Universidad
Carlos III de Madrid

SOLUCIONES MATCH-ON-DEVICE Y PROCESSING-ON-DEVICE EN SISTEMAS MÓVILES

GRADO EN INGENIERÍA TELEMÁTICA

AUTOR: Raúl Serna Marín

TUTOR: Raúl Sánchez Reíllo

Leganés, 20 de junio de 2013

Título: Soluciones Match-On-Device y Processing-On-Device en sistemas móviles

Autor: Raúl Serna Marín

Tutor: Raúl Sánchez Reíllo

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día __ de Julio de 2013 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

Quiero empezar agradeciendo el enorme apoyo que me han prestado mis padres Ángela y Juan Antonio a lo largo de este trabajo y en definitiva a lo largo de toda la carrera. Siempre han estado allí cuando lo necesitaba dándome grandes consejos. Gracias.

También he de dar las gracias a mis amigos, ellos saben quién son, por soportarme como soy y apoyarme continuamente pese a mi constante indisponibilidad.

No me voy a olvidar de mi tutor Raúl Sánchez Reíllo, por su gran carácter y motivación, y también por permitirme alcanzar mi sueño. Esto tampoco hubiera sido posible sin la ayuda constante de Judith, quien, junto a mi tutor, me ha guiado de forma incansable a lo largo de la realización de este TFG. Muchas gracias a los dos.

Por último, quiero dar las gracias a un nuevo grupo de amigos surgido durante la realización del TFG: los gutitos. Gracias por esos tutoriales y esos partidos de baloncesto que me relajaban y me ayudaban a proseguir. Y gracias también por el constante ánimo y buen ambiente existente en el grupo, que hacían que los días fueran un poco mejores.

RESUMEN

La identificación biométrica está comenzando a ganarse un hueco en nuestras vidas. Cada vez son más las instituciones donde se autentica a las personas utilizando dispositivos biométricos. Su excelente tasa de reconocimiento, unido a la comodidad que ofrece al usuario, junto a su dificultad de suplantación de identidad, hacen de la identificación biométrica una opción muy atractiva a la hora de reconocer personas en el ámbito de la seguridad. Sin embargo, dado el gran número de fabricantes que existen, es necesario definir un estándar para poder compatibilizar todas las tecnologías en una sola. Nace así el estándar BioAPI.

Por otro lado, el mercado actual de telefonía se ha incrementado vertiginosamente con la llegada de los teléfonos móviles conocidos como “smartphones”. Estos dispositivos se han convertido en herramientas indispensables para la población, haciendo que la mayoría de personas los lleven siempre consigo. Además, posibilitan la realización de operaciones que antes sólo los ordenadores eran capaces de ejecutar.

Uniendo estos puntos anteriores, surge este Trabajo de Fin de Grado, cuyo objetivo es comprobar las propuestas al nuevo estándar BioAPI. Para verificar la compatibilidad que ofrece estas nuevas propuestas, se implementará un sistema basado en reconocimiento por huellas dactilares, formado por un ordenador y un teléfono móvil, con el requisito de que las funcionalidades biométricas se ejecuten en el dispositivo móvil.

ABSTRACT

Biometric identification is starting to make itself a space in our lives. Every day there are more institutions in which people authentication is performed using biometric devices. Its excellent recognition rates, besides the comfort that offers to the user, and its strength against identity theft, make the biometric identification a truly attractive option when comes to recognizing people in the security scope. However, due to the great number of manufacturers in the market, it is necessary to define a standard in order to make all different technologies compatible. In this context, BioAPI standard arises.

On the other hand, the current telephony market has experienced a vertiginous increase with the arrival of the cell phones known as “smartphones”. These devices have become indispensable tools for most of the population, provoking that most of the people carry one with them. Furthermore, they make possible to perform operations that before could only computers can carried out.

If we join up the above mentioned points, this Bachelor’s Thesis emerges, whose main objective is to check the new standard BioAPI’s approaches. To verify the compatibility of these approaches, we will implement a fingerprint recognition system, formed by a personal computer and mobile device, with the requirement of running the biometric functionalities on the mobile device.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

ADT: Android Development Tools

API: Application Programming Interface

BDB: Biometric Data Block

BioAPI: Biometric Application Programming Interface

BIR: Biometric Information Record

BSP: Biometric Service Provider

CBEFF: Common Biometric Exchange Formats Framework

DB: Data Base

FBI: Federal Bureau of Investigation

GB: Giga Bytes

GUI: Graphical User Interface

IDE: Integrated Development Environment

JVM: Java Virtual Machine

MAC: Message Authentication Code

MB: Mega Bytes

MINDTCT: MINutiae DeTeCT

NIST: National Institute of Standards and Technology

OO: Object-Oriented

PC: Personal Computer

RAM: Random Access Memory

SB: Security Block

SBH: Standard Biometric Header

SD: Security Digital

SGL: Scene Graph Library

SPI: Service Provider Interface

SSL: Secure Sockets Layer

TCP: Transmission Control Protocol

TFG: Trabajo de Fin de Grado

UDP: User Datagram Protocol

USB: Universal Serial Bus

1. INTRODUCCIÓN

En este documento se presentará una implementación de la propuesta 4.0. del estándar BioAPI. Para poder comprobar su correcto funcionamiento, se creará un sistema biométrico basado en huellas dactilares compuesto de un ordenador trabajando sobre C# y de un teléfono móvil Android, donde se realizarán la mayoría de funcionalidades. Se establecerá una forma de comunicación entre ambos dispositivos para posibilitar el intercambio de información y verificar la estabilidad del sistema.

1.1. MOTIVACIÓN

La motivación principal es la posibilidad de trabajar con la plataforma Android en dispositivos móviles. Con el auge y la cada vez mayor importancia que esta plataforma está ganando en el mercado, su aprendizaje ofrecerá al alumno conocimientos que serán de mucha utilidad en un futuro próximo, cuando se encuentre desarrollando su carrera profesional. Dado que está habiendo un crecimiento exponencial del número de aplicaciones Android, la demanda de desarrolladores aumenta vertiginosamente por lo que aprender a utilizar este sistema operativo puede ser de gran utilidad.

Por otro lado, las tecnologías biométricas también suscitan un gran interés en el alumno. Esto es debido principalmente a que la biometría es una ciencia que se encuentra en pleno desarrollo en la actualidad. Cada vez son más los sensores de huellas dactilares o herramientas de reconocimiento facial que se pueden encontrar en aeropuertos, instituciones públicas e instituciones privadas. No es de extrañar que alguien se sienta atraído hacia este sector en auge, posibilitando con su aprendizaje la posibilidad de continuar sus estudios en un ámbito similar.

Por último, aunque no menos importante, la idea de ayudar a definir y probar una nueva versión de una norma biométrica dentro de un grupo de investigación resulta una gran motivación. Aprender cómo funcionan los comités de estandarización, trabajar en conjunto con otros desarrolladores dentro de un mismo proyecto, analizar las diferentes versiones por las que pasa un documento y observar cómo este se desarrolla desde un prototipo inicial hasta una versión final constituyen una fuerte atracción para el autor de este TFG.

1.2. OBJETIVOS

El objetivo principal de este Trabajo de Fin de Grado es comprobar el funcionamiento de la propuesta 4.0 del estándar BioAPI sobre dispositivos móviles. Este estándar define una serie de interfaces que permiten la comunicación entre las aplicaciones del nivel superior y los dispositivos biométricos en un sistema. La versión actual aceptada de esta norma es la 3.0. Sin embargo, el grupo de investigación donde se ha realizado este TFG ha decidido actualizarla debido a que posee demasiadas incoherencias. De esta forma, el alumno debe ayudar a corregir los principales errores y fallos de esta nueva beta de la norma, así como verificar que funciona correctamente en un sistema real.

Este sistema estará formado por dos dispositivos diferentes, con el fin de constatar implícitamente que la compatibilidad entre ellos es idónea. Se desarrollarán las funcionalidades biométricas de procesado y comparación de huellas dactilares sobre el

dispositivo móvil, dejando al ordenador realizar la captura de huellas. Además, se deberá prestar atención a la interoperabilidad entre los diferentes módulos que propone este estándar.

Para poder confirmar que la norma se implementa correctamente, se necesitará desarrollar una aplicación que pueda interaccionar con el usuario. Además, se generará una comunicación fiable entre ambas máquinas que permita la transmisión de datos biométricos.

1.3. ESTRUCTURA DEL DOCUMENTO

Este documento está formado por dos capítulos iniciales donde se analizan las tecnologías utilizadas para desarrollar este TFG. A continuación se estudiarán las herramientas de desarrollo necesarias para llevar a cabo este proyecto: la plataforma Android y el lenguaje de programación C#.

Tras esto, se procederá 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.

Por último, se comprobará el correcto funcionamiento del sistema completo para acabar finalizando con una breve conclusión y las líneas futuras que se derivan de este Trabajo de Fin de Grado.

2. RECONOCIMIENTO DE HUELLAS DACTILARES

Dado a que el sistema biométrico a implementar está basado en el reconocimiento de huellas dactilares, es necesario preparar al lector en este campo antes de introducirse en el diseño y desarrollo de la aplicación. Por lo tanto, en este capítulo, se analizará esta modalidad biométrica, comenzando con una breve introducción a la biometría para centrarse en las características de las huellas dactilares, así como en los diferentes tipos de sensores existentes en el mercado. Finalmente, se analizarán los diferentes algoritmos usados en el procesado y comparación de las imágenes capturadas.

2.1. INTRODUCCIÓN A LA BIOMETRÍA

La biometría es la ciencia que clasifica, registra e identifica a las personas mediante sus características físicas y/o de comportamiento. La identificación biométrica es una tecnología nueva, todavía en desarrollo que no ha alcanzado su madurez. Pese a ello, cada vez más gobiernos y empresas están comenzando a utilizarlo, ya que mejoran notablemente los sistemas de seguridad. A lo largo de este apartado describiremos en que consiste la biometría, recorreremos su historia a lo largo de los años para terminar analizando su funcionamiento, técnicas y métodos más importantes:

2.1.1. ¿Qué es la biometría?

Como se ha comentado en el párrafo anterior, la biometría es el estudio de métodos de reconocimiento basados en la extracción de alguna de las características propias del ser humano, ya sean físicas (huellas dactilares, el iris del ojo, los patrones faciales) o de comportamiento (el paso al andar, el tecleo en un ordenador o la firma). Estas características intrínsecas reciben el nombre de *identificadores biométricos*.

La biometría se ha adaptado a la tecnología de información, creándose así los términos *autenticación e identificación biométrica* en el ámbito de la seguridad. Se define como el uso de diferentes algoritmos matemáticos y estadísticos con el objetivo de verificar o identificar correctamente a un usuario.

Para que un sistema biométrico sea clasificado como tal, tanto los indicadores biométricos como el propio sistema deben cumplir una serie de requisitos [1]:

- *Universalidad*: las características se deben poder extraer de cualquier usuario.
- *Unicidad*: no pueden existir dos sujetos con las mismas características.
- *Estabilidad*: las características deben permanecer constantes frente a determinados factores (tiempo, enfermedades, etc).
- *Facilidad de captura*: deben existir métodos que capturen la característica de forma sencilla.
- *Rendimiento*: el porcentaje de reconocimiento válido debe ser lo suficientemente alto.
- *Aceptabilidad de los usuarios:* sistemas de difícil usabilidad o dolorosos no serán aceptados.
- *Robustez frente al fraude:* el sistema debe ser capaz de resistir un intento de burla del sistema (huellas de látex, grabaciones de voz, etc.).

Los sistemas biométricos se componen de tres elementos: los dispositivos de captación de características, el software o algoritmos que interpreta estos indicadores y los transforman a una secuencia numérica y las bases de datos que almacenan estas secuencias.

2.1.2. Historia de la biometría

Aunque la biometría está en pleno auge actualmente, el reconocimiento de personas mediante sus características personales lleva haciéndose muchísimos años atrás. Muchas son las referencias de personas, que en la antigüedad, han sido identificados por diversas características físicas y morfológicas como cicatrices, medidas, color de los ojos, etc. Una característica física muy utilizada para el reconocimiento de una persona es su rostro. Desde épocas muy remotas hasta la actualidad, los seres humanos hemos utilizado las características faciales para distinguir a una persona determinada de las demás.

Dejando a un lado esta habilidad innata, la biometría como ciencia de estudio de métodos de reconocimiento humano nace en el siglo XIX en Europa. Es un policía francés quien, con el objetivo de identificar a criminales reincidentes, decide realizar numerosas medidas sobre partes del cuello y la cabeza, cicatrices, tatuajes y características individuales del sospechoso. Nace así el sistema de identificación Antropométrica de Bertillon en 1982, primer sistema biométrico propiamente dicho.

Una década más tarde, en 1892, un antropólogo inglés llamado Francis Galton (primo de Charles Darwin) realizó las primeras pruebas satisfactorias de identificación de criminales mediante la huella dactilar. Este método fue imponiéndose poco a poco. Pese a que el método identificativo de Bertillon había producido grandes mejoras en la detención de criminales, este sistema era difícil y tedioso de medir, por lo que diferentes oficiales solían conseguir diferentes medidas del mismo criminal. Además, ocurrió que dos personas llegaron a tener el mismo conjunto de medidas. En consecuencia, este sistema fue desbancado por la identificación por huellas dactilares y sigue en cabeza en la actualidad [2] [3].

2.1.3. Etapas de la identificación biométrica

Aun existiendo varios métodos de identificación biométrica, el flujo de funcionamiento de un sistema es siempre el mismo. De esta forma, tal como muestra la Figura 1, un sistema consta de dos fases totalmente diferenciadas: la fase de inscripción o reclutamiento y la fase de verificación. Ningún sistema será biométrico si le falta alguna de las dos etapas.

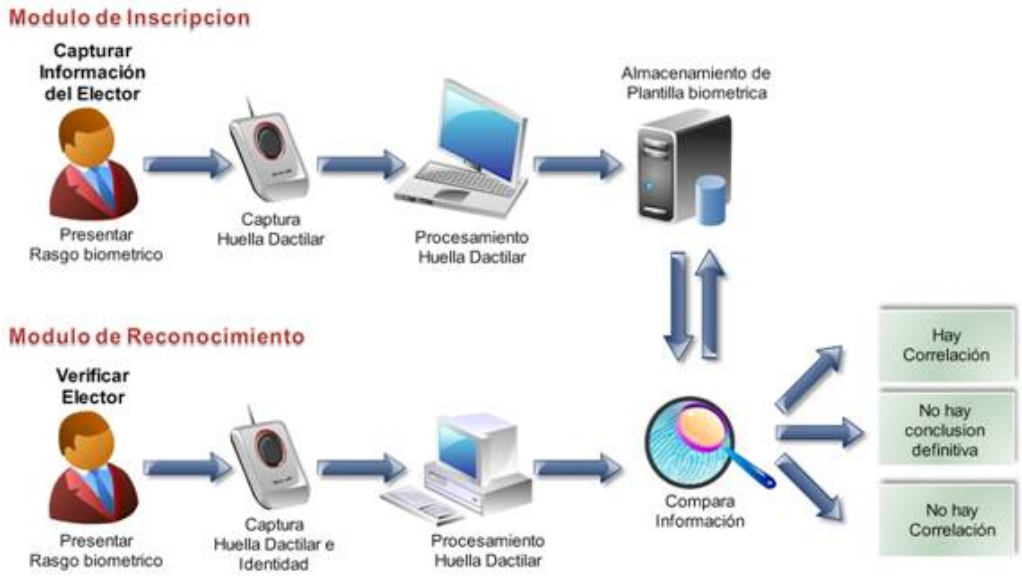

Figura 1: Etapas de la identificación biométrica [4]

Reclutamiento

En esta fase, se toman una o más muestras del usuario mediante un dispositivo de captura, se procesan mediante un algoritmo que extrae sus características y genera un patrón que se almacenará en una base de datos. Este proceso se hace de forma supervisada, es decir, debe haber una persona encargada de controlar que la captura de datos es correcta además de validar la identidad del usuario. Usualmente también se suele enseñar al usuario el funcionamiento y solucionar todas las dudas posibles.

Verificación

En esta fase el usuario introduce de nuevo la característica a comparar y del mismo modo que en la fase anterior, las muestras son procesadas. A la salida del procesamiento, la muestra procesada se compara con la almacenada en la fase de reclutamiento devolviendo un resultado positivo (el usuario fue verificado) o falso (el sistema rechazó al usuario).

Para la comparación se utiliza un umbral de semejanza, si la muestra introducida lo supera, el usuario es validado y si se queda por debajo, el usuario es rechazado. Cómo establecer este umbral es por consiguiente muy importante.

El detalle con el que se comparan las muestras también depende del tamaño del patrón almacenado. Es decir, dependiendo del tamaño de la base de datos del sistema, se crea un patrón u otro. Así, si tenemos una base de datos de gran capacidad (varios PCs, servidores), el patrón contiene muchas características por lo que la comparación es más minuciosa. Si por el otro lado, tenemos una base de datos más ligera (un móvil), el patrón tiene en cuenta un número menor de datos y la comparación no será tan precisa.

En la fase de verificación, los sistemas biométricos siempre ofrecen dos funcionalidades:

- **Reconocimiento:** el objetivo de esta funcionalidad es identificar a un usuario dentro de la base de datos donde se encuentran todos los usuarios del sistema. Es una identificación *uno-a-muchos*. Esta modalidad se caracteriza por la necesidad de comparar el usuario con todos los patrones de la base de datos.

- **Autenticación:** a diferencia de la funcionalidad anterior, aquí el sistema sólo está interesado en verificar si el usuario es realmente quien dice ser. Es una identificación uno-a-uno.

2.1.4. Modalidades biométricas

Como hemos venido comentando a lo largo del capítulo, existen diferentes técnicas de identificación del ser humano. Actualmente se encuentran en desarrollo muchas otras. Entre las más importantes destacan las huellas dactilares, el reconocimiento por iris, el reconocimiento facial, la autenticación de la persona mediante el patrón de sus venas, la geometría de la mano, el reconocimiento de voz y la firma manuscrita.

La elección de una modalidad u otra depende de la ocasión: en una situación nos interesará tener resultados altamente válidos mientras que en otro entorno diferente lo que se busque sea la aceptabilidad por parte del usuario y rapidez de captura.

En resumen, la identificación biométrica es una ciencia que está aún en desarrollo pero que está comenzando a ganarse un puesto en el ámbito de la autenticación y seguridad. Cuenta con numerosas y variadas técnicas de reconocimiento que van desde las huellas dactilares hasta la firma manuscrita. No existe una modalidad perfecta si no que hay que tomar la decisión de cual usar dependiendo de los factores del entorno.

En este trabajo de fin de grado, hemos realizado una implementación biométrica usando huellas dactilares ya que es la modalidad más empleada en el mundo además de que ofrece unos resultados muy buenos. Es por ello que en las siguientes líneas nos centraremos en detalle en esta modalidad con el objetivo de preparar al lector para el diseño y desarrollo de nuestra aplicación.

2.2. LA HUELLA DACTILAR

En este apartado se describirá en detalle en que consiste una huella dactilar. Una huella dactilar es una representación de la piel del dedo, formada por una sucesión *crestas* (elevaciones de piel) separadas entre sí por *valles* (espacios entre crestas). Las crestas sufren una serie de puntos singulares, tales como *terminaciones* y/o *bifurcaciones*, que son las que hacen que sean representativas de cada usuario. Una terminación es donde las líneas terminan abruptamente mientras que una bifurcación es donde se abren, tal como muestra la Figura 2. A estos puntos singulares se les denomina *minucias*, y mediante su tipo, localización y orientación, se puede llegar a identificar a una persona. [5] [6]

Figura 2: Representación de una huella dactilar [7]

Encontrar más de dos personas en la Tierra que poseen un número alto de minucias del mismo tipo, tamaño, ubicación y orientación en el dedo es prácticamente imposible. Por otro lado, las minucias se encuentran en capas inferiores a la piel por lo que cuando esta se daña levemente (una herida por ejemplo) no se ven afectadas y aunque lo hicieran, poseen una gran capacidad regenerativa gracias a la que obtienen sus antiguas características en poco tiempo.

Estas minucias suelen ser suficientes para identificar a un ser humano. Sin embargo también existen otros puntos singulares que sirven para el reconocimiento dactilar: el *core* y el *delta*. El core es el punto de máxima curvatura de las crestas mientras que la delta es el lugar donde las crestas forman triángulos concéntricos (ver siguiente Figura 3).

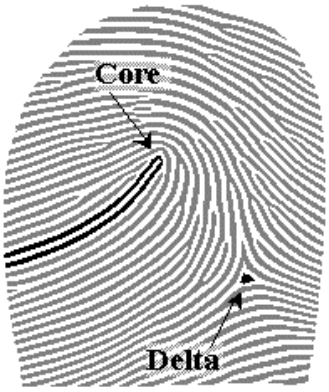

Figura 3: Representación del core y el delta [8]

Existe por último una clasificación de las huellas dactilares para ayudar a reducir el espacio de búsqueda de identidad. Estos tres tipos de huellas son: [5]

- **En arco:** este patrón es el menos común de los tres, se da en menos del 5% de la población mundial. En este patrón, las crestas se asemejan a una ola del mar: entran por un lado del dedo, se levantan hacia arriba formando un leve arco en la mitad de la huella y salen de nuevo por el otro lado
- **En espiral:** se encuentra en un 25% de la población. Como su propio nombre indica, las crestas forman una espiral alrededor del centro del dedo.
- **En bucle:** es el tipo de patrón más común ya que se encuentra en un 70 % de la población. En este caso, las crestas entran por un lado del dedo, dan una vuelta

completa y salen por el mismo lado, a diferencia del patrón en arco que salía por el otro.

Figura 4: Tres tipos de patrones existentes [9] [10] [11]

2.3. DISPOSITIVOS DE IDENTIFICACIÓN BIOMÉTRICA

Existen diferentes dispositivos usados para identificar al ser humano mediante su huella dactilar. Su función es capturar la imagen de la huella para aplicar los algoritmos que procesen esas digitalizaciones. Actualmente se encuentra en desarrollo un gran número de sensores. La gran mayoría de dispositivos se basan en obtener las crestas y valles de la huella mediante las diferencias existentes entre ellas. A continuación se muestran los tipos de sensores más importantes: [12]

- **Sensor óptico**, que usa luz visible para capturar la huella y digitalizarla. El sensor óptico emite luz, la cual es reflejada por las crestas y absorbida por los valles, capturándose así una imagen en el sensor.
- **Sensor capacitivo**. Se basa, como su propio nombre indica, en los principios de capacitancia que actúan sobre las diferentes capas cutáneas.
- **Sensor por presión.** Se apoya en el flujo de corriente que recibe del dedo cuando éste hace una presión sobre el sensor. Las crestas ofrecen más presión mientras que los valles menos, por lo que se inyectan diferentes corrientes que el sensor detectará, obteniéndose la imagen deseada.
- **Sensores ultrasónicos**. Tienen un funcionamiento similar al de los murciélagos: envían ultrasonidos al dedo y analizan los ecos que devuelven las crestas y los valles.
- **Sensores térmicos**, basados en las diferencias de temperatura entre las crestas y los valles.

En la actualidad, la tecnología ultrasónica se considera la más precisa de las tecnologías dactilares. Sin embargo, son dispositivos de gran tamaño y coste, por lo que se suele optar por sensores ópticos.

2.4. ALGORITMOS DE IDENTIFICACIÓN

En esta sección se discuten los algoritmos utilizados para el procesado de las imágenes así como su posterior comparación con otras huellas con el objetivo de identificar al usuario. Se emplearán dos algoritmos: *MINDTCT* [13] y *BOZORTH3* [14]. El primero se encarga de obtener las minucias mediante un pre-procesado de la imagen mientras que el segundo es el algoritmo encargado de comparar dos conjuntos de minucias retornando un valor que indica la similitud entre las huellas a comparar.

2.4.1. MINDTCT

Es un algoritmo que, como su propio nombre indica (**MIN**utiae **D**e**T**e**CT**), se encarga de detectar las minucias de una huella dactilar. Este extractor de patrones fue creado por el Instituto Nacional de Estándares y Tecnologías (NIST) por orden del FBI (Federal Bureau of Investigation) con el objetivo de desarrollar un algoritmo de procesado rápido y preciso de imágenes [15].

El algoritmo MINDTCT sigue una serie de etapas desde que recibe la imagen de la huella dactilar del sensor hasta que genera el archivo de salida de minucias. Estas etapas se pueden ver en la siguiente Figura 5 y serán analizadas a lo largo de esta sección [13]:

Figura 5: Diagrama de flujo del pre-procesado y extracción de minucias

PASO 1: Imagen de la huella de entrada:

En el primer paso del proceso, el algoritmo recibe una imagen del sensor biométrico, la convierte a escala de grises para trabajar con ella.

PASO 2: Generación de mapas de calidad de la imagen

Debido a que la calidad de la imagen de una huella dactilar puede variar, es necesario ser capaz de analizar la imagen y determinar las áreas que están degradadas. Estas áreas más degradadas presentarán dificultades al algoritmo para encontrar minucias y pueden dar lugar a fallos en el algoritmo. Por esta razón se debe analizar la calidad de la imagen.

Para analizar la calidad de la imagen existen diferentes métodos: determinar la dirección del flujo de las crestas de la imagen, detectar las regiones de bajo contraste, las de flujo de crestas bajo y las de alta curvatura. Todas estas características en conjunto ayudan a representar los niveles de calidad de la imagen. A continuación se verán con más detalle:

- **Mapa de orientación**: El propósito de este mapa es representar las áreas de la imagen con suficiente estructura de crestas. Las crestas bien formadas y claramente visibles son esenciales para detectar de forma fiable los puntos singulares (terminaciones y bifurcaciones). Ver Figura 6.

Imagen de entrada de la huella Salida del mapa de direcciones

Figura 6: Entrada y salida del mapa de direcciones [13]

- **Mapa de bajo contraste:** En las imágenes suelen existir regiones de bajo contraste, donde resulta muy difícil, si no imposible, determinar exactamente los flujos de las crestas. Este proceso se encarga de separar o diferenciar los bloques de bajo contraste de los que poseen unas crestas bien definidas para favorecer la localización de minucias.

Figura 7: Resultado del mapa de bajo contraste [13]

- **Mapa de bajo flujo:** Durante la generación del mapa de orientación puede ocurrir que algunos bloques no tengan un flujo de crestas dominante, lo cual suele relacionarse con zonas de baja calidad. En este punto se marcan las zonas de flujo no dominante, para señalar que las minucias encontradas en ellos son poco fiables. En la Figura 8, a los bloques de bajo flujo se les asigna una cruz blanca.

