Universidad de las Ciencias Informáticas

Facultad 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Sistema para garantizar la confidencialidad de los datos de múltiples usuarios en dispositivos móviles con sistema operativo *Android* v4.0

Autores: Alberto Baxter Barrizonte

Osmanis del Risco Ferral

Tutores: Ing. Yoanny Torres Rubio

Ing. Ernesto Avilés Vázquez

Co-tutor: Ing. Dainiel Viltre Guillen

La Habana, Cuba

Junio, 2015

"Año 57 de la Revolución"

Declaración de autoría

Declaramos por este medio ser autores del trabajo final de tesis Sistema para garantizar la confidencialidad de los datos de múltiples usuarios en dispositivos móviles con sistema operativo *Android* v4.0 y que autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmamos la presente declaración jurada de autoría en la Ciudad de La Habana a los _____días del mes de _____ del año _____.

_____

Alberto Baxter Barrizonte

Firma del Autor

_____

Osmanis del Risco Ferral

Firma del Autor

_____

Ernesto Avilés Vázquez

Firma del Tutor

_____

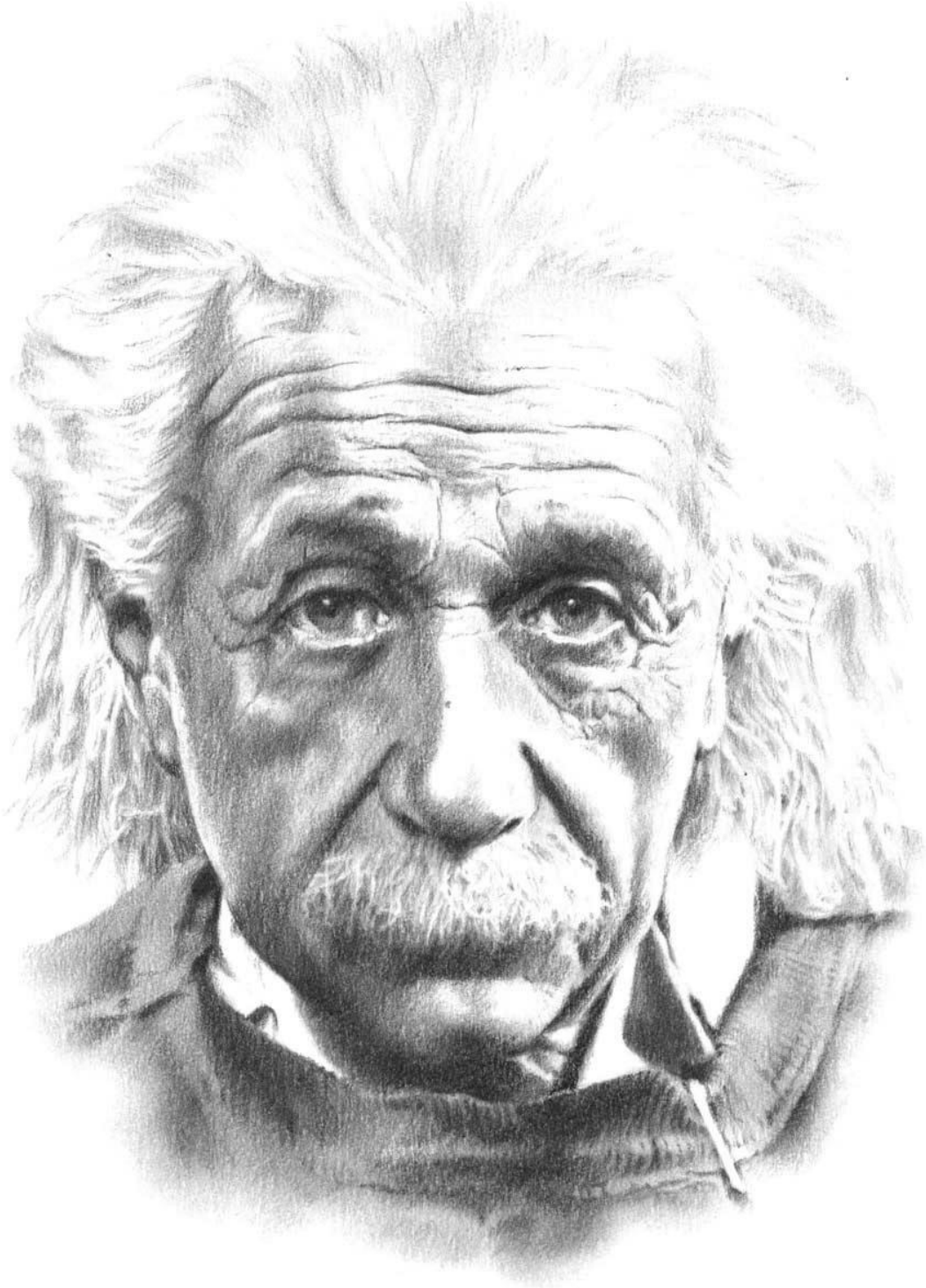
Yoanny Torres Rubio

Firma del Tutor

_____

Dainiel Viltre Guillen

Firma del Co-Tutor



"No intente convertirse en un hombre de éxito, sino más bien intente convertirse en un hombre de principios."

Albert Einstein

Datos de contactos

Tutores

Nombre y Apellidos: Yoanny Torres Rubio.

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

E-mail: ytrubio@uci.cu

Graduado en el 2011 de Ingeniero en Ciencias Informáticas. Actualmente trabajador del Centro de Telemática. Departamento de Desarrollo de Componentes, desempeña el cargo de Especialista “B” en Ciencias Informáticas.

Nombre y Apellidos: Ernesto Avilés Vázquez

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

E-mail: aviles@uci.cu

Graduado en el 2010 de Ingeniero en Ciencias Informáticas. Actualmente trabajador del Centro de Telemática. Departamento de Desarrollo de Componentes, desempeña el cargo de Especialista “B” en Ciencias Informáticas.

Co-tutor

Nombre y Apellidos: Dainiel Viltre Guillen.

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

E-mail: viltre@uci.cu

Graduado en el 2013 de Ingeniero en Ciencias Informáticas. Actualmente trabajador del Centro de Informática Industrial. Departamento de Desarrollo de Aplicaciones.

Dedico este trabajo a toda mi familia y amigos, en especial a mis padres y mis hermanos por haber sido mi inspiración y por quienes hoy me he convertido finalmente en un profesional, por el amor que me han brindado y por toda la confianza que siempre han depositado en mí.

Osmanis del Risco Ferral.

Esta tesis va dedicada a mis padres, que han hecho todo lo posible porque yo día a día sea una mejor persona.

Alberto Baxter Barrizonte.

Agradecimientos

A mis padres, por apoyarme en todo momento, por sus consejos, por sus valores, por su motivación constante que me ha permitido seguir adelante en cada minuto de mi vida, por brindarme todo el amor de padres que un hijo puede necesitar.

A mi hermanito Osniel, gracias al amor que le tengo puse mi empeño en esta carrera y a pesar de ser un niño de solo 10 años le doy millones de gracias por haberse preocupado por mí en cada momento de mi carrera y le prometo que siempre estaré a su lado.

A mi hermano Dainiel, por el cariño que le tengo, gracias por su apoyo en mis estudios, por estar siempre a mi lado en las buenas y en las malas desde niños, siempre lo tendré presente.

A mi abuela por haberme bridado tanto amor, mis primos Reinier, Josvel y Daike, a mis tías en particular Maritza y Annerys, a mi tío Luis que siempre se ha preocupado por mí.

Muchas gracias a mi dúo de tesis por ser un gran amigo y un buen compañero.

A mis tutores que siempre estuvieron presente brindándome apoyo incondicional.

