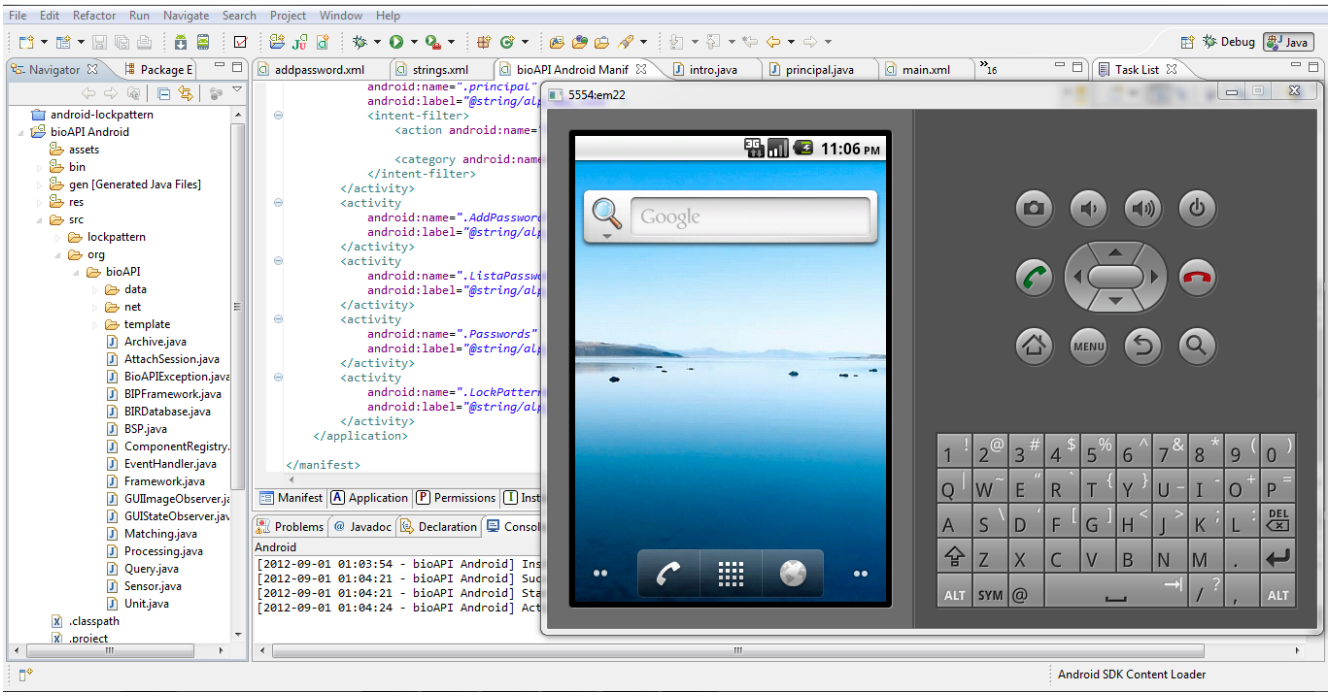

Figura 12: Entorno de desarollo Eclipse

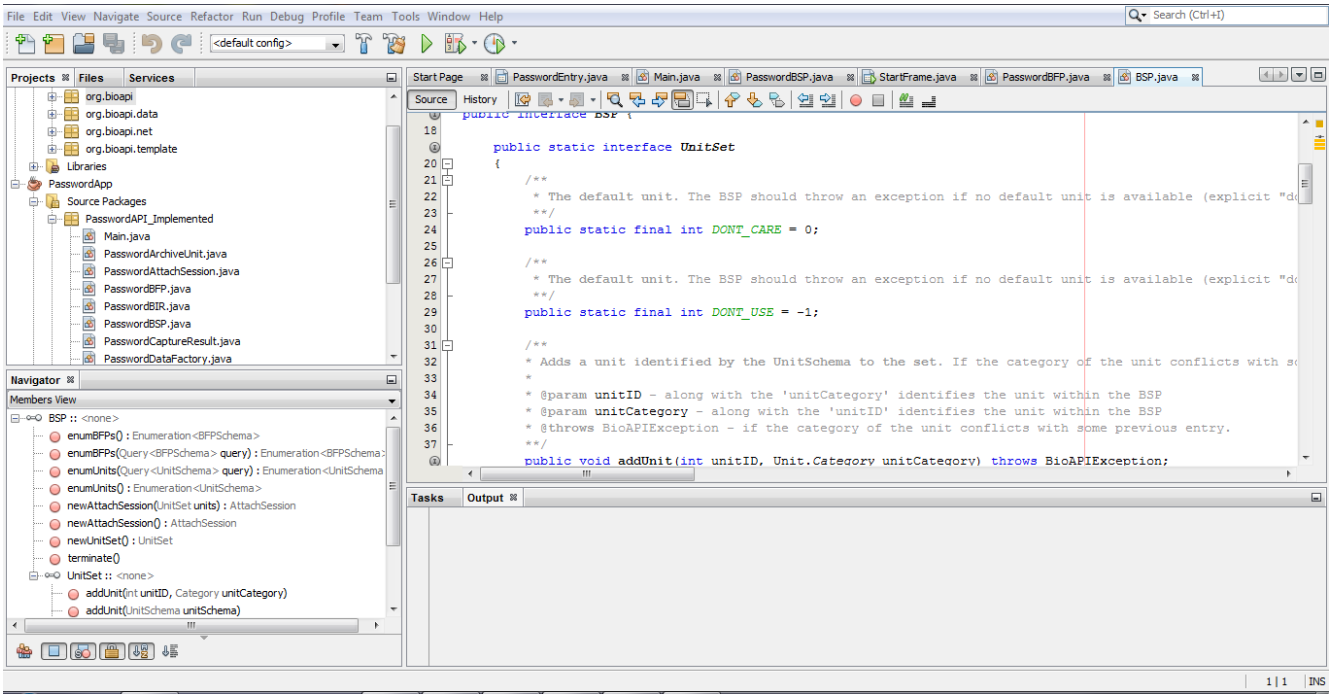

Figura 13: Entorno de desarrollo NetBeans

4 Diseño de la solución técnica.

En este capítulo se dan a conocer los pasos seguidos para lograr los objetivos descritos al inicio del trabajo. Se planteará la solución, teniendo en cuenta las limitaciones que posen los dispositivos disponibles, y finalmente se explicara el proceso de diseño de la aplicación.

4.1. Planteamiento del problema

A continuación se describirán distintas situaciones en las que se interesante almacenar información con un alto grado de seguridad, almacenándola en un dispositivo y limitando el acceso a su consulta a la persona dueña del dispositivo.

- **Entorno empresarial:** En multitud de empresas existe personal cualificado encargado de poseer información exclusiva y al mismo tiempo debe disponer de una alta accesibilidad a esos datos. De este modo se podría permitir que el usuario y propietario del dispositivo de empresa tenga siempre a mano la información necesaria y solamente accesible por esta misma persona, evitando su acceso por terceros ante un robo o pérdida del teléfono.
- **Aplicaciones comerciales:** Millones de personas en el mundo tienen que hacer uso de contraseñas para identificarse y acceder a datos personales y de alto valor, como pueden ser cuentas bancarias, datos o trámites con el gobierno o ministerios, etc. Mediante la inclusión de este tipo de aplicaciones se evitaría tener que memorizar millones de diferentes contraseñas las cuales pueden ser olvidadas o hackeadas.

4.2 Planteamiento de la solución

Una vez expuesto el problema, se debe buscar una solución económica, eficiente, que se adapte a nuestras necesidades y siempre guardando una gran sencillez.

Debido a que existe un alto porcentaje de la población que cuenta con un Smartphone y las múltiples herramientas con las que contamos para desarrollar aplicaciones para estos teléfonos, se ha decidido optar por utilizar una plataforma móvil para el desarrollo de una aplicación que incluya un acceso mediante el desbloqueo por patrón que, como ya se dijo anteriormente, será una función que sustituya a una identificación biométrica.

Para acceder a los datos almacenados por el teléfono, se optará por almacenarlos en una base de datos que se ejecutará en una vista del teléfono tras desbloquear la pantalla mediante el patrón y proceder a la consulta de los mismos.

Una vez elegida la plataforma y el método por el que se almacena y solicita la información, solo queda desarrollar y conseguir una aplicación robusta y sin errores.

4.3 Requisitos y marco regulatorio

Para el desarrollo de la aplicación debemos tener en cuenta si existe algún tipo de restricción que aplique al problema, así como alguna normativa legal.

4.3.1 Requisitos

Los requisitos de la aplicación vienen principalmente determinados por las limitaciones de hardware y software del dispositivo. A nivel de hardware, como ya hemos expresado anteriormente varias veces, nos encontramos con la dificultad de que no existe dispositivo en el mercado con un sensor biométrico incorporado (lector de huellas dactilares, escáner de iris, etc.) por ello mismo se optó por realizar un patrón de desbloqueo. Esto se considera como una solución transitoria válida ya que cumple con las premisas principales del reconocimiento biométrico, ya que en el caso de la Biometría, tras realizar reconocimiento biométrico el sensor varias capturas, realiza un patrón con aquellos datos necesarios, distinguibles y únicos de cada persona. Para el caso del patrón de desbloqueo el único requisito será el de poseer un dispositivo con una pantalla táctil, donde en el mercado actual más de la mitad de los dispositivos son táctiles con una tendencia ascendente. En cuanto a las resoluciones de pantalla en los distintos tamaños de Smartphones, no existe problema alguno gracias a la mejora introducida a partir de la versión Android 1.6 Donut, en la cual se incluyó la posibilidad de que el propio sistema adapta la aplicación al tamaño de la pantalla.