Figura 8: Resultados del mapa de bajo flujo [13]

- **Mapa de curvatura:** Otra parte problemática se encuentra en las áreas de alta curvatura. Estas áreas se encuentran principalmente en las regiones del *core* y *delta* de la huella. El *mapa de curvatura* se encarga de marcar estos bloques. La calidad de las minucias detectadas en estos bloques se reduce debido a que también son una parte poco fiable de la imagen.

Figura 9: Resultados del mapa de alta curvatura [13]

Mapa de calidad: La información de los cuatro mapas anteriores se integra en uno solo, el *mapa de calidad*, el cual contiene 5 niveles de calidad. Estos niveles de calidad, que son asignados a cada bloque, se determinan basándose en su proximidad a los demás bloques (ver Figura 10)

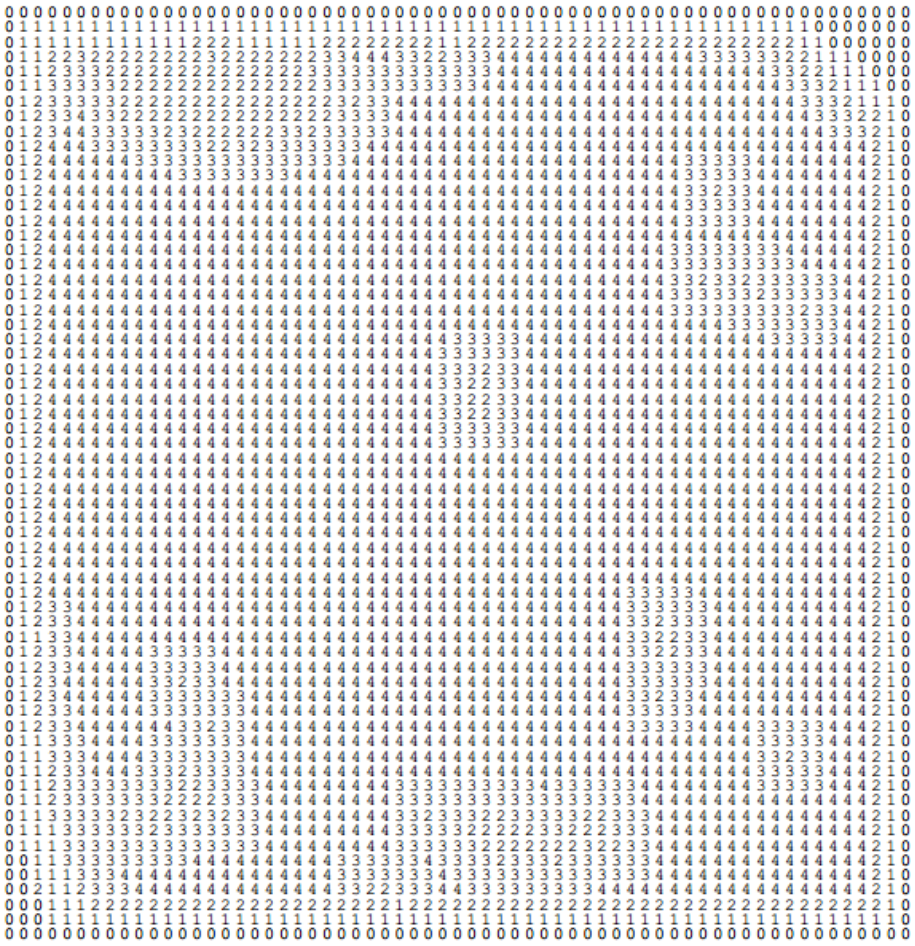

Figura 10: Resultado final del mapa de calidad [13]

PASO 3: Binarización de la imagen

El algoritmo para detectar las minucias necesita trabajar con imágenes binarias donde los píxeles blancos representan los valles y los negros las crestas. Por tanto, este proceso se encarga de generar una imagen binaria a partir de una imagen en escala de grises.

Para realizar esta binarización, el algoritmo se basa en la orientación del flujo de crestas. El algoritmo se centra en un pixel y si no se detecta ningún flujo en él, al pixel se le asigna el color blanco. Si por el contrario se encuentra en un flujo, se le asigna el color negro. El resultado se puede observar en la Figura 11.

Figura 11: Binarización de la huella dactilar [13]

La binarización de la imagen es un paso crítico para la exitosa detección de minucias en este algoritmo. Los resultados de la binarización necesitan ser *robustos* y *fiables*. Es necesario preservar tanta información de la imagen como sea posible para una precisa detección de minucias. Sin embargo, preservando información también preservamos las áreas degradadas y dañadas, lo que es indeseable. Por tanto, se debe buscar un equilibrio entre ambas partes.

PASO 4: Detección de minucias

Como se ha comentado a lo largo de este capítulo, las minucias son los puntos singulares donde las crestas acaban o se bifurcan. En este paso, el algoritmo escanea la imagen binaria de la huella comparando cada bloque de 6 píxeles con los patrones que utiliza y posee el algoritmo. Un ejemplo de estos patrones se encuentra en la Figura 12. De esta forma, si la comparación es positiva, quiere decir que se ha encontrado una terminación o una bifurcación.

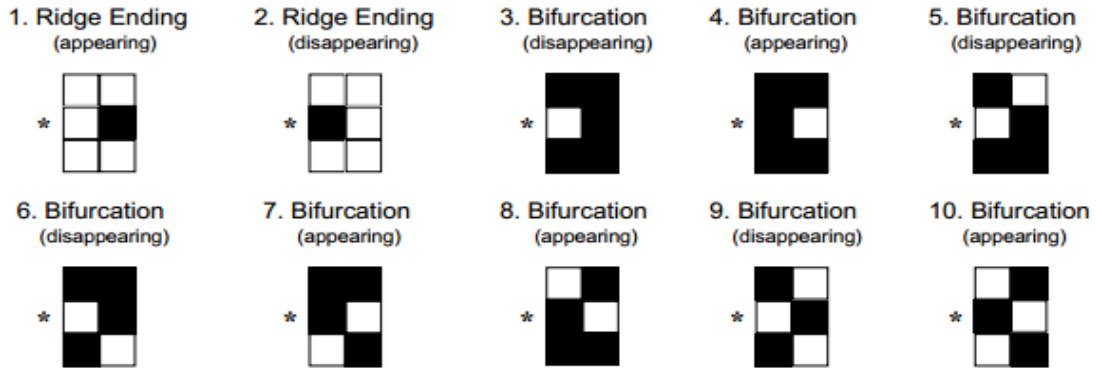

Figura 12: Patrones de píxeles utilizados para detectar minucias [13]

PASO 5: Eliminación de falsas minucias

Gracias al PASO 4: Detección de minucias, las minucias candidatas son detectadas mediante simples patrones de 6 píxeles por lo que tal grado de detección minimiza la posibilidad de perder minucias potenciales. Sin embargo, tiene como desventaja que también se detectan muchas falsas minucias. Este paso se encarga de intentar eliminarlas.

PASO 6: Recuento de crestas vecinas

Es muy frecuente que los algoritmos comparadores (BOZORTH3, por ejemplo) necesiten información sobre las minucias vecinas de la minucia que analizan en un momento dado. En este paso se efectúa el recuento de las crestas que hay entre una minucia y sus vecinas.

PASO 7: Evaluación de la calidad de la minucia

En este paso el objetivo es evaluar la calidad de cada minucia detectada. Partiendo del hecho de que las falsas minucias deberían estar asignadas con una calidad más baja, una robusta medida de calidad podría ayudar a desechar las restantes. En este paso se computan estas calidades aplicando una serie de algoritmos matemáticos y se obtienen unos valores de calidad en un rango de 0.01 a 0.99, donde un valor bajo representa una minucia detectada en una región de baja calidad, mientras que un valor alto, lo contrario.

PASO 8: Archivo de salida de las minucias

En este último paso, el algoritmo MINDTCT recoge las minucias resultantes de la huella, mostradas en la Figura 13, y las guarda en un archivo de texto. La información resultante asociada a cada minucia es su posición X, su posición Y, el ángulo de dirección y la calidad de la minucia.

Figura 13: Minucias resultants [13]

2.4.2. BOZORTH3

BOZORTH3 es un algoritmo *comparador*, es decir, se encarga de comparar diferentes huellas con el objetivo de buscar casos coincidentes. Fue desarrollado por Allan S. Bozorth por orden del FBI aunque el NIST lo cogió más tarde y lo mejoró notablemente.

Como se ha comentado, el algoritmo MINDTCT se encarga de extraer las minucias de una imagen de huella dactilar. El algoritmo BOZORTH3 recibirá estas minucias y las comparará con otras muestras de huellas dactilares. Este comparador sólo necesita la posición X, la posición Y, la orientación y calidad de cada minucia. Una de las características más importantes de este algoritmo es que es invariante a la traslación y rotación de la huella.

Al igual que MINDTCT, este algoritmo se compone de tres etapas importantes, tal y como muestra la siguiente Figura 14 y que explicaremos a continuación [14]:

Figura 14: Diagrama de flujo del algoritmo BOZORTH3

PASO 1: Construcción de las tablas de comparación de minucias

En este primer paso, el objetivo es construir una tabla para la huella tomada al usuario y otra para la huella de la base de datos con la que se va a comparar.

BOZORTH3 se encarga de calcular las medidas relativas de cada minucia de una huella con las otras minucias de la misma huella. Estas medidas relativas se guardan en una *tabla de comparación de minucias*.

Las medidas tomadas consisten en la distancia entre la minucia que nos interesa y las demás, representadas en la Figura 15 como $d(P_m)$. También se calcula el ángulo entre la orientación de la minucia (flecha negra) y la línea que une ambas minucias. Estos ángulos están representados como $\beta_1\ (P_m)$ y $\beta_2(P_m)$. El objetivo de la medición de estas medidas es que el algoritmo permanezca invariante a la traslación de la imagen (gracias a la distancia d) y a la orientación (gracias a β_1 y β_2).

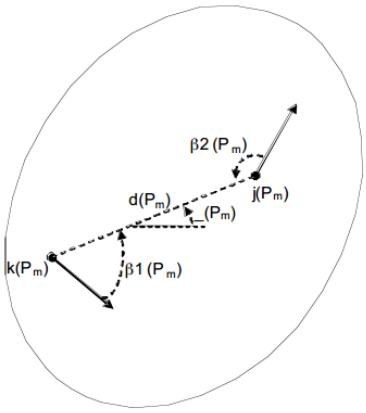

Figura 15: Comparación de minucias de huella dactilar [14]

Cada dos minucias comparadas, se introduce en la tabla una nueva entrada que tendrá la distancia entre las minucias, los ángulos comentados en el párrafo anterior así como información propia de la minucia. Cada minucia es una fila y cada huella dactilar una tabla, por lo que tendremos tantas tablas como huellas queramos comparar.

PASO 2: Construcción de la tabla de compatibilidad

El siguiente paso del algoritmo consiste en buscar entradas compatibles entre las tablas de comparación de dos diferentes huellas. Para encontrar esta compatibilidad se realizan 3 test. El primero comprueba si ambas distancias tienen una tolerancia específica. Los 2 otros test se corresponden con los dos ángulos y comprueban que estén por debajo de una tolerancia o límite. Estos niveles de tolerancia se deben a la elasticidad de la piel. Si el resultado es positivo, se inserta en la tabla de compatibilidad una nueva entrada consistente en dos pares de minucias, un par de la huella introducida y otro de la galería.

PASO 3: Recorrido de la tabla de compatibilidad

Al alcanzar este último paso, tenemos una tabla de compatibilidad completamente construida que consiste en una lista de asociaciones compatibles entre dos huellas. Para saber si realmente coinciden, se construye un *grafo de compatibilidad* con estas asociaciones. Así, recorriendo el grafo en busca del camino más largo de asociaciones compatibles. Se obtendría puntuación, dando al camino más largo la mayor, y al más corto, la menor. Cuanto mayor sea el número de enlaces compatibles, mayor será su puntuación y por tanto, mayor será la probabilidad de que las dos huellas sean de la misma persona.

Al terminar, el algoritmo BOZORTH3 devuelve un resultado numérico. Cuanto más alto sea este resultado, más probable es que las dos huellas sean coincidentes. Es importante decidir un umbral adecuado por el cual, si el resultado está por debajo, la comparación es incorrecta y si lo supera, fue correcta. Este límite cambiará dependiendo de las condiciones del sistema.

A modo de resumen, en este capítulo se ha detallado en qué consiste la modalidad de reconocimiento de huellas dactilares. Los algoritmos estudiados han sido los proporcionados para este TFG. Sin embargo, existen más tipos de algoritmos para la autenticación por huellas dactilares. La elección de uno u otro dependerá de la aplicación a implementar.

3. BIOMETRIC APLICACTION PROGRAM INTERFACE (BioAPI)

La especificación de BioAPI es un estándar biométrico que define una Interfaz de Programación de Aplicaciones (API) y una Interfaz de Proveedores de Servicios (SPI) mediante las cuales las aplicaciones biométricas son capaces de comunicarse con un gran rango de tecnologías biométricas de diferentes proveedores. Su función es asegurar una interoperabilidad entre aplicaciones y dispositivos de diferentes proveedores.

En este capítulo, se hará un breve repaso a la historia del estándar BioAPI, se verá en detalle su arquitectura y una vez entendida, se expondrán los elementos más importantes que la forman.

3.1. HISTORIA

La historia de BioAPI comienza en 1998, cuando varias empresas de la industria tecnológica deciden crear un consorcio americano llamado BioAPI con el objetivo de desarrollar un estándar biométrico. Al finalizar el año, este grupo desarrolla una arquitectura para el prototipo y comenzaban a definir componentes asociados. Era el inicio de BioAPI 1.0.

Sin embargo, este estándar inicial no duró demasiado tiempo ya que el grupo de estándares ANSI INCITS comenzó a trabajar sobre él, lanzando en 2001 una versión definitiva sobre la que trabajar, BioAPI 1.1. Esta nueva versión ya implementaba una especificación estable sobre la que apoyarse, aunque poseía todavía varios errores importantes como una gestión inadecuada del recolector de basura y una asignación dinámica de recursos incorrecta. Además presentaba varios fallos de interoperabilidad [16].

Acto seguido del lanzamiento de BioAPI 1.1, el grupo de trabajo internacional ISO/IEC JTC1 SC37 comenzó a trabajar sobre BioAPI y ambos grupos se unieron con el fin de solucionar estos problemas. Surge así, en el año 2002, la versión 2.0 de la especificación, conocida como la norma ISO/IEC 19784-1. Aunque se consiguió solventar los problemas anteriores, el mayor inconveniente ahora era la Interfaz Gráfica de Usuario (GUI) ya que la aplicación no permitía controlar lo que ocurría en los niveles inferiores de la implementación. Para solucionar este problema se añadió funcionalidades de *callback* al GUI por la cual las capas inferiores envían información a las superiores. Nace así BioAPI 2.1 reflejado en ISO/IEC 19784-1 Amendment 1 (2007).

En estos momentos, BioAPI estaba pensado para desarrollarse en sistemas operativos de gran capacidad como podían ser ordenadores. No obstante, comenzó a gestarse la idea de portarlo a otras plataformas de menor complejidad, como dispositivos móviles. Se origina así BioAPi Framework-Free (ISO/IEC 19784-1 Amd. 2 en el año 2009), donde se elimina el elemento intermediario que se encargaba de la comunicación entre las capas superiores e inferiores.

Poco después de lanzar esta última versión, el consorcio se da cuenta que aun habiendo resuelto todos los problemas que existían, no estaban teniendo en cuenta la seguridad dentro de la especificación, lo que dio lugar a una última especificación en ISO/IEC 19784-1 Amd 3 (2010).

En 2011, se produjo una revisión de los documentos con el objetivo de unir e integrar todas las especificaciones en una sola. En la actualidad todavía se encuentra en desarrollo, previéndose que se terminará en 2015, pasando a ser BioAPI 3.0.

Todas las anteriores normas están definidas basándose en el lenguaje de programación C ya que éste ofrece una buena gestión de la memoria, eficiencia y alta portabilidad. Sin embargo, en paralelo a estos acontecimientos, se comenzó a pensar en crear una norma BioAPI orientada a objetos (OO), ya que esta filosofía facilita enormemente la comprensión y creación de código.

Es en la universidad de Purdue (Estados Unidos) donde nace la primera especificación de OO BioAPI definida en el lenguaje multiplataforma Java. Poco después, el departamento de Tecnología Electrónica de la Universidad Carlos III de Madrid, comienza a implementar la misma norma pero basándose en el lenguaje C#. Trabajando en conjunto con la U. de Purdue, surge así la especificación ISO/IEC 30106-X. En la actualidad, ambas universidades se encuentran todavía realizando esta especificación, implementando nuevas versiones y mejoras, y se prevé que se termine de definir a mediados del 2015.

En el presente proyecto de fin de grado, se implementa y desarrolla el código basado en esta última especificación (C# y Java) dado que se ha trabajado dentro del Grupo Universitario de Tecnologías de Identificación perteneciente al departamento de tecnología de la Universidad Carlos III. En los siguientes apartados, se detallarán las características de la versión 4.0 de la norma BioAPI ya que el presente TFG se basó en ella.

3.2. ARQUITECTURA

La arquitectura de BioAPI se muestra en la Figura 16. La aplicación biométrica, en el nivel más alto, se comunica mediante la interfaz API con el *Framework*. El Framework se encarga de comunicar las aplicaciones con los *Biometric Service Providers* (BSPs). La función de los BSPs es proveer a las aplicaciones de los servicios biométricos que se requieran (identificación, verificación, almacenamiento, etc). El Framework se comunica con las aplicaciones utilizando la interfaz API y con los BSPs utiliza la interfaz SPI. Para que los BSPs puedan ofrecer sus servicios es necesario que se comuniquen con los dispositivos biométricos, que se encuentra en el nivel más bajo.

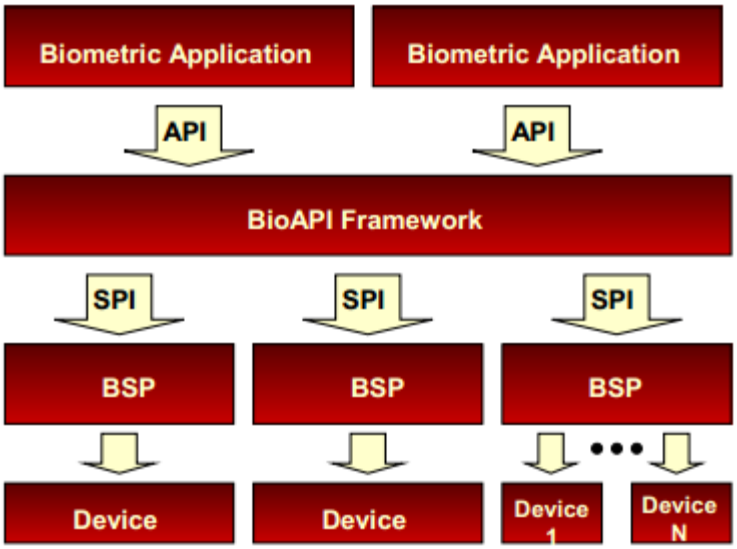

Figura 16: Arquitectura de BioAPI [17]

Con esta arquitectura, el funcionamiento es el siguiente: la aplicación solicita un servicio biométrico al Framework por medio de la interfaz API (la captura de una huella, el almacenamiento de una imagen, el procesado de una muestra, etc). Dependiendo del servicio, el Framework decidirá que BSP es el correcto para desempeñar el trabajo. Una vez decidido, el Framework instala y conecta el BSP a la aplicación para que esta pueda acceder a sus funciones. El BSP finalmente realiza el servicio que se le ha pedido y devuelve el resultado hacia arriba, de vuelta a la aplicación.

Existe la posibilidad de que una nueva aplicación necesite el BSP que otra esté usando o que una aplicación necesite usar más de dos BSPs al mismo tiempo. BioAPI soporta estas opciones.

3.3. BSPs Y UNIDADES

Los BSPs se encargan de realizar las operaciones pedidas por la aplicación. Para ello deben realizar diferentes acciones dependiendo del servicio deseado. Con este propósito, los BSPs gestionan una serie de unidades que encapsulan el software y hardware en diferentes bloques dependiendo de su funcionalidad. Surgen así las Unidades de BioAPI y son cuatro: sensor, archivo, procesamiento y comparación. Un BSP tiene acceso a 4 unidades diferentes, si se necesitan más, será necesario utilizar otro BSP adicional. A continuación las veremos en detalle:

3.3.1. Unidad de Archivo

Esta unidad representa la funcionalidad de almacenado en un BSP. Implementa una serie de métodos que son necesarios para gestionar el almacenamiento de imágenes o usuarios en cualquier aplicación biométrica. Entre las funciones que utiliza destacan: abrir bases de datos, cerrarlas, guardar muestras biométricas o usuarios, cargar muestras para usarlas o eliminarlas (así como eliminar usuarios).

3.3.2. Unidad de Sensor

Representa el grupo de operaciones biométricas que tienen como objetivo la captura de las muestras. Así, las funciones que implementa esta unidad son entre otras, el calibrado del sensor si este lo soporta, la captura de la muestra, etc.

3.3.3. Unidad de Procesamiento

Esta unidad se encarga de llevar a cabo todas las operaciones relacionadas con el procesamiento de las muestras biométricas.

En el caso del presente trabajo de fin de grado, como se ha visto, utilizamos el algoritmo MINDTCT para procesar la huella dactilar en crudo consiguiendo así un conjunto de minucias. La unidad de Procesamiento es la encargada de recibir esa imagen en crudo, utilizar internamente este algoritmo y retornar la muestra procesada.

3.3.4. Unidad de Comparación

Esta unidad incluye todos los métodos necesarios para realizar la función de comparación de muestras biométricas, tanto identificación como verificación.

En este trabajo se ha empleado el algoritmo BOZORTH3 para la unidad de comparación de huellas dactilares.

En resumen, estas cuatro unidades son gestionadas por el BSP, que sabe a qué función de cada bloque llamar para realizar el servicio que la aplicación le pidió. Las unidades son independientes entre sí, es decir, no se comunican entre ellas, es el BSP quien las maneja según lo solicitado por el Framework.

3.4. FRAMEWORK

En este momento, se conoce el funcionamiento de los BSPs y las Unidades pero, ¿quién se encarga de gestionar ambos elementos? El Framework. Como se vio en el apartado ARQUITECTURA, el Framework es la capa intermedia que permite comunicar los niveles inferiores con los superiores. Además de esta responsabilidad, para el correcto funcionamiento del sistema biométrico, el Framework tiene que asegurar un registro de los componentes y gestionar la instalación de BSPs y Unidades [18].

El registro de componentes es un campo del estándar que contiene información sobre todos los elementos conectados al Framework, es decir, BSPs, Unidades e incluso información sobre el propio Framework. El propósito de este registro es que la aplicación pueda obtener toda la información posible de lo que ocurre en las capas inferiores, como por ejemplo, que BSPs se encuentran instalados y que Unidades son accesibles desde estos BSPs. Este registro se usa principalmente en aplicaciones complejas que requieren el uso de varios BSPs y Unidades, siendo importante guardar la información de los diferentes elementos.

Por otro lado, la gestión de la instalación de BSPs y Unidades es necesaria siempre que una aplicación quiere acceder a las funcionalidades biométricas. A continuación se muestra en la

siguiente Figura 17, el proceso que sigue el Framework para gestionar de forma óptima estas conexiones:

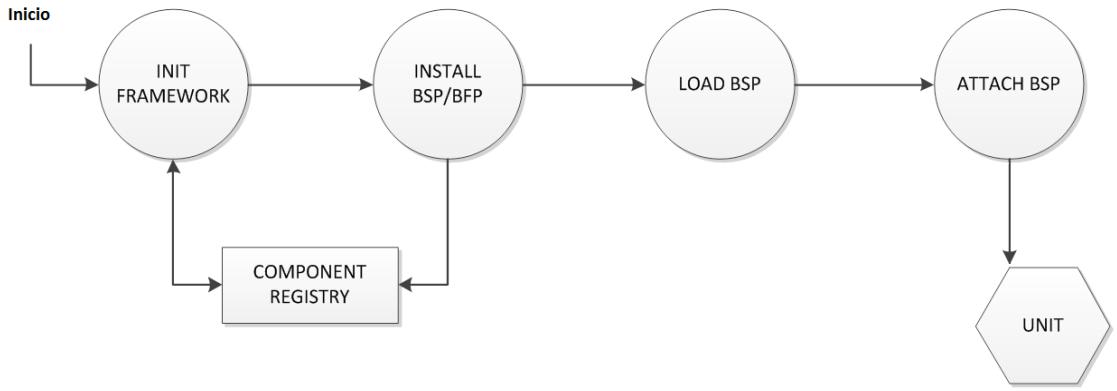

Figura 17: Proceso de instalación y asociación de los elementos biométricos [19]

En el primer paso, la aplicación inicia el Framework, quien envía información sobre su estado al registro de componentes. Asimismo, se encarga de instalar y cargar el BSP adecuado al servicio pedido por la aplicación. En el paso de instalación del BSP, éste envía su información al registro. Tras cargar el BSP, se asocia con las Unidades que necesite. En este momento se establece una *Asociación de Sesión* (attach session) que permanecerá invariante hasta que el Framework decida desasociarla (dettach session). Mientras que los componentes están ligados, es decir, mientras que la sesión está activa, existe una comunicación directa entre el Framework, los BSPs instalados y las Unidades asociadas, y es a partir de aquí, cuando se deben iniciar las operaciones biométricas.

3.5. BioAPI FRAMEWORK FREE

Como se vio en el apartado 3.1, existen dos implementaciones diferentes en BioAPI: BioAPI Framework y BioAPI Framework Free.

BioAPI Framework Free elimina el elemento Framework de la arquitectura, dejando a los BSPs como encargados de comunicarse con la aplicación. Al desaparecer el Framework, también desaparece la interfaz API, y los BSPs deben encargarse directamente de su propia gestión, por lo que generalmente, la mayoría de sistemas que implementan Framework-free son aplicaciones sencillas de un único proveedor.

El funcionamiento de BioAPI Framework Free se basa en permitir a las aplicaciones instanciar directamente los BSPs, que se comunican a su vez con las diferentes Unidades. Esto se representa en la Figura 18:

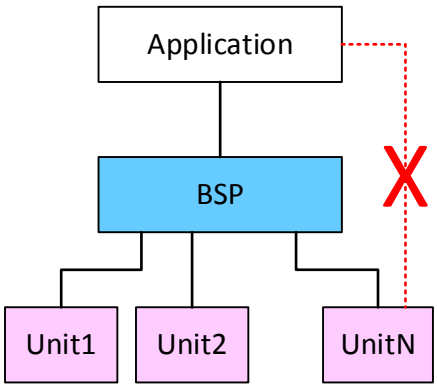

Figura 18: Aplicación Framework Free con un único BSP [19]

En el caso de la imagen anterior, la aplicación sólo se comunica con un BSP. En el caso de que la aplicación necesite emplear varios BSPs, cada BSP actúa como una biblioteca que será usado por la aplicación (Figura 19). Si por ejemplo, una aplicación debe usar dos sensores diferentes, procesar y verificar las muestras, necesitará 3 BSPs: uno para cada sensor y el tercero encargado de gestionar las Unidades de Procesamiento y Comparación.