A Dios, a mis compañeros en la UCI y en mi localidad donde vivo, a los que me enseñaron lo que es un concepto de amigo, gracias a aquellos que por cosas de la vida no están presentes en estos momentos, sin embargo mis respeto hacia ellos por saber guardar silencio en momentos y situaciones, eso más que ser un amigo es ser un hombre. De corazón muchas gracias.

También le brindo mis agradecimientos al profesor Abel por sus consejos, a Caudia mi cuñi que le tengo un gran aprecio, a Brenda mi segunda cuñi, a Yailén una gran amiga, a Eider que durante todo el tiempo que hemos compartido nunca me ha fallado, se ha convertido en mi mejor amigo, le doy muchas gracias por todo su apoyo y por saber valorar nuestra amistad, a Danaray la China muchas gracias por todos sus consejos y su apoyo incondicional, siempre la tendré como una hermana, a Roniel el Fi muchas gracias por ser un buen compañero en todo este tiempo.

A Dashiel, Raidel, El zurdo, Pepe, Ernesto, David, a mis compañeros de apartamento y a los del barrio Kadir y Jorge Luis, en fin a todas las personas que no he mencionado muchas gracias.

Osmanis del Risco Ferral.

Agradecimientos

A mis padres por el apoyo incondicional que me han brindado en todo momento.

A toda mi familia por preocuparse por mí y brindarme buenos consejos.

A mis compañeros de la escuela que han compartido cinco años a mi lado en las buenas y en las malas.

A mis tutores y cotutor por haberme guiado con su ayuda.

Alberto Baxter Barrizonte.

Resumen

La protección de los datos es uno de los temas más importantes a tratar en el mundo de la informática, ya que el uso indebido de información privada puede causar efectos indeseables en usuarios de cualquier sistema.

La presente investigación tiene como objetivo principal la elaboración de un sistema informático libre que permita garantizar la confidencialidad de los datos personales de múltiples usuarios, que operan en un mismo dispositivo con sistema operativo *Android*.

Esta solución surge a partir de la necesidad de factores como evitar el acceso a datos confidenciales como documentos, mensajes, imágenes o videos y evitar robos de información. Para ello se seleccionó como metodología de desarrollo XP, se realizó un estudio de los principales aspectos del sistema operativo de dispositivos móviles *Android* y se investigó sobre algunos de los métodos de cifrado más usados en la actualidad en particular el algoritmo AES e IDEA, con el objetivo de lograr mantener la confidencialidad de los datos.

Se obtuvo como resultado una aplicación capaz de brindarle confidencialidad a los datos privados de los usuarios a través de las características de cifrado que brindan los algoritmos AES e IDEA y varias herramientas que facilitaron el desarrollo.

Palabras clave: *Android,* cifrado, confidencialidad, descifrado, información, móvil.

Índice

Índice de figuras

Índice de tablas

Introducción

En el transcurso de los años, la informática y las tecnologías han obtenido avances significativos lo que ha traído aparejado consigo grandes avances en el desarrollo de los dispositivos tecnológicos. El auge de la tecnología ha permitido un auténtico desenvolvimiento en el desarrollo de los dispositivos móviles, que desde sus inicios en la década de los 90 han evolucionado enormemente.

La línea entre lo que define qué es un dispositivo móvil y lo que no lo es puede ser un poco difusa, pero en general, se pueden definir como aquellos micro-ordenadores que son lo suficientemente ligeros como para ser transportados por una persona y que disponen de capacidades de procesamiento, conectividad a *Internet*, memoria y la capacidad de batería suficiente como para poder funcionar de forma autónoma (Tardáguila, 2006).

Normalmente, son versiones limitadas en prestaciones y por tanto en funcionalidades, de los ordenadores portátiles o de sobremesa. Por cierto, los ordenadores portátiles no se consideran como dispositivos móviles, ya que consumen más batería y suelen ser un poco más pesados de lo que se espera de algo pensado para llevar siempre encima (Tardáguila, 2006).

Los dispositivos móviles se han convertido en una parte inseparable de la mayoría de las personas, ya que son pequeños y pueden portarse fácilmente durante el traslado. El punto fuerte de estos es que ofrecen funcionalidades similares a las de los ordenadores personales. Entre las funcionalidades más comunes se tienen: editar documentos de textos, hojas de cálculo, leer libros en formato pdf o chm, recibir y enviar correos electrónicos, manejar una agenda, sincronizar datos con el ordenador, navegar por *Internet*, utilizar un GPS y algunas otras; en definitiva, se puede hacer casi lo mismo que en un ordenador, pero con una pantalla más pequeña (Tardáguila, 2006).

Al igual que las computadoras personales los dispositivos móviles también son controlados por sistemas operativos. Los sistemas operativos móviles centran más su atención a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos. A medida que los dispositivos móviles ascienden en popularidad, los sistemas operativos con los que funcionan adquieren mayor importancia.

En febrero del 2015 se realizó un estudio referente a los sistemas operativos móviles más difundidos en el mundo, del cual se arribó a la conclusión que *Android* es el sistema operativo móvil más usado con un 47.45% de usabilidad, a este le sigue iOS con un 42.59% y el resto con un 9.96% (Móviles, 2015), en la Figura 1 se pueden apreciar los datos mencionados.

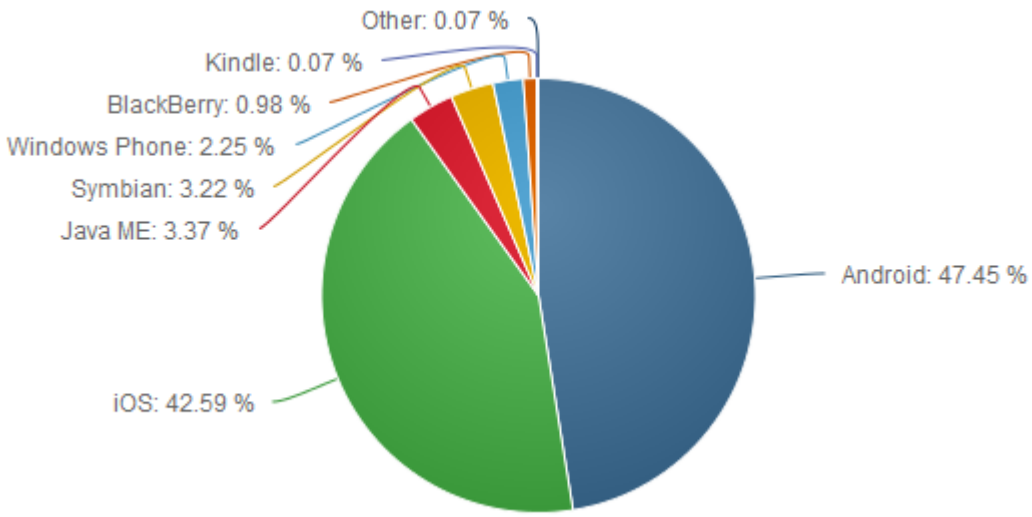


Figura 1. Dispositivos móviles más usados en febrero de 2015

Los dispositivos móviles, en especial los que cuentan con sistema operativo *Android* que es el más difundido según las estadísticas vistas anteriormente, han dejado de ser un mero instrumento usado como agenda personal o para la comunicación interpersonal. Estos debido a sus infinidades de aplicaciones ya forman parte de la vida cotidiana, de los espacios de interacción social, de los centros de trabajos, de las empresas y de los ritos cotidianos en los que las personas se construyen como sujetos y como sociedades (Inteco, 2012).

Debido a que las funcionalidades y aplicaciones brindadas por dispositivos con sistema operativo *Android* son similares a las de un ordenador, es muy importante llevar de la mano la seguridad y protección de la información personal. Este tema es importante tratarlo con todas las precauciones pertinentes, ya que puede suceder que la confidencialidad de los datos sea quebrantada. La divulgación ilícita y el uso indebido de la información personal de los usuarios, siempre ha traído consigo perjudiciales consecuencias en la vida de las personas.