A nivel de software, las posibles restricciones residen en lo referente a las distintas versiones de Android anteriormente descritas en el punto 3.2.2. Como se podía observar en el gráfico de la cuota de mercado de las distintas versiones, en el caso de desarrollar para Android 2.2 Froyo, más de un 90% del total de los dispositivos serán compatibles con nuestra aplicación. Por ello es muy importante elegir la versión adecuada, ya que, cuanta más nueva sea la versión, menor será la cuota de dispositivos compatibles.

4.3.2 Marco regulatorio

Como se expuso con anterioridad en los planteamientos del problema, la implementación de este sistema podría servir para identificar personas en aplicaciones comerciales, y para ello se debería adaptar la legislación vigente sobre la Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal, (LOPD), puesto que las empresas privadas tanto como administraciones públicas tendrían acceso a unos datos e información personales como lo son las características biométricas de cada individuo.

Esto supondría un gran avance por parte de la identificación personal, pero supondría un problema en cuanto a la violación de los derechos sobre sus datos personales si esta ley no fuese adaptada al marco social y las empresas no se ajustaran a esa legislación.

4.4 Diseño

Antes de comenzar a describir nada, se realizará una breve explicación de las 3 partes en las que está dividida la aplicación de este trabajo.

En primer lugar nos encontramos con la aplicación biométrica, que será la que solicite información, mediante un API (Application Programming Interface) al sensor biométrico una vez llegado al punto de acceso restringido. La aplicación es programada para invocar las funciones del API.

El siguiente eslabón es el BioAPI Framework y es en este punto donde nos encontramos con las funciones y clases propias de la norma BioAPI, que una vez adaptada a código Android, será lo siguiente que nos encontremos. El BioAPI Framework es el corazón de la norma, ya que es el que gestiona los distintos BSPs y mapea las llamadas a funciones de los BSPs por parte de los APIs direccionando al correcto BSP.

La aplicación solicitaría información mediante el API y es el BioAPI Framework el intermediario de toda información. El BioAPI Framework soporta las funciones solicitadas por el API.

Por último nos encontramos con el BSP, lo que podríamos decir que son los “drivers” del dispositivo, y son el último eslabón de esta cadena, pues son los encargados de obtener las funciones biométricas (capturas biométricas, extracción de características comunes, búsqueda, emparejamiento, etc.). Una vez que la aplicación no necesita el uso de el BSP, es desligado y desactivado.

Una aplicación puede acceder a las funcionalidades del BSP (siempre a través del BioAPI Framework) solamente después de que el BSP ha sido instalado y ligado.

También cabe destacar que una aplicación puede cargar y unirse a más de un BSP cada vez y también un BSP puede estar ligado a más de una aplicación al mismo tiempo.

A continuación se puede observar un esquema de lo descrito anteriormente:

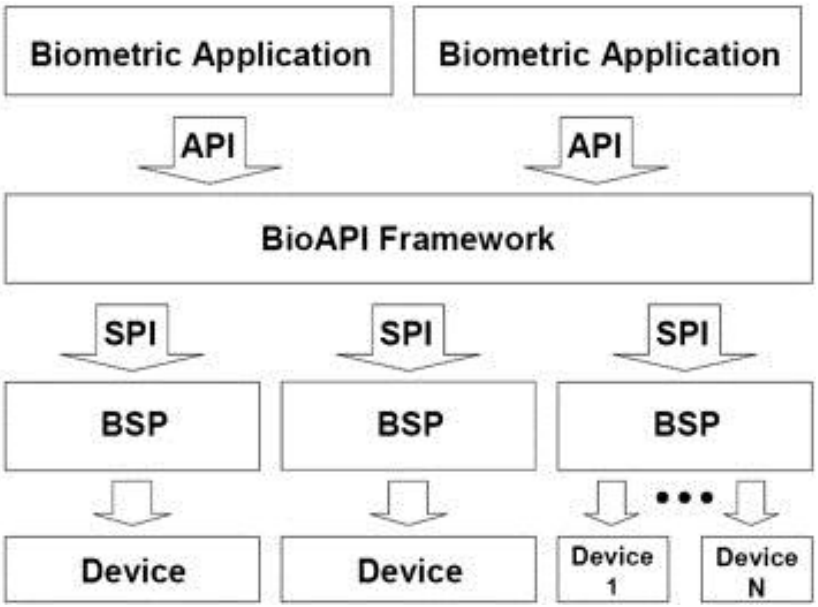

Figura 14: Estructura BioAPI [12]

4.4.1 Aplicación

Una vez descrita su función principal, hay que hacer ciertas aclaraciones para realizar un diseño sencillo de nuestra aplicación.

En primer lugar cabe diferenciar entre las posibilidades que nos brinda el BioAPI Framework en cuanto a su desarrollo y a las funciones que posee:

- **Reclutar (Enroll)**: Este sería el primer paso en cualquier aplicación biométrica, el cual podríamos decir que se trata del "registro" del usuario, por el cual la aplicación solicita que el sensor capture y realice un patrón del usuario, para posteriormente almacenar este patrón y compararlo posteriormente.
- **Identificar**: En este caso, la aplicación solicita al BSP la captura, para posteriormente, con los datos obtenidos, compararlos con todos los datos existentes en la base de datos, y, en el caso de coincidir con alguno de todos ellos, permitir al acceso a esa persona, o denegárselo en el caso de que no se haya registrado con anterioridad.

- **Verificar**: Esta función es más concreta que la anterior y es solicitada tras averiguar anteriormente que usuario esta identificándose. La aplicación solicita al sensor que capture una imagen, para posteriormente compararla con el patrón del usuario concreto y determinar si se le permite el acceso o no.

Una vez aclaradas estas funciones, se ha dado por supuesto que en nuestro caso sólo existirá un usuario, el cual es el propietario del dispositivo, por lo cual no se utilizará la funcionalidad de identificar, pero aclarar que esa funcionalidad está disponible para implementarse en el futuro.

En el siguiente paso se describirá el diseño de la interfaz con el usuario el cual deberá ser sencillo y muy claro. Una idea muy clara sobre la primera pantalla en aparecer se muestra en la siguiente imagen:

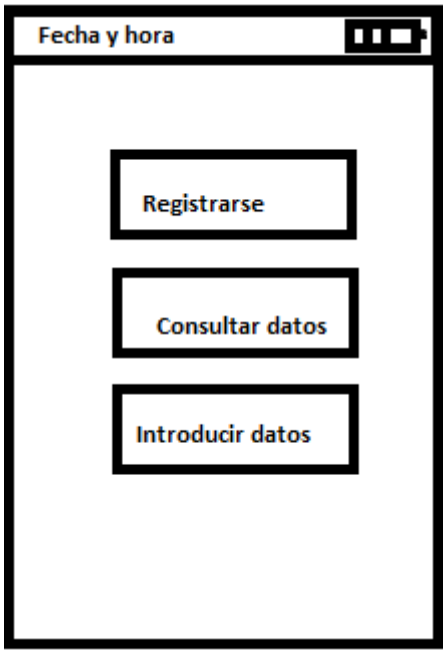

Figura 15: Diseño vista principal

En la imagen se observan 3 botones los cuales se relacionan con el Registro de usuario, el siguiente sería la identificación del usuario, y por último el almacenamiento de datos por parte del usuario.

En cuanto al primer botón, al pulsarlo debe aparecer otra pantalla nueva, donde deba introducir su patrón de desbloqueo, que, tras terminarlo exitosamente, deberá introducir el mismo una vez más, para evitar confusiones y almacenar el patrón. Un ejemplo sería el siguiente:

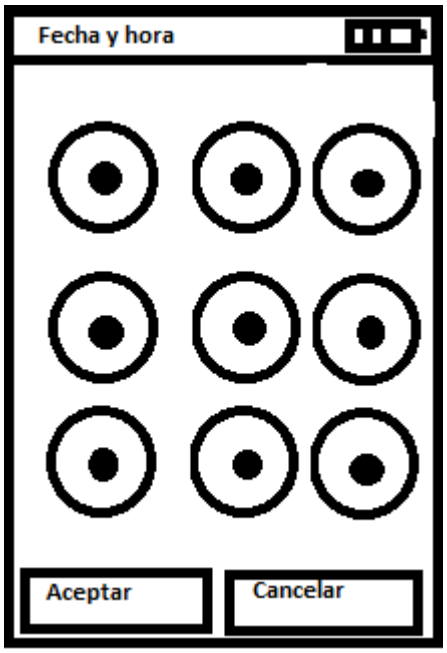

Figura 16: Diseño registro de patrón

Una vez realizada esta función por primera vez, el usuario podría pulsar el segundo botón e identificarse con el patrón, que tras ser satisfactorio, entraríamos en una nueva pantalla con el siguiente posible diseño:

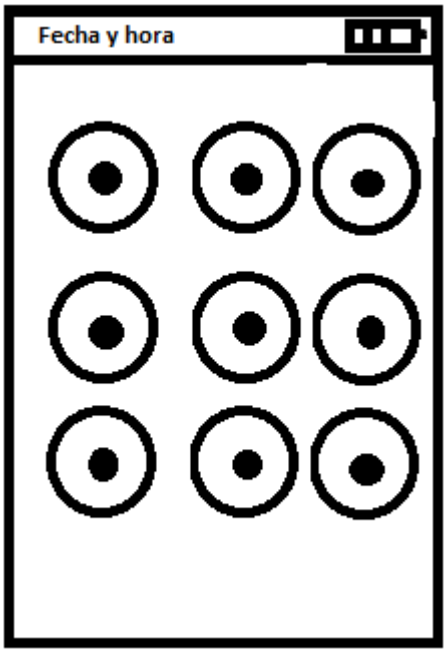

Figura 17: Diseño verificar patrón

Como se pueden ver aparecen varios cuadros de texto, donde el usuario podrá, leer, modificar o borrar sus datos almacenados. En la siguiente imagen se puede ver un diseño como ejemplo de esta vista:

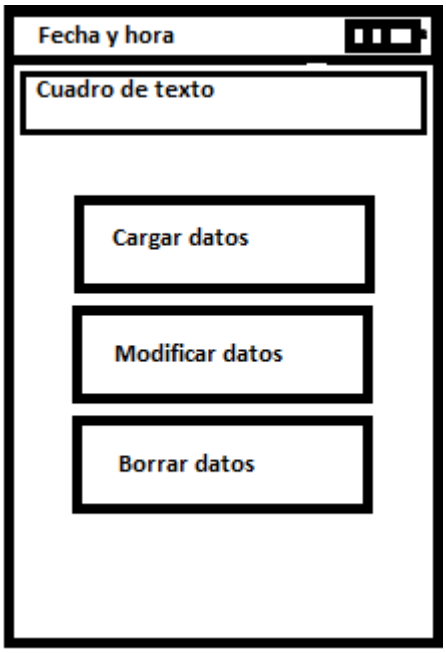

Figura 18: Diseño vista consultar datos

Por último, nos queda describir la funcionalidad del último botón que tras ser pulsado observamos un diseño muy similar al anterior:

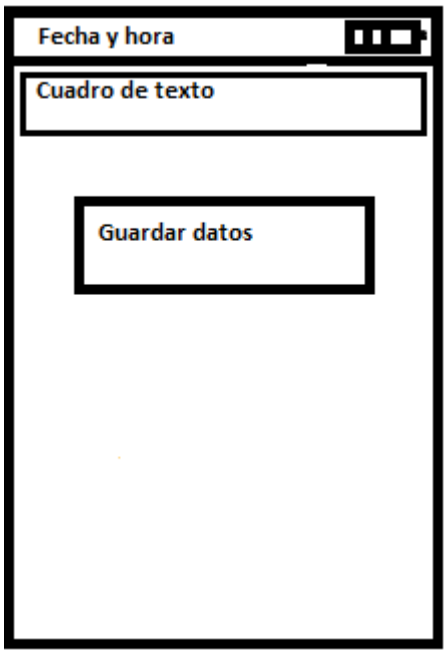

Figura 19: Diseño vista Guardado de datos

En esta última pantalla es destacable la mayor sencillez, debido a que su única utilidad sea la de almacenar los datos correspondientes, pero ningún usuario podrá acceder a la lectura de los datos almacenados en esta pantalla.

Diseñadas las interfaces, queda por aclarar su estructura final, donde podemos verlo en el siguiente diagrama de flujo:

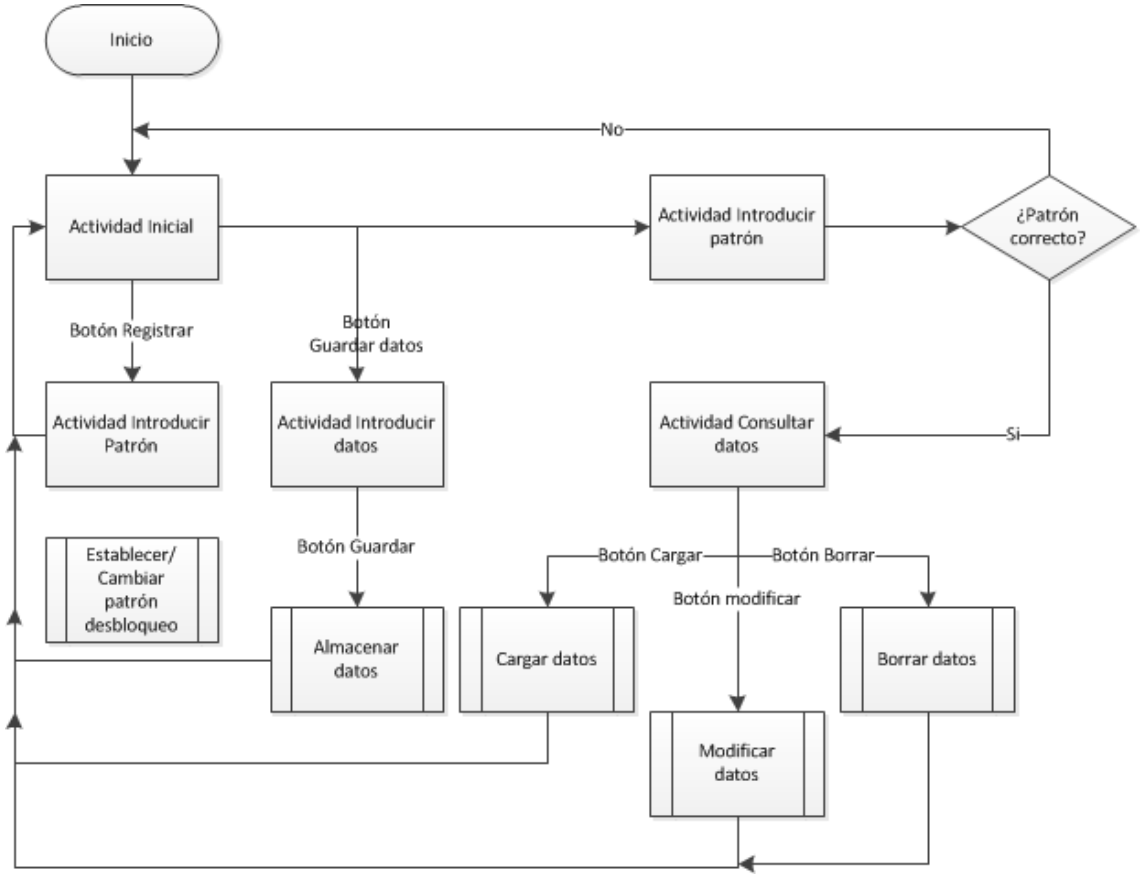

Figura 20: Diagrama de flujo de la aplicación móvil

4.4.2 BioAPI Framework

En cuanto al diseño de esta parte, se ha optado por realizar una adaptación del mismo código implementado en lenguaje Java, manteniendo su misma estructura, para no modificar ninguna de sus funcionalidades ni dar lugar a error.

Como ya se indicó anteriormente, el Framework es una de las partes más importantes de este desarrollo, pues es esta parte la que gestiona el flujo de datos entre aplicación y sensor, así como controla las sesiones realizadas entre dispositivo y aplicación, es decir, decide si un sensor puede unirse en una sesión a una aplicación biométrica dependiendo de si el sensor se encuentra ocupado o en reposo.

Como se puede imaginar, al adaptar esta parte de lenguaje Java a lenguaje Android, se tuvieron que aprender los fundamentos del lenguaje Java [13] y compararlos con los de Android, para no cometer ningún error en el portado del código.

En cuanto al portado del código, siempre se realizó con sumo cuidado, interfaz por interfaz siempre cumpliendo con los requisitos impuestos por la norma bioAPI (ISO/IEC WD 19784-1). En cuanto al porte se tuvo que tener exclusivo cuidado con las siguientes clases:

- **Archive:** Esta interfaz es la encargada de realizar las llamadas de métodos que gestionan la base de datos. Se implementó a modo de que utilizara el paquete de Android encargado de las bases de datos en lugar del paquete java.
- **BioAPIException:** En este caso, se tuvo que tener exclusivo cuidado pues es la interfaz encargada de lanzar los métodos cuando se ha producido una mal función del sistema.
- **BIRDatabase:** Esta interfaz produjo pequeños errores al principio. Se solucionaron mediante la inclusión del tipo de datos que tenía que soportar, que era el formato de los patrones realizados por la aplicación Android.
- **BSP:** En cuanto a este tipo, se tuvo que esperar a la posterior realización del BSP con lenguaje Android (el cual se explicará brevemente en el siguiente punto) y su adaptación al mismo.
- **Processing:** En esta interfaz tuvieron que modificarse ciertos detalles y puntualidades de su código, pues no estaba implementado para su desarrollo conjunto con la pantalla táctil de un dispositivo móvil.
- **Sensor:** Es un caso similar al anterior, pero en este caso eran las llamadas a capturar cualquier tipo de movimiento de la mano del usuario en la pantalla táctil las que fueron modificadas.

Cabe destacar, que todas las interfaces del Framework no han sido nombradas en este punto. Aquí solo han sido nombradas aquellas que suponían una dificultad al adaptarlas al lenguaje Android. Tanto estas como el resto de interfaces que componen el Framework, serán descritas en el capítulo de desarrollo.

Una vez realizado el porte exitosamente, se procede a la realización del BSP, que es la parte encargada de adaptar el elemento físico a este desarrollo: el sensor.

4.4.3 BSP

Supone una obligación introducir qué se espera de esta parte, pues es la que pone la guinda al pastel, y hace que el sensor opere sin ningún problema en conjunto con el resto del sistema.