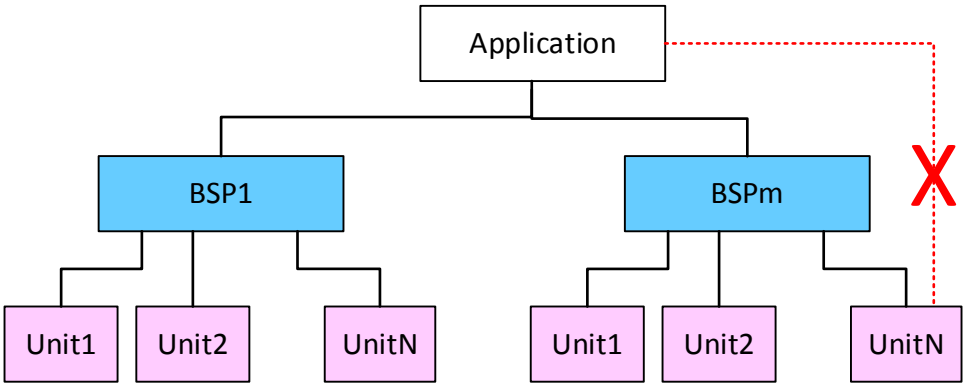

Figura 19: Aplicación Framework Free con varios BSPs [19]

Tal y como se vio en el apartado 3.4, el elemento Framework, además de gestionar la comunicación, también tenía otras funciones: el registro de los componentes y la óptima instalación y asociación de los BSPs y las Unidades. Al desaparecer el Framework, no es posible llevar un registro de información de componentes por lo que las aplicaciones Framework-Free serán aplicaciones sencillas que no requieran el uso de muchos elementos. Por otro lado, la instalación y asociación queda ahora en manos del BSP.

Debido a que en la actualidad BioAPI Framework se encuentra aún en versiones iniciales y a que la complejidad de este trabajo de fin de grado no requiere el uso de este Framework, la aplicación desarrollada en el presente documento está implementada usando BioAPI Framework-Free.

3.6. BIR (BIOMETRIC IDENTIFICATION RECORD)

El *BIR* o *Registro de Identificación Biométrica* es una encapsulación de las imágenes biométricas, ya sean imágenes en crudo o imágenes ya procesadas. Cuando en el apartado 3.3, respecto a las Unidades, se hablaba de "enviar las muestras o imágenes de unos métodos a otros", estas imágenes no se pueden transmitir por las diferentes funciones de forma simple, sin codificar, sino que necesitan ser encapsuladas dentro de una estructura de datos definida previamente para permitir la interoperabilidad entre ellas.

Para construir un BIR es necesario seguir un estándar llamado *CBEFF* (Common Biometric Exchange Formats Framework). CBEFF define los registros de información biométrica (BIR) como un conjunto de bytes, similar a una trama de información. Su estructura, representada en la Figura 20, está formada por una cabecera donde se guardará información sobre la muestra (SBH), los datos de la muestra (BDB) y un bloque final de seguridad (SB).

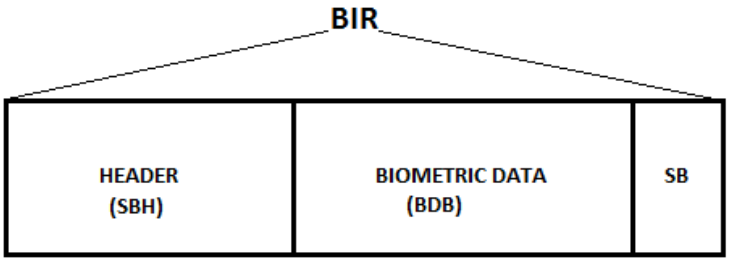

Figura 20: Representación de la estructura de un objeto BIR

A continuación se estudiará cada bloque en detalle.

3.6.1. Standard Biometric Header (SBH)

La Standard Biometric Header (SBH) se corresponde con la cabecera de información del BIR. Está formada por diferentes campos, como se observa en la Figura 21:

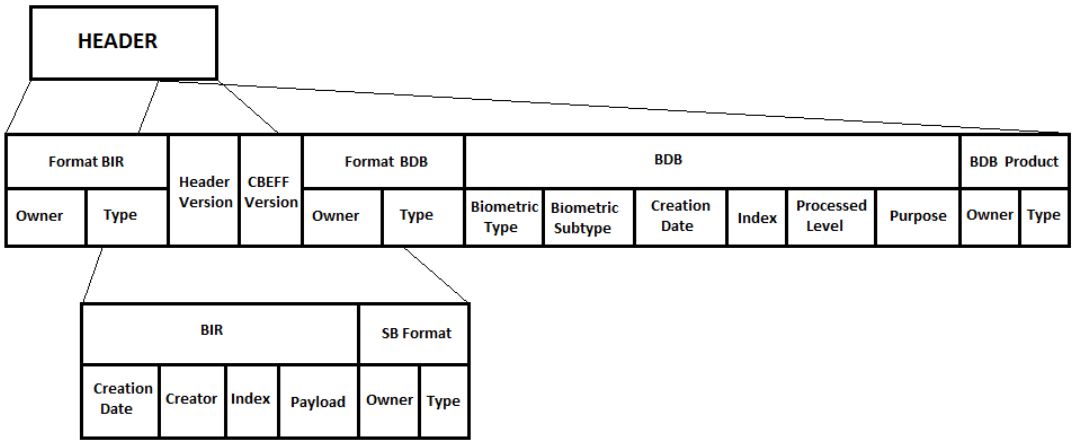

Figura 21: Representación de la cabecera SBH de un objeto BIR

En la cabecera existen tres partes diferenciadas: la información del BIR, la información del BDB y la del bloque de seguridad SB. BDB se refiere a la información de las muestras mientras que el BIR hace referencia a la estructura que encapsula el propio BDB. Cabe destacar que cada campo tiene un tamaño mínimo obligatorio, medido en bytes, que puede ser ampliado si se requiere. A continuación se presentan estos campos:

- **Format BIR**: Estructura *Format* compuesta por los campos *Owner* y *Type.* El campo *owner* denota el proveedor, grupo de trabajo o consorcio que crea el BIR mientras que el campo *type* indica el formato del BIR según lo que especifique el owner.
- **Header** y **CBEFF versión**: Estos dos campos especifican la versión de la cabecera y del formato CBEFF que se está usando.
- **Format BDB:** Se indica el formato del bloque de información BDB.
- **Biometric Type:** Campo que define qué tipo de modalidad biométrica tiene el BDB. Estas modalidades pueden ser: iris, rostro, huella, retina, geometría de la mano, firma, teclado, movimiento de labios, venas, ADN o forma de andar.
- **Biometric Subtype:** Dentro de cada modalidad, existen subtipos. Por ejemplo, dentro de la huella dactilar, podemos tener el dedo índice, el corazón, el pulgar, etc. Este campo se encarga de establecer esta subcategoría en el caso de que la hubiera.
- **Creation Date (BDB):** Representa la fecha de adquisición de la muestra.
- **Index (BDB):** Campo que define el nombre de la muestra para diferenciarla dentro de la base de datos.
- **Processed level:** Establece el nivel de procesado de la muestra. Estos niveles pueden ser: crudo (raw), intermedio o pre-procesado (intermediate) y procesado (processed).
- **Purpose:** Campo que define el propósito de la acción a llevar a cabo por el BSP. Pueden ser: verificar, identificar, reclutar, reclutar para verificación o reclutar para identificación.
- **Product BDB:** Indica el formato del producto del BDB.
- **Creation Date (BIR):** Representa la fecha de creación del BIR. Cabe mencionar que la fecha de creación del BDB puede ser diferente a la del BIR.
- **Creator:** Representa el proveedor o grupo de trabajo creador del BIR. Es una representación binaria, similar a una firma.
- **Payload:** Permite tener una lista de BIRs.
- **Format SB:** Mismo funcionamiento que Format BDB/BIR pero aplicado al bloque de seguridad SB.

3.6.2. Biometric Data Block (BDB)

Una vez definida la cabecera del BIR, queda por ver la información biométrica. Este bloque contiene los datos biométricos. Estos datos pueden ser tanto la imagen en crudo sin modificar como la imagen procesada (minucias). Generalmente esta imagen se codificará como un array de bytes.

3.6.3. Security Block (SB)

BioAPI también garantiza la seguridad de la información. En este último bloque se guardan todos los campos relacionados con esta seguridad. Para garantizarla, BioAPI tiene dos opciones. La primera consiste en la encriptación de la información biométrica o BDB. La

segunda se basa en utilizar un mecanismo de autentificación llamado MAC (Message Authentication Code). Para ello, se guardan los parámetros necesarios en una porción del SB y utilizando el algoritmo de MAC, se autentican los datos.

En la actualidad este bloque no se encuentra completamente desarrollado por lo que en este trabajo se ha optado por no implementarlo.

Para finalizar este capítulo y a modo de resumen, BioAPI es un estándar con gran potencial que permite la interoperabilidad entre elementos y la compatibilidad entre diferentes sistemas y dispositivos. Para el desarrollo del sistema, ha sido necesario basarse estrictamente en la norma proporcionada, dificultando en muchas ocasiones la creación de éste. Hay que destacar por tanto, que implementar el código BioAPI y verificar su correcto funcionamiento ha sido el problema más complejo que se ha encontrado a lo largo de este proyecto.

4. HERRAMIENTAS DE DESARROLLO

En este capítulo se abordarán las herramientas de desarrollo que han permitido realizar este trabajo de fin de grado. Como se comentó en la introducción, la aplicación del presente proyecto se basa en una comunicación PC-móvil y dado que BioAPI se encuentra en la actualidad definida en C# y Java, se ha decidido utilizar el lenguaje C# para implementar el código en el PC y el software de desarrollo Android sobre Java para el móvil. En consecuencia en este apartado se verá brevemente el lenguaje de programación Java para introducir al lector en el entorno de desarrollo Android. Además, se comentará brevemente el entorno de desarrollo utilizado. Por último, se hablará del lenguaje C# y sus principales características, finalizando con una escueta explicación del editor de código empleado en este caso.

4.1. JAVA

Java es un lenguaje de programación desarrollado por James Gosling, perteneciente a la compañía Sun MicroSystems. Se define como un lenguaje compilado e interpretado ya que todo programa de Java debe ser compilado antes de ejecutarse, generando un código propio de Java, llamado *bytecode,* que será interpretado por la máquina virtual de Java (JVM). Esta característica hace que Java sea independiente de la plataforma que lo ejecuta, es decir, el programa sólo necesita que se escriba una vez para ser ejecutado en cualquier dispositivo. Java ha desarrollado el eslogan "Write Once, Run Anywhere" en referencia a esta propiedad.

Java es un lenguaje de *propósito general*, ya que puede ser usado para diferentes tareas, *concurrente*, es decir, permite la simultaneidad de diferentes hilos de ejecución y *orientado a objetos*. [20]

4.2. ANDROID

Android es un sistema operativo libre, gratuito y multiplataforma basado en Linux y orientado a dispositivos móviles. Las aplicaciones de Android se basan en bibliotecas desarrolladas por la empresa Google y están escritas en el lenguaje de programación Java.

4.2.1. Historia

La historia de Android comienza en 2003, cuando una pequeña empresa de desarrollo de teléfonos móviles, llamada Android Inc y formada por cuatro integrantes, es adquirida por Google. A finales del 2007, Google anuncia oficialmente la creación de un sistema operativo llamado Android. Al mismo tiempo, se dio a conocer el consorcio Open Handler Alliance, formado por compañías de renombre como Google, HTC, Samsung o Qualcomm. Esta alianza nació con el fin de desarrollar estándares abiertos para los dispositivos móviles. En Octubre de 2008 sacaron al mercado el primer dispositivo Android, el HTC Dream. [21]

Desde este primer móvil, el sistema operativo Android ha ido actualizándose con el objetivo de mejorar el sistema operativo, añadiendo nuevas funcionalidades y mejorando la interacción con el usuario. Cada conjunto de actualizaciones recibe un nombre cuya letra inicial está en orden alfabético. Además, ese nombre es un postre en inglés. En las siguientes figuras podemos ver las diferentes actualizaciones que ha habido (Figura 22) y su impacto en el

mercado (Figura 23). Hemos considerado que dado el carácter de este proyecto, no es necesario explicar o detallar cada una de las versiones por separado.

Figura 22: Diferentes actualizaciones de Android [22]

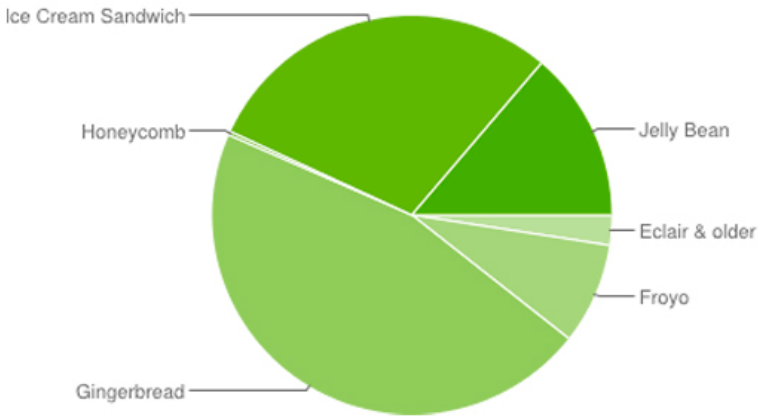

Figura 23: Fragmentación Enero 2013 de la plataforma Android [23]

4.2.2. Arquitectura Android

En las próximas líneas, se presentará una visión global de la arquitectura usada en Android. La siguiente Figura 24 se muestra las capas que conforman Android y como en cualquier pila de software, las aplicaciones inferiores utilizan servicios ofrecidos por las capas superiores. A continuación, se detallará brevemente en que consiste cada nivel: [24]

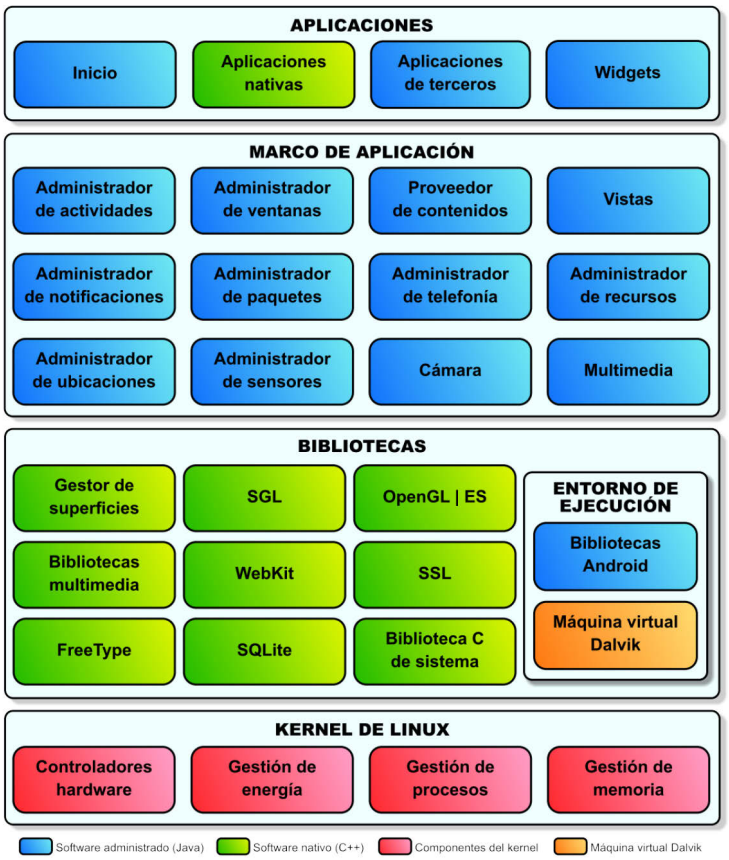

Figura 24: Arquitectura de Android [25]

Kernel de Linux

El núcleo del sistema Android es el kernel 2.6 de Linux y está adaptado a las características del hardware que se instalará en él, generalmente smarthpones y tabletas. Funciona como una capa de abstracción para el hardware disponible de modo que las aplicaciones de la última capa puedan acceder sin ningún problema al hardware instalado en el dispositivo, siempre que se hayan instalado los drivers necesarios.

Bibliotecas

La capa superior al kernel está formada por las bibliotecas nativas de Android. Están implementadas en C/C++ y proporcionan un gran número de funcionalidades a las aplicaciones.

Entre las más importantes destacan un gestor de interfaz gráfica (*gestor de superficies)*, un motor gráfico de 2D (*SGL*) y otro de 3D (*OpenGL*), un framework que facilita el uso de audio y video (*OpenCore* o *bibliotecas multimedia),* un motor de navegadores web (*WebKit*), un protocolo de seguridad (*SSL*), una biblioteca de fuentes y textos (*FreeType*), un gestor de bases de datos relacionales (*SQLite*) y por último una biblioteca que incluye todas las funciones de C.

Entorno de ejecución

En el mismo nivel que las bibliotecas antes mencionadas, está el entorno de ejecución o runtime de Android. La primera parte del entorno, la forman un conjunto de bibliotecas que contienen prácticamente todas las funcionalidades de Java aparte de otras específicas de

Android. La segunda parte corresponde a la máquina virtual Dalvik, optimizada para poder ejecutar múltiples aplicaciones en Android.

Marco de la aplicación

Es un framework formado por todas las herramientas de desarrollo de cualquier aplicación, es decir, todas las clases y servicios que el desarrollador necesita para implementar una aplicación.

Aplicaciones

En el último nivel se encuentran las aplicaciones incluidas por defecto así como las instaladas por el usuario. No es necesario notar que las propias aplicaciones utilizan los servicios y bibliotecas de las capas inferiores.

4.3. ECLIPSE

Eclipse ha sido la herramienta de desarrollo de código Android. Es un Entorno de Desarrollo Integrado (IDE) multiplataforma y de código abierto. Como IDE, proporciona un editor de código con resaltado de sintaxis, un compilador en tiempo real, un depurador y las herramientas necesarias para crear una interfaz gráfica (GUI) de forma sencilla e intuitiva.

Para poder programar en Android, es necesario añadir a Eclipse un plug-in llamado ADT (*Android Development Tools*) que permite utilizar las herramientas y bibliotecas básicas de Android para comenzar a crear aplicaciones móviles. Además, facilita enormemente la creación de interfaces gráficas para dispositivos Android.

4.4. C#

Como se comentó en la Introducción del documento, este trabajo fin de grado consiste en implementar un sistema biométrico basándose en el estándar BioAPI utilizando un ordenador y un móvil. En los apartados anteriores, se ha hablado sobre Android, la plataforma de desarrollo de Java que se utilizará para implementar el código en el móvil.

Debido a que en la actualidad, el grupo se encuentra trabajando en el desarrollo de BioAPI C#, se ha decidido desarrollar la aplicación del PC siguiendo este lenguaje de programación.

En las siguientes líneas de este reducido capítulo, se hablará brevemente sobre este lenguaje y se comentarán las diferencias más importantes respecto a Java.

4.4.1. Características de C#

C# (*C sharp*) es un lenguaje de programación multi-paradigma y orientado a objetos y a componentes, estandarizado por el gigante Microsoft como parte de su iniciativa .NET. Es un lenguaje imperativo y fuertemente restrictivo, que evoluciona de lenguajes los programación C y C++ pero incorporando características típicas de Java. Como características principales podemos citar su sencillez de uso, la recolección automática de basura, su facilidad de migración a otros lenguajes y su alta productividad. Nació como un lenguaje donde se quería combinar las mejores características de los lenguajes existentes como C, C++ o Java. [26]

4.5. VISUAL STUDIO

Al igual que para desarrollar aplicaciones en Android existe el entorno de desarrollo Eclipse, para la creación de aplicaciones en C# se utiliza otro entorno diferente proporcionado por la multinacional Microsoft: Visual Studio. Este IDE soporta varios lenguajes de programación entre los que destacan Visual Basic, C++ o C#.
Del mismo modo que el ADT de Android, Visual Studio cuenta con herramientas que facilitan la creación de interfaces gráficas. En este caso, son interfaces orientadas a aplicaciones de escritorio en Windows.

5. DISEÑO DE LA SOLUCIÓN TÉCNICA

Tras haber examinado en detalle los capítulos anteriores, el lector ya puede considerarse apto de indagar sin problemas en el diseño y desarrollo de la solución. En este capítulo se planteará el problema a resolver de este trabajo de fin de grado, se proporcionará la solución elegida, basándose en los requisitos y restricciones externas, para terminar diseñando el sistema final.

5.1. PLANTEAMIENTO DEL PROBLEMA

La especificación BioAPI en la que se encontraba trabajando el grupo donde se desarrolla este TFG era la versión 3.0, la cual presentaba muchas dificultades ya que era incomprensible para los desarrolladores, incompatible con todo tipo de programas y estaba obsoleta. En consecuencia, se propusieron nuevos cambios importantes en esta versión con el fin de intentar mejorar el resultado global de la norma, dando como resultado la versión 4.0 de BioAPI.

A continuación se detallarán los diferentes problemas que existían en esta versión.

Tal y como se comentó en el apartado 3.1, inicialmente BioAPI se desarrolló en C. El grupo en el que se desarrolla este TFG comenzó a trabajar con BioAPI orientada a objetos, basándose en los lenguajes de programación Java y C#, siempre intentando seguir la arquitectura de la norma definida en C. En instantes anteriores al comienzo de este TFG, el grupo se encontraba desarrollando la versión 3.0 de la norma BioAPI Java. Sin embargo, se descubrió que esta beta no se basaba en la estructura original de la especificación C. Además, se tomaba excesivas licencias y dejaba de respetar las ideas originales que dieron nombre a BioAPI. En ese momento hubo que replantearse desde el principio la existencia del estándar BioAPI en el paradigma de la programación orientada a objetos.

Dado que la norma basada en Java era muy incoherente y no se vislumbraba ninguna manera de mejorarla, se decidió dejar ésta a un lado y avanzar desde la especificación BioAPI C#. Para ello, hubo que revisar punto por punto toda la normativa comparándola con BioAPI C y descartando aquellas partes que no tuvieran correspondencia.

Todos estos nuevos cambios y mejoras en la estructura y arquitectura de la norma se definieron en una nueva versión, la beta 4 de BioAPI C#.

Debido a que se produjeron tantos cambios, es muy difícil mentarlos punto por punto. A continuación se mencionan los más significativos:

La estructura de los BIR. En la beta 3.0, su estructura era muy simple como para contener la complejidad de los múltiples tipos de datos biométricos que existían. En consiguiente, se redefinió esta estructura en un formato con muchos más campos y más complejo. Además, en esta nueva versión, es necesario almacenar en los dispositivos estas estructuras en forma de array, por lo que se definieron dos nuevos métodos para la conversión de BIR a byte array y su opuesto.

Eliminación de las clases no usadas. En la versión 3.0 muchas de las clases definidas se no se usaban en ningún caso. Existían numerosas interfaces que sólo devolvían un parámetro ya definido; por ejemplo, la función Identify, del tipo ProcessingProcessResult, tenía un único cometido que era devolver un BIR. En este caso la simplificación consistió en eliminar la interfaz ProcessingProcessResult y cambiar el tipo de la función Identify a BIR. Este error no sólo existía en interfaces; había clases sin métodos que simplemente devolvían un tipo de dato determinado. En estos casos, se eliminaron las clases y se sustituyeron en el código los tipos de los datos correspondientes. Aplicando este proceso de forma reiterada, el código quedó considerablemente simplificado.

Por otro lado, algunas clases tenían una función pensada para Java, pero obviaban que dicha función ya era asumida por otras partes del código. La clase DataFactory, por ejemplo, originalmente diseñada para mejorar la seguridad del sistema, no estaba implementada ya que la propia estructura de BioAPI orientado a objetos garantizaba por sí sola la seguridad necesaria, y por tanto se optó por su eliminación en la versión 4.

Nuevas clases añadidas. Se añadieron nuevos parámetros y clases con la intención de mejorar la norma. Los más importantes son las clases *DataFormat* y *DataTypes*. La primera surgió cuando los desarrolladores se dieron cuenta que había un gran número de implementaciones de diferentes *Format* (ver apartado 3.6.1). Para cada tipo de Format (*ProductFormat*, *BIRFormat*, *BDBFormat*, etc), existía una interfaz. Esto daba como resultado numerosas clases que repetían el mismo código. Se tomó la decisión de juntar todas las interfaces en una sola, DataFormat, que contendría el par Owner/Type pero sin diferenciar de los demás. Respecto a la clase DataTypes, sucedió algo parecido. En la versión 3.0 existía un gran número de enumeraciones de diferentes tipos de datos a lo largo del código. En cada clase, había enumeraciones desperdigadas. Se decidió juntarlas todas en una nueva interfaz, que representara todos los tipos y que se pudiera acceder a ella de forma simple e intuitiva.

Cambio de los parámetros de los métodos. La mayoría de métodos de la nueva propuesta 4.0 han cambiado sus parámetros de retorno así como los que reciben. El fallo más importante que se encontró en la versión 3.0 es que la mayoría de los métodos retornaban muchos parámetros a la vez y había que ir accediendo a ellos uno por uno para construir un BIR. Se decidió que en lugar de esto, se devolvieran BIRs directamente, consiguiendo que la comunicación entre métodos se realizara mediante estas estructuras.

Fechas. En la versión 3.0 del estándar BioAPI se utilizaban dos tipos de clases para generar fechas. La primera clase era *Date*, que representaba el año, mes y día. La segunda clase era *Time*, utilizada para las horas, minutos y segundos. En la nueva beta, se ha decidido juntar ambas clases en una sola, llamada *Date*. Además, se han incluido métodos booleanos para posibilitar la comparación de fechas.

La estructura de las bases de datos. En la versión 3.0 de la norma, se estandarizaba el acceso a las bases de datos, es decir, definía como se debía realizar todo acceso, modificación o eliminación de datos. En la versión 4.0, este camino de entrada ya no está normalizado y lo único que se tiene que implementar basándose en la norma son las funciones para abrir y cerrar las bases de datos.