Existen variadas maneras de violar la confidencialidad e integridad de la información en un dispositivo móvil. Las amenazas más comunes a las que se ven expuestos estos son la pérdida o robo del dispositivo, infecciones por virus o *malware* vía *Internet*, robo de información vía *Bluetooth* y acceso a datos confidenciales de conversaciones, imágenes o vídeos sin autorización del propietario (Inteco, 2012).

Existen recomendaciones para incrementar la seguridad de la información personal y prevenir posibles riesgos. Algunas de estas son que al reciclar un dispositivo móvil es necesario cerciorarse de eliminar todo el contenido personal de las memorias, instalar y mantener actualizado algún antivirus, usar PIN para el acceso y tras la inactividad de los dispositivos y hacer copias de seguridad para restablecer el sistema en caso de fallos o pérdidas de información (Inteco, 2012).

Resulta que en las medidas de seguridad y protección de la información descritas hasta ahora solo se tiene en cuenta el uso del dispositivo móvil para un único usuario, es decir, los dispositivos como el PDA, el teléfono, el *tablet* o la consola son para uso individual, al fin y al cabo para eso es que fueron creados. Debido al incremento continuo de funcionalidades que ofrecen los dispositivos *Android*; existen empresas, instituciones, grupos de desarrollo e incluso hogares donde más de un usuario hacen uso de un mismo dispositivo por diferentes motivos. Esto trae como consecuencia que los datos que se almacenen por cada uno de los usuarios puedan ser accedidos y modificados por el resto sin la autorización del propietario de estos, es decir la privacidad es nula.

Tanto los usuarios de dispositivos móviles de uso personal como los de dispositivos de uso común pueden tener almacenada información no relevante, pero también puede darse el caso que la información sea sensible y que la divulgación de la misma pueda ocasionar inconvenientes. Entre los riesgos que pude causar no proteger los datos personales de un usuario en un dispositivo de uso colectivo está el robo de identidad, pues una vez que el atacante posee los datos sensibles de una persona puede usarlos para dañar su reputación. Otro de los riesgos es la pérdida o robo de la información, ya que el atacante puede extraer o eliminar la información accedida. La divulgación no autorizada, la alteración y modificación también son riesgos a los cuales puede estar expuesta la información que no posea protección en un dispositivo móvil.

Actualmente los dispositivos con sistema operativo *Android* v4.0 no cuentan con una funcionalidad con soporte para varios usuarios que permita que los usuarios tengan dentro del teléfono un sector seguro, protegido criptográficamente donde puedan almacenar todo tipo de información que sea considerada confidencial. A través de esto se identificó la necesidad de contar con un espacio en la memoria del dispositivo para almacenar información privada de los distintos usuarios y mantener dicha información en un ámbito confidencial.

Una vez analizada la problemática anterior, se propone como **problema a resolver**: ¿Cómo garantizar la confidencialidad de los datos personales de múltiples usuarios, que operan en un mismo dispositivo con sistema operativo *Android*?

De esta forma, se plantea como **objeto de estudio**, la protección de la confidencialidad de la información.

Se ha seleccionado como el **campo de acción**, cifrado y descifrado de la información confidencial en dispositivos móviles con sistema operativo *Android*.

Se tendrá como **objetivo general de investigación**:

Desarrollar un sistema informático que permita garantizar la confidencialidad de los datos personales de múltiples usuarios, que operan en un mismo dispositivo con sistema operativo *Android*.

Se plantean como **objetivos específicos**:

- Realizar un análisis de los principales conceptos asociados a la criptografía.
- Realizar el análisis y el diseño de la aplicación a desarrollar.
- Implementar una librería criptográfica compatible con el API de Java para el cifrado y descifrado de la información.
- Implementar una aplicación multicapa que permita que varios usuarios interactúen con el mismo dispositivo asegurando la confidencialidad de la información de cada uno.
- Evaluar la calidad del sistema a través de las pruebas del *software.*

Para el cumplimiento de los objetivos planteados anteriormente se trazan las siguientes **tareas de la investigación**:

- Análisis de aplicaciones criptográficas en *Android* para verificar si solucionan al problema de la investigación.
- Estudio del lenguaje de programación Java y del metalenguaje estándar XML, además del sistema de gestión de bases de datos SQLite para llevar a cabo los objetivos específicos planteados.
- Análisis de los algoritmos criptográficos de claves pública y privada para seleccionar los más adecuados a utilizar en el desarrollo del sistema.
- Análisis de la metodología de desarrollo de *software* a utilizar para guiar todas las fases del desarrollo del sistema.
- Selección de la arquitectura de *software* que más se ajuste al sistema a desarrollar.
- Estudio de los patrones de diseño más adecuados para la construcción del sistema propuesto.
- Modelado de la base de datos para una correcta estructuración de la información.
- Desarrollo de una librería criptográfica para realizar el proceso de cifrado y descifrado de los datos.
- Estudio del *framework* JUnit para mediante su uso evaluar si el funcionamiento de cada uno de los métodos de las clases no presenta errores.
- Desarrollo de pruebas de aceptación al sistema para garantizar la calidad del mismo.

Como **idea a defender** se plantea que, la realización de una aplicación *Android* que permita gestionar cuentas de usuarios, contribuirá a garantizar la confidencialidad de los datos personales de los usuarios que operan en el dispositivo.

Para el desarrollo de la investigación se hizo uso de métodos científicos ejemplos de estos son los métodos empíricos y los teóricos.

Los **métodos teóricos** que se utilizaron para darle cumplimiento a las tareas planteadas son el **Analítico-Sintético** para identificar los conceptos y las definiciones más importantes relacionadas con el cifrado y descifrado de información además de la tecnología estudiada. También el **Histórico-Lógico** para lograr una mayor comprensión del estado actual de las herramientas que garantizan confidencialidad en dispositivos móviles a partir del análisis de su evolución y las etapas principales por las que han transitado, tomar lo positivo de los sistemas que presenten características comunes para una buena fundamentación y posterior implementación.

Se utilizaron también **métodos empíricos** como el de **Observación** el cual posibilitó obtener el conocimiento debido acerca del comportamiento de los sistemas de protección de información existentes hasta el momento. La **Revisión de Documentos** para lograr una mayor comprensión referente al contenido de la protección de la confidencialidad de la información, así como para determinar la forma de trabajo. Durante la captura de requisitos la **Entrevista** permitió obtener información sobre cómo es que el cliente desea el producto final.

Para lograr una mayor comprensión del trabajo realizado se decidió estructurarlo de la siguiente manera:

Capítulo 1: Fundamentación teórica de la investigación.

Este capítulo hace un análisis de los principales conceptos relacionados con el objeto de estudio y el campo de acción. Además contiene un análisis de aplicaciones similares vinculadas al campo de acción. También presenta la metodología de desarrollo de *software* utilizada además de las herramientas, lenguajes y tecnologías a tener en cuenta para el proceso de desarrollo de soluciones basadas en *Android*.

Capítulo 2: Características del sistema.

En este capítulo se presenta una propuesta de solución a la problemática anteriormente planteada, se desglosan las características funcionales y no funcionales que debe cumplir el sistema y las personas que este involucra. También se muestran las historias de usuario, el plan de iteraciones, plan de duración de las iteraciones y el plan de entregas como artefactos generados durante el desarrollo de la solución.

Capítulo 3: Diseño del sistema.

Este capítulo elabora una propuesta del sistema a desarrollar mediante la creación de los artefactos correspondientes a la fase de diseño según plantea la metodología XP. Se muestra

la estructura de la base de datos con la que cuenta el sistema, también se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización de la herramienta.

Capítulo 4: Implementación y pruebas.