Esta parte es la encargada de realizar las operaciones solicitadas por la aplicación cuando el Framework realiza las llamadas a las funciones. Por esto mismo, las funciones que posea el BSP deben tener una estrecha relación con el Framework.

El BSP debe contar con una clase fundamental que sea la encargada de ligarse a una aplicación. Esto es, cuando una aplicación solicita tomar una muestra biométrica, ya sea por registro o identificación, el Framework deberá elegir un sensor que esté libre para realizar la función en el caso de que existan varios. El sensor debe ser capaz de saber qué operación se le solicita que realice, y qué aplicación requiere los datos. Para ello el BSP contará con esta clase que será la encargada de obtener estos datos.

También debe contar el BSP con una clase que sea la encargada de obtener el tipo de información biométrica, ya sea para capturarla del sensor o para leerla de la base de datos y compararla con la obtenida por el sensor.

Por supuesto, el BSP debe contar con otra clase que implemente las llamadas a las funciones Registrar (Enroll()) y Verificar (Verify()). Estos métodos van a poseer una estrecha relación con la clase comentada con anterioridad.

Por último deberá contar con una clase que responda a las llamadas de captura por parte del sensor. Esto quiere decir, que sea capaz de devolver un patrón cuando se le solicita una captura tanto para el registro del usuario como para la comparación.

Para finalizar, tras esta breve descripción realizada cabe destacar el enorme esfuerzo realizado para diseñar e implementar esta parte, pues se debe actuar conforme a dos partes: debe siempre cumplir la norma bioAPI funcionando acorde con el Framework y debe de saber solicitar la información a la captura de datos del móvil mediante la pantalla táctil.

Una descripción más detallada de las partes que componen el BSP será expuesta en el capítulo 5 de desarrollo.

5 Desarrollo

5.1 Aplicación móvil

Como se vio en el apartado anterior, el diseño de esta aplicación sirve para almacenar datos y para ser más concretos, se ha decidido por implementar un desarrollo que permita al usuario guardar contraseñas junto con sus respectivos identificadores. Un ejemplo sería el identificador del correo electrónico usuario@correo.com y su respectiva contraseña.

A continuación se describirán las distintas aplicaciones con ilustraciones e imágenes que faciliten la comprensión de este trabajo.

5.1.1 Actividad principal

El punto de partida, es una aplicación Hello World de Android. Tras consultar y aprender los conceptos básicos de programación gracias a la documentación existente en internet y la bibliografía podemos desarrollar una pantalla inicial con 3 botones que haga las llamadas pertinentes hacia otras actividades o hacia las APIs del bioAPI Framework.

En esta actividad principal, podemos observar la distribución de los botones que es idéntica a la descripción realizada en cuanto al diseño en el punto 4.4.1. Al ejecutar la aplicación por primera vez podemos ver lo ilustrado en la imagen a continuación:

Figura 21: Actividad principal

5.1.2 Actividad "Registrarse"

En el caso de pulsar el botón "Registrar", la aplicación llamará a la siguiente actividad a ejecución junto con el envío del método Enroll() como principal característica hacia el BioAPI Framework. La función que realiza este método se comentará con mayor detalle más adelante. La siguiente actividad en aparecer será la encargada de realizar la captura del patrón. La clase LockPatternActivity será la encargada de solicitar al usuario que diseñe un patrón que una como mínimo cuatro puntos. En caso de no unir tres puntos, el patrón será erróneo y se solicitará al usuario la introducción de uno nuevo.

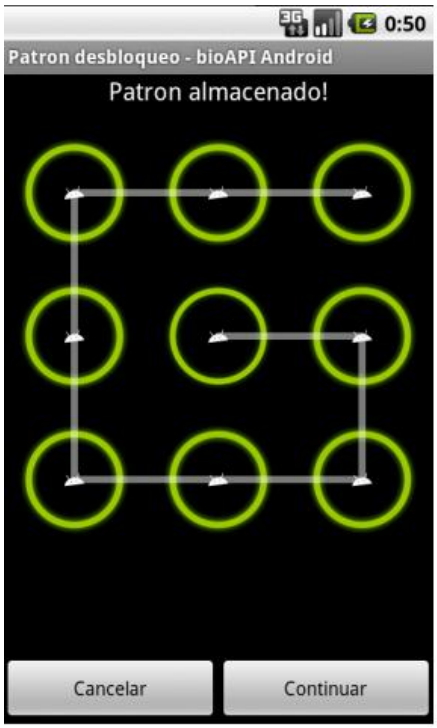

Figura 22: Actividad registrarse [14]

Una vez realizado un patrón con éxito, se deberá pulsar el botón de "Aceptar", tras el cual se reiniciará la pantalla y se volverá a solicitar al usuario que introduzca el mismo patrón una vez más. Una vez que el patrón realizado es el mismo, se le permitirá al usuario la posibilidad de pulsar una vez más el botón "Aceptar" y guardar de este modo su patrón de desbloqueo en la base de datos. Finalmente se volverá a la pantalla inicial.

5.1.3 Actividad "Guardar contraseña"

En este momento, el usuario dispone de tres cuadros de texto, que le servirán para introducir sus contraseñas y guardarlas de una manera segura. El primer cuadro

de texto es un diálogo donde el usuario debe introducir un número, el cual será el que indique su posición en la lista de la base de datos. El siguiente diálogo de texto sirve para introducir una pista o indicador de a qué pertenece la contraseña en concreto (puede ser tarjeta de crédito, cuenta de correo, etc.).

Por último, el cuadro de texto que nos encontramos al final, es donde se introducirá la susodicha contraseña.

Tras estar todo debidamente cumplimentado, se pulsará el botón Guardar para almacenar los datos, y finalmente aparecerá un cuadro de diálogo, informando de que la operación ha sido realizada con éxito.

Por último, cabe destacar que el número introducido en el primer cuadro de texto será el encargado de marcar la dirección de los datos guardados en la base de datos. Si existiera el caso de que ya existe un dato escrito en una dirección, y se intenta escribir encima, esto será imposible, debido a que el sistema sólo permite el almacenado siempre y cuando ese número este vacío. Esto se debe para evitar que alguien sin permiso sobre escriba o modifique sin permiso los datos introducidos por el usuario con anterioridad.

Figura 23: Actividad guardar contraseñas [15]

Por poner un ejemplo, supongamos que el usuario tiene almacenado en la posición “1” el identificador “correo electrónico” y la contraseña “contraseña”. Si algún usuario por error introdujera el identificador numero 1 y ponga otros datos distintos en los campos de identificador y contraseña, al pulsar el botón “Guardar” le saldrá el mensaje como válido, pero los datos no se habrán sobre escrito, pues la función sobre

escribir solamente está disponible tras la identificación del usuario mediante su patrón en otra vista que se explicará a continuación.

5.1.4 Actividad "Cargar contraseña"

Esta actividad es ejecutada tras pulsar el botón "Ver contraseñas" en la vista principal. En primer lugar cabe destacar la aparición de la vista de verificación de patrón, donde el usuario deberá introducir el patrón. Esta actividad llamará al método Verify() del BSP que se explicará más adelante.

Aquí podemos ver una imagen de cómo sería esta vista:

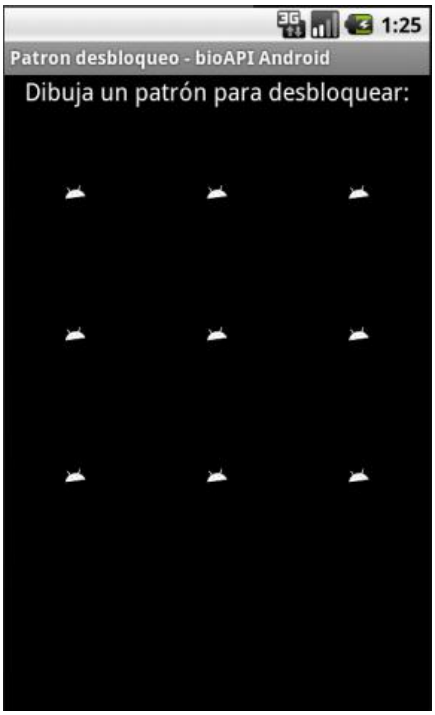

Figura 24: Actividad verificar patrón

El usuario tiene 5 intentos para introducir su patrón con éxito, en caso de no lograrlo, será devuelto siempre a la vista principal.

En caso de introducirlo correctamente, accederá a la Actividad Cargar contraseñas que podemos ver a continuación:

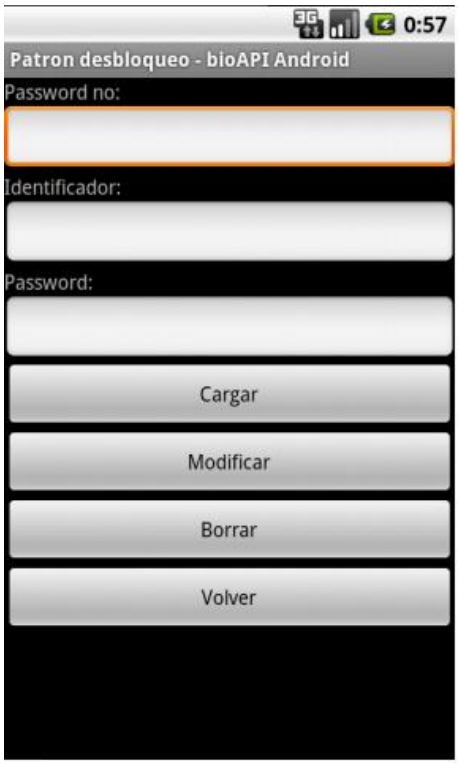

Figura 25: Actividad cargar contraseñas

Como podemos observar, esta actividad contiene también 3 cuadros de texto idénticos a los explicados en el punto anterior.