5.2. PLANTEAMIENTO DE LA SOLUCIÓN

El objetivo de este Trabajo de Fin de Grado es probar que los cambios de la norma BioAPI 4.0 que se quieren proponer a los comités de estandarización son válidos, solucionan los problemas y son compatibles con las versiones anteriores. Con este objetivo en mente, se han propuesto diferentes requisitos que el sistema a desarrollar debe cumplir así como restricciones externas que limitan su funcionamiento. Teniendo en cuenta ambos factores, se desarrollará un sistema global cuyo fin sea comprobar el correcto funcionamiento de esta nueva especificación.

5.3. REQUISITOS DEL SISTEMA

En primer lugar, el sistema debe implementar correctamente las operaciones biométricas de captura de huellas dactilares, reclutamiento y verificación de usuarios, según lo establecido en la versión 4.0 de la norma BioAPI.

En segundo lugar, la estructura de los BIRs debe seguir el formato CBEFF 19875, explicado en el apartado 3.6.1.

En tercer lugar, el sistema tiene que estar formado por distintos dispositivos de proveedores diferentes, con el objetivo de comparar la correcta interoperabilidad de la especificación. Por lo tanto, se rechaza la idea de implementar el sistema sobre un solo equipo. Además, uno de los dispositivos debe ser un teléfono móvil y en él deben ejecutarse las funcionalidades de reclutamiento y verificación.

En cuarto lugar, es necesario que los dispositivos posean diferentes sistemas operativos y entornos de desarrollo. Dado que el grupo donde se ha realizado este documento está trabajando en la especificación BioAPI Java y C#, lo adecuado sería que se usen estos lenguajes de programación.

En base a estos tres requisitos, el sistema debe gestionar dos tipos de datos: huellas dactilares y usuarios. Además, la gestión debe separarse, es decir, las huellas dactilares necesitan ser manejadas en un dispositivo mientras que los usuarios deben ser administrados en otro.

Además, el sistema debe proporcionar una interfaz para que el usuario pueda interaccionar con él y comprobar en un contexto real que las funcionalidades biométricas funcionan correctamente.

Para poder integrar lo antes comentado en un único sistema, se debe crear una comunicación real entre ambos aparatos que permita transferir datos entre los dos dispositivos, dotando de estabilidad al sistema global.

Como consecuencia de los anteriores puntos, se ha decidido desarrollar el sistema usando un ordenador ejecutando el código en C# y un teléfono móvil Android.

5.4. RESTRICCIONES EXTERNAS

Las restricciones anteriores vienen dadas por los requisitos y objetivos de este proyecto. Sin embargo, en este apartado se detallan las restricciones encontradas por las limitaciones hardware y software de los dispositivos proporcionados a la hora de realizar el TFG.

La captura de huellas dactilares debe efectuarse en el ordenador debido a que en la actualidad no existen sensores en el mercado que tengan una interfaz compatible con teléfonos móviles. Sin embargo, los equipos proporcionados no están adaptados a ningún lector biométrico, por lo que será necesario simular la captura de huellas dactilares en el ordenador.

Las funcionalidades de procesamiento de huellas dactilares y verificación de usuarios deben realizarse en el dispositivo móvil Android. Para implementar estas operaciones se proporcionan los algoritmos MINDTCT y BOZORTH3.

Respecto a la comunicación entre los dispositivos, la única restricción existente es que la transmisión se debe efectuar libre de errores y sin pérdidas.

5.5. DISEÑO DE LA SOLUCIÓN TÉCNICA

A modo de resumen, uniendo los requisitos y las restricciones anteriores, se ha propuesto la implementación de un sistema formado por un ordenador que se encargue de capturar las huellas dactilares de los usuarios y un teléfono móvil cuya función sea el procesamiento y almacenamiento de esas huellas junto con la verificación de usuarios. Para posibilitar la transmisión de información, se crea una comunicación entre ambos dispositivos.

En ésta comunicación, el ordenador envía las imágenes de las huellas dactilares al dispositivo móvil para que éste pueda realizar las operaciones antes mencionadas. Cuando termina de ejecutarlas, retorna los resultados obtenidos de nuevo al ordenador. A continuación, en la Figura 25, se añade el diagrama de flujo para facilitar la comprensión y ofrecer una visión global de la lógica del sistema.

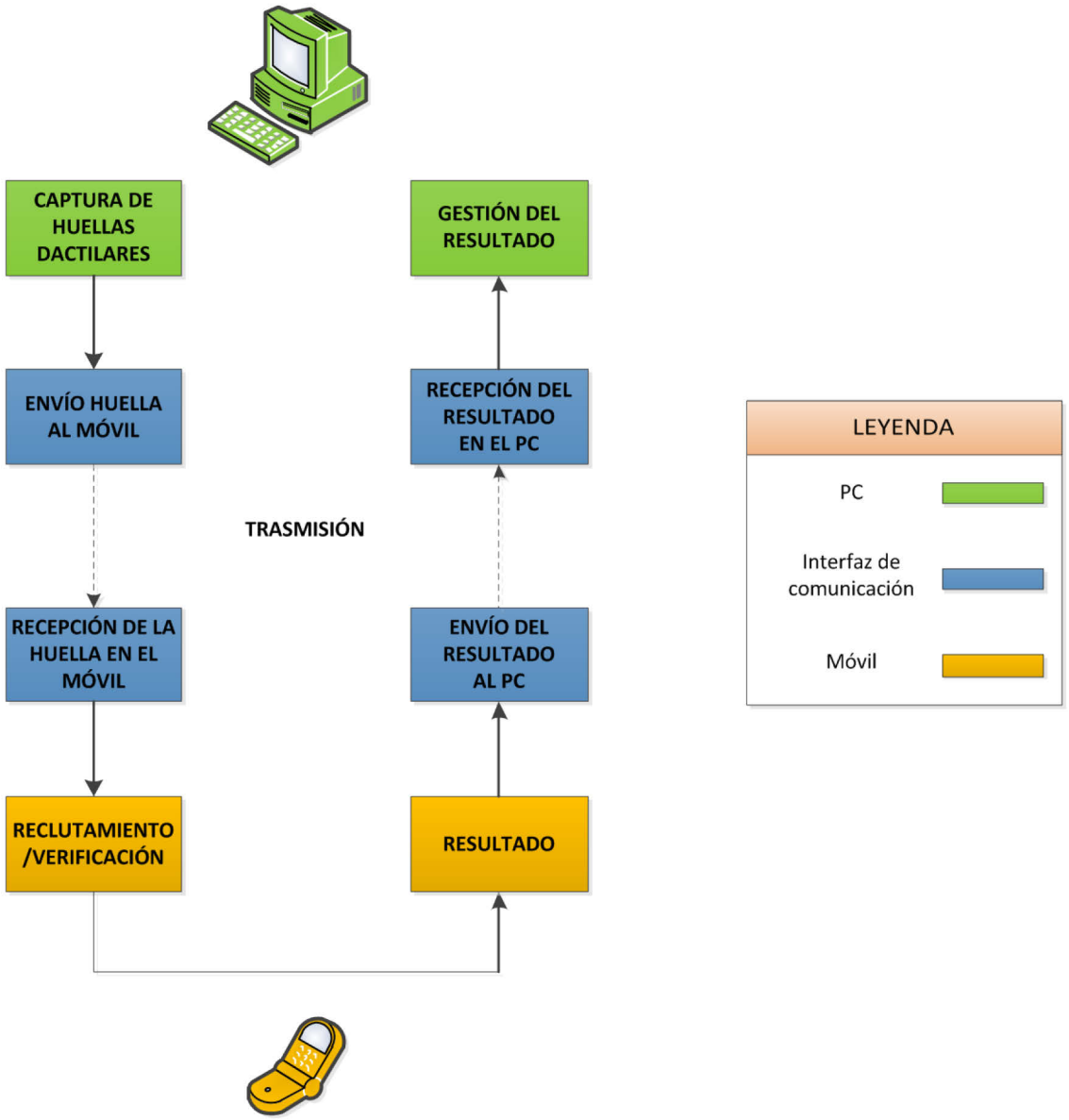

Figura 25: Diagrama de flujo del sistema

Como se puede observar, la aplicación en el ordenador, ejecutada en código C#, se comunica con la interfaz encargada de gestionar la transmisión de información. Ésta, a su vez, se comunica con el dispositivo móvil, que integra sus funcionalidades en Android.

Diseñado el sistema global, a continuación se pasará a diseñar los componentes que lo forman. Como se puede observar, existen tres módulos diferenciados: la aplicación que interacciona con el usuario, las funcionalidades biométricas y la comunicación entre dispositivos.

5.5.1. Aplicación

En primer lugar, es imprescindible definir qué tipo de aplicación se puede crear para que se simule un correcto funcionamiento de las funcionalidades biométricas. Esto se corresponde con la creación de una interfaz que interactúe con el usuario.

En la actualidad, existen multitud de aplicaciones que requieren de autenticación para ejecutar operaciones delicadas, como puede ser posibilitar el acceso a una página web o permitir una compra por Internet. Debido a que la verificación por huella dactilar es una funcionalidad que ofrece una gran seguridad, se pensó en una aplicación que necesitara una gran protección. Tras varias opciones estudiadas, se determinó que un ejemplo perfecto sería simular el funcionamiento de un banco, dónde el usuario necesitara verificarse para realizar las operaciones pertinentes.

Una vez llegado a este punto, tras darse cuenta que un sistema bancario debe proporcionar una máxima seguridad a sus usuarios y que este puede ser objeto de numerosos ataques, surge un nuevo requisito, no definido en el apartado anterior: la autenticación desarrollada para el sistema debe ser de *triple factor*, es decir, para autenticar al cliente, éste necesitara de tres factores: lo que el usuario posee, un dispositivo móvil; lo que sabe, su número de usuario en el sistema y lo que es, su huella dactilar. Por lo tanto, en las operaciones delicadas éste requerirá tanto de su número asignado como de su teléfono, además de su huella dactilar. Se ha elegido esta opción para ofrecer una seguridad máxima a la hora de ejecutar las diferentes funcionalidades críticas.

A la hora de decidir qué funcionalidades debía proporcionar este sistema bancario al usuario, se decidió elegir las básicas: ingreso y reintegro de dinero, registro de usuarios en el sistema, además de la opción informativa de mostrar el saldo disponible. La decisión de realizar sólo estas cuatro operaciones se basó en el hecho de que el trabajo de fin de grado se trataba de probar las propuestas de la nueva norma BioAPI, por lo que estas funcionalidades eran suficientes para lograr este fin.

Dado que el sistema bancario debe manejar nombres de usuarios y sumas de dinero, es necesario implementar también una gestión de colecciones de nombres y sus correspondientes saldos. No hay que confundirse entre la funcionalidad biométrica *reclutamiento de usuarios* y la gestión bancaria de nombres de usuarios, ya que son operaciones diferentes. Esta última se utiliza simplemente para posibilitar una autenticación de *triple factor* en el sistema bancario.

Por último, el almacenamiento de las huellas dactilares y de los usuarios se gestionará mediante directorios en los dispositivos. Dado que existen dos tipos de datos a tratar, existirán dos directorios en el sistema, cada uno situado en un dispositivo diferente. Se ha decidido no implementar bases de datos debido a que su complejidad queda fuera del ámbito de este TFG.

5.5.2. Sistema Biométrico

A continuación se procederá a describir las diferentes funcionalidades biométricas que se desarrollarán en el sistema. Tal y como se ha explicado, estas operaciones deben basarse en la norma BioAPI utilizando los cambios propuestos.

Captura de huellas dactilares

En primer lugar se encuentra la captura de huellas dactilares en el ordenador. Este proceso consta de dos pasos: la digitalización de la huella dactilar así como su posterior almacenamiento. Como se limitó en el apartado 5.4, el sistema a implementar carece de lector

de huellas, por lo que se deberá simular su comportamiento. Dado que el usuario no podrá introducir sus propias huellas, se decidió saltarse el primer paso del proceso de captura física y ofrecer directamente al usuario las imágenes de las huellas almacenadas en un directorio para que este tuviera la opción de elegir. Así pues, el primer directorio se encuentra en el ordenador y es el encargado de almacenar las huellas dactilares. Por supuesto, estas huellas no son las del usuario, si no que pertenecen a otras personas y fueron incluidas en momentos anteriores. Se decidió además, incluir diferentes huellas de una misma persona, con diferentes calidades y orientaciones. En conjunto, lo que se pretende es hacer una simulación lo más real posible para que el usuario no eche en falta en ningún momento el sensor en este sistema.

En la Figura 26, se puede observar el diagrama de flujo de los dos sistemas de capturas, el real y el simulado, donde el simulado necesita huellas dactilares del real para poder funcionar.

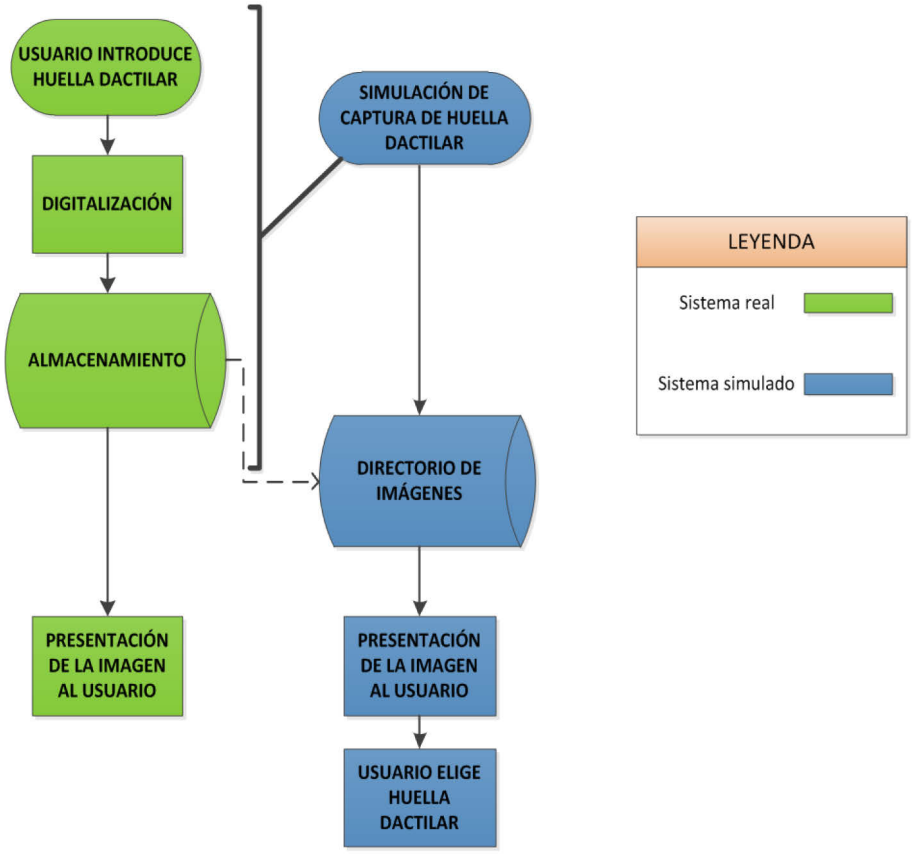

Figura 26: Diagrama de flujo de la funcionalidad *captura de huellas dactilares*

Como se puede ver, el proceso de captura real queda simplificado a un solo paso en el simulado, donde se ofrece directamente al usuario un directorio con huellas digitales. La línea discontinua muestra que es necesario pasar las imágenes de un sistema de captura real a un directorio del sistema simulado.

Reclutamiento de usuarios

En segundo lugar está la funcionalidad reclutamiento de usuarios en el dispositivo móvil. Esta operación biométrica consta de dos fases: inicialmente, se procesa la huella dactilar, recibida desde el ordenador, utilizando el algoritmo MINDTCT visto en el apartado 2.4.1. Este

algoritmo, de forma resumida, recibe una huella dactilar en crudo, ejecuta sus operaciones internas y retorna una serie de minucias. La siguiente fase de la funcionalidad es el posterior almacenamiento de estas minucias. Por lo tanto, cuando se habla de reclutar usuarios, realmente lo que se reclutan son las huellas procesadas de los usuarios. Es necesario destacar que para almacenar las minucias entra en juego el segundo directorio del sistema, que se localizará en el dispositivo móvil. A diferencia del anterior implementado en el ordenador, el sistema debe crear este directorio en tiempo de ejecución al almacenarse el primer conjunto de minucias. En la Figura 27, se muestra el flujo de este proceso, suponiendo que la huella dactilar ya se encuentra en el dispositivo móvil.

Figura 27: Diagrama de flujo de la funcionalidad *reclutamiento de usuarios*

Verificación de usuarios

En último lugar, se encuentra la función de verificación de usuarios. Esta funcionalidad estriba en la comparación de la huella dactilar de un usuario con la huella de quien dice ser. Para ello, cuando el beneficiario intenta autenticarse, es decir, afirma ser una determinada persona que está registrada en el sistema, por lo tanto sus minucias estarán guardadas en el dispositivo móvil habiendo realizado la operación anterior. Así pues, el usuario dice ser alguien que está en el sistema introduciendo un nombre de usuario. Para comprobar que esto es cierto, introduce su huella dactilar en el ordenador, se envía al móvil y se compara esta huella y la ya almacenada correspondiente al número de usuario que introdujo. A continuación se presenta el diagrama de flujo de esta operación en la Figura 28, suponiéndose que los datos necesarios ya se encuentran en el dispositivo móvil.

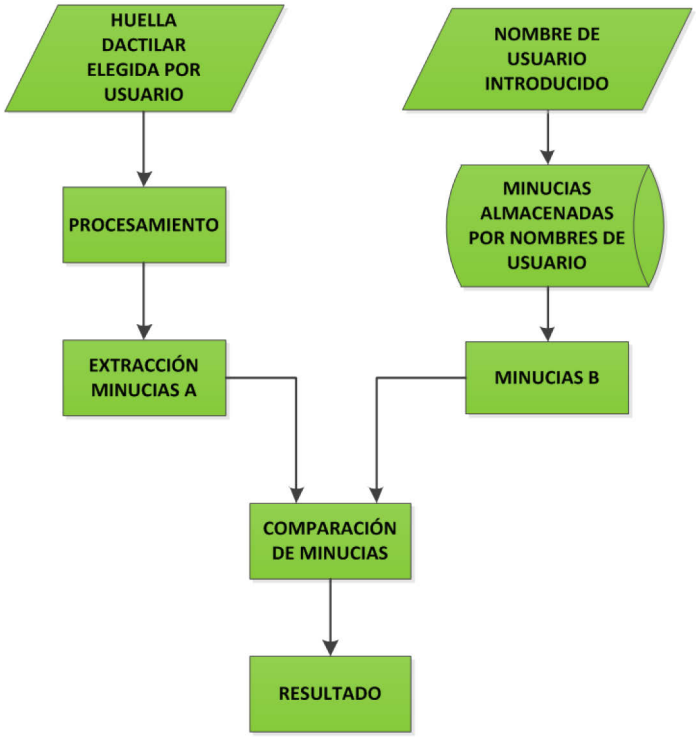

Figura 28: Diagrama de flujo de la funcionalidad *verificación de usuarios*

5.5.3. Comunicación ordenador-móvil

En este punto del trabajo, ya está diseñada la aplicación interfaz, así como el código biométrico que propicia el funcionamiento de las operaciones de captura, reclutamiento y verificación. Por lo tanto, lo único que falta es esbozar la comunicación que permita conectar los dispositivos y transmitir la información entre ellos. Por consiguiente, en este apartado se expondrán las diferentes soluciones encontradas a la hora de elegir qué tipo de comunicación utilizar. Una vez tomada la decisión, se pasará a decidir el protocolo de comunicación a implementar entre el ordenador y el dispositivo móvil.

Tipo de comunicación

Para elegir el tipo de comunicación a desarrollar, primero hay que analizar las diferentes opciones que existen. En la actualidad, se encuentran un gran número de posibilidades tanto inalámbricas como cableadas. Sin embargo, teniendo en vista el sistema a crear, la lista se reduce ampliamente. A continuación se debatirán las más importantes:

En primer lugar, se consideró crear una comunicación por medio del estándar Bluetooth. Este planteamiento se rechazó debido a que lo portátiles a los que se tuvo acceso carecían de conectividad Bluetooth.

En segundo lugar, se pensó la idea de utilizar una comunicación cableada basada en USB. Tanto Android como C# tienen bibliotecas que proporcionan una conexión basada en este estándar. Además, la comunicación es estable y suficientemente rápida como para ser considerada. Sin embargo, este planteamiento se declinó debido a la limitación de distancia

que impone el cable USB. Dado que se busca la comodidad del usuario, el cable USB no era la opción ideal.

En último lugar aparece la comunicación basada en Wi-Fi. Del mismo modo que con el estándar USB, Android y C# ofrecen bibliotecas para su implementación. Debido a que actualmente las conexiones inalámbricas son las más usadas [27] y proporcionan una alta comodidad y velocidad, se ha decidido desarrollar la comunicación sobre la tecnología Wi-Fi.

Protocolo de transmisión

En este apartado se decidirá qué protocolo de transmisión se utilizará en la aplicación y cómo se diseñará para posibilitar la transferencia de datos entre el ordenador y el móvil.

A la hora de plantear qué protocolo sería el más óptimo para el sistema a desarrollar, se pensó en los protocolos UDP y TCP. La principal ventaja del primero es que no se establece ninguna conexión por lo que no añade retardos aumentando la velocidad de transmisión. Como desventaja, no proporciona fiabilidad o garantía alguna ya que los datagramas se pueden perder por el camino o llegar desordenados. En el caso de TCP, éste establece una conexión entre las dos máquinas y asegura que todos los datos llegarán sin pérdidas y en el mismo orden en que se enviaron. Para alcanzar esta garantía sacrifica eficiencia y velocidad de transmisión.

Con el objetivo de decidir qué protocolo es el mejor, es necesario analizar los requisitos del sistema que se intenta implementar en el presente proyecto. En primer lugar, es necesario establecer una conexión entre el ordenador y el móvil antes de cualquier transmisión. Además, según se explicó, es necesario que la comunicación sea libre de pérdidas y errores. El PC debe enviar imágenes al móvil para que éste las procese byte a byte. Por lo tanto, es necesario garantizar que no se pierde o se desordena ningún byte de información por el camino. Como consecuencia directa, el protocolo vencedor es TCP.

Una vez tomadas las decisiones de diseño globales, se pasó al desarrollo de las mismas. A lo largo del desarrollo surgieron varias decisiones de diseño menores, que se comentarán en el siguiente capítulo, donde además, se desarrollará con mayor amplitud la implementación del sistema.

6. DESARROLLO DE LA SOLUCIÓN

En este capítulo se describirá paso a paso la implementación técnica de este proyecto. Una vez definidos los requisitos y diseñado el sistema global, sólo falta su creación. En las siguientes páginas, se explicará cómo se ha realizado, qué problemas se han encontrado y como se han solucionado. El apartado se dividirá en tres módulos bien diferenciados: la aplicación interfaz, la creación del sistema biométrico que implemente las funcionalidades de captura, reclutamiento y verificación y, finalmente, la comunicación cliente-servidor.

6.1. APLICACIÓN

Este apartado tratará sobre el desarrollo de la aplicación que interactúa con el usuario. Dado los requisitos del sistema a implementar, éste necesitará dos interfaces: una para el ordenador y otra para el dispositivo móvil.

Como se definió en el apartado 5.5.1, la aplicación consiste en un sistema bancario donde se ofrece al usuario la posibilidad de realizar las operaciones de reintegro e ingreso de dinero, llamadas *crédito* y *débito*, mostrar su saldo disponible y un registro de nuevos usuarios en el sistema.

Debido a las características de nuestro sistema, se consideró que lo lógico sería la creación de una interfaz principal en el ordenador donde se le proporcionara al usuario las operaciones antes descritas para que interactuara con ellas. El móvil tendría un rol más informativo de cara al usuario, implementando una interfaz mucho más sencilla.

Además, como consecuencia de la autenticación *triple factor*, a cada usuario se le asignará un nombre en el sistema que será necesario para su verificación. Por lo tanto, como se comentó en la sección 5.5.1, es necesario implementar una gestión de nombres. Se decidió que fuera realizada en el ordenador, mientras que el manejo de sus correspondientes saldos se haría en el dispositivo móvil. Los factores que influyeron para tomar la decisión de implementar una gestión en cada máquina fue que el saldo era un tipo de dato delicado que necesitaba cierta protección. Dado que es en el móvil donde se realizan las operaciones delicadas y que este dispositivo suele ser personal y privado, se eligió realizar su gestión en él. Asimismo, a causa de que sería en el ordenador donde se pediría el nombre al usuario, lo óptimo sería manejar los nombres de usuario en el PC.

En las siguientes líneas, se separará entre la aplicación en el ordenador y la aplicación en el móvil, describiéndose la creación de ambas, con los problemas que han surgido y su solución encontrada.

6.1.1. Aplicación en el ordenador

El objetivo principal de esta sección es crear una interfaz agradable e intuitiva para el usuario en el ordenador y que cumpla lo especificado en el apartado de requisitos del sistema. El objetivo secundario es gestionar de forma óptima los nombres de usuario. A continuación se explicarán en detalle.

Gestión de nombres de usuario

En esta aplicación no se permitirá al usuario crear su propio nombre si no que será el sistema quien lo generará y se lo asignará. Se ha decidido que sea el sistema quien cree el nombre y no el propio usuario debido a la formalidad propia de los bancos que no permitirían el uso de nombre de usuarios fuera del formato establecido. Antes de comenzar a estudiar la mejor gestión posible, es necesario destacar que se decidió remplazar los nombres por números con el objetivo de simplificar su manejo. Por lo tanto, a partir de ahora, se hablará de números y no de nombres.

Tal y como se comentó, la gestión de los números se realizará en el ordenador. La característica más importante de estos datos es que tienen que ser persistentes frente al cierre de la aplicación o, incluso, del ordenador. Por ello, es necesario analizar que herramientas de gestión de datos permanentes ofrece C#. En los comienzos, surgió la idea de utilizar bases de datos o ficheros de texto. Sin embargo, se rechazó dado que el sistema se complicaba mucho para simplemente almacenar una colección de números. La idea ganadora fue utilizar un tipo de variable que en C# se conoce como *Setting.* En resumidas cuentas, esta variable permanece persistente aunque la aplicación se cierre o el ordenador se apague, permitiendo su acceso y modificación en tiempo de ejecución.

Debido a que se trabaja con números, se decidió no ir almacenando los diferentes números en un array si no gestionarlos a través de un índice, es decir, un simple número positivo. Cada vez que un usuario nuevo se registra correctamente, se incrementa el índice una unidad, se almacena de nuevo en el sistema y se muestra por pantalla. Para comprobar si, en las operaciones de crédito, débito o mostrar saldo, el número introducido por el usuario existe, simplemente éste debe ser menor que el índice que se gestiona. Dado que el sistema no permite suprimir usuarios, esta solución, tal y como se ha probado, funciona perfectamente para el sistema y la aplicación desarrollada.