En este capítulo se exponen las principales características de la implementación. Se procede a desarrollar las tareas de la ingeniería que responden a las historias de usuario abordadas en cada iteración, luego mediante las pruebas se verifica que el producto resultante cumpla con los requerimientos definidos.

Capítulo 1: Fundamentación teórica de la investigación

1.1 Introducción

En el presente capítulo se realiza un estudio sobre la fundamentación teórica de la investigación que se desarrolla, tratando los conceptos fundamentales que se manejan durante el proceso de elaboración del *software*. Se aborda sobre los algoritmos criptográficos más difundidos en la actualidad. También se realiza un estudio de las distintas propuestas tecnológicas que se utilizarán en el desarrollo del sistema. Así como la herramienta CASE y la metodología a utilizar en el desarrollo de este trabajo.

1.2 Conceptos Fundamentales

1.2.1 Criptografía

La criptografía es el nombre genérico con el que se designan dos disciplinas opuestas y a la vez complementarias: criptografía y criptoanálisis. La criptografía se ocupa del diseño de procedimientos para cifrar; es decir, para enmascarar una determinada información de carácter confidencial. El criptoanálisis se ocupa de romper esos procedimientos para así recuperar la información. Ambas disciplinas siempre se han desarrollado de forma paralela, pues cualquier método de cifrado lleva siempre emparejado su criptoanálisis correspondiente. El objetivo final que persigue la criptografía es diseñar, implementar, implantar y hacer uso de sistemas criptográficos para dotar de alguna forma de seguridad (Pastor Franco, 1998).

1.2.2 Cifrado

Cifrar es transformar información con el fin de protegerla**.** Al aplicar cifrado, un mensaje se altera hasta volverse irreconocible o incomprensible, pero la información no se pierde, sino que puede recuperarse más adelante.

El método usado para crear y recuperar un mensaje cifrado se denomina descifrar, es un secreto compartido entre quien envía el mensaje y quien lo recibe. A la disciplina que estudia estos métodos matemáticos se le llama criptografía (Fabri, 2013).

1.2.3 Ataque criptográfico

Cuando alguien que no es el receptor intenta descifrar el mensaje sin conocimiento de la técnica empleada o de la clave requerida para descifrarlo, se habla de ataque criptográfico o criptoanálisis. Su objetivo es romper el cifrado de los mensajes (Fabri, 2013).

1.2.4 Aseguramiento de recursos o de la información

El aseguramiento de la información no es más que mecanismos que se establecen para lograr resguardar y proteger la información con el objetivo de mantener la confidencialidad, la disponibilidad e integridad de la misma, de manera que la seguridad de la información sea confiable e invulnerable ante cualquier ataque (Información, 2015).

1.2.5 Confidencialidad

Es la propiedad de prevenir la divulgación de información a personas o sistemas no autorizados. A groso modo, la confidencialidad es el acceso a la información únicamente por personas que cuenten con la debida autorización (Confidencialidad, 2015).

1.3 Estudio de aplicaciones similares vinculadas al campo de acción

En este epígrafe se presentan algunas de las aplicaciones *Android* existentes hasta el momento que brindan protección a la información personal de usuarios que operan en dispositivos móviles.

1.3.1 Universal Encryption App

Su principal función es la de proteger con una contraseña cualquier archivo o carpeta de cualquier tipo que se tenga en el dispositivo, como fotos, archivos de audio, archivos de texto, etc. Una vez cifrados, sólo quien conozca la contraseña podrá verlos. Pero también dispone de otras utilidades, como generador y administrador de contraseñas, o cifrado de texto (Android, 2015).

1.3.2 Encryption Manager

Presenta facilidad para descifrar archivos para su modificación y que la propia aplicación se encarga de volver a cifrar una vez guardado, borrando cualquier rastro. Se puede acceder a los archivos cifrados desde dentro de la propia aplicación o en el navegador de archivos del dispositivo, introduciendo la contraseña de cada uno individualmente (Android, 2015).

1.3.3 File Locker

Consiste simplemente en añadir contraseñas a los archivos o carpetas, sin más utilidades adicionales. Se puede acceder a los archivos a través de la aplicación, que muestra todos los archivos cifrados y se puede cifrar incluso los nombres de los archivos, para que no den pista de su contenido. Se puede elegir que envíe recordatorios en caso de que se haya descifrado un archivo que se quiera volver a cifrar más tarde, para que el usuario no olvide que la protección aún no ha sido restaurada (Android, 2015).

1.3.4 Encrypt It

Esta aplicación es para los casos que simplemente se quiera mantener a salvo textos de carácter privado (Android, 2015).

1.3.5 Boxcryptor Classic

Aplicación especializada en la nube, de forma que automáticamente, antes de que un archivo se suba a la nube, se le aplicará un filtro que lo caracteriza con una contraseña, que será necesaria para poder verlo. De esta forma, si alguien piratea la carpeta en la nube, se verá con la dificultad de tener que piratear también la contraseña, en caso de que sea capaz de hacerlo (Android, 2015).

1.3.6 Necesidad una nueva herramienta

Las aplicaciones descritas anteriormente propician la seguridad y protección a la información sensible almacenada en un dispositivo a través del cifrado, sin embargo no son capaces de gestionar múltiples cuentas de usuarios, para las personas que posean información cifrada dentro del dispositivo. Las cuentas de usuario permiten que distintas personas utilicen un mismo dispositivo o equipo y que cada una disponga de carpetas propias y de opciones de configuración personalizadas.

Otro inconveniente de algunas de las aplicaciones antes mencionadas, es que el usuario debe ingresar una contraseña que puede ser distinta o no, cada vez que desee proteger un fichero o una carpeta, este proceso en ocasiones puede ser un poco engorroso, ya que si el usuario utiliza más de una contraseña para proteger varios archivos, puede suceder que olvide alguna propiciando que se pierdan los datos. Como inconveniente económico está presente que la versión gratuita de *Encryption Manager* tiene la limitación de 5 archivos cifrados (Android, 2015). Debido a los motivos mencionados surge la necesidad de crear la aplicación *Android* denominada “Datos Protegidos”.

1.4 Algoritmos de criptografía

Para el desarrollo de la presente investigación se debe hacer uso de algoritmos de criptografía lo suficientemente resistentes como para garantizar la confidencialidad de los datos de los usuarios, pero que a su vez no consuman mucho recursos en cuanto al *hardware* del dispositivo, ya que esta es una limitación presente en los dispositivos móviles. El impacto de elegir los algoritmos más adecuados es trascendental para el éxito del *software.* A continuación se describen una serie de algoritmos para seleccionar los más adecuados de acuerdo a algunos criterios como seguridad, velocidad de procesamiento, simplicidad del diseño y flexibilidad.

1.4.1 Algoritmos de cifrado simétricos

Estos implementan un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario y éste lo descifra con la misma clave.

1.4.1.1 Data Encryption Standard (DES)

DES es un algoritmo que toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud. Presenta un tamaño de bloque de 64 bits. Utiliza también una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en el cifrado. La clave mide 64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad y después son descartados. Por tanto, la longitud de clave efectiva en DES es de 56 bits y así es como se suele especificar. La flexibilidad de DES reside en que el mismo algoritmo puede ser utilizado tanto para cifrar como para descifrar, simplemente invirtiendo el orden de las 16 subclaves obtenidas a partir de la clave de cifrado (Paar, 2009).

1.4.1.2 International Data Encryption Algorithm (IDEA)

IDEA opera con bloques de 64 bits usando una clave de 128 bits y consiste de ocho transformaciones idénticas (cada una llamada una ronda) y una transformación de salida (llamada media ronda). El proceso para cifrar y descifrar es similar. Gran parte de la seguridad de IDEA deriva del intercalado de operaciones de distintos grupos adición y multiplicación modular y O-exclusivo (XOR) bit a bit que son algebraicamente "incompatibles" en cierta forma (Hüseyin, 2003).