En el caso de que el usuario desee cargar la información de una contraseña, tan sólo debe introducir el número correspondiente y pulsar el botón “Cargar”. En el caso de que ese número este vació, se le indicará al usuario mediante un cuadro de diálogo emergente que la contraseña solicitada no existe. En caso contrario, aparecerán en el resto de cuadros de texto la información que guardó con anterioridad.

Se puede observar un ejemplo en la siguiente figura:

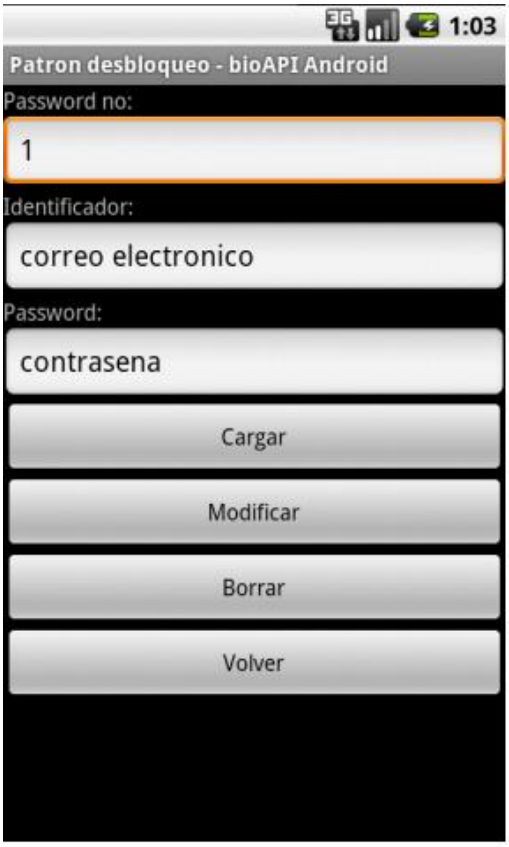

Figura 26: Actividad con contraseña cargada

El proceso de borrado de contraseñas es muy similar al de cargado. El usuario tan sólo debe introducir el número de contraseña determinado en el primer cuadro de texto y pulsar el botón “Borrar”. En caso de que la posición de contraseña esté ya vacía, se le indicará al usuario mediante un diálogo emergente. En caso contrario aparecerá un cuadro de diálogo para indicar al usuario que el proceso de borrado ha sido satisfactorio.

Por último hay que nombrar la función del botón “Modificar”. Como se explicó anteriormente, se ha incluido en esta vista para proteger de intrusos la modificación de información restringida. Si el usuario está interesado en modificar algún dato, deberá introducir el número de contraseña a modificar, en segundo lugar rellenar el resto de campos y por último pulsar el botón “Modificar”. Un cuadro de diálogo mostrará que la operación ha sido realizada con éxito.

5.2 BSP

En este momento se explicarán los métodos de los que consta esta parte y hacen posible el funcionamiento de todo el sistema.

El BSP está compuesto por el siguiente número de clases:

- PatternArchiveUnit.java
- PatternAttachSession.java
- PatternBFP.java
- PatternBIR.java
- PatternBSP.java
- PatternCaptureResult.java
- PatternDataFactory.java
- PatternMatchingunit.java
- PatternSensorUnit.java

5.2.1 PatternArchiveUnit

La función principal que cumple esta clase es la de gestionar las bases de datos, en concreto la creación de las mismas, y el almacenamiento y lectura de los datos.

Como podemos observar en la siguiente declaración del método insertRecord(String pattern) almacena el patrón introducido por el usuario:

```
public boolean insertRecord(String pattern)
    {
        try{
           if(!con.isClosed()){
            stmt= con.createStatement();
            stmt.executeUpdate("INSERT INTO "+DBname+ " VALUES (
'"+pattern+"')");
        }
         return true;
      }
         catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
             return false;
         }

                }
```

5.2.2 PatternAttachSession

Esta clase tiene la función principal de obtener los datos del sensor solicitados por la aplicación. Estos datos serán concretamente, que sensor es el solicitado y a qué aplicación está ligado el sensor requerido.

En cuanto a este desarrollo, es obvio que no existe sensor biométrico en el dispositivo móvil como se explicó anteriormente, y por ello mismo el ligado se realiza con la pantalla táctil.

5.2.3 PatternBFP

En este punto nos encontramos con una de las clases más importantes de todo este trabajo. En el interior de esta clase se encuentra la implementación de los métodos Enroll() y Verify() cuyas funciones ya se describieron en el punto 4.4.1.

En cuanto a la función de registro (Enroll()) el primer paso que realiza es concretar con que sensor se va a realizar la captura de datos. Una vez capturados comprobará que no existe un patrón existente en la base de datos que posea la misma estructura que el último introducido. Para nuestro desarrollo esta función solo tendrá utilidad en el caso de que el usuario quiera restablecer el patrón y evitar que la nueva introducida sea la misma que la ya existente. En el caso de implementar una solución con varios usuarios mediante el método identificar, la función que se realiza es que varios usuarios repitan el mismo patrón.

En cuanto a la función que realiza el método de verificación de usuario (Verify()) es la de comprobar si el patrón introducido coincide con el patrón almacenado en la base de datos. Podemos ver en el siguiente extracto de código, como al comprobar que efectivamente coinciden los patrones, el método devolverá un valor “true”, en caso contrario devolverá “false”:

```
public boolean Verify() throws BioAPIException
	{


		_bir= (PatternBIR)
_birdatabase.getSingleBIR(entries[0]).getValue();
		if(session.verify(null,_bir, null,-1,null).getResult()==true){
		 return true;
		}
		 return false;
			}
```

5.2.4 PatternBIR

En esta clase se implementan los métodos para obtener los datos y el patrón almacenados en la base de datos, la calibración del sensor, el tiempo límite para realizar una captura y también se implementa la relación con la clase PatternArchiveUnit.java para obtener los datos del sensor y de unión con la aplicación.

5.2.5 PatternBSP

En este punto se implementan todos los métodos y llamadas al sensor acerca del formato de la captura.

5.2.6 PatternCaptureResult

En este punto se declaran los métodos requeridos para que el sensor, en nuestro caso la pantalla, obtenga los datos necesarios para realizar el patrón en la captura del sensor.

5.2.7 PatternSensorUnit

Esta clase realiza las llamadas necesarias para que el sensor comience y tome una captura del elemento a medir. En este caso tomará la medida del recorrido del dedo del usuario en la pantalla.

5.3. BioAPI Framework

En cuanto a la descripción del Framework, será meramente descriptiva debido a que la adaptación a código Android no cambia ninguna funcionalidad respecto a la descrita en la normativa ISO/IEC WD 19784-1: Especificación del BioAPI. Como se describió anteriormente las clases y métodos incorporadas en esta parte del trabajo guardan una estrecha relación con las llamadas a métodos tanto por parte de la aplicación como por parte del BSP. Es esta función la que permite adaptar cualquier sensor y cumpla las funcionalidades con toda aplicación que se ajuste a la norma.

A continuación se numeran todas las clases que componen esta parte:

- **Archive**: Representa la funcionalidad que está disponible cuando la unidad de archivo se adjunta a la sesión. También es la encargada de de hacer las llamadas a los métodos para crear y borrar las bases de datos
- **AttachSession**: Representa la parte del BioAPI adjunta a la sesión BSP. Expone grupos de operaciones biométricas disponibles a través de Archive, Process, Matching y las interfaces del sensor. El objeto Session también proporciona control de la herramienta a través de la interfaz de la unidad
- **BioAPIException**: Representa la condición excepcional que se produce durante la operación de un sistema biométrico.
- **BIPFramework**: Define las extensiones BIP para BioAPI.
- **BIRDatabase**: Representa una base de datos abierta que se gestiona internamente por el BSP. La base de datos contiene los registros donde cada uno incluye un BIR y la clave característica de cada uno que es el UUID.
- **BSP**: Representa el proveedor de servicios biométricos. El BSP es la fábrica de objetos de sesión que proporcionan acceso a las operaciones biométricas.

- **ComponenRegistry**: Representa la interfaz para el registro de componentes. El programa de instalación añade, borra y modifica los BSP y los registros de BFP en el registro de componente administrado por el Framework.
- **EventHandler**: La aplicación implementa la interfaz EventHandler para controlar los eventos reportados por el BSP. La aplicación debe evitar cualquier llamada a BioAPI mientras que el cuerpo del controlador de eventos es ejecutado, en caso contrario una condición de interbloqueo puede ocurrir. Un caso de uso posible del controlador de eventos es el de sincronización de ejecución de la aplicación con el estado de la unidad biométrica. Por ejemplo, la aplicación concreta del controlador de eventos puede proporcionar el método waitForSample () que bloquea la ejecución de la llamada hasta que el evento SOURCE_PRESENT es recibida por la unidad particular.
- **Framework**: Representa el sistema biométrico. El sistema biométrico es el conjunto jerárquico que el nodo raíz es el componente Framework. El Framework controla los BSPs que impulsan sensores biométricos e implementan diversas tecnologías biométricas.
- **GUIImageObserver**: Representa la devolución de llamada que la aplicación biométrica suministra para permitir a un BSP transmitir datos de pantalla, en forma de una secuencia de mapas de bits, a la aplicación a través del marco.
- **GUIStateObserver**: Representa una devolución de llamada que proporciona un permiso al BSP para pasar información a la aplicación a través del Framework y recibir respuesta.
- **Matching**: Representa el conjunto de operaciones biométricas que están disponibles cuando la categoría Matching del sensor solicitado está ligado a la sesión.
- **Processing**: Representa el grupo de operaciones biométricas que están disponibles cuando la unidad de procesado de información está vinculado a la sesión.
- **Query**: La aplicación proporciona la clase para implementar esta interfaz para filtrar las enumeraciones devueltas por el Framework y otros objetos.
- **Sensor**: Representa el grupo de operaciones biométricas que están disponibles cuando la categoría Sensor está vinculada a la sesión.
- **Unit**: Representa la unidad genérica seleccionada para una sesión adjunta.