Interfaz

Para lograr implementar la interfaz, se pasaron por varios prototipos y se tomaron varias decisiones hasta modelar el resultado final. En las siguientes líneas, se explicará la versión más importante que se desarrolló antes de llegar a la interfaz final. Esta versión fue finalmente desechada y no se guardaron capturas de las ventanas, por lo que se muestran bocetos de la interfaz.

En esta primera versión, tal y como muestra la Figura 29, la interfaz constaba de una ventana inicial donde el usuario podía registrarse o autenticarse. La función *registro* se corresponde con la funcionalidad biométrica de reclutamiento de usuarios mientras que *autenticarse* se corresponde con la de verificación. Dependiendo de la opción elegida se abrían dos nuevos tipos de ventana donde se pedía al usuario su huella dactilar, además del número de usuario si se encontraba en autenticación. Una vez autenticado, se le mostraba una pantalla principal con las diferentes funcionalidades bancarias: crédito, débito y mostrar saldo.

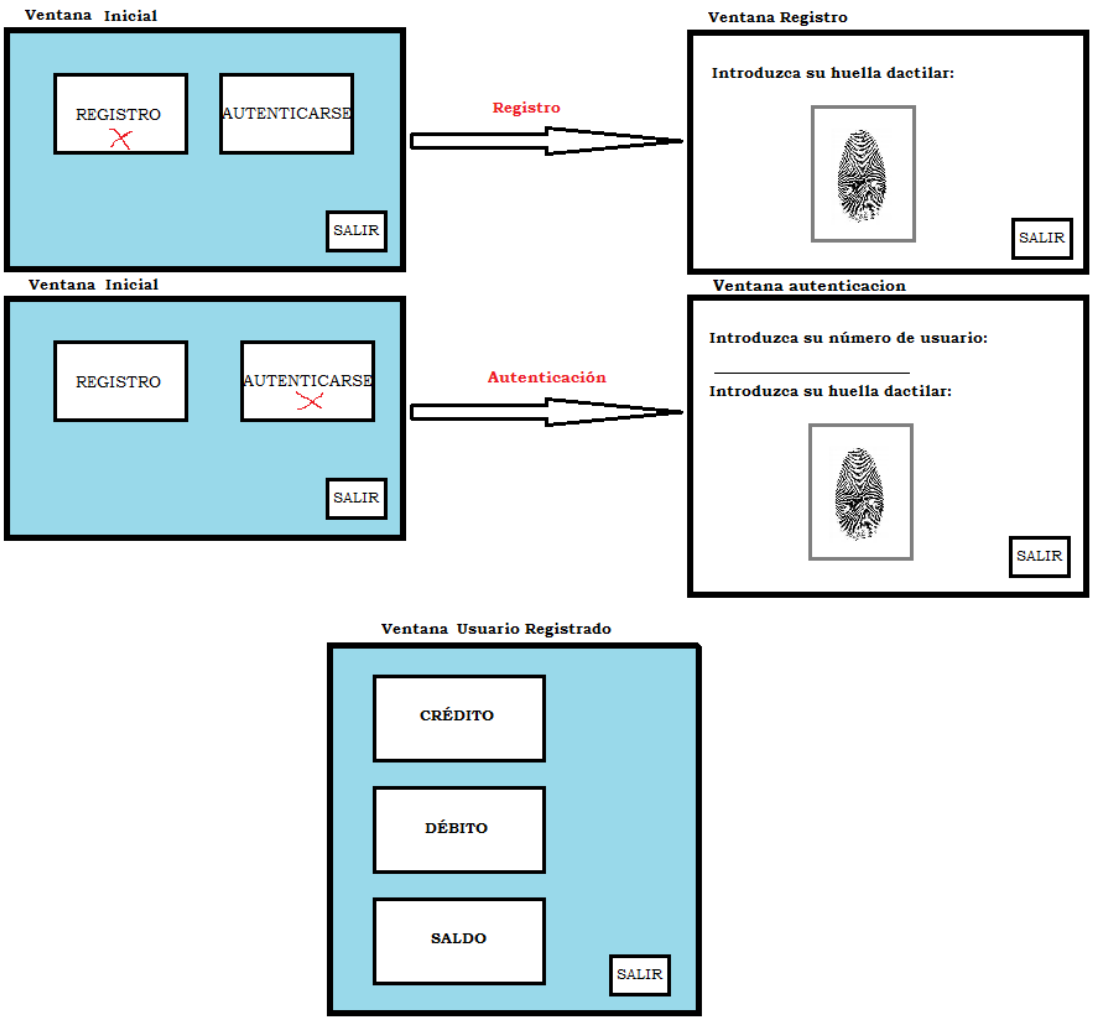

Figura 29: Diseño de las ventanas del primer prototipo de interfaz del ordenador

Respecto al flujo del proceso de la aplicación de este primer prototipo, se muestra en el diagrama de la Figura 30. En primer lugar, el usuario entra en la ventana de acceso. A partir de ahí, tiene dos caminos: el registro o la autenticación. Si elige el primero, salta a una pantalla donde se le pide introducir la huella dactilar para almacenarla en el dispositivo móvil. Éste registra al usuario retornándole un número de identificación. Una vez registrado, el usuario ya puede identificarse en el PC con ese número y su huella. Al hacerlo, accede al menú principal donde aparecen las opciones de crédito, débito o mostrar saldo. En esta ventana, el usuario ya está identificado por lo que se puede mover en libertad sin necesidad de ingresar de nuevo su huella dactilar. En el caso de pinchar en cualquiera de las opciones, se envía al móvil la orden y este retorna el resultado.

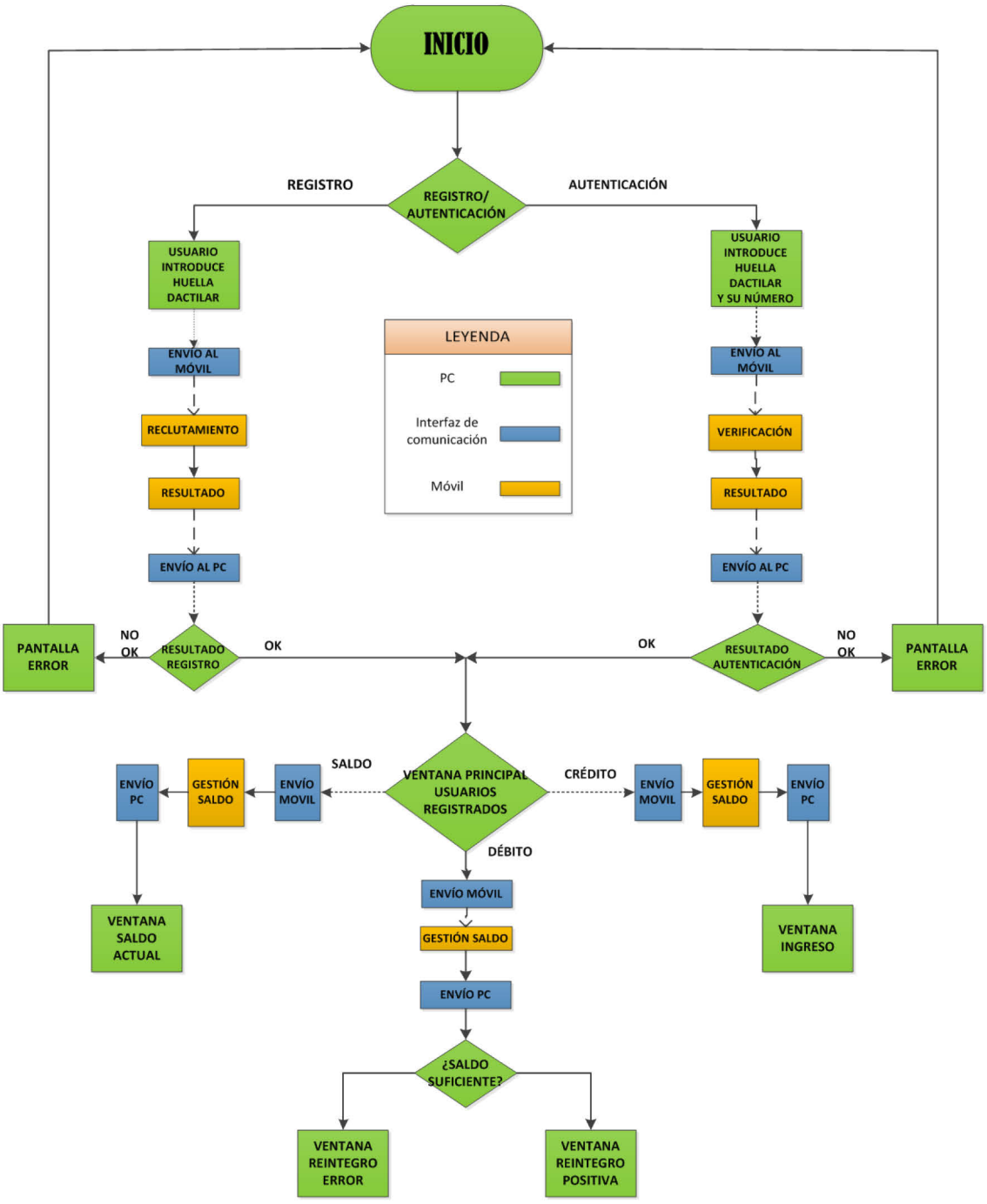

Figura 30: Diagrama de flujo del primer prototipo de interfaz

Esta primera idea de interfaz, aunque es intuitiva y fácil de usar, no cumple el requisito de ofrecer una máxima seguridad al usuario ya que verificar a éste una sola vez en toda la sesión no proporciona gran protección frente a ataques externos.

Es por esta razón que se debe pensar en un tipo de interfaz que pida en todo momento identificarse al usuario a la hora de ejecutar operaciones delicadas. De esta forma, se añade una protección extra ya que cualquier operación crítica necesita ser confirmada por el dispositivo móvil.

En consecuencia, se ha determinado crear una interfaz simple, dónde todas las operaciones aparezcan en la misma pantalla principal. De este modo, existe una ventana principal con

cuatro botones representando las cuatro funcionalidades monetarias y dependiendo de cuál pinche el usuario, saltará a otras ventanas secundarias donde se le pidan los datos biométricos para autenticarse. En el caso de la opción *mostrar saldo* no se requiere identificar al cliente.

A continuación, se mostrarán las diferentes ventanas ya implementadas en lenguaje de programación C# y se comentarán sus principales características.

Ventana principal

El resultado de esta primera ventana se puede ver en la Figura 31. Aparte de los cuatro botones antes comentados, tiene un quinto botón que cierra la aplicación en caso de que el usuario haga click sobre él. Además, al ser la pantalla que da la bienvenida, se muestra el nombre y logo de la aplicación.

Figura 31: Ventana principal de la interfaz del ordenador

Ventana de *Registro*

La funcionalidad *registro* permite reclutar la huella de un nuevo usuario. Para ello, el usuario necesita introducir su huella dactilar. Una vez introducida, el sistema le asignará un nombre de usuario con el que podrá autenticarse en futuras operaciones. Este nombre se le mostrará al usuario cuando el móvil retorne el resultado del reclutamiento.

El resultado final se puede ver en la siguiente Figura 32, donde se muestran dos casos: en el primero, el usuario no ha introducido ninguna huella, por lo que el botón *Registrarse* esta deshabilitado, y en el segundo, el botón se encuentra activado y listo. Pulsando el botón *seleccione su huella dactilar,* se abrirá el directorio donde se encuentren las huellas dactilares.

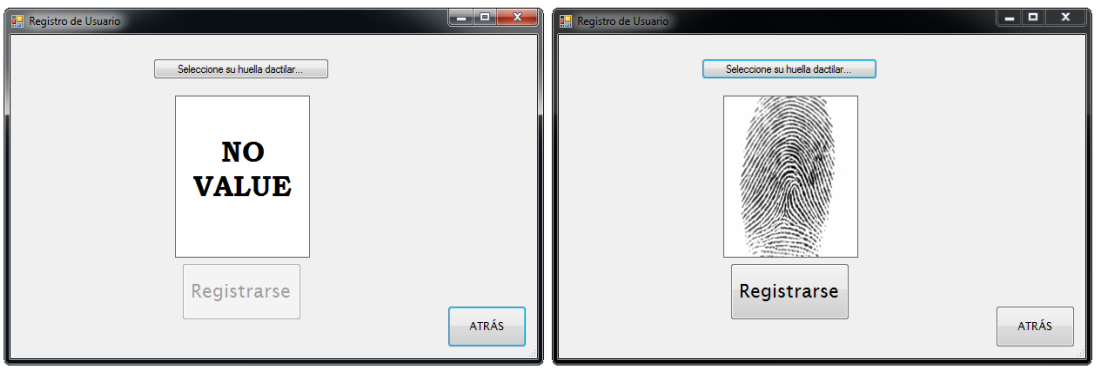

Figura 32: Ventana de Registro Usuario del ordenador

Cuando se hace click sobre el botón *registrarse*, se establece la comunicación y hasta que el móvil no se conecte al ordenador, no se iniciará la operación.

A continuación, en la Figura 33, se muestra el diagrama de flujo de esta funcionalidad de la interfaz.

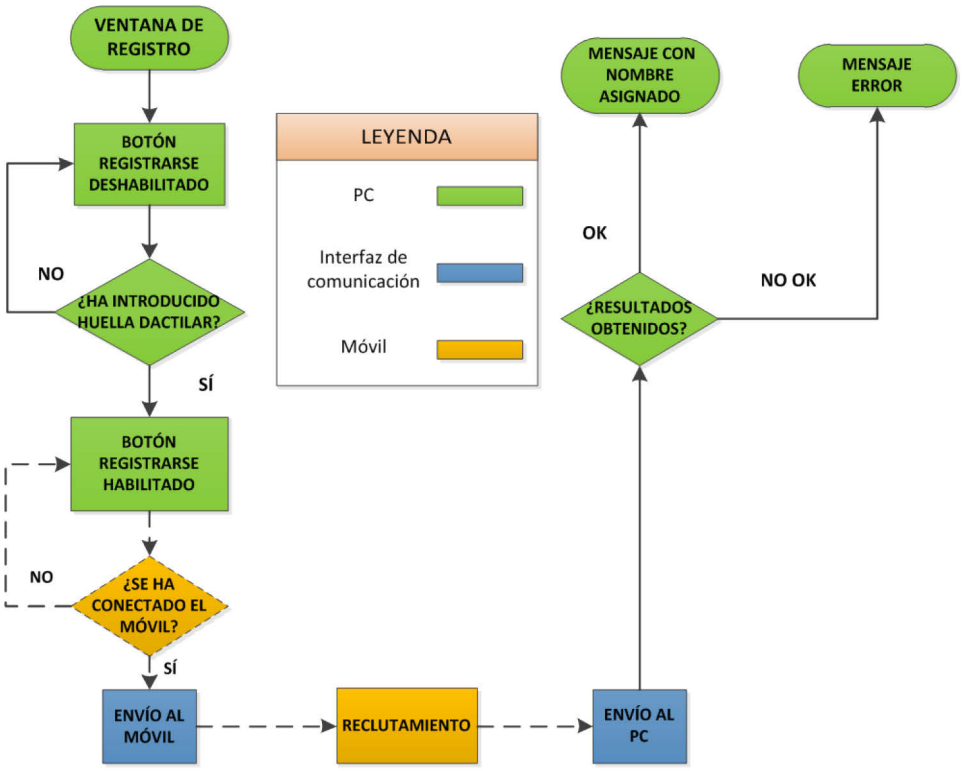

Figura 33: Diagrama de flujo de la ventana Registro Usuario

Ventana Crédito/Débito

Estas funcionalidades utilizan por debajo la funcionalidad biométrica de *verificación de usuarios*, donde es necesario tanto el número de usuario como la huella dactilar de éste para autenticarle.

Las ventanas de *crédito* y *débito* son muy similares, la única diferencia es que la primera consiste en ingresar una cantidad de dinero mientras que la segunda consiste en extraerla. Este apartado se centrará en ésta última.

Respecto al diseño de la interfaz, podemos verlo en la Figura 34. La ventana está formada por dos bloques de texto a rellenar así como la opción de elegir la huella del usuario. Del mismo modo que en la funcionalidad de *registro,* hasta que el usuario no ha rellenado los tres campos, no se le permite realizar la acción.

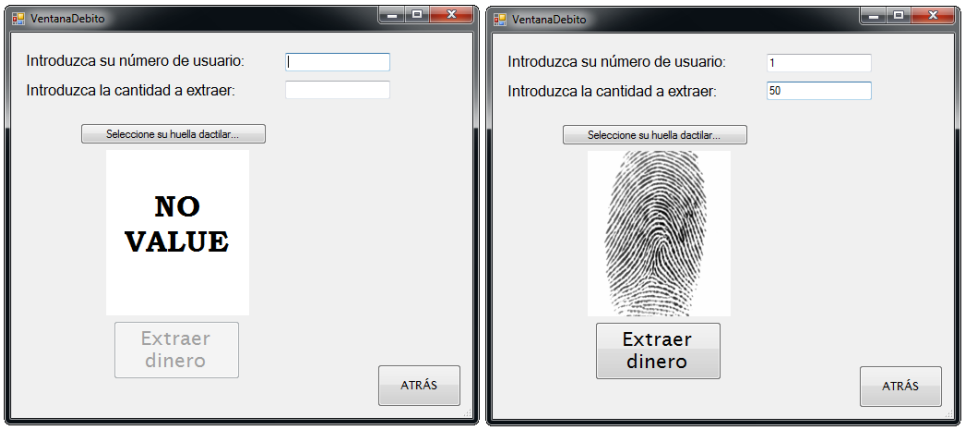

Figura 34: Ventana de Débito del ordenador

Considerando la lógica de la ventana, una vez que el usuario ha completado todos los campos, éste pulsa el botón *extraer dinero.* Para que se realice la operación, el móvil debe conectarse al ordenador. A partir de aquí, los datos introducidos se envían al móvil, quien realiza la operación biométrica *verificación*, gestiona el saldo del usuario y retorna los resultados. En esta funcionalidad, es necesario comprobar también que la cantidad de dinero que introduce el usuario es menor que el saldo actual de éste. Dependiendo de los resultados obtenidos, el ordenador mostrará un tipo de mensaje u otro. Cabe destacar que en la operación opuesta *crédito,* no hay que comprobar este último paso. Para facilitar la comprensión de este proceso, se incluye Figura 35.

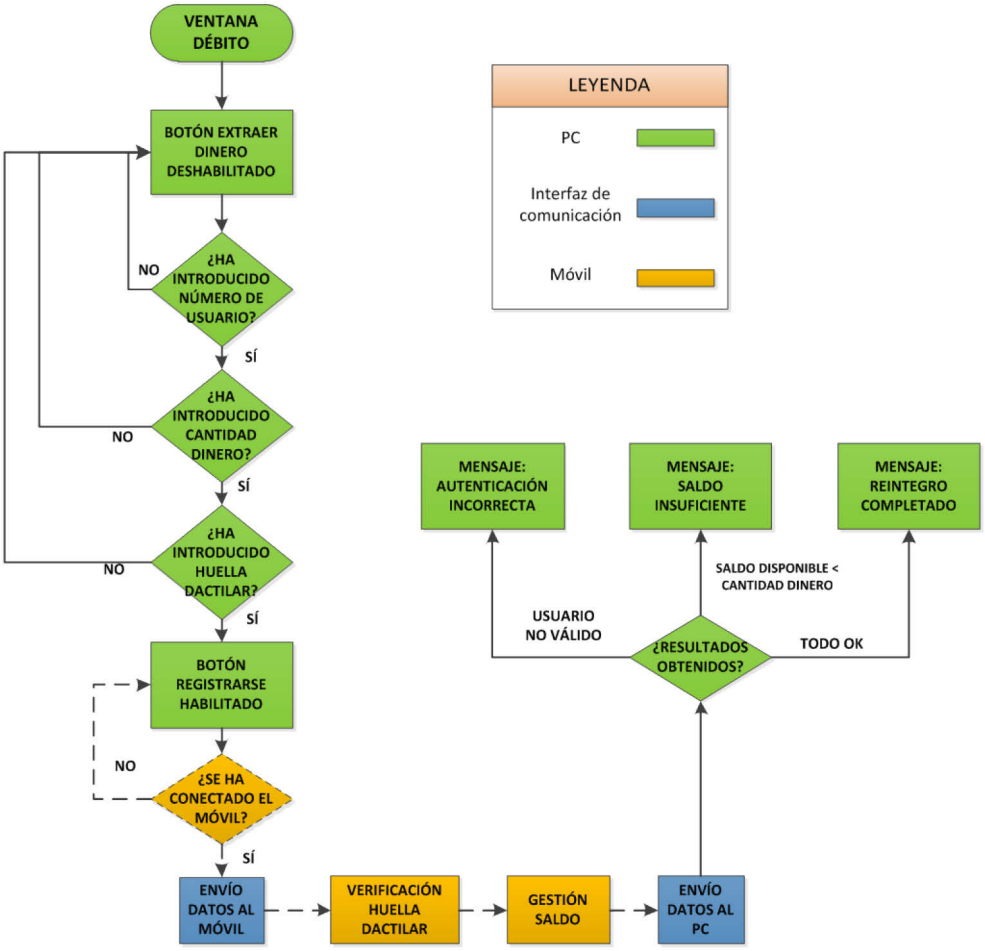

Figura 35: Diagrama de flujo de la ventana Débito

Ventana Saldo

Esta es la ventana más sencilla de la interfaz ya que no se requiere verificar al usuario para ejecutarla, sólo se necesita el número de éste. Una vez introducido, se manda al móvil y éste retorna el saldo del usuario. El diseño que tiene la ventana se puede ver en la Figura 36.

Figura 36: Ventana Mostrar Saldo del ordenador

A continuación se muestra el diagrama de flujo de esta operación (Figura 37).

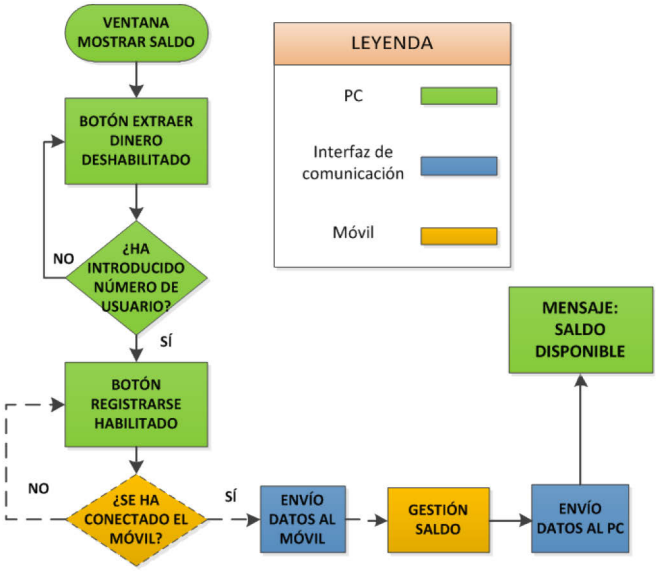

Figura 37: Diagrama de flujo de ventana Mostrar Saldo

6.1.2. Aplicación en el dispositivo móvil

En este apartado se detallará cómo se ha implementado la aplicación usuario en el dispositivo móvil, los problemas encontrados y sus soluciones. Para desarrollar correctamente la aplicación que interaccione con el usuario son necesarios dos módulos concretos: la gestión del saldo disponible de los usuarios y la creación de la interfaz gráfica.

Gestión del saldo de los usuarios

Del mismo modo que en la gestión de números de usuario realizada en el ordenador, para el manejo del saldo se evaluaron varias alternativas. Debido a la gran similitud entre Java y C#, las mismas ideas que se rechazaron en el ordenador, bases de datos y ficheros de textos, fueron desechadas aquí. La decisión vencedora fue utilizar una clase en Android llamada *SharedPreferences* que, similar a la idea utilizada en C#, permite almacenar diferentes tipos de datos de la aplicación de forma persistente.

El mayor problema encontrado aquí es que *SharedPreferences* no permite gestionar colecciones de números, y los saldos son exactamente eso. En cambio, sí que posibilita la interacción con cadenas de Strings. La solución elegida fue convertir cada saldo a String y almacenarlo en un array de ellos. Para acceder a ellos, se realiza el proceso inverso.

Por otro lado, surgía un nuevo problema: teniendo en el dispositivo móvil la colección de saldos y el número introducido por el usuario, ¿cómo se puede saber de toda la colección de saldos cual es el que le corresponde a este número? La solución encontrada es muy simple: dado que este número es un índice, los saldos son un array de enteros y no se permite suprimir usuarios ni saldos, realizando la sentencia

$$saldoUsuario = arraySaldos\ [numeroUsuario]$$

se asocia al usuario con su correspondiente suma de dinero.

Por último, una vez que se conoce como acceder a la variable, es necesario estudiar cómo se gestionará el saldo en las diferentes funcionalidades de *registro*, *crédito/débito* y *mostrar saldo*. En la primera, cada vez que un usuario nuevo se registra, se añade una nueva posición en la colección con la cantidad inicial de 1000 euros. En *crédito/débito* se actualiza el saldo correspondiente al usuario dependiendo de la operación. En *Mostrar saldo*, simplemente se accede a la posición exacta del usuario determinado y se envía su valor al ordenador.

Interfaz

En este caso, la interfaz que interactúa con el usuario es mucho más simple ya que el objetivo del dispositivo móvil está más focalizado en recibir las órdenes del ordenador y ejecutar las operaciones biométricas pertinentes. Por lo tanto, esta interfaz es más informativa y el usuario interacciona muy poco con ella.

La primera decisión tomada fue que existiera sólo una ventana con dos botones, uno que permitiera ver el saldo disponible del usuario y otro para iniciar la comunicación con el ordenador.

La lógica de la aplicación depende de que botón se pulse. Si es el primero, se le pedirá al usuario introducir su número asignado para mostrarle su saldo disponible. Como se puede observar, este botón es informativo y se implementó con la idea de que el usuario no necesitara iniciar una comunicación entre el ordenador y el móvil para obtener su saldo. El segundo botón es necesario pulsarlo cuando se quiera realizar las funcionalidades de *registro, crédito, débito* o *mostrar saldo* en el ordenador. Este botón hace que el móvil se conecte al ordenador ya que es necesario establecer una comunicación cada vez que se quiera enviar datos entre el PC y el móvil. Una vez que se conecta, comenzará a recibir la información desde el ordenador, realizará la funcionalidad biométrica pertinente, actualizará la posición determinada de la colección de saldos y retornará los resultados al PC, a la vez que son mostrados por medio de un mensaje en el propio móvil.