Este algoritmo presenta diferencias notables con el DES, que lo hacen más atractivo:

- El espacio de claves es mucho más grande: $2^{128} \approx 3.4 \times 10^{38}$.
- Todas las operaciones son algebraicas.
- No existen operaciones a nivel de bits, facilitando su programación en alto nivel.
- Es más eficiente que los algoritmos de tipo Feistel, porque a cada vuelta se modifican todos los bits de bloque y no solamente la mitad.
- Se pueden utilizar todos los modos de operación definidos para el DES.

1.4.1.3 Advanced Encryption Standard (AES)

La estructura del algoritmo AES está formada por un conjunto de rondas, entendiendo por rondas un conjunto de reiteraciones de 4 funciones matemáticas diferentes e invertibles. AES presenta un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 o 256 bits. Este es un sistema simétrico de cifrado por bloques, por tanto utiliza la misma clave para el proceso de cifrado como para el proceso de descifrado.

La mayoría de los cálculos del algoritmo AES se hacen en un campo finito determinado, este algoritmo tiene una descripción matemática muy ordenada lo que lo hace eficiente y a la misma vez robusto ante cualquier ataque (Rijmen, 2002).

1.4.2 Algoritmos de cifrado asimétricos

Estos implementan un método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Si el remitente usa la clave pública del destinatario para cifrar el mensaje, una vez cifrado, sólo la clave privada del destinatario podrá descifrar este mensaje, ya que es el único que la conoce. Por tanto se logra la confidencialidad del envío del mensaje y nadie salvo el destinatario puede descifrarlo.

1.4.2.1 Diffie-Hellman

Se emplea generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión (establecer clave de sesión). Siendo no autenticado, sin embargo, provee las bases para varios protocolos autenticados. Su seguridad radica en la extrema dificultad (conjeturada, no demostrada) de calcular logaritmos discretos en un cuerpo finito. El sistema se basa en la idea de que dos interlocutores pueden generar conjuntamente una clave compartida sin que un intruso que esté escuchando las comunicaciones pueda llegar a obtenerla.

Para ello cada interlocutor elige un número público y un número secreto. Usando una fórmula matemática, que incluye la exponenciación, cada interlocutor hace una serie de operaciones con los dos números públicos y el secreto. A continuación los interlocutores se intercambian los resultados de forma pública. En teoría revertir esta función es tan difícil como calcular un

logaritmo discreto (Un millón de millones de cuadrillones más costosa que la exponenciación usada para transformar los números). Por eso se dice que este número es el resultado de aplicar una función unidireccional al número secreto.

A continuación ambos interlocutores utilizan por separado una fórmula matemática que combina los dos números transformados con su número secreto y al final los dos llegan al mismo número resultado que será la clave compartida (Vanstone, 1997).

1.4.2.2 Rivest, Shamir y Adleman (RSA)

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200} y se prevé que su tamaño crezca con el aumento de la capacidad de cálculo de los ordenadores.

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.

Se cree que RSA será seguro mientras no se conozcan formas rápidas de descomponer un número grande en producto de primos. La computación cuántica podría proveer de una solución a este problema de factorización (Cormen, 2011).

1.4.2.3 ElGamal

En el procedimiento de cifrado/descifrado ElGamal se refiere a un esquema de cifrado basado en problemas matemáticos de logaritmos discretos. Es un algoritmo de criptografía asimétrica basado en la idea de Diffie-Hellman y que funciona de una forma parecida a este algoritmo discreto.

El algoritmo de ElGamal puede ser utilizado tanto para generar firmas digitales como para cifrar o descifrar, fue descrito por Taher Elgamal en 1984 y se usa en *software* GNU *Privacy Guard*, versiones recientes de PGP y otros sistemas criptográficos. Este algoritmo no está bajo ninguna patente lo que lo hace de uso libre.

La seguridad del algoritmo se basa en la suposición que la función utilizada es de un sólo sentido y la dificultad de calcular un logaritmo discreto. El procedimiento de cifrado y descifrado está basado en cálculos sobre un grupo cíclico cualquiera **G** lo que lleva a que la seguridad del mismo dependa de la dificultad de calcular logaritmos discretos en **G** (Cormen, 2011)**.**

1.4.3 Algoritmo seleccionado

Pese a que los algoritmos asimétricos pudieran proveer una implementación y velocidad de cálculo mejor que los algoritmos simétricos, la realidad es otra: El cifrado simétrico es más rápido que el cifrado asimétrico, debido a la simplicidad de los algoritmos de clave privada necesarios para un determinado nivel de seguridad. Los algoritmos de cifrado asimétricos son más complejos, debido a que sus claves deben ser largas y se tardan bastante tiempo en aplicarlas. El cifrado asimétrico se basa en funciones matemáticas muy complejas, computacionalmente intensivo y no es muy eficiente para los dispositivos móviles (Salama, 2010), además es necesario seleccionar un algoritmo que se pueda implementar en un dispositivo móvil y no consuma demasiados recursos al ser estos limitados en el mismo. El cifrado asimétrico normalmente lleva 100 veces más tiempo para cifrar un mensaje dado que el simétrico (Cormen, 2011).

Por lo anterior, la categoría de algoritmos simétricos es la más apta para comenzar a probar y garantizar la confidencialidad a la información. Luego del análisis de algunos de los algoritmos de cifrado simétricos más usados en la actualidad, se decide hacer uso de los algoritmos AES e IDEA. Se utilizó más de un algoritmo a petición del cliente, para brindar al usuario la posibilidad de seleccionar el que más se ajuste a sus necesidades en caso que este posea conocimientos de criptografía. El algoritmo AES fue seleccionado teniendo en cuenta el alto grado de seguridad y confianza que posee debido a su nivel de fortaleza, este consume pocos recursos en cuanto a hardware y software, posee un rápido nivel de procesamiento y su diseño es simple. La selección del algoritmo IDEA está enmarcada a que posee una velocidad rápida de procesamiento, su diseño es simple y su longitud de clave dificulta en la práctica un ataque por fuerza bruta (Salama, 2010).

A continuación se muestra una tabla comparativa entre los algoritmos simétricos descritos en el epígrafe anterior, donde se muestra algunos de los criterios por los cuales se procedió a la elección de los algoritmos:

Tabla 1. Comparación de Algoritmos Criptográficos Simétricos

Algoritmo	Tamaño de bloques	Longitud de clave	Velocidad de procesamiento	Simplicidad del diseño	Flexibilidad
DES	64	56	Rápido	Simple	Si
IDEA	64	128	Rápido	Simple	Si
AES	128	128,192,256	Rápido	Simple	Si

1.5 Lenguajes

1.5.1 Java

Java es un lenguaje de programación orientado a objetos. Se creó con cinco objetivos principales: usar el paradigma de programación orientada a objetos, permitir la ejecución de un mismo programa en múltiples sistemas operativos, incluir por defecto soporte para trabajo en red, diseñarse para ejecutar código en sistemas remotos de forma segura, ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos.

Una de sus características es la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este lenguaje es independiente de la plataforma y posee un entorno de ejecución ligero y gratuito. El entorno de ejecución Java se ha convertido en un componente habitual en los computadores de usuario de los sistemas operativos más usados en el mundo. Muchas aplicaciones Java lo incluyen dentro del propio paquete de la aplicación de modo que se pueda ejecutar en cualquier ordenador (Java, 2015). Además es el lenguaje utilizado para el desarrollo de aplicaciones *Android*, en la investigación se hace uso de su versión 1.7.

1.5.2 Extensible Markup Language (XML)

Es un metalenguaje estándar y extensible de etiquetas que permite definir la gramática de lenguajes específicos. Permite al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de éstos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello.