6. Resultados y evaluación

En la siguiente sección se va a realizar la evaluación del desarrollo utilizando dos modos distintos. En el primero se mostrarán imágenes tomadas desde el simulador, y en el segundo caso se mostrarán imágenes tomadas desde un dispositivo real.

El objetivo es demostrar el funcionamiento de la aplicación y el correcto funcionamiento de su base de datos en un entorno real.

6.1. Pruebas con el emulador

En la siguiente figura se muestra el emulador de Android con la pantalla principal que simula un entorno de Android con la versión 2.2 (Froyo).

Figura 27: Vista principal en el emulador

En la siguiente figura, se muestra el simulador cuando la aplicación solicita la introducción de un patrón al usuario una vez pulsado el botón de registrarse:

Figura 28: Vista registrar en el emulador virtual

Como se puede ver, se le muestra al usuario un mensaje de que el patrón se ha guardado en la memoria local de la aplicación. Tras pulsar el botón “Continuar” deberá introducir el mismo patrón una vez más, y como se explicó con anterioridad, en caso de que coincida con el patrón guardado, se le indica al usuario que la operación ha sido realizada con éxito y tras pulsar el botón “Continuar” de nuevo, se guardará el patrón en la base de datos y se volverá a la pantalla inicial.

El siguiente paso que se debe realizar es el de guardar datos. Como se explicó anteriormente, el usuario debe introducir un número en el primer cuadro de texto y recordarlo pues este será el que indique la posición donde se almacene en la base de datos. Los siguientes campos a rellenar son los respectivos a un identificador, para ayudar a recordar a que pertenece la contraseña y el último campo que es donde se debe almacenar la contraseña.

En la siguiente imagen se puede observar el dialogo de texto tras almacenar satisfactoriamente la información.

Figura 29: Vista en el emulador virtual con contraseña guardada

Una vez que el usuario vuelva a la pantalla principal, puede consultar sus datos almacenador pulsando el botón “Ver Contraseñas”. En el caso de que el usuario lo pulse, le será solicitado que introduzca el patrón de desbloqueo para acceder a la pantalla de consulta de datos.

Podemos observar en la siguiente imagen el caso de que el usuario haya introducido un patrón erróneo:

Figura 30: Vista verificación errónea en emulador virtual

Como ya se explicó con anterioridad, el usuario dispone de 5 intentos, en caso de no introducir el patrón correcto, se le devolverá siempre a la pantalla principal.

En el caso de introducir el patrón correcto, se le permitirá el acceso a la pantalla de consulta de datos, donde podrá cargar, modificar o borrar contraseñas. Para cargar y borrar, el usuario necesitara indicar el número de contraseña donde se encuentra posicionado en la base de datos. En el caso de borrar datos, la aplicación tomará el número, accederá a esa posición en la base y procederá a borrar todos sus campos y a liberar ese número. Desde ese mismo instante, si el usuario pretende cargar los datos con ese número, la aplicación le indicará que la contraseña no existe.

En el caso de la carga de datos, se accederá a la posición indicada y la aplicación le mostrará en cada respectivo campo la información almacenada. El usuario podrá modificar los campos “Identificador” y “Contraseña” y pulsar el botón “Modificar” para guardar estos nuevos datos en la misma posición.

En la figura 31 podemos observar un ejemplo de carga de datos:

Figura 31: Vista contraseña cargada en emulador virtual

6.2. Pruebas con un dispositivo real

Las pruebas en un dispositivo real se llevarán a cabo en un Tablet Szenio 1106, dispositivo que cuenta con la versión 2.3.1 (Gingerbread) de Android.

En esta parte nos vamos a centrar en mostrar los posibles errores o diálogos emergentes que nos devuelve la aplicación en las diferentes situaciones, ya que la funcionalidad normal, se mostró en el ejemplo anterior.

Cuando el usuario realiza la función de registro, como ya se ha explicado, debe de introducir el mismo patrón dos veces. En la siguiente imagen se podrá observar cuando la aplicación solicita al usuario la introducción de su patrón elegido una vez más. También podemos observar como el botón de “Aceptar” se encontrará con el botón “Confirmar” deshabilitado. Esto se debe a que el propio sistema no habilitará el botón hasta que el patrón introducido, no coincida con el almacenado en la memoria local, ya que, al pulsar el botón de confirmación, es cuando el sistema procede al almacenamiento del patrón en la base de datos.

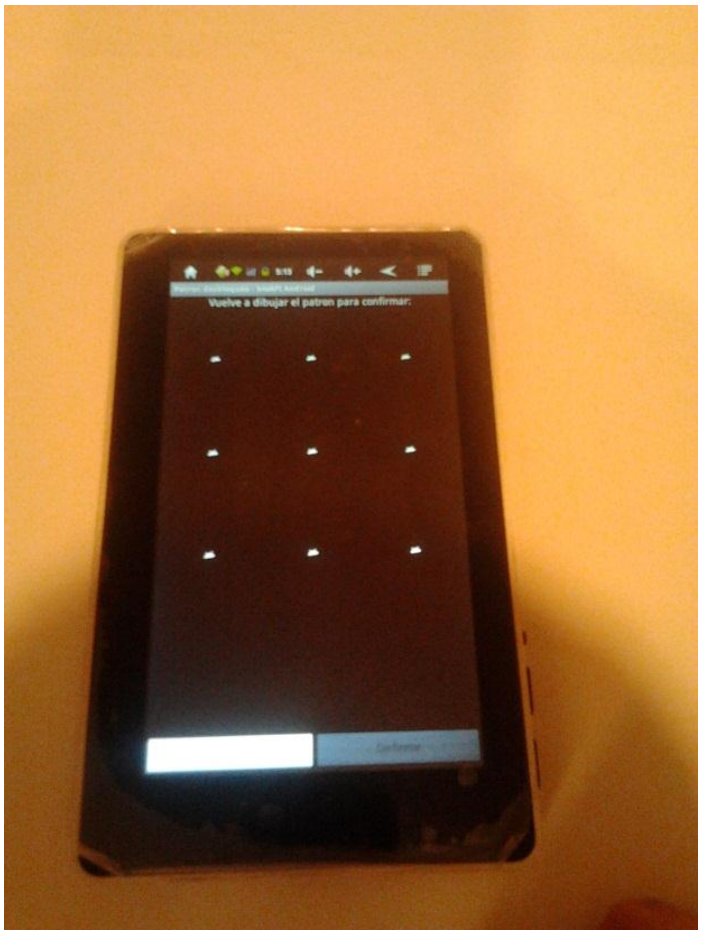

Figura 32: Vista registrarse en dispositivo real

Para el siguiente ejemplo se mostrará cuando el usuario pulse el botón “Guardar Contraseñas” y tras realizar el almacenamiento de unos datos correctamente, se mostrará un diálogo emergente en el que se le informe que ha realizado la operación correctamente:

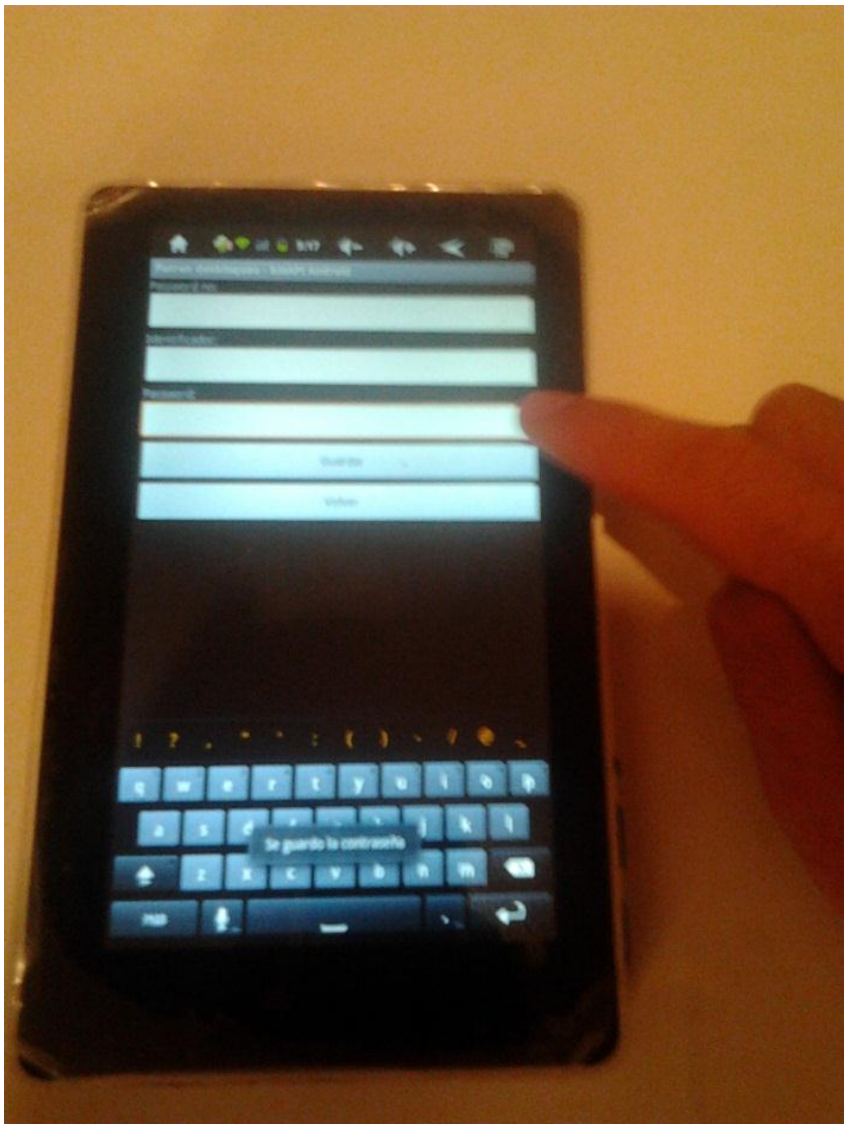

Figura 33: Vista guardar contraseñas en dispositivo real

Una vez almacenados los datos, el usuario podrá volver a la pantalla principal, pulsar el botón “Ver Contraseñas” e introducir el patrón de desbloqueo. Si el patrón introducido es el correcto, se desbloqueará y le dará acceso a la pantalla de consulta de datos. En ella podrá cargar los datos introduciendo el número como se explicó en el punto anterior. En caso de que el numero de contraseña no exista se le indicará al usuario mediante un diálogo emergente como el que se muestra en la siguiente imagen.

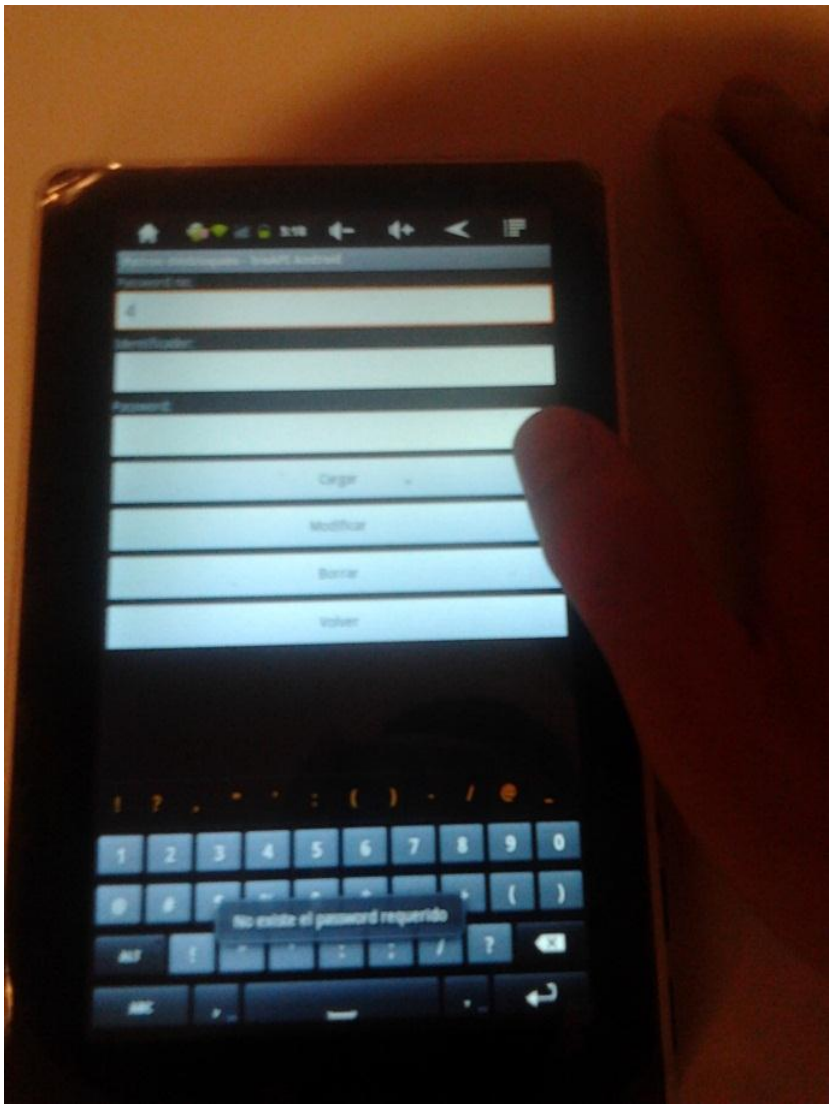

Figura 34: Vista carga errónea en dispositivo real

En el caso de modificar datos, la operación será similar a la del guardado de los mismos, solo que el usuario deberá realizar la modificación exclusivamente en esta pantalla, para evitar el intrusismo en la modificación de datos. El proceso es el mismo que para el guardado, solamente que en este caso, una vez rellenados correctamente todos los campos, el usuario deberá pulsar el botón “Modificar”. En el caso de que no existan datos en esa posición, no habrá datos para modificar y se le notificará al usuario la no existencia de los mismos en tal posición. Con esto se explica, que el usuario no podrá realizar el almacenamiento de datos pulsando el botón de modificar.

Por último, se mostrará el ejemplo por el cual el usuario borra datos, y tras introducir el número de contraseña existente en la lista, y pulsar el botón “Borrar”, se mostrará un mensaje en un diálogo emergente el cual notifique al usuario la correcta eliminación de los datos. Un ejemplo es el siguiente:

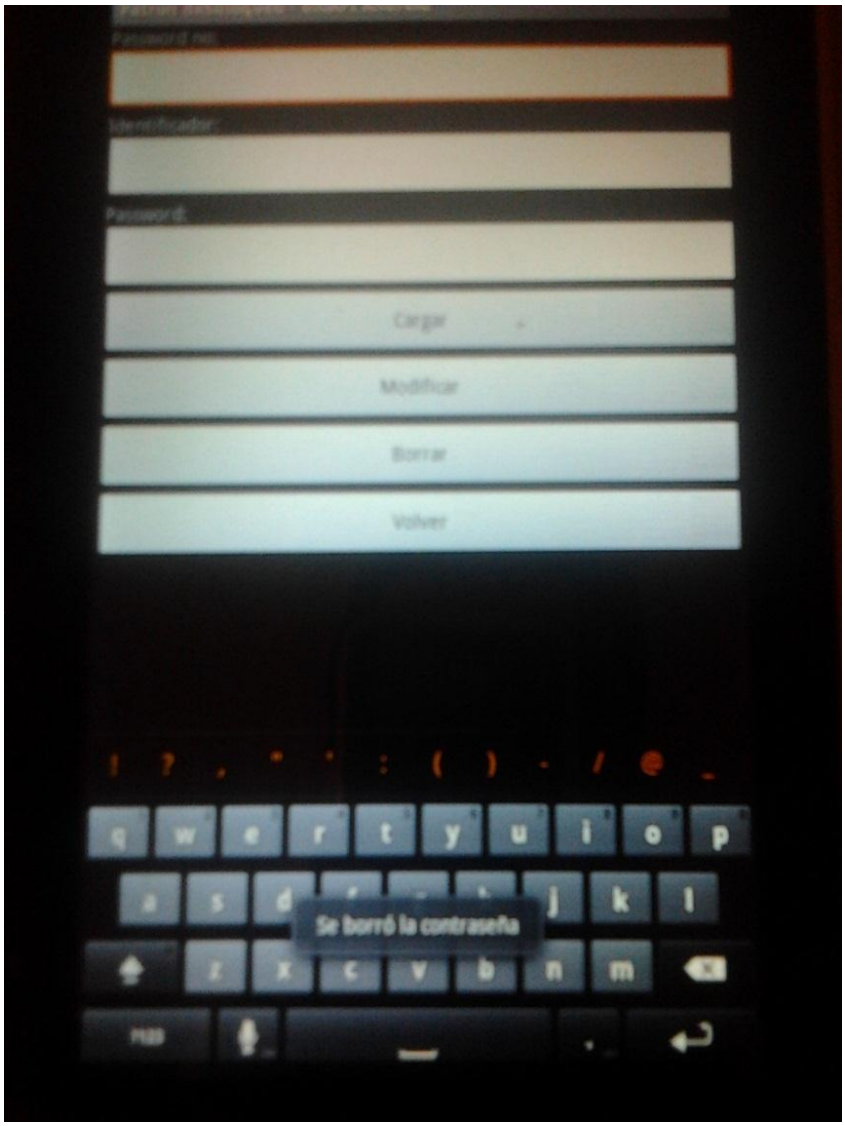

Figura 35: Vista datos borrados

7 Conclusiones y líneas futuras

7.1 Conclusiones

En este trabajo se ha creado una aplicación prototipo de un sistema de almacenamiento seguro de datos mediante identificación biométrica en la plataforma Android. Esta aplicación nos permite almacenar datos confidenciales mediante la limitación al acceso de los mismos con la identificación por parte del usuario con un patrón que solo él conoce o posee. El primer objetivo ha sido la búsqueda de la sencillez pero que al mismo tiempo cumpla las funciones especificadas y sea efectiva. El desarrollo completo de la aplicación se ha llevado a cabo con herramientas de licencia gratuita.