Respecto a la interfaz, se adjunta su apariencia en la Figura 38 si se pulsa el botón *Mostrar saldo*.

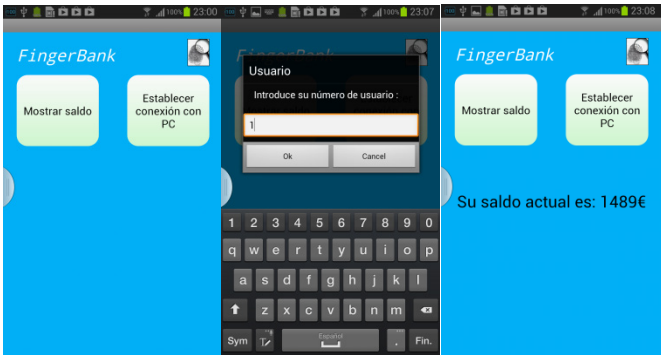

Figura 38: Flujo de ventanas de Mostrar Saldo en dispositivo móvil

Para el caso en el que se pulse el botón *Establecer conexión con PC*, el móvil comenzará a recibir información y ejecutar las operaciones pertinentes. Dado que el usuario no interacciona

en este punto con el sistema, se le mostrará simplemente una ventana de espera y las funcionalidades que se van haciendo. Finalmente, cuando el móvil termina, aparece una imagen con el resultado obtenido de la operación biométrica y un mensaje informando que los datos han retornado al ordenador. A continuación, en la Figura 39, se incluye las ventanas correspondientes al proceso de la operación *crédito.*

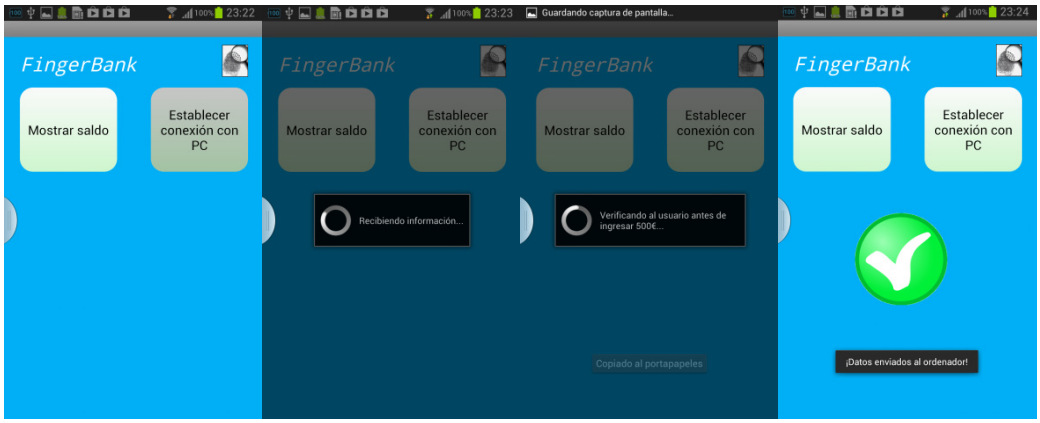

Figura 39: Flujo de ventanas de Establecer conexión con PC

Para finalizar, es necesario comentar que este módulo ha sido el más sencillo de implementar, pese a los problemas acontecidos. Fue el último bloque desarrollado por lo que la mayoría de los errores estaban ya corregidos. Además, gracias a las herramientas gráficas que ofrecen los entornos Visual Studio C# y ADT Android en Eclipse, la creación de interfaces se simplificó considerablemente.

6.2. SISTEMA BIOMÉTRICO

En este apartado se explicará en detalle los problemas encontrados a la hora de implementar las funcionalidades biométricas basándose en la especificación BioAPI 4.0.

Es imprescindible comentar que tal y como se ha podido ver a lo largo de este documento, el sistema biométrico se compone de dos partes diferenciadas. Por un lado, se encuentra el BSP gestionando Unidades definidas utilizando la norma BioAPI. Por otro lado, están los algoritmos de procesamiento y verificación, MINDTCT y BOZORTH3, respectivamente. La creación de estos algoritmos quedaba fuera del objetivo del proyecto por lo que fueron proporcionados ya implementados en Android por el grupo de investigación. Sin embargo, hubo que adaptarlos a los objetivos de este TFG. Por lo tanto, cuando se hable de ambos algoritmos en conjunto con el BSP, se presupone que estos funcionan y ya han sido adaptados correctamente.

Por lo tanto, esta sección abordará la creación del BSP y de sus respectivas Unidades. Además, dado que la propuesta para la versión 4.0 de la norma se centra principalmente en la transmisión de información biométrica entre las diferentes partes del sistema utilizando la estructura BIR, se explicará cómo se implementa esta estructura y sus métodos más importantes.

A diferencia de los apartados anteriores, en esta sección no se han tomado decisiones sobre qué procedimiento elegir ya que hay que basarse rigurosamente en la especificación BioAPI y ésta no da opción a la duda.

6.2.1. BIR

Uno de los requisitos más importantes que se definieron en el apartado 5.3, es que la creación de los BIRs se basara en el formato CBEFF 19875. Según se vio en la sección 3.6, este formato establece el tipo de datos que debe contener cada BIR. Así, esta estructura estaría formada por una cabecera, los datos biométricos y un bloque de seguridad. Este último bloque se obviará ya que todavía se está definiendo esa parte de la norma. Además, según obliga la norma BioAPI, la transmisión de imágenes entre dispositivos se debe realizar mediante BIRs, es decir, aunque se ha mencionado que se envían imágenes de huellas dactilares entre las dos máquinas, en realidad se envían BIRs que encapsulan esas imágenes en su interior. Por lo tanto, el ordenador se encargará de crear el BIR, encapsular la imagen y enviarlo. El móvil lo recibirá y ejecutará las funcionalidades necesarias. Lo que se pretende con esto es mejorar la compatibilidad entre dispositivos definiendo una estructura única que todos deben cumplir.

Para implementar esta estructura en código Android o C#, el BIR se representa en una clase. Los tipos de datos del BIR serán las propiedades de esta clase y para acceder a ellos se utilizarán los métodos de acceso *getters* y *setters.* El constructor asigna los diferentes tipos de datos de la cabecera SDH dependiendo del contexto de la aplicación. En consecuencia, cuando el BSP instancie esta clase se obtendrá un BIR con sus campos ya rellenados correctamente dependiendo de la operación elegida. Lo único necesario, será introducir *a posteriori* en el campo *BDBData* la imagen de la huella dactilar (creando un BIR en crudo) o el array de minucias (creando un BIR procesado).

A modo de ejemplo, se mencionan las propiedades más importantes de la cabecera que se asignan en el contexto de este sistema biométrico.

- **Biometric Type**: dado que es un sistema basado en huellas dactilares, este campo se rellena como un tipo de dato *Finger*.
- **Biometric Subtype**: subtipo de huellas dactilares, *RightFinger*.
- **BDB/BIR Index**: se rellena con el número que el usuario introdujo en el ordenador. Por tanto, cuando se hable del nombre del BIR será equivalente al nombre del usuario.
- **BDB/BIR Creation Date**: se escribe la fecha real de creación del BIR. Para ello se utiliza una biblioteca de C# y Java que proporciona la hora, el día, el mes y el año actual.
- **Processed Level**: Dependerá de dónde se encuentre el BIR. Cuando se cree en el ordenador, antes de ser enviado al móvil, será *Raw* (crudo, en inglés), mientras que cuando se procese la imagen en el dispositivo móvil, este campo pasará a ser *Processed.*
- **Purpose**: dependerá de la acción que vaya a realizar el sistema, si se debe reclutar a los usuarios, se rellenará con *Enrol*. Si por el contrario se quiere verificar dos huellas dactilares, *Verify*.

Como se pudo ver en el apartado de BioAPI, existen más campos que se asignan a un BIR pero estos son los más importantes y representativos.

Otro cambio en el que insiste esta nueva versión de la norma es que en la funcionalidad biométrica de *reclutamiento* se almacenen en el directorio del móvil BIRs que contengan minucias encapsuladas, en lugar de las propias minucias. En el capítulo de diseño, se ha hablado de guardar minucias en lugar de BIRs porque facilita más la comprensión y no hay diferencias significativas entre las dos opciones. Asimismo, la especificación define que los BIRs se deben almacenar en forma de array por lo que se necesita implementar un método que convierta de BIR a array, llamado *BIRToArray().* Por otro lado, la norma determina que el BSP a la hora de realizar operación de *verificación* obtendrá del directorio este array de BIR. En consecuencia, se debe añadir otro método complementario que transforme el array a una estructura BIR para luego poder trabajar con él: *BIRFromArray()*.

Estos métodos se implementan en el dispositivo móvil Android y para crearlos hay que basarse en la propiedad que dicta que todo tipo de variables tienen un tamaño fijo de bytes en los lenguajes de programación. Así, aunque el BIR esté formado por diferentes tipos de datos, estos siempre se pueden convertir a bytes utilizando las máscaras necesarias. Del mismo modo, se puede utilizar un método complementario para pasar de bytes a diferentes tipos de datos definidos en la estructura BIR. Se considera que queda fuera del ámbito de este escrito explicar en detalle en qué consisten estos métodos ya que se complicaría mucho el documento.

Esta clase fue la primera en implementarse debido a que el BSP y sus Unidades intercambiaban BIRs entre sus diferentes módulos. Aunque parece que introduciendo el concepto de BIR se dificulta mucho el manejo de datos biométricos, realmente este concepto es clave en el hecho de proporcionar una compatibilidad entre dispositivos y diferentes partes del sistema, consiguiéndose que se logre transmitir siempre el mismo tipo de estructura con el mismo tipo de datos en las mismas posiciones, siendo el interior de éstos lo único variable.

6.2.2. BSP

En las siguientes líneas se detallará al lector la implementación del BSP y sus respectivas Unidades para conseguir ejecutar correctamente las funcionalidades biométricas de *captura, reclutamiento* y *verificación.*

Según lo planteado en el apartado 5.3, la primera operación se simulará en el ordenador mientras que las demás se ejecutarán sobre el dispositivo móvil. Por lo tanto existirán dos BSPs, uno implementado en el ordenador cuya única función será simular la captura de huellas y otro en el dispositivo móvil que se encargará de realizar las operaciones de reclutamiento y verificación. En consecuencia, a continuación el apartado se dividirá en dos módulos: el BSP del ordenador y el BSP del móvil.

BSP en el ordenador

Este BSP se encarga inicialmente de gestionar la captura de huellas y de su posterior encapsulación de la imagen en una estructura BIR. Tras esto, debe transformar este BIR a array utilizando el método *BIRToArray()* para poder enviar estos datos al móvil. En un proceso de

captura real, este BSP necesitaría implementar la Unidad de Sensor. No obstante, en este sistema se simula la lectura de huellas dactilares por lo que esta Unidad desaparece.

En conjunto, es un BSP muy sencillo, que no necesita gestionar ninguna Unidad para realizar su funcionalidad.

BSP en el dispositivo móvil

En este apartado se analizarán los diferentes elementos y Unidades necesarios para desarrollar el BSP que se implementa en el teléfono móvil.

Como se ha detallado anteriormente, el usuario puede elegir entre dos funcionalidades biométricas: reclutar sus huellas a modo de registro o verificar que es quien dice ser. En la norma biométrica, el BSP es quien implementa estos métodos y es dentro de ellos dónde llama a las Unidades necesarias. Por lo tanto, a continuación se separarán las funcionalidades y se explicarán su funcionamiento por separado, comentándose qué tipo de Unidades se gestionará en cada operación y su propósito. En este apartado se presupone que tanto la imagen de la huella dactilar como el número del usuario ya están en el dispositivo móvil.

Reclutamiento

Esta funcionalidad consiste en el procesado de la huella dactilar capturada en el PC así como el almacenamiento de las muestras procesadas en el dispositivo. Por esta razón, el método de reclutamiento (o *Enrol* en inglés), utiliza la Unidad de Procesamiento y la Unidad de Archivo.

La Unidad de Procesamiento es una clase que implementa el método de procesado, quien recibe el BIR en crudo, es decir, con la imagen de la huella que se capturó en el ordenador sin modificar. Este método introduce la imagen de la huella incrustada en el BIR en otro método interno que contiene el algoritmo MINDTCT. Cuando termina de procesar la huella, devuelve las minucias detectadas. A partir de estas minucias, se construye un BIR procesado formado por una cabecera y las minucias extraídas. Finalmente, este método retorna el BIR y el BSP lo recoge. El diagrama de flujo de la Figura 40 muestra este proceso:

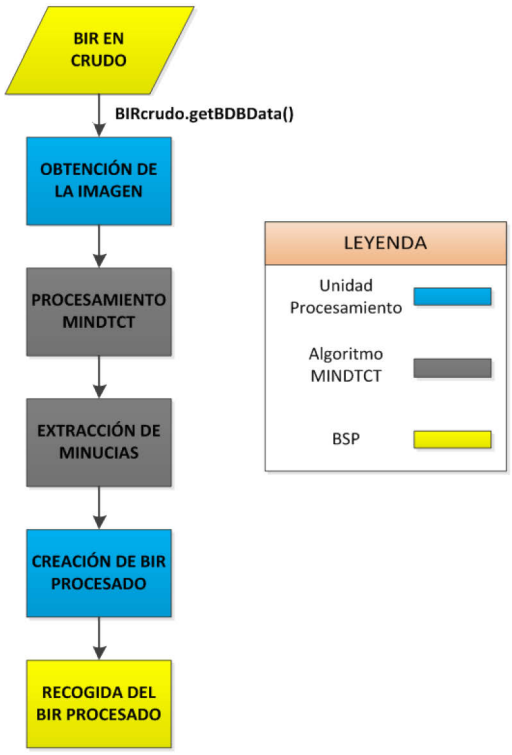

Figura 40: Diagrama de flujo de la Unidad de Procesamiento en funcionalidad Reclutamiento

A continuación, el BSP llama a la Unidad de Archivo para poder guardar este BIR procesado. La Unidad de Archivo recibe el BIR. Para poder escribirlos en el directorio, es necesario convertir la estructura del BIR en un array con el objetivo de almacenarlo en la ruta especificada. Una vez que el BIR se haya guardado, el método devuelve el nombre con el que se guardó para que lo trate el BSP. El BSP recibirá este nombre y lo promoverá hacia la aplicación móvil.

La funcionalidad *reclutamiento de usuarios* ya estaría implementada.

Verificación

Esta funcionalidad estriba en la comparación de la huella dactilar de un usuario con la huella de quien dice ser.

El método de verificación (o *Verify* en inglés) que implementa el BSP, utiliza las unidades de Archivo, Procesamiento y Comparación. Para el correcto funcionamiento, debe recibir de la aplicación una imagen en crudo de la huella dactilar y el número del usuario que se quiere identificar. Ambos campos irán encapsulados en un mismo BIR. Dado que el algoritmo BOZORTH3 trabaja con dos tipos de minucias, el BSP juega con dos BIRs: el BIR procesado que se encuentra en el directorio almacenado y el BIR capturado en el PC.

La primera Unidad necesaria es la Unidad de Archive ya que el BSP necesita cargar el BIR procesado correspondiente al número del usuario que se quiere verificar. Para cargarlo, se utiliza el método inverso de almacenamiento, es decir, se le pasa un nombre y se recibe el array que representa el BIR. Finalmente este array se convierte a una estructura BIR.

El algoritmo de verificación BOZORTH3 trabaja con minucias y no con imágenes en crudo por lo que la segunda Unidad que se utiliza es la Unidad de Procesamiento. Ésta recibe el BIR con la

huella capturada en el PC encapsulada dentro y procesa la imagen como se explicó en la funcionalidad Reclutamiento.

En este momento, entra en juego la Unidad de Comparación que se encarga de recibir ambos BIRs procesados, extraer sus minucias y enviarlas al método que implementa el algoritmo BOZORTH3. Tras hacer la comparación, este método interno devuelve un resultado numérico que la Unidad interpreta y lo convierte a un dato booleano para que el BSP lo pueda manejar. Este proceso se puede ver en la Figura 41: Diagrama de flujo de la Unidad de Comparación en funcionalidad Verificación.

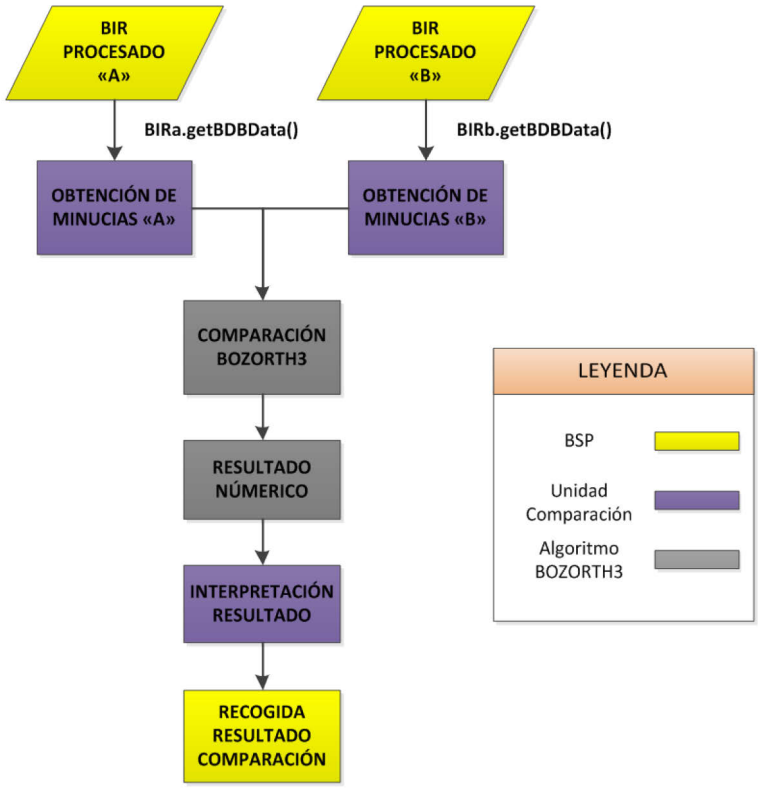

Figura 41: Diagrama de flujo de la Unidad de Comparación en funcionalidad Verificación

Se incluye a continuación, Figura 42, un diagrama de flujo para facilitar la comprensión del proceso, comprimiéndose las operaciones de cada Unidad en una sola caja.

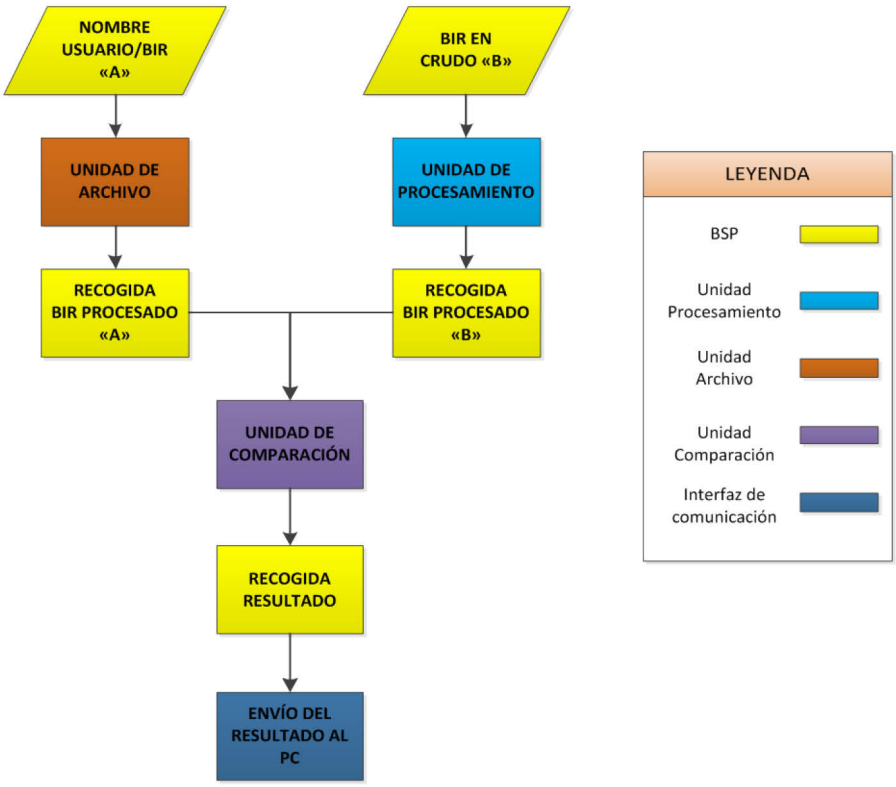

Figura 42: Diagrama de flujo completo de la funcionalidad Verificación

Integrando todas las funcionalidades en una y comprobando su correcto funcionamiento, da como resultado el esperado BSP. Como se puede observar, no ha sido una tarea nada sencilla de realizar ya que para lograr este resultado final, se han pasado por muchas etapas: inicialmente, lo más complicado fue la comprensión absoluta de una norma, no definida totalmente y en constante desarrollo. Una vez que se tuvo una visión global, comenzar a crear código Android desde cero basándose en una especificación dirigida a C# fue una labor sin duda ardua. Al finalizar la implementación de las diferentes Unidades, otro inconveniente crítico que se encontró fue integrar todo el código creado en uno solo. Aunque por separado, todas las funcionalidades funcionaban correctamente, al juntarlas en el BSP, surgieron nuevos errores que hubo que solventar.

Es por ello que la implementación del BSP basándose en la propuesta para la norma BioAPI 4.0 sobre un dispositivo móvil Android ha sido sin lugar a dudas el mayor rompecabezas de resolver.

6.3. COMUNICACIÓN ORDENADOR-MÓVIL

Recapitulando el diseño de la comunicación elegida en el apartado 5.5.3 entre el ordenador y el móvil, ésta se basará en la tecnología Wi-Fi utilizando el protocolo de transmisión TCP. Llegados a este punto, sólo queda desarrollar cómo se implementará este protocolo, es decir, definir las reglas y normas de la transmisión. Una vez definidos, se incluirá una breve descripción de cómo funciona la interfaz de comunicación de este protocolo.

6.3.1. Diseño del protocolo TCP

En una conexión TCP es habitual que la arquitectura del sistema sea cliente-servidor puesto que otro sistema diferente causará muchos problemas a la hora de implementarlo. En este tipo de arquitectura, el servidor se encarga de crear la conexión y escuchar la red hasta que un cliente se conecta y realiza las peticiones. En un primer momento, se pensó erróneamente en utilizar el móvil como servidor y el PC como cliente. Esta idea se rechazó al caer en la cuenta de que un dispositivo móvil funcionando como servidor tendría un consumo excesivo de la batería. Además, el dispositivo móvil solicita al ordenador que le envíe la imagen de la huella por lo que los roles deben ser móvil-cliente y ordenador-servidor.

En las siguientes líneas, se procede a diseñar el protocolo de comunicación usado. Debido a que existen cuatro operaciones diferentes en la aplicación servidor (mostrar saldo, crédito, débito y registro de usuarios), el protocolo variará dependiendo de qué funcionalidad se quiera utilizar.

Antes de comenzar, es necesario explicar que para recibir datos TCP es necesario conocer *a priori* el tamaño de la trama que contiene esos datos. En el caso de que sea un campo de tamaño variable, como lo es la imagen de la huella dactilar, será necesario transmitir antes un campo con su peso. La imagen se encapsula dentro del BIR y es éste en forma de array el que se envía del ordenador. Todos los demás campos adicionales tendrán una dimensión fija que conocerán anticipadamente el cliente y el servidor.

Inicialmente, el ordenador necesita advertir al móvil sobre la operación que se llevará a cabo para que éste pueda interactuar correctamente con todos los datos recibidos. Por ende, aparte del tamaño del BIR que contiene la imagen, también se requerirá un campo con la operación elegida por el usuario.

Respecto al cuerpo del mensaje que se enviará depende de la funcionalidad seleccionada por el usuario. Si elige *registro de usuarios*, el dispositivo móvil necesitará únicamente el BIR con la imagen de la huella dactilar. Si su elección es la opción de *crédito* o *débito*, el móvil precisará de la imagen, el número de usuario (encapsulado dentro del BIR) y la cantidad de dinero a ingresar o restar. Por último, si opta por la funcionalidad de *mostrar saldo*, sólo será necesario enviar el número de usuario como un campo más ya que esta opción no requiere verificación. En la siguiente Figura 43 se puede ver las diferentes tramas a enviar.

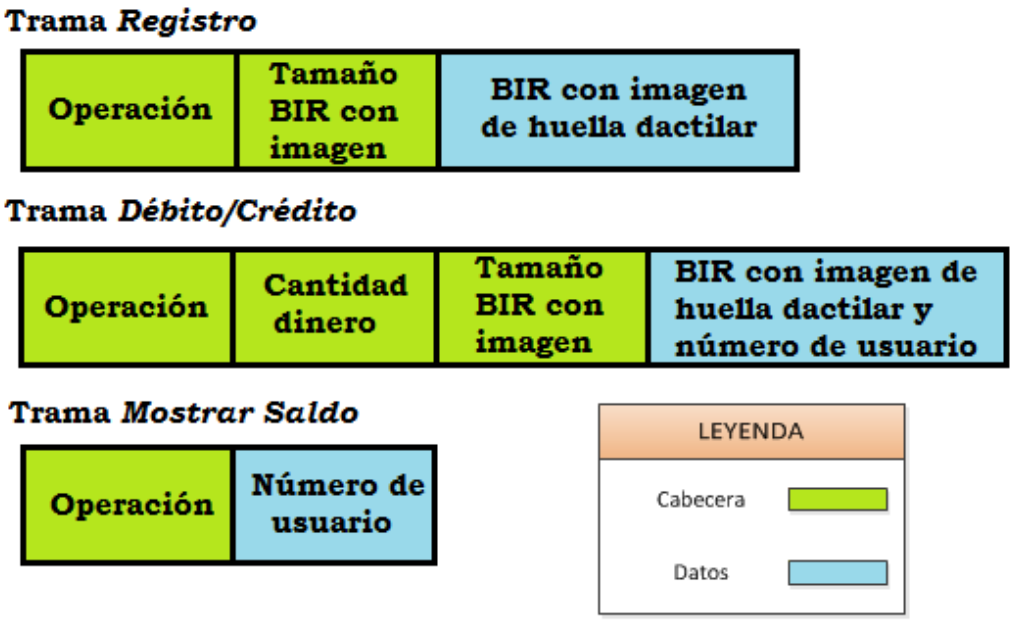

Figura 43: Representación del tipo de trama en función de la operación elegida por el usuario

El móvil, dependiendo de la operación recibida en el datagrama se comportará de una u otra manera ejecutando las operaciones pertinentes. Cuando termine, retornará un segmento con un solo campo representando el resultado (semejante a un ACK) y el ordenador sabrá cómo manejarlo.