Todo lo que se puede definir vía XML también se puede hacer directamente desde código Java. En cuestiones técnicas, el hecho de tener separado el código de diseño y el código de funcionalidad, es que si en cualquier momento se necesita editar la interfaz de usuario, esta tarea se facilita si se tiene un archivo con la estructura que caracteriza a XML (XML, 2015).

En *Android* se utiliza XML para la declaración de *layouts* y otros elementos de los que hará uso la aplicación para su correcto funcionamiento. De esta forma, características como la instanciación dinámica se reserva para escenarios más complejos en dónde los elementos de las interfaces de usuarios no se conocen en tiempo de compilación. En el desarrollo del sistema para cumplir con los objetivos trazados se utilizó la versión 1.0 de este metalenguaje.

1.5.3 Lenguaje de Modelado Unificado (UML)

Para el desarrollo de la aplicación se utilizó UML v2.4.1, como el lenguaje con que se modelaron los artefactos que se crean en el proceso de desarrollo del *software*. UML es un lenguaje de construcción de modelos para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de *software*, con los que se construyen mayormente sistemas orientados a objetos.

Constituye una forma de modelar elementos conceptuales como los procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reutilizables (Pooley, 2002).

1.6 Herramientas y tecnologías

1.6.1 Eclipse

Es un entorno de desarrollo integrado (IDE), de código abierto y multiplataforma, es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. No es más que un IDE en el que se encuentran todas las herramientas y funciones necesarias para el desarrollo de aplicaciones, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar (Eclipse, 2015). En el desarrollo de la investigación se hace uso de la versión 3.8.0.

1.6.2 Android SDK

El SDK (*Software Development Kit*) de *Android*, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen Linux (cualquier distribución moderna), Mac OS X 10.4.9 o posterior y *Windows* XP o posterior. La plataforma integral de desarrollo soportada oficialmente es Eclipse junto con el complemento ADT, aunque también puede utilizarse un editor de texto para escribir ficheros Java, XML y utilizar comandos en un terminal para esto se necesitan los paquetes JDK para crear y depurar aplicaciones. Además, pueden controlarse dispositivos *Android* que estén conectados (es decir, reiniciarlos, instalar aplicaciones en remoto, etc.) (SDK, 2015). En el desarrollo de la investigación se hace uso de la versión 16.0.2.

1.6.3 Android Developer Tools (ADT)

ADT es un *plugin* para Eclipse que proporciona un conjunto de herramientas que se integran con el IDE de Eclipse. Ofrece acceso a muchas características que le ayudan a desarrollar

aplicaciones de *Android*. ADT proporciona acceso GUI a muchas de las herramientas del SDK de línea de comandos, así como una herramienta de diseño de interfaz de usuario para creación rápida de prototipos, diseño y construcción de la interfaz de usuario de la aplicación (Tools, 2015). En el desarrollo de la investigación se hace uso de la versión 21.1.0.

1.6.4 SQLite

SQLite es un sistema de gestión de base de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

A diferencia de los sistemas de gestión de base de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción (SQLite, 2015). SQLite es el sistema de gestión de base de datos que *Android* usa para el manejo de los datos, por tal motivo se hace uso de la versión 3.8.10.2 en la construcción de este sistema.

1.6.5 JUnit

JUnit es un conjunto de clases (*framework*) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente (JUnit, 2015). La versión de JUnit usada es la 4.10.

1.6.6 Java Develpment Kit (JDK)

El JDK es una pieza fundamental para el ecosistema Java, pues incluye numerosas utilidades que permiten compilar, probar y distribuir aplicaciones. El componente más famoso es *javac*, el compilador que convierte el código fuente en código ejecutable por el ordenador. Pero hay más, como *javadoc*, que crea documentación a partir de los comentarios en el código y *jar*, que empaqueta y comprime los archivos. JDK incluye también la máquina virtual de Java, el componente conocido como JRE (*Java Runtime Environment*), indispensable para abrir

aplicaciones Java en el navegador y el escritorio. Por otro lado, el JDK no es un kit de desarrollo completo (SDK), pues carece de otras herramientas, siendo la más notable un IDE (Java, 2015). La versión del JDK usada es la 1.7.

1.6.7 Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Está diseñado para una amplia gama de usuarios, incluidos los Ingenieros de *software*, Analistas de sistemas, Analistas de negocios, Sistema de arquitectos y todo aquel que esté interesado en la construcción de sistemas de *software* a gran escala mediante el uso fiable; es orientado a objetos (Paradigm, 2015). La versión de la herramienta usada es 8.0.

1.7 Metodología de desarrollo de software

El proceso de desarrollar *software* no es una tarea fácil, se debe contar con un transcurso bien detallado y para esto se necesita aplicar una metodología que sea capaz de llevar a cabo el control total del producto. Las metodologías de desarrollo surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un sistema.

Las metodologías tienen un papel fundamental, ya que el éxito y la calidad del producto dependen en gran parte de la metodología escogida, sea tradicional o ágil. Esta debe ser capaz de guiar y organizar las actividades que conlleven a cumplir las metas trazadas y que satisfaga más allá de las necesidades definidas al inicio del proyecto.

1.7.1 Extreme Programming (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requerimientos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Letelier, 2006).

A continuación se presentan algunas características de XP que se adaptan a las necesidades de un proyecto, así como a las condiciones de trabajo:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera es revisado y discutido mientras se escribe.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores sean detectados.
- Simplicidad en el código: la programación extrema apuesta que es más sencillo hacer un código simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar funciones complicadas y quizás nunca utilizarlas.

La metodología de desarrollo ágil XP fue seleccionada para guiar todas las fases de evolución del sistema, ya que la misma está concebida para ser utilizada dentro de proyectos pequeños y de desarrollo rápido, se adapta perfectamente al presente desarrollo debido a que el periodo de entrega del producto oscila alrededor de 6 meses y el cliente está vinculado al equipo de desarrollo. Esta metodología está dirigida a grupos de trabajo pequeños y con pocos roles, en este caso el equipo de desarrollo está compuesto por 2 integrantes. A medida que el proyecto avanza pueden surgir nuevas expectativas o ideas que pueden ser incorporadas fácilmente permitiéndole mayor adaptabilidad al producto, con la metodología XP esto es completamente factible pues esta se adapta perfectamente a los proyectos cuyos requerimientos cambian a menudo.

1.8 Conclusiones del capítulo

Después de realizar un estudio referente a la criptografía y sus conceptos fundamentales se decide hacer uso de algoritmos AES e IDEA debido a su rapidez, simplicidad de diseño, longitud de clave, tamaño de bloques y flexibilidad. El estudio de algunas de las herramientas similares referentes al campo de acción arrojó como resultado la necesidad de la creación de una nueva herramienta. Luego del análisis de las características que brinda XP, se decidió hacer uso de esta como metodología para guiar el proceso de desarrollo del sistema.

Capítulo 2: Características del sistema.

2.1 Introducción al capítulo

Este capítulo tiene como objetivo presentar las características principales del sistema a desarrollar, los requerimientos funcionales y no funcionales que debe cumplir y las personas que involucra. También incluye las historias de usuario, el plan de iteraciones y el plan de entregas como artefactos generados durante el desarrollo de la solución.

2.2 Propuesta del sistema

Se propone la implementación de un sistema informático que sea capaz de brindar confidencialidad a la información de múltiples usuarios que operen en un mismo dispositivo. El sistema contará con restricciones de acceso, de acuerdo al rol que posea el usuario que interactúe con él. Existirán solo dos roles, el rol administrador y el rol registrado. El usuario que cuente con el rol administrador, es el encargado de gestionar las cuentas de usuarios y realizar tareas como eliminar cuentas, bloquear usuarios, permitir o no la creación de nuevas cuentas además de reiniciar el sistema en caso de ser necesario. La aplicación solo admitirá la existencia de un único usuario con este rol. El usuario que posea el rol registrado tendrá privilegios sobre el sistema que le permitirán el uso pleno de las funcionalidades de la aplicación. Los usuarios serán libres de crear nuevas cuentas mientras el administrador lo permita.