El segundo objetivo, mucho más ambicioso que el primero, es realizar este tipo de aplicación siguiendo los estándares aplicables para el caso en el que en el futuro la autenticación del usuario se hiciese de forma biométrica. Esto impone el uso de BioAPI para el desarrollo de la aplicación. Sin embargo, debido a que se va a trabajar en la plataforma Android, habrá que utilizar una versión de BioAPI en lenguaje Java, la cual, aunque existe una primera versión, no es ni completa, ni detallada, ni correcta. Además la norma que la rige todavía está en desarrollo. Por lo tanto ese segundo objetivo era hacer una implementación de BioAPI Java que funcionase correctamente y especialmente en la plataforma Android.

Los mayores problemas que han surgido a lo largo de la realización del trabajo vienen derivados de la falta de experiencia y conocimiento sobre la plataforma Android, el diseño de las interfaces gráficas mediante lenguaje XML y relacionarlas con los elementos de programación Android, como botones, cuadros de texto y demás, fueron de lo más complicado de lograr. Para el desarrollo de esta aplicación, fueron de gran ayuda los tutoriales existentes en la red, tanto en español como en ingles. Otro de los puntos de mayor dificultad fue la comprensión y adaptación de la norma BioAPI, debido al gran número de funciones que contiene y todas las condiciones que deben cumplirse en cuanto a su código.

Gracias a las decisiones de diseño, como por ejemplo la combinación de actividades y almacenamiento en una base de datos, han dado como resultado una aplicación muy fluida que cumple con los requisitos buscados por el usuario, un almacenamiento de sus datos y restringir su acceso a personas ajenas. Además de cumplir los requerimientos, se ha querido conseguir que la aplicación sea intuitiva para el usuario, con el objetivo de que no exista complejidad en cuanto a su uso.

En cuanto al mercado, existen aplicaciones similares en cuanto al almacenamiento de datos, pero no existe ninguna que implemente un desbloqueo de los mismos mediante el patrón. Sin embargo, en cuanto al desbloqueo por patrón, es más corriente su uso para

desbloquear el uso del teléfono por completo. Con la mezcla de estas dos aplicaciones el resultado obtenido es innovador en cuanto al tipo de aplicaciones.

En conclusión se puede decir que el alumno partía de unos conocimientos básicos en cuanto a la programación de plataformas móviles y tras la realización del trabajo se han logrado el conocimiento y la posesión de las herramientas necesarias para el desarrollo de cualquier idea, poniendo en práctica los mismos métodos utilizados a lo largo de este trabajo cumpliendo de este modo el objetivo principal, el cual no es otro que aprender. Esto no es sólo aplicable al lenguaje de programación Java, sino también al desarrollo de soluciones en la plataforma Android, al uso de interfaces estándar y al desarrollo futuro de aplicaciones biométricas.

7.2 Líneas futuras

Un trabajo como el realizado en este documento es una fuente de posibles líneas de trabajo futuro, tanto en la mejora de la aplicación móvil, como en modificaciones de la identificación de usuario, incluyendo mejoras de rendimiento conjunto.

Como posible mejora para el futuro, se podría indicar en la mejora de los dispositivos móviles mediante la inclusión de un sensor biométrico para la identificación de usuarios, en lugar del método adaptado en este desarrollo mediante el uso de la pantalla táctil.

Otra mejora sería el desarrollo de múltiples aplicaciones utilizando el BSP ya generado, demostrando la interoperabilidad de los BSPs, y por tanto la viabilidad del API definido.

Respecto a las líneas futuras en cuanto a la versión de Android, para esta aplicación se contó con un dispositivo adaptado con una versión 2.3 de Android, pero en el mercado actual, el 90% de los Tablets están adaptados con la versión 3.0 exclusiva para este tipo de dispositivos. Como ya se explicó, la primera versión conjunta para Smartphones y Tablets es la versión 4.0 Ice Cream Sandwich. Esta versión, solo cuenta con una cuota inferior al 20% lo cual hace muy interesante adaptar la aplicación a esta versión.

Bibliografía

[1] Evolución de los terminales móviles. http://es.wikipedia.org/wiki/Historia_del_tel%C3%A9fono_m%C3%B3vil, Julio 2012

[2] Altran Smart society. http://altransmart.wordpress.com/, Agosto 2012

[3] Sistemas operativos móviles. http://www.taringa.net/posts/info/8589949/Sistemas-operativos-moviles-_Comparacion_.html, Julio 2012

[4] Ágora Sic Divulgación. Identificación Biométrica y su unión con las Tarjetas Inteligentes. Volumen 19, Abril 2000.

[5] Ed. Brunette –Programación Android. Anaya Multimedia, 2011.

[6] Breve historia de Android. http://www.elandroidelibre.com/2011/08/la-historia-y-los-comienzos-de-android-el-sistema-operativo-de-google.html, Julio 2012

[7] Primer teléfono Android. http://hothardware.com/newsimages/Item9070/T-Mobile-G1-2.jpg, Julio 2012

[8] Versiones Android. http://es.wikipedia.org/wiki/Android, Julio 2012

[9] Pasado, presente y futuro de Android. http://www.ohmyphone.com/android/sistema-operativo-android/historia-android/, Julio 2012

[10] Android developer. http://developer.android.com, Agosto 2012

[11] Activity|Android developers. http://developer.android.com/reference/android/app/Activity.html, Julio 2012

[12] Norma BioAPI. http://en.wikipedia.org/wiki/BioAPI, Julio 2012

[13] Paul J. Deitel. JAVA: Como programar (Quinta edición). Prentice Hall, 1997

[14] Hai. Bison. [Library] lockpattern activity for Android. http://code.google.com/p/android-lockpattern/ , Abril 2012.

[15] Almacenamiento en una base de datos SQLite. http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=145&inicio=, Agosto 2012.

[16] Programar en Android. http://android-so.com/programar-en-android. Marzo 2012.

[17] Hands-on with the HTC Magic Android phone .http://www.cnet.com.au/hands-on-with-the-htc-magic-android-phone-339295016.htm Agosto 2012.

[18] Seguridad en Android, el patrón de desbloqueo http://www.androidzona.net/seguridad-en-android-el-patron-de-desbloqueo/. Agosto 2012.

[19] Comparación de los sistemas operativos móviles. http://es.engadget.com/2009/03/19/la-gran-comparacion-de-los-sistemas-operativos-moviles/

[20] Android versión history. http://en.wikipedia.org/wiki/Android_version_history Agosto 2012.

ANEXO 1. Presupuesto y planificación del trabajo

A continuación se va a llevar a cabo un desglose de las tareas que se han realizado a lo largo de este trabajo fin de grado, lo que facilitará posteriormente un cálculo aproximado sobre su coste.

Debido a la complejidad de un trabajo de estas características se ha optado por desglosarlo en distintas fases, las cuales se van a comentar a continuación:

Fase 1: Documentación inicial

I. Estudio de la plataforma Android y del entorno de desarrollo (30 horas)
II. Preparación de las herramientas de trabajo (5 horas)
III. Búsqueda y realización de tutoriales y aplicaciones sencillas. (30 horas)
IV. Asistencia a charlas y presentaciones sobre Android (5 horas)

Fase 2: Desarrollo de la aplicación

I. Actividad principal (3 horas)
II. Actividad de registro (80 horas)
III. Actividades secundarias (7 horas)
IV. Interconexión de todas las actividades (30 horas)

Fase 3: Pruebas en un dispositivo real

V. Pruebas de campo en un Samsung Galaxy Ace (20 horas)
VI. Corrección y depuración (20 horas)

Fase 4: Elaboración de la memoria

VII. Redacción de la memoria (85 horas)
VIII. Corrección y maquetación (15 horas)

Tabla 1: Desglose de tareas

FASES	HORAS EMPLEADAS
Documentación inicial	70
Desarrollo de la aplicación	120
Pruebas en un dispositivo real	40
Elaboración de la memoria	100
TOTAL	**310**

PRESUPUESTO DEL TRABAJO FIN DE GRADO

COSTES MATERIALES

Los materiales necesarios han sido un ordenador de altas prestaciones para un correcto funcionamiento del emulador, y un dispositivo Android. Considerando un periodo de amortización de cada uno de los dos dispositivos de 3 años y teniendo en cuenta el tiempo del proyecto, los costes materiales quedan como se expone en la Tabla 2.

Tabla 2: Costes materiales

CONCEPTO	PRECIO (€)
Ordenador de altas prestaciones	200
Samsung Galaxy Ace	12.5
TOTAL	**212.5**

COSTES DE PERSONAL

Para la realización de este trabajo, ha sido necesaria la presencia de un jefe de proyecto y un ingeniero.

Tabla 3: Costes de personal

OCUPACIÓN	HORAS	PRECIO/HORA	IMPORTE (€)
Jefe de proyecto	30	50	1500
Ingeniero	280	30	8400
TOTAL	**310**		**9900**

COSTES TOTALES

Tabla 4: Costes totales

CONCEPTO	PRECIO (€)
Costes de materiales	212.5
Costes de personal	9900
Costes indirectos (20%)	1942.5
Subtotal	12055
I.V.A. (21%)	2531.55
TOTAL	**14586.55**

El coste total del proyecto es de *catorce mil quinientos ochenta y seis euros con cincuenta y cinco céntimos.*

Leganés, 5 de septiembre de 2012

El ingeniero