Como se puede observar, se conoce de antemano el tamaño de todos los campos exceptuando el de la imagen. En consecuencia, tanto el ordenador como el móvil saben a priori estos tamaños y su orden en el datagrama por lo que pueden tratar los datos de forma correcta.

En este instante, ya estaría definido el protocolo. Para ver su funcionamiento dentro de la red, se añade la Figura 44, que muestra la comunicación entre el servidor y el cliente.

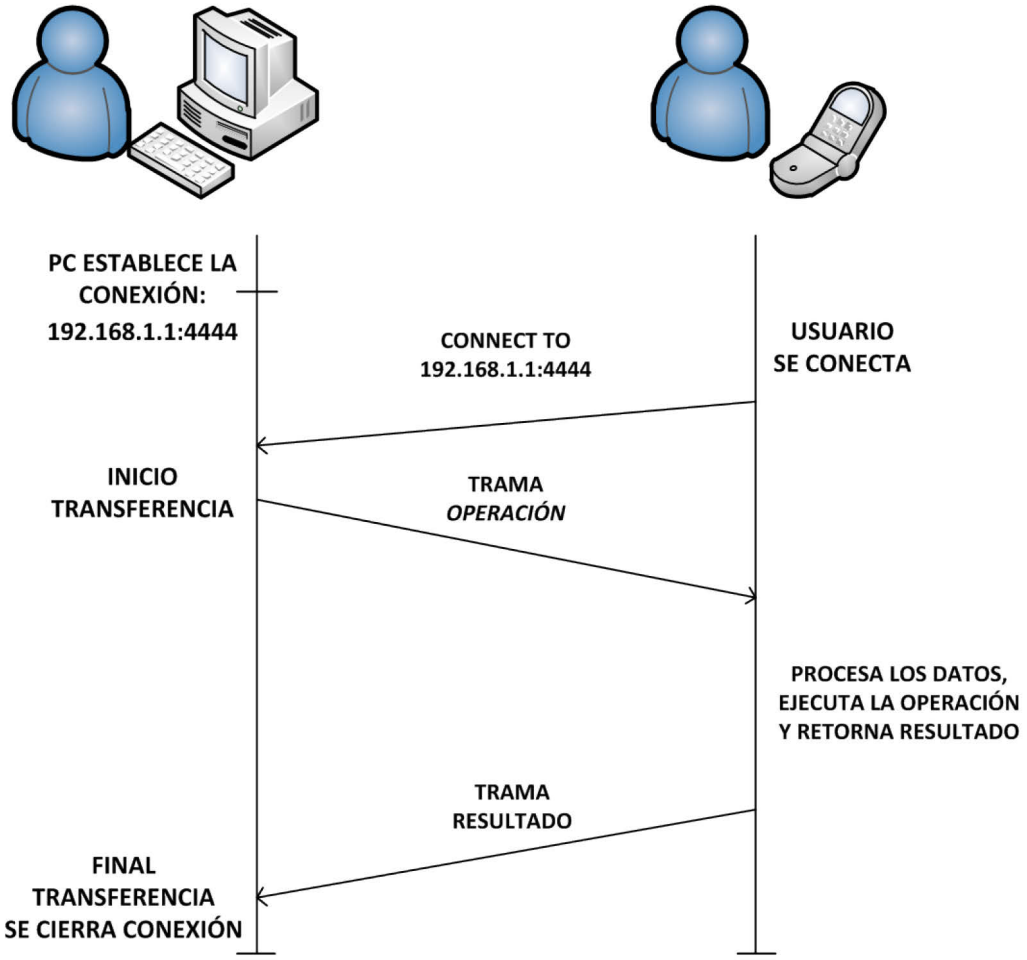

Figura 44: Funcionamiento del protocolo de transmisión

Como se puede observar, el servidor establece la conexión y el móvil se conectará cada vez que se quiera realizar una operación. El ordenador inicia la transferencia enviando el tipo de trama determinado dependiendo de la función elegida por el usuario. El móvil la recibe, la interpreta y ejecuta la operación pertinente, retornando un resultado. Este resultado es similar a un ACK y le dirá al ordenador el resultado de la transmisión, de la operación biométrica y de la gestión de saldo. Finalmente, el servidor cierra la conexión.

6.3.2. Implementación de la interfaz de comunicación entre Android y C#

En este apartado se describirá como se ha implementado la interfaz de comunicación en el ordenador y en el móvil. Esta interfaz es la capa que se encarga de enviar las tramas e interpretar los resultados, tanto en el ordenador como en el dispositivo móvil. En la Figura 45, se puede observar las diferentes interfaces del sistema al completo.

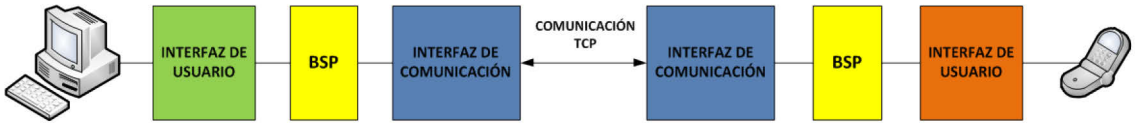

Figura 45: Diferentes interfaces del sistema

A continuación se dividirá el apartado en dos procesos: el envío de la trama de operación desde el ordenador y la recepción en el móvil (dirección descendente), y la transmisión de la trama que contiene el resultado desde el móvil junto a la posterior recepción en el ordenador (dirección ascendente).

Dirección descendente de la comunicación TCP

A continuación explicará los pasos que sigue el proceso desde que el usuario elige una operación monetaria hasta que el móvil la interpreta correctamente una vez recibida. Para facilitar la comprensión, se incluye la Figura 46, donde se omite la presencia de los BSPs.

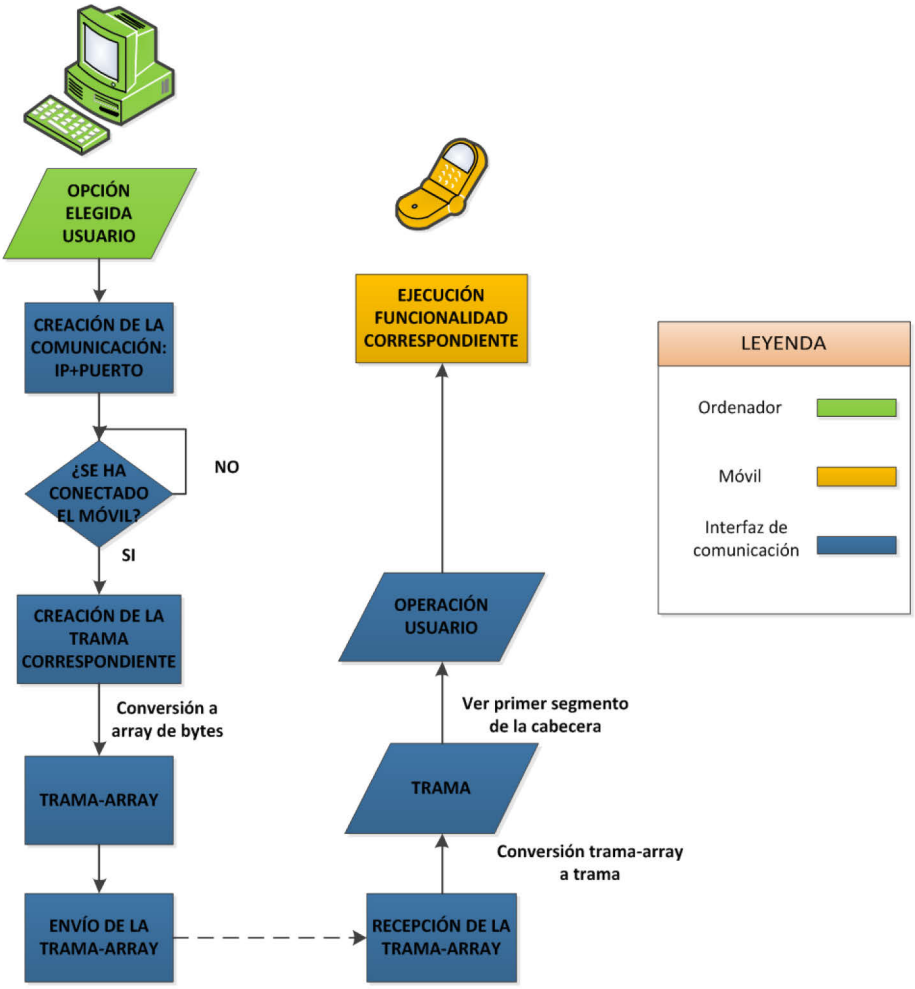

Figura 46: Diagrama de flujo de la interfaz de comunicación en el sentido descendente

En primer lugar, el ordenador instancia la interfaz e inicia la comunicación en la red. Para ello, el servidor debe definir una dirección IP y un puerto TCP. Una vez que éste ha iniciado la conexión, el móvil se debe conectar y establecer la comunicación. A continuación, el ordenador puede proceder a crear la trama correspondiente a la opción elegida por el usuario. Hay que mencionar que la comunicación se basa en el uso de *Sockets* y que estos utilizan array de bytes para intercambiar información. Finalmente, el ordenador envía este array de bytes al dispositivo móvil. Dado que el protocolo está definido antes de iniciar la comunicación, el móvil sabe cómo interpretar los datos para acceder a la cabecera y ejecutar las operaciones solicitadas.

Dirección ascendente de la comunicación TCP

En esta sección se explicará el proceso que sigue desde que el móvil se prepara a enviar los resultados hasta que el ordenador los interpreta. Como en el apartado anterior, se incluye un diagrama de flujo (Figura 47).

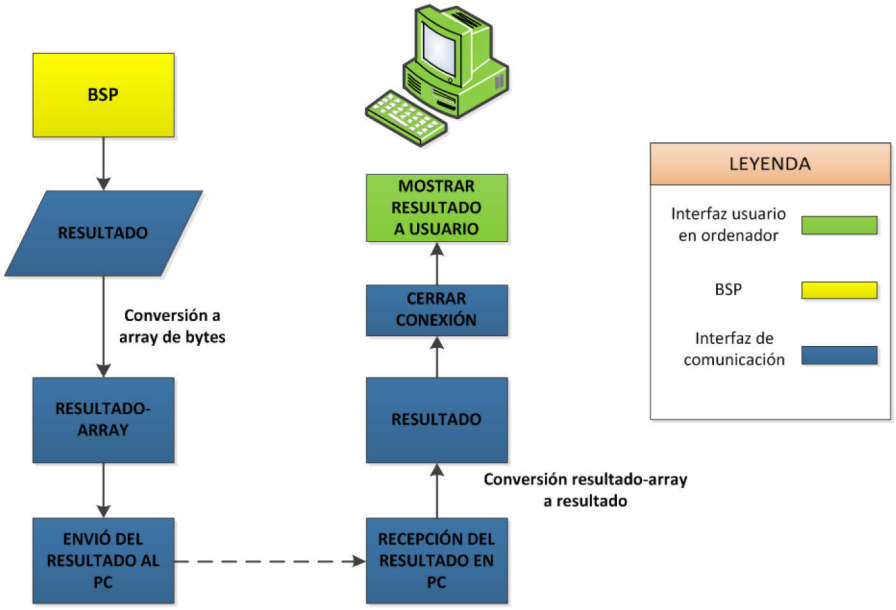

Figura 47: Diagrama de flujo de la interfaz de comunicación en el sentido ascendente

En la recepción, la interfaz de comunicación en el móvil recibe el resultado del BSP en forma de número. Dependiendo de la funcionalidad, este número puede variar. A continuación se muestran los diferentes valores que puede tener en función de la operación elegida.

- Registro de usuario:
  - Reclutamiento inválido: $resultado = -1$
  - Reclutamiento válido: $resultado = numeroUsuario$
- Crédito:
  - Verificación de usuario inválida: $resultado = -1$
  - Verificación correcta, dinero ingresado: $resultado = 0$
- Débito:
  - Verificación de usuario inválida: $resultado = -1$
  - Verificación correcta pero saldo insuficiente: $resultado = -2$
  - Verificación correcta, dinero extraído: $resultado = 0$
- Mostrar saldo:
  - Saldo disponible: $resultado = saldoDisponible$

Dado que la conexión está todavía establecida, el móvil lo envía sin retardos. En este caso no se envían más campos: sólo este número en forma de array de bytes. Cuando llega al ordenador, éste conoce cuál fue la funcionalidad requerida por lo que sabe cómo interpretar el valor y enviárselo a la interfaz de usuario para que lo muestre por pantalla al usuario. En un último paso, el ordenador cierra la conexión con la red.

La implementación de este protocolo fue una tarea complicada a la hora de elegir qué cabeceras introducir para no sobrecargar la trama total. En conjunto, este módulo se tardó tiempo en construir debido al gran número de decisiones que se tuvieron que tomar.

7. PRUEBAS

En este capítulo se detallan las pruebas realizadas para comprobar el funcionamiento del sistema biométrico desarrollado verificando implícitamente que la implementación de la propuesta al estándar BioAPI 4.0. funciona correctamente. Para ello, se propondrán diferentes casos que probarán el sistema y se analizará el resultado obtenido. Esta evaluación se realizará sobre un ordenador y un dispositivo móvil Android.

El apartado se dividirá en dos tipos de pruebas: las pruebas de funcionalidad y las de rendimiento del sistema. Las primeras consisten en comprobar el correcto funcionamiento de la aplicación suponiendo diferentes entradas del usuario. En las segundas se analizarán los tiempos y eficiencia del sistema completo.

Por último, se compararán los resultados obtenidos en un teléfono móvil con los alcanzados en un ordenador para observar las diferencias de rendimiento entre ambos dispositivos.

7.1. PLATAFORMAS Y TIPO DE IMÁGENES EMPLEADAS PARA LAS PRUEBAS

Plataformas

Como ya se ha comentado, se precisa para las pruebas del sistema de un ordenador y de un móvil.

Para obtener los tiempos de transmisión de la comunicación se ha utilizado un PC con las siguientes características:

- **Procesador**: Intel® Core™ 2 Duo T6500 @ 2.10 GHz
- **RAM**: 3,00 GB
- **Disco duro**: 200 GB
- **Tarjeta gráfica**: 512 MB

Para conseguir los tiempos de los algoritmos de procesamiento y comparación MINDTCT y BOZORTH3 se usó un PC con las siguientes características:

- **Procesador**: Intel Core 2 Duo E6850 Processor (3.0 GHz, 4 MB L2 cache, 1333 MHz FSB)
- **RAM**: 4,00 GB
- **Disco duro:** 250 GB
- **Tarjeta gráfica:** 256 MB

En el caso del móvil se ha dispuesto de un móvil Sony Xperia U, cuyas características son:

- **Procesador:** NovaThor U8500 dual-core @ 1GHz
- **RAM:** 512 MB
- **Memoria total:** 4 GB
- **Conectividad Wi-Fi:** 802.11b/g/n

Tipo de imágenes

Para las pruebas realizadas se han utilizado un tipo de imagen .PNG en escala de grises con dimensiones de 388 x 374, por lo que el tamaño total es de 145112 bytes. Se disponen de nueve imágenes del mismo candidato, capturadas en distintos momentos, siendo un total de 4 usuarios diferentes, por lo que tenemos en total una base de datos de 36 imágenes.

7.2. PRUEBAS DE FUNCIONALIDAD

Una vez que el sistema está completamente implementado, se debe comprobar su funcionamiento. Dado que existe una interfaz con la que el usuario puede interactuar, se evaluarán las diferentes opciones por medio de ésta. A continuación se plantean los diferentes casos límite a los que se expone al sistema.

7.2.1. Reclutamiento del primer usuario

La primera prueba consiste en el registro del primer usuario dentro del sistema.

Resultado esperado

El resultado esperado es la correcta creación del directorio en la ruta de la SD externa

"/data/huellasprocesadas/"

en el dispositivo móvil, un reclutamiento válido de su huella dactilar así como la asignación del número de usuario "1" en el ordenador.

Resultado obtenido

La Figura 48 se corresponde con la interfaz del ordenador y se puede comprobar que tanto el reclutamiento como la asignación del número de usuario fueron correctos.

Figura 48: Resultado obtenido en el ordenador en el caso de *primer usuario registrado*

En la segunda imagen (Figura 49), se puede comprobar que la creación del directorio en el dispositivo móvil fue correcta y que contiene el BIR con las minucias creadas correctamente, en formato .bir. Esta comprobación se realizó de manera manual, observando varios BIRs en un procesador de texto hexadecimal y comprobando byte a byte que el resultado era correcto.

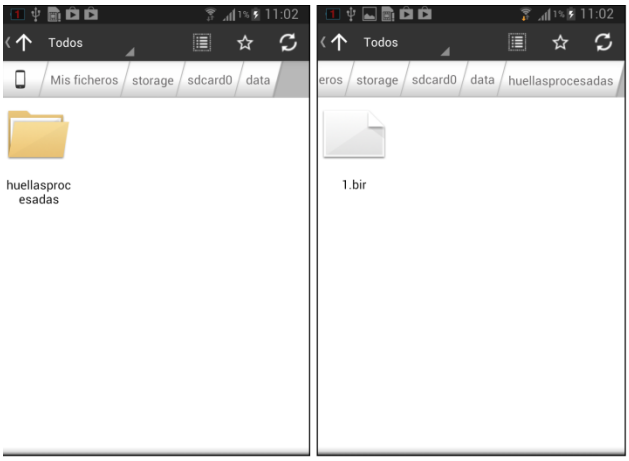

Figura 49: Resultado obtenido en el dispositivo móvil en el caso de *primer usuario registrado*

7.2.2. Verificación correcta del usuario

En este apartado, se comprobará que el usuario antes registrado es quien dice ser. Para ello se utilizará la opción bancaria de *Crédito,* que necesita verificar al usuario antes de ingresar el dinero requerido. Dado que en el directorio de huellas dactilares del ordenador existen diferentes muestras para un solo usuario, utilizaremos dos huellas diferentes para además comprobar el correcto funcionamiento de los algoritmos.

Resultado esperado

Se espera que la verificación sea correcta, permitiendo al usuario ingresar el dinero requerido. Para ello, se mostrará tanto el resultado obtenido en la interfaz del ordenador como el saldo disponible antes y después de la operación en el dispositivo móvil.

Resultado obtenido

La Figura 50 se corresponde con la ventana donde el usuario introduce la huella dactilar para la verificación junto con el resultado obtenido en el ordenador.

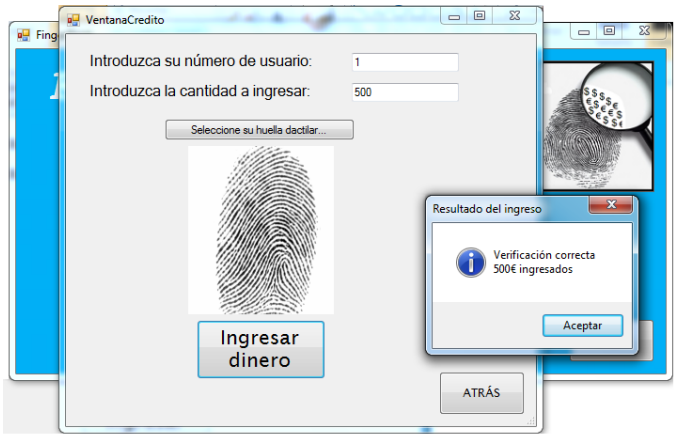

Figura 50: Resultado obtenido en el ordenador en el caso de *verificación correcta*

La Figura 51 se corresponde al saldo antes y después de la operación en el dispositivo móvil.

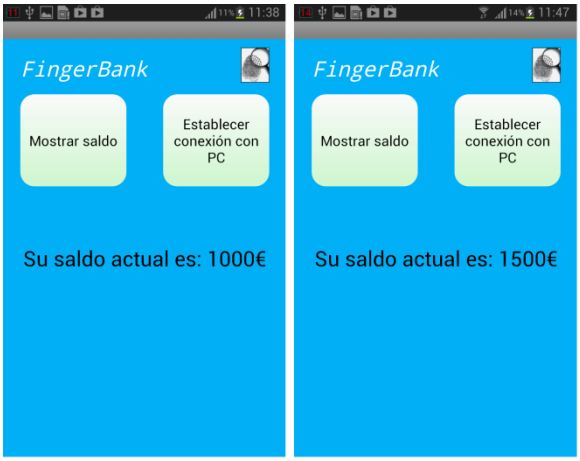

Figura 51: Resultado obtenido en el dispositivo móvil en el caso de *verificación correcta*

7.2.3. Verificación incorrecta del usuario

Del mismo modo que antes y usando la misma operación bancaria, se comprobará que ocurre si se elige una huella dactilar de otro usuario.

Resultado esperado

Se espera que el sistema no permita al usuario realizar la operación de *Crédito* dado que la comprobación del algoritmo de comparación BOZORTH3 invalidará al usuario.

Resultado obtenido

Al igual que en el caso anterior, la Figura 52 se corresponde con la ventana donde se introduce la huella así como el resultado obtenido después de verificarla.

Figura 52: Resultado obtenido en el ordenador en el caso de *verificación incorrecta*

No se añade una imagen sobre la interfaz del móvil ya que muestra el mismo saldo que en el caso anterior debido a que la operación no se ha realizado.

7.2.4. Funcionalidad *Débito* cuando el saldo disponible es insuficiente

Debido a que el usuario puede introducir una cantidad de dinero más grande que su saldo disponible, en este apartado se comprobará el funcionamiento de esta operación.

Resultado esperado

El resultado esperado es que el sistema no permita extraer el saldo avisando al usuario.

Resultado obtenido

La Figura 53 representa la respuesta del ordenador al efectuar la operación:

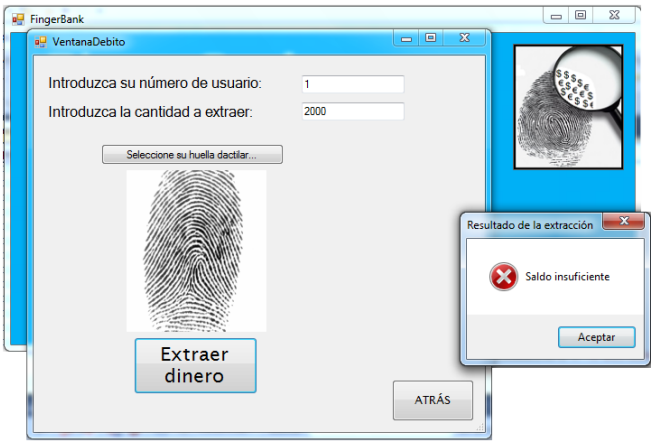

Figura 53: Resultado obtenido en el ordenador en el caso de *saldo insuficiente*

No se añade una imagen sobre la interfaz del móvil ya que muestra el mismo saldo que en el caso anterior debido a que la operación no se ha realizado.

7.2.5. Número de usuario inválido

En este apartado se comprobará que ocurre si en alguna operación se introduce un número de usuario que no existe en el sistema. Para ello, se ha utilizado la funcionalidad *Mostrar saldo* del ordenador y del dispositivo móvil.

Resultado esperado

Debido a que la gestión de usuarios ocurre en el ordenador, se espera que sea la aplicación de éste quién no deje al usuario ejecutar la operación. En el dispositivo móvil simplemente debería aparecer un error.

Resultado obtenido

El resultado obtenido en la interfaz del ordenador se puede ver en la siguiente Figura 54.

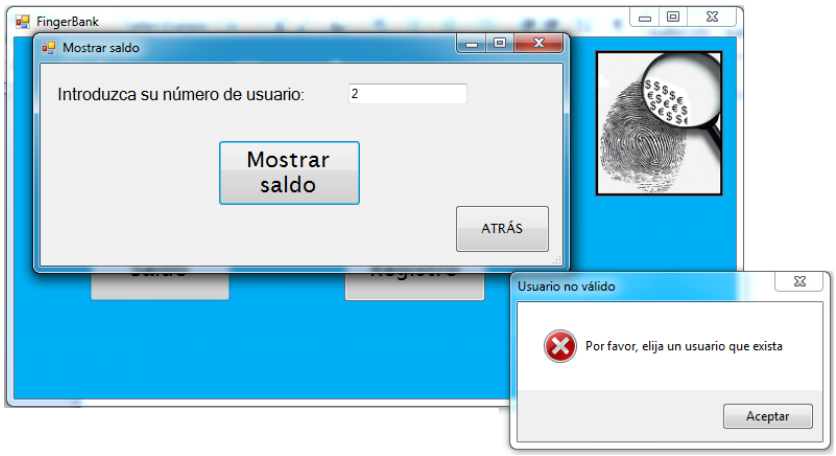

Figura 54: Resultado obtenido en el ordenador en el caso de *usuario no válido*

En el dispositivo móvil, el resultado se muestra en la Figura 55.

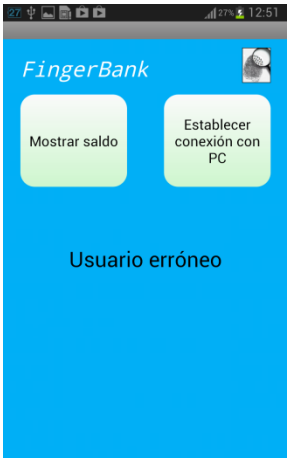

Figura 55: Resultado obtenido en el dispositivo móvil en el caso de *usuario inválido*

7.3. PRUEBAS DE RENDIMIENTO

En este apartado se analizarán los tiempos que tarda el sistema en ejecutar una operación concreta. Se estudiarán los tiempos obtenidos en la transmisión de la información así como los de los algoritmos MINDTCT y BOZORTH3 en un teléfono móvil Android. El último apartado se corresponde con una comparación final de estos algoritmos entre un ordenador y un dispositivo móvil.

7.3.1. Resultados obtenidos en la transmisión de información

El protocolo de comunicación TCP se caracteriza por ofrecer una fiabilidad en la transmisión. Para ello, como desventaja, añade ciertos retardos al iniciar la comunicación. En este apartado se analizarán estos retardos para las cuatro operaciones bancarias y se investigará si los tiempos son muy diferentes dependiendo de la trama enviada (ver Figura 43 para los diferentes tipos de trama).