El sistema poseerá tres funcionalidades fundamentales destinadas a los usuarios con el rol registrado que son las que justifican la importancia de la investigación. La primera de ellas es la protección de ficheros privados, en esta, la aplicación brindará la opción de importar ficheros que sean considerados como información confidencial, el proceso de cifrado de la información quedará como una operación totalmente transparente para el usuario y se le dará la sensación de que está navegando por directorios físicos reales. La segunda es la creación de una lista de contactos privada, ofreciendo el privilegio de que el usuario cree nuevos contactos o importe los contactos de su lista nativa. La tercera es la creación de una lista de mensajes privados, en esta se le brindará la opción de seleccionar de sus mensajes en el dispositivo, los que considere confidenciales y moverlos hacia su cuenta privada.

La información de cada usuario permanecerá cifrada con una clave privada que solo él conocerá y se descifrará en el momento que el usuario esté autenticado y desee examinarla. El algoritmo por defecto que llevará a cabo la lógica de cifrado es el algoritmo AES, aunque también se brindará la opción de escoger el algoritmo IDEA durante el proceso de registro de una nueva cuenta.

La interfaz gráfica será amigable y permitirá a los usuarios una fácil interacción con el sistema. También incluirá la ayuda para los usuarios que no posean experiencia haciendo uso del sistema.

2.3 Definición de audiencia

Definir la audiencia a la que va dirigida el sistema implementado constituye uno de los pasos fundamentales para alcanzar el éxito en los objetivos propuestos. La audiencia es el público hacia el cual está orientada la aplicación. Este sistema informático va dirigido especialmente a los usuarios que operan con dispositivos móviles con sistema operativo *Android* v4.0.

2.4 Usuarios relacionados con el sistema

Se denomina usuario a cualquier persona relacionada con el sistema, ya sea vinculada al desarrollo del mismo o que de una forma u otra interactúa con la aplicación, incluyendo a los que mantendrán el sistema funcionando y actualizado.
Los usuarios relacionados con la aplicación son las personas que de manera directa o indirecta interactúan con la misma, tanto los que obtienen resultados de valor con los procesos que se ejecutan como los que no obtienen ningún resultado. En el proyecto se definieron los siguientes usuarios:

Tabla 2. Personas relacionadas con el sistema

Persona	Descripción
Usuario registrado	Es la persona que tiene una cuenta de usuario en la aplicación y solicita la autenticación de sus credenciales para acceder a su sesión.
Usuario *root* o administrador	Es la persona que tiene los privilegios de crear, modificar, bloquear y eliminar las cuentas de usuarios, permitir o no la creación de nuevas cuentas en la aplicación además de reiniciar el sistema en caso de ser necesario.

2.5 Características funcionales y no funcionales del sistema

Antes de poder plantear una solución a un problema siempre se ha hecho preciso llevar a cabo una reunión con el cliente, para llegar a acuerdos y establecer las características que se quiera que posea el producto. De ahí es común establecer en los primeros encuentros que se tienen

con los clientes, peticiones informales de las características del *software* y diseños que a la larga aportarán mucho a la concepción del *software*.

La metodología XP propone que se realice un levantamiento de las características del sistema en las primeras reuniones, en las que se tomen todos los aspectos relacionados con las necesidades del sistema a desarrollar. En estos encuentros se pueden llevar a cabo prácticas muy efectivas como es el caso de las entrevistas con el cliente.

2.5.1 Características funcionales del sistema

Las funcionalidades del sistema son las capacidades o condiciones que el sistema debe cumplir. El sistema a desarrollar posee las siguientes funcionalidades:

- F1: Registrar usuario.
- F2: Autenticar usuario.
- F3: Gestionar usuarios.
  - Listar usuarios.
  - Eliminar usuario.
  - Bloquear usuario.
  - Desbloquear usuario.
  - Cambiar contraseña.
  - Mostrar información de usuario.
- F4: Iniciar sesión.
- F5: Cargar datos.
  - Descifrar datos.
  - Cargar datos.
- F6: Gestionar ficheros.
  - Eliminar fichero.
  - Renombrar fichero.
  - Copiar fichero.
  - Cortar fichero.
  - Exportar a memoria del teléfono.
  - Importar desde memoria del teléfono.
  - Mostrar detalles de fichero.
- F7: Gestionar directorios.
  - Eliminar carpeta.
  - Renombrar carpeta.
  - Copiar carpeta.

- Cortar carpeta.
- Mostrar detalles de carpetas.
- Crear nueva carpeta.
- Buscar.
- Actualizar directorio.

- F8: Ejecutar fichero.
- F9: Guardar datos.
  - Cifrar datos.
  - Guardar datos.
- F10: Mostrar ruta.
- F11: Gestionar contactos.
  - Listar contactos del dispositivo.
  - Importar contacto del dispositivo.
  - Eliminar contacto del dispositivo.
  - Listar contactos privados.
  - Adicionar contacto privado.
  - Eliminar contacto privado.
  - Editar contacto privado.
  - Llamar a contacto privado.
- F12: Gestionar mensajes.
  - Listar mensajes del dispositivo.
  - Importar mensaje del dispositivo.
  - Eliminar mensaje del dispositivo.
  - Listar mensajes privados.
  - Eliminar mensaje privado.
- F13: Reiniciar aplicación.
- F14: Mostrar disponibilidad de memoria.
- F15: Gestionar creación de cuentas.
- F16: Cerrar sesión.

2.5.2 Características no funcionales del sistema

Usabilidad

Para lograr una mejor usabilidad del sistema se debe tener en cuenta algunos elementos de diseño como encabezados de pantalla para guiar al usuario, estilos y formatos de textos sencillos y entendibles. Las interfaces de usuario con las que cuenta el sistema fueron diseñadas mediante las normas de diseño UI establecidas por *Android* (Interface, 2015).

Además se verifica la información de todos los campos de entrada de datos y en caso de posibles errores se muestra un mensaje que ayude a corregirlos. También se muestran mensajes de confirmación en caso que se desee eliminar información. Los colores utilizados deben ser claros como por ejemplo el verde o el azul, para lograr una mejor aceptación del producto por parte del cliente.

El sistema cuenta con una ayuda para que el usuario comprenda el objetivo de cada funcionalidad, además podrá ser utilizado por cualquier usuario con las siguientes características:

- Conocimientos básicos relativos al uso de un teléfono inteligente.
- Conocimientos básicos del sistema operativo *Android*.

Seguridad

Los requisitos vinculados a la seguridad de la aplicación son los que generan mayores riesgos si no se manejan correctamente. Estos pueden ser tratados en tres aspectos diferentes: confidencialidad, integridad y disponibilidad.

- **Confidencialidad**

Los datos serán almacenados de forma segura y protegidos con algoritmos de cifrado AES e IDEA. El acceso a los datos será controlado, de manera que cada usuario solo pueda acceder a sus datos personales mediante en inicio en su sesión y no pueda tener acceso a los datos de otros usuarios registrados en la aplicación. El contenido de la base de datos permanecerá cifrado, de manera que si se intenta acceder a este fichero, no se podrá adquirir ninguna información.

- **Integridad**

El sistema no garantiza la integridad de la información, debido a que se puede acceder a la base de datos y modificar su información.

- **Disponibilidad**

El sistema no garantiza la disponibilidad de la información, ya que los datos generados pueden ser borrados.

Hardware

Para la instalación y buen funcionamiento de la aplicación en un entorno emulado se debe disponer de una computadora de 1GB de RAM o superior y un procesador con velocidad de 1.7 GHz o superior. El emulador requiere de la versión 4.0 del sistema operativo *Android*.