Para poder medir los tiempos exactos, se crea una variable en el ordenador con el tiempo real antes de iniciar la comunicación y se envía en la cabecera de la trama. Cuando el móvil la reciba, la mostrará por pantalla directamente. Este dato mostrará el tiempo de transmisión del sentido descendente de la comunicación. Para calcular el tiempo que tarda el móvil en retornar el resultado de vuelta al ordenador, se realizará de la misma forma. Dado que este tiempo es siempre el mismo para las diferentes operaciones ya que la trama de retorno no varía, se analizará sólo en el primer apartado. Se ha elegido la opción de incluir la variable tiempo en la trama con el objetivo de conseguir el tiempo exacto, sacrificando por otro lado el tener una cabecera mayor. Los tiempos de gestión interna de saldos en el dispositivo móvil se suponen despreciables.

Operación Crédito/Débito

Para la operación de Crédito o Débito, la trama está formada por una cabecera con tres campos más el BIR conteniendo la huella dactilar. El tamaño total de la cabecera del protocolo si se tiene en cuenta la variable tiempo introducida es de 20 bytes. El tamaño de la cabecera del BIR es de 47 bytes, por lo que el tamaño total de la trama será de 145179 bytes. Se han realizado diferentes transmisiones de la misma trama para generar un tiempo medio. En la Tabla 1, se puede ver cuatro tiempos de cuatro transmisiones diferentes de la misma trama.

Tabla 1: Tiempos obtenidos para la transmisión TCP en la operación Crédito/Débito

	Tiempo (milisegundos)
Transmisión 1	376
Transmisión 2	356
Transmisión 3	388
Transmisión 4	421

A partir de estos datos, se puede establecer un tiempo medio de transmisión de *385,25 milisegundos*.

Para el caso del valor de retorno, los tiempos obtenidos se adjuntan en la Tabla 2.

Tabla 2: Tiempos obtenidos para la transmisión TCP en el resultado de retorno

	Tiempo (milisegundos)
Transmisión 1	186
Transmisión 2	216
Transmisión 3	168
Transmisión 4	204

Se obtiene un valor medio de 193.5 milisegundos. En conjunto con el resultado que envía el móvil de vuelta al ordenador, el tiempo de transmisión total medio es de *578.5 milisegundos.*

Operación Registro de Usuario

En este caso, la trama está formada por una cabecera con dos campos y el BIR conteniendo la huella dactilar. En total, la trama tiene un tamaño de 145175 bytes. Al igual que en el caso anterior, se han realizado cuatro envíos diferentes de la misma trama (Tabla 3).

Tabla 3: Tiempos obtenidos para la transmisión TCP en la operación *Registro de Usuario*

	Tiempo (milisegundos)
Transmisión 1	322
Transmisión 2	353
Transmisión 3	375
Transmisión 4	311

Se obtiene un tiempo medio de transmisión de *340.25 milisegundos*.

En conjunto con el tiempo de retorno, el valor medio total es de *533.75 milisegundos.*

Operación Mostrar Saldo

En este caso, la trama está formada simplemente por una cabecera de un solo campo y los datos, siendo el número de usuario. El tamaño total si se cuenta la variable tiempo en la cabecera es de 16 bytes. En la siguiente Tabla 4 se muestran los resultados de cuatro transmisiones.

Tabla 4: Tiempos obtenidos para la transmisión TCP en la operación *Mostrar Saldo*

	Tiempo (milisegundos)
Transmisión 1	288
Transmisión 2	256
Transmisión 3	334
Transmisión 4	276

Se obtiene un tiempo de transmisión medio de *288.5 milisegundos*.

En conjunto con el tiempo de retorno, el valor medio total es de *482 milisegundos.*

Por un lado, se puede observar como al disminuir el tamaño de la trama disminuye sensiblemente el tiempo medio de transmisión.

Por otro lado, el tiempo total medio de transmisión de los cuatro tipos de trama es de *531.42 milisegundos*. Como se puede ver, este resultado es bastante insignificante, por lo que no afectan gravemente al rendimiento del sistema.

7.3.2. Resultados obtenidos con el algoritmo de detección de minucias MINDTCT.

En este apartado se analizará el tiempo que tarda el algoritmo MINDTCT en procesar las huellas y detectar las minucias en el teléfono móvil. Para medir los tiempos, se crea un tiempo de inicio antes de que comience el algoritmo y uno de fin al terminarlo. Dado que el directorio de huellas contiene 4 tipos de huellas de diferentes candidatos y de cada huella existen varias muestras, se ha optado por sacar 2 tiempos de cada tipo de huella. Se muestra en la Tabla 5 (el primer número es el número de candidato mientras que el segundo es el número de la muestra).

Tabla 5: Tiempos obtenidos en el dispositivo móvil con el algoritmo MINDTCT

	Tiempo total (milisegundos)
Candidato 1_1	2090
Candidato 1_2	1895
Candidato 2_1	2018
Candidato 2_2	1835
Candidato 3_1	1982
Candidato 3_2	1954
Candidato 4_1	2036
Candidato 4_2	1842

El tiempo medio de detección del algoritmo MINDTCT obtenido es *1956,5 milisegundos*.

Pese a que este tiempo afecta considerablemente al sistema global, se puede afirmar que es un resultado muy aceptable ya que el algoritmo se ejecuta sobre un dispositivo móvil y éste tiene mucha menos capacidad que un ordenador.

7.3.3. Resultados obtenidos con el algoritmo de comparación de BOZORTH3

En este apartado se analizarán los diferentes tiempos obtenidos al compulsar dos tipos de huellas dactilares en un dispositivo móvil. Además, se expondrá también el resultado numérico que retorna el algoritmo BOZORTH3. Para ello, se comparan las minucias de los candidatos anteriores entre ellas (Tabla 6).

Tabla 6: Tiempos y resultados obtenidos en el dispositivo móvil con el algoritmo BOZORTH3

Candidato A	Candidato B	Tiempo (milisegundos)	Resultado
1_1	1_1	4828	509
1_1	2_1	422	9
1_1	3_1	516	13
1_1	4_1	516	10
2_1	2_1	1084	293

2_1	3_1	1011	23
2_1	4_1	442	7
3_1	3_1	8142	474
3_1	4_1	808	18
4_1	4_1	5048	393
1_2	1_2	1256	476
1_2	1_1	1008	160
1_2	2_1	448	16
1_2	3_1	525	15
1_2	4_1	322	16
2_2	1_2	418	11
2_2	1_1	444	8
2_2	2_1	460	57
2_2	3_1	148	9
2_2	4_1	118	4
3_2	3_2	3866	488
3_2	1_2	514	10
3_2	2_2	438	8
3_2	1_1	663	10
3_2	2_1	1098	22
3_2	3_1	4010	233
3_2	4_1	680	14
4_2	4_2	3600	446
4_2	1_2	556	19
4_2	2_2	477	8
4_2	3_2	622	25
4_2	1_1	418	15
4_2	2_1	817	10
4_2	3_1	1502	22
4_2	4_1	7013	114

El tiempo medio obtenido es *1549,66 milisegundos*. De la misma forma que el caso anterior, este proceso afecta al rendimiento global del sistema.

Además, se puede observar que dos huellas iguales tienen un tiempo de comparación mucho más alto que dos huellas diferentes. Esto se debe a que el algoritmo encuentra un mayor número de coincidencias por lo que tiene que analizar mayor número de minucias.

Por otro lado, el resultado numérico que retorna el algoritmo es mayor cuanto más parecidas sean las huellas. Este resultado ayuda a que se decida el umbral que hay que fijar dependiendo del objetivo de la aplicación que se busque. Si se aumenta, se incrementa la *tasa de falso rechazo*, es decir, hay más probabilidad de rechazar al usuario correcto. Esto suele necesitarse en sistemas de seguridad muy restrictivos donde no se pueden permitir aceptar a un individuo que no debe ser aceptado. Si este límite se reduce, ocurre exactamente esto: aumenta *la tasa de falsa aceptación*, o lo que es lo mismo, aumenta la probabilidad de verificar correctamente

a un candidato erróneo. Es por ello que hay que elegir sabiamente este límite y dependerá del contexto donde se quiera instalar el sistema.

En este punto, podemos conocer el tiempo medio que tarda cada operación bancaria en ejecutarse en el sistema. Los resultados se muestran en la siguiente Tabla 7.

Tabla 7: Tiempos medios totales de las operaciones bancarias del sistema

	Tiempo medio (milisegundos)
Crédito	3891,41
Débito	3891,41
Registro de usuarios	2490,25
Mostrar saldo	482,0

Se puede observar que para las operaciones de Crédito/Débito se obtienen los tiempos más altos debido a que se realiza tanto la detección de minucias con el algoritmo MINDTCT como la comparación de estas minucias con el algoritmo BOZORTH3.

Es perfectamente posible implementar un sistema de estas características en un ámbito comercial ya que el tiempo máximo que puede tardar una operación en realizarse es de *3.9 segundos*, un resultado más que aceptable.

7.4. COMPARACIÓN CON RESULTADOS OBTENIDOS EN C#

En este apartado se compararán los tiempos obtenidos de los algoritmos anteriores utilizando ahora el lenguaje de programación C# sobre un ordenador. Se utilizan dos documentos escritos por el grupo donde se ha realizado este TFG, donde se comparan diferentes bases de datos de huellas dactilares y sus respectivos tiempos obtenidos con los algoritmos MINDTCT y BOZORTH3.

7.4.1. Resultados obtenidos con el algoritmo de detección de minucias MINDTCT.

A continuación se mostrarán los tiempos medios obtenidos utilizando el algoritmo de procesamiento y detección MINDTCT [28]. En la Tabla 8, se obtienen diferentes tiempos correspondientes a diferentes bases de datos (DB) obtenidos por medio de distintos sensores.

Tabla 8: Tiempos obtenidos en un ordenador con el algoritmo MINDTCT

	Tiempo medio (milisegundos)
DB1	14,63
DB2	268,65

DB3	206,22
Media	163,17

7.4.2. Resultados obtenidos con el algoritmo de comparación BOZORTH3

En este apartado, se mostrarán los tiempos medios de comparación de huellas dactilares sobre un ordenador utilizando C# [29]. Para la obtención de estos tiempos medios (Tabla 9), se han utilizado las mismas bases de datos y sensores que en la tabla anterior.

Tabla 9: Tiempos obtenidos en un ordenador con el algoritmo BOZORTH3

	Tiempo medio (milisegundos)
DB1	3,414
DB2	237,608
DB3	147,040
Media	129,354

7.4.3. Comparación de tiempos entre Android y C#

En las siguientes tablas se comparan los tiempos medios obtenidos en un dispositivo móvil con los tiempos conseguidos en un ordenador, tanto para el algoritmo MINDTCT (Tabla 10) como para BOZORTH3 (Tabla 11).

Tabla 10: Diferencia de tiempos medios entre Android y C# utilizando el algoritmo MINDTCT

Tiempo medio en Android (ms)	Tiempo medio en C# (ms)
1956,5	163,17

Tabla 11: Diferencia de tiempos medios entre Android y C# utilizando el algoritmo BOZORTH3

Tiempo medio en Android (ms)	Tiempo medio en C# (ms)
1549,66	129,354

Como se aprecia, el tiempo obtenido en un móvil Android es considerablemente mayor que en un ordenador. Sin embargo, pese a que pueden parecer resultados muy poco eficientes, estos tiempos son admisibles dado que un dispositivo móvil carece de un procesador tan potente como el de un PC.

8. CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

8.1. CONCLUSIONES

Con la finalización de este trabajo de fin de grado, se ha conseguido implementar, de forma totalmente funcional y estable, un sistema biométrico de huellas dactilares basado en la propuesta del estándar BioAPI 4.0 sobre un dispositivo móvil y un ordenador. Se puede afirmar sin duda que esta beta presenta grandes ventajas respecto a la versión 3.0 y que el objetivo de mejorar BioAPI se ha cumplido. De este modo, esta propuesta ofrece una menor complejidad de código, siendo bastante más simple e intuitiva de entender. Así mismo, existe una mayor compatibilidad entre dispositivos así como interoperabilidad entre módulos posibilitando la implementación de un sistema biométrico muy versátil.

Además, se ha desarrollado una aplicación bancaria que permite realizar diferentes operaciones una vez que el usuario ha sido reclutado y verificado correctamente, añadiendo un alto grado de protección.

Los mayores problemas han aparecido a la hora de implementar las funcionalidades biométricas de *reclutamiento* y *verificación* de usuarios en el dispositivo móvil basándose estrictamente en la propuesta de la norma BioAPI 4.0. Esto se debe al desconocimiento inicial con el que se empezó este trabajo. Con el paso del tiempo se fue cogiendo experiencia y entendiendo los diferentes módulos que componen el estándar, permitiendo una implementación final satisfactoria.

Respecto al mercado, se ha demostrado que las propuestas de la norma BioAPI 4.0. son factibles por lo que se puede pensar en llevarla a comités de normalización para su aprobación, y que posteriormente sea empleado por diferentes fabricantes y así proporcionar interoperabilidad entre los diferentes productos de estos y entre ellos. Sin embargo, esta versión todavía necesita pulir determinados detalles y definir nuevos conceptos para poder ser una versión final. El mayor problema que sigue existiendo en esta propuesta de la norma es que todavía es bastante compleja de entender. Los nuevos desarrolladores necesitan de un tiempo largo para estudiarla, adaptarse a ella y comenzar a utilizarla. La solución sería llegar a un compromiso óptimo entre usabilidad y versatilidad, es decir, la norma debe ser lo suficientemente amplia para que cubra todos los casos que se pueden dar en cualquier sistema biométrico pero que a su vez, sea lo bastantemente simple para que los fabricantes y desarrolladores no necesiten pelearse con ésta cada vez que necesiten usarla.

En conclusión y como opinión personal, cuando el alumno comenzó este TFG carecía del conocimiento necesario del tema a tratar. No sólo en el manejo de estándares y normas, sino también en la utilización de tecnologías biométricas así como en programación Android y C#. Sin embargo, varios meses y muchas horas después, finaliza el trabajo con muchos más conocimientos con los que habría contado, con todos los objetivos cumplidos, sintiéndose capaz de continuar trabajando e investigando en este nuevo sector, totalmente atractivo para él.

8.2. LINEAS FUTURAS

Respecto a las líneas futuras, existen diferentes mejoras que pueden implementarse en el sistema. Estos avances se pueden realizar en cualquier parte del sistema: en las aplicaciones bancarias, en el algoritmo biométrico o en la comunicación entre dispositivos.

En primer lugar, se podría eliminar la autenticación de *triple factor* y pasar a usar una de *doble factor* donde el usuario sólo necesite el dispositivo móvil y su huella dactilar para realizar las operaciones bancarias. El problema que se encuentra aquí es como saber que usuario está en el sistema en un instante determinado. Dado que el móvil es personal e intransferible, la solución podría residir en él. De esta forma, cada vez que el usuario se conecta al sistema por medio del dispositivo móvil, éste enviaría una trama de inicio de comunicación al ordenador incluyendo su nombre de usuario. El ordenador comprobaría entonces que el usuario que se intenta conectar existe en el sistema y se establecería la conexión.

La ventaja principal es que el usuario ya no necesitaría recordar su número asignado por el sistema, brindando una gran comodidad a éste. Sin embargo, se renunciaría a una protección extra frente a ataques de terceros, pudiendo éstos hacerse pasar por un usuario cuyo nombre esté incluido en la aplicación. En consecuencia, esta mejora podría ser muy interesante si se pensara en un nuevo contexto donde no fuera necesaria tanta seguridad como en un sistema bancario.

En segundo lugar, una mejora que añadiría un mejor rendimiento y mayor fluidez al sistema global sería evitar la necesidad de que el dispositivo móvil se conecte al ordenador cada vez que se quiere iniciar una operación bancaria. Lo ideal sería que fuera el móvil quien iniciara la comunicación con el ordenador cuando ambos dispositivos se encontraran en la misma red. De esta forma, se evitaría al usuario pulsar el botón de *establecer conexión con el PC* en cada actividad.

El mayor problema que existe es el de cómo saber si ambos dispositivos se encuentran dentro del rango definido por la red Wi-Fi. La solución encontrada consiste en iniciar la aplicación cuando se esté cerca del ordenador y que el teléfono escuche la red hasta que se establezca la conexión. La desventaja principal es que el dispositivo móvil carece de una batería duradera y escuchar una comunicación Wi-Fi durante mucho tiempo afectaría críticamente a su duración total. Esto se podría evitar utilizando un time-out por el cual si la escucha permanece activa durante más de X segundos, se cierra pasado ese intervalo de tiempo.

Otra desventaja que se desprende de utilizar iniciar la comunicación con el móvil es que una vez establecida la conexión con el PC, si el usuario no ejecuta ninguna operación, ésta permanecerá abierta y la batería del móvil se descargará con mucha rapidez.

En conjunto, esta mejora se podría considerar si se utilizara un dispositivo móvil con una optimización efectiva del rendimiento de la batería.

En tercer lugar, no puede olvidarse que el sistema implementado sería completamente funcional si se utilizara un dispositivo de captura de huellas dactilares en lugar de simular este sensor. Para ello, en primer lugar, se necesitaría un sensor que pudiéramos conectar con el

ordenador. En segundo lugar y más importante, sería necesario mejorar el código biométrico del ordenador, introduciendo la Unidad del Sensor y sus correspondientes funcionalidades definidas en la norma BioAPI. En consecuencia, se conseguiría implementar un sistema biométrico real que podría ser utilizado en un futuro cercano.

Estas son algunas de las mejoras que se pueden implementar. Este sistema, a pesar de tener las carencias que se han comentado, ha cumplido el objetivo que nos fijamos al comienzo de este TFG: estudiar y demostrar la viabilidad de los cambios que se proponen para una nueva versión de BioAPI sobre un dispositivo móvil y un ordenador. Así mismo, esta aplicación deja abiertas varias líneas de mejora en la aplicación diseñada, tanto en el ordenador servidor como en el móvil, pasando por la comunicación entre ellos.

BIBLIOGRAFÍA

[1] K. Karu, S. Chen, A.K. Jain N.K. Ratha, *A real-time matching system for large fingerprint databases.*: IEEE TPAMI, 1996.

[2] Wikipedia - Alphone Bertillon. [Online]. Disponible en: http://en.wikipedia.org/wiki/Alphonse_Bertillon#Biography [Último acceso: 03 Abril 2013].

[3] Wikipedia - Francis Galton. [Online]. Disponible en: http://en.wikipedia.org/wiki/Francis_Galton [Último acceso: 12 Abril 2013]

[4] Monografías - Biometría y voto electrónico. [Online]. Disponible en: http://www.monografias.com/trabajos82/biometria-y-voto-electronico/image013.jpg [Último acceso: 12 Abril 2013].

[5] Wikipedia - Fingerprint recognition. [Online]. Disponible en: http://en.wikipedia.org/wiki/Fingerprint_recognition [Último acceso: 25 Abril 2013].

[6] J. Lucena, LM. Planchat, A. Cuquerella, L. Ferreiro y MA. Vázquez M. Subirana, *Medicina forense en imágenes: La huella perdida. Identificación personal utilizando un dedo parcialmente amputado encontrado en la escena de un robo.*, 2005.

[7] Monografías - Características de la huella . [Online]. Disponible en: http://www.monografias.com/trabajos57/huellas-lofoscopicas/hu2.jpg [Último acceso: 25 Abril 2013].

[8] Representación del Core y Delta de una huella dactilar. [Online]. Disponible en: http://www.anilaggrawal.com/ij/vol_002_no_001/papers/pics/paper005_image15.gif [Último acceso: 25 Abril 2013].

[9] Patrón de huella dactilar en Arco [Online]. Disponible en: http://ridgesandfurrows.homestead.com/files/Arch.jpg [Último acceso: 25 Abril 2013].

[10] Patrón de huella dactilar en Espiral [Online]. Disponible en: http://ridgesandfurrows.homestead.com/files/whorl.jpg [Último acceso: 25 Abril 2013].

[11] Patrón de huella dactilar en Bucle [Online]. Disponible en: http://ridgesandfurrows.homestead.com/files/loop.jpg [Último acceso: 25 Abril 2013].

[12] Saeed Jalilzadeh. Majid Meghdadi, "Validity and Acceptability of Results in Fingerprint Scanners," *WSEAS*, vol. 7th, p. 8, Oct. 2005.

[13] NIST - Algoritmo MINDTCT [Online]. Disponible en: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=51097 [Último acceso: 28 Abril 2013].

[14] NIST - Algoritmo BOZORTH3 [Online]. Disponible en: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=51096 [Último acceso: 28 Abril 2013].

[15] Vierito - Procesado de la imagen de huellas dactilares. [Online]. Disponible en: http://vierito.es/wordpress/2008/10/24/fprint-procesado-de-la-imagen-huellas-dactilares/ [Último acceso: 29 Abril 2013].

[16] Raul Sanchez-Reillo, "ES NB Position on CBEFF and BioAPI related standards (19784-1 and 30106-x)", Reunión de comité de estandarización ISO/IEC JTC 1/SC 37, Winchester (Reino Unido). Junio, 2013.

[17] W3 - BioAPI. [Online]. Disponible en: http://www.w3.org/2008/08/siv/Slides/Daon/BioAPI-Tilton-2009-full.pdf [Último acceso: 2 Mayo 2013].

[18] Rubén Romero García. e-archivo.uc3m. [Online]. Disponible en: http://e-archivo.uc3m.es/bitstream/10016/16893/1/MemoriaTFG_Ruben_Romero_Garcia.pdf [Último acceso: 4 Mayo 2013].

[19] Norma ISO-IEC_30106-3 Proposed Alternative v3, 2013.

[20] Wikipedia - Java. [Online]. Disponible en: http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n) [Último acceso: 28 Mayo 2013].

[21] El Androide Libre - La historia y los comienzos de Android. [Online]. Disponible en: http://www.elandroidelibre.com/2011/08/la-historia-y-los-comienzos-de-android-el-sistema-operativo-de-google.html [Último acceso: 2 Junio 2013].

[22] Imagen de Versiones de Android [Online]. Disponible en: http://i.emezeta.com/weblog/ganar-dinero-android/versiones-android.png?1 [Último acceso: 2 Junio 2013].

[23] Fragmentación del sistema operativo Android, Enero 2013 [Online]. Disponible en: http://cdn5.andro4all.com/wp-content/blogs.dir/28/files/2013/02/Fragmentacion-Android-Enero-2013.jpg [Último acceso: 2 Junio 2013].

[24] Arquitectura de Android. [Online]. Disponible en: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android [Último acceso: 2 Junio 2013].

[25] columna80.files.wordpress - Pila de Android [Online]. Disponible en: http://columna80.files.wordpress.com/2011/02/0013-01-pila-software-android.png [Último acceso: 2 Junio 2013].

[26] Jose Antonio Gonzalez, *El lenguaje de programación C#*. [Online]. Disponible en: http://dis.um.es/~bmoros/privado/bibliografia/LibroCsharp.pdf [Último acceso: 3 Junio 2013].

[27] "Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación (TIC) en los Hogares," Instituto Nacional de Estadística, Encuesta 2012.

[28] Raúl Sánchez Reillo Eugenio Hernández, "Informe de tiempos del algoritmo MINDTCT,"

Universidad Carlos III de Madrid, Interno.

[29] "Estudio de la influencia del número de minucias en el software NIGOS" Universidad de Granada, Universidad de Pamplona y Universidad Carlos III de Madrid, Proyecto CAB - Centro de Autenticación Biométrica 2012.

ANEXO 1. PRESUPUESTO

En este anexo se desglosarán las distintas tareas habidas a lo largo de este proyecto con el objetivo de calcular posteriormente su coste total.

Primero, se detallarán las diferentes fases por las que pasó este TFG y las horas empleadas en cada actividad. A continuación, se procederá a calcular su presupuesto total teniendo en cuenta todos los factores que contribuyeron en este proyecto.

Fase 1: Documentación inicial

I. Estudio de la plataforma Android y del entorno de desarrollo (25 horas)
II. Estudio del lenguaje de programación C# y del entorno de desarrollo (10 horas)
III. Preparación de las herramientas de trabajo (5 horas)
IV. Búsqueda y realización de tutoriales y aplicaciones sencillas. (35 horas)
V. Asistencia a charlas y presentaciones sobre Android (10 horas)

Fase 2: Desarrollo de la aplicación

I. Sistema biométrico en dispositivo móvil Android (50 horas)
II. Sistema biométrico en ordenador (10 horas)
III. Interfaz gráfica (20 horas)
IV. Comunicación cliente-servidor (30 horas)
V. Interconexión de todos los módulos (40 horas)

Fase 3: Pruebas en un dispositivo real

I. Pruebas en un Sony Xperia U (20 horas)
II. Corrección y depuración (10 horas)

Fase 4: Elaboración de la memoria

I. Redacción de la memoria (70 horas)
II. Corrección y maquetación (10 horas)

Tabla 12: Desglose de tareas

FASES	HORAS EMPLEADAS
Documentación inicial	85
Desarrollo de la aplicación	150
Pruebas en un dispositivo real	30
Elaboración de la memoria	80
TOTAL	**345**

COSTES DE MATERIAL

En este apartado se analizarán los costes materiales, formados por costes de hardware, licencias de software y los recursos de oficinas o fungibles.

Empezando por el hardware, los materiales necesarios han sido un ordenador de altas prestaciones, necesario para ejecutar el entorno de desarrollo Eclipse y Visual Studio 2010, y un teléfono móvil Sony Xperia U para realizar las pruebas sobre un dispositivo real.

Sobre el software de pago utilizado, han sido necesarias las licencias de Windows 7, Visual Studio 2010 y Microsoft Office 2010.

Por último, en cuanto a los recursos de oficina sólo se tiene en cuenta la conexión a Internet, que se calcula en euros/mes con una duración aproximada de 2 meses (345 horas).

Tabla 13: Costes materiales

RECURSOS	PRECIO (€)
Ordenador	400
Sony Xperia U	100
Licencia Windows 7	100
Licencia Visual Studio 2010	600
Licencia Microsoft Office 2010	60
Conexión a Internet	30 (€/mes)
TOTAL	**1320**

COSTES DE PERSONAL

En este apartado se detallarán los costes referidos a las dos personas que han intervenido en la creación de este proyecto: el ingeniero y el jefe de proyecto.

Tabla 14: Coste de personal

OCUPACIÓN	HORAS	PRECIO/HORA	IMPORTE TOTAL (€)
Jefe de proyecto	25	80	2000
Ingeniero	345	40	13800
TOTAL	**370**		**15800**

COSTES TOTALES

Tabla 15: Costes totales

CONCEPTO	PRECIO (€)
Coste de material	1320
Coste de personal	15800
Costes indirectos (20 %)	3424
Subtotal	20544
IVA (21%)	4314.24
TOTAL	**24858.24**

El coste total del proyecto es de *veinticuatro mil ochocientos cincuenta y ocho euros con veinticuatro céntimos*

Leganés, 20 de Junio de 2013

El ingeniero