Para la instalación y buen funcionamiento de la aplicación en un dispositivo móvil se debe contar con 512 MB de RAM o superior, almacenamiento interno y obligatoriamente el dispositivo debe contar con tarjeta Micro SD con capacidad de 2GB como mínimo requerimiento.

Software

La instalación de la aplicación en un entorno emulado requiere de la máquina virtual de Java en su versión 1.7 o superior, SDK de *Android* v16.0.2 y el *plugin* de Eclipse ADT v21.1.0. La instalación de la aplicación en un dispositivo real requiere de una versión del sistema operativo *Android* v4.0.

2.6 Fase de exploración.

El ciclo de vida de un proyecto realizado con la metodología XP inicia con la fase de exploración. En esta fase, los clientes plantean a grandes rasgos las historias de usuario. Al finalizar el equipo cuenta con suficiente material de trabajo como para producir una primera entrega. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y tecnologías que serán utilizadas en el proyecto.

2.6.1 Historias de usuario

Las historias de usuario (HU) constituyen el único artefacto de requerimientos generado por la metodología XP. De forma similar a los casos de uso que construyen otras metodologías, intentan describir los requerimientos del sistema pero en muy pocas líneas, de manera clara y sin utilizar lenguaje técnico. Se redactan desde la perspectiva del cliente y el contenido que abarcan debe ser sencillo pero concreto.

En el proceso de desarrollo de un *software*, las HU resultan claves para lograr una comprensión adecuada de las necesidades del cliente. Además, ayudan a estimar tiempo de desarrollo y sirven de base para realizar las pruebas de aceptación. Generan poca documentación lo que constituye una forma rápida de administrar los requerimientos y responder ante las modificaciones que estos puedan sufrir durante el ciclo de desarrollo del sistema.

Las HU se clasifican según:

La prioridad en el negocio:

Alta: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.

Media: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en su desarrollo:

Alto: Cuando en la implementación de las HU se consideran la posible existencia de errores que conlleven a la inoperatividad del código.

Medio: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

Bajo: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Las HU son representadas mediante tablas divididas por las siguientes secciones:

- **Número:** Esta sección representa el número, incremental en el tiempo, de la HU que se describe.
- **Nombre de Historia de Usuario:** Identifica la HU que se describe entre los desarrolladores y el cliente.
- **Modificación de Historia de Usuario:** Número de sección que representa si la HU se le realizó alguna modificación con respecto al estado anterior.
- **Usuario:** Los programadores responsables de la historia de usuario.
- **Iteración asignada:** Número de la iteración donde va a desarrollarse la HU.
- **Prioridad en negocio:** Se le otorga una prioridad (Alta, Media, Baja) a las HU de acuerdo a la necesidad de desarrollo.
- **Riesgo en Desarrollo:** Se le otorga una medida de (Alto, Medio, Bajo), a la ocurrencia de errores en el proceso de desarrollo de la HU.
- **Puntos Estimados:** Es el tiempo estimado en semanas que se demorará el desarrollo de la HU.
- **Puntos Reales:** Representa el tiempo que se demoró en realidad el desarrollo de la HU.
- **Descripción:** Breve descripción de la HU.
- **Observaciones:** Señalamiento o advertencia del sistema.

Para contribuir con la implementación del sistema propuesto se identificaron las HU que se detallan a continuación.

Tabla 3. HU para la F1

Historia de usuario	
Número: 1	**Nombre de la HU:** Registrar usuario
Modificación de historia de usuario: Ninguna	
Usuario: Osmanis del Risco Ferral Alberto Baxter Barrizonte	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 0.2
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Registra un nuevo usuario en la aplicación	
Observaciones: Hace referencia a la funcionalidad F1	
Prototipo: 	

Tabla 4. HU para la F2

Historia de usuario	
Número: 2	**Nombre de la HU:** Autenticar usuario
Modificación de historia de usuario: Ninguna	
Usuario: Osmanis del Risco Ferral Alberto Baxter Barrizonte	**Iteración asignada:** 1

Prioridad en negocio: Alta	**Puntos estimados:** 0.2
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Permite la autenticación de los usuarios registrados a priori en la aplicación	
Observaciones: Hace referencia a la funcionalidad F2	
Prototipo: 	

En el **Anexo II** se muestran las restantes HU.

2.7 Fase de planificación

En esta fase el cliente establece la prioridad que tendrá cada HU según sus necesidades más inmediatas, luego los programadores realizan una estimación del esfuerzo que se necesita para cada una de ellas. La fase de planeamiento toma un par de días. Se deben incluir varias iteraciones para lograr un buen producto. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a tres semanas en ejecución (Letelier, 2006).

2.7.1 Estimación de esfuerzo

La estimación por esfuerzo consiste en asignarle puntos a las HU y cada uno de estos puntos son equivalentes a una semana. Los programadores se basan para realizar la estimación en la complejidad que pueden tener las HU y en el tiempo hábil para el desarrollo de la aplicación. Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen

utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos (Letelier, 2006).

Para el logro del sistema se ha realizado la estimación de esfuerzo por cada HU, teniendo en cuenta la complejidad de cada una de las funcionalidades requeridas por el cliente. A continuación se muestran las estimaciones de esfuerzo para cada HU:

Tabla 5. Estimación de esfuerzos por historias de usuario

Historia de usuario	Puntos de estimación
Registrar usuario	0.2
Autenticar usuario	0.2
Gestionar usuarios	1.2
Iniciar sesión	0.2
Cargar datos	0.8
Gestionar ficheros	1.4
Gestionar directorios	1.6
Ejecutar fichero	0.2
Guardar datos	0.8
Mostrar ruta	0.2
Gestionar contactos	1.6
Gestionar mensajes	1
Reiniciar aplicación	0.2
Mostrar disponibilidad de memoria	0.2
Gestionar creación de cuentas	0.2
Cerrar sesión	0.2
	10.2

2.7.2 Plan de iteraciones

La elaboración de las HU y la estimación de los esfuerzos para la realización de cada una de ellas, sirven de base para la planificación de la implementación del sistema. Durante esta etapa, cada HU se trasforma en tareas de desarrollo, que abarcan un período del ciclo de vida del *software* o iteración. Por tanto, un plan de iteraciones, es el artefacto a través del cual se seleccionan las HU que serán implementadas en cada iteración del sistema. La implementación del sistema propuesto por esta investigación constará de cuatro iteraciones.

Iteración 1: En esta iteración se implementan las HU que permiten construir el marco de trabajo del negocio, para luego ser utilizado en el resto del proyecto, las principales son las HU

1, 2, 3, 4 y 5. Con esta iteración se obtiene la primera versión de la aplicación la cual se utilizará para mostrar al cliente y permitirá al grupo de trabajo tener una retroalimentación.

Iteración 2: En esta iteración se realiza la implementación de las HU 6 y 7. De esta forma se obtiene la segunda versión del sistema. Esta segunda iteración al igual que las sucesivas, es mostrada al cliente con el objetivo de evaluar las aceptaciones por parte del cliente con el *software*.

Iteración 3: En esta iteración se realiza la implementación de la HU 8, 9, 10 y 11. De esta forma se obtiene la tercera iteración.

Iteración 4: En esta iteración se realiza la implementación de la HU 12, 13, 14, 15 y 16. De esta forma se obtiene la cuarta y última iteración del sistema.

2.7.3 Plan de duración de las iteraciones

El plan de duración de las iteraciones es otro artefacto que se genera durante la fase de planificación. Se crea uno diferente para cada equipo de desarrollo. Su objetivo es mostrar, para cada iteración, el orden en que serán implementadas las HU y su duración total. El desarrollo de la solución propuesta está a cargo de un único equipo de desarrollo, para el cual, el plan de duración de las iteraciones quedaría de la siguiente manera:

Tabla 6. Plan de duración de las iteraciones

Iteraciones	Orden de las HU a implementar	Duración (semanas)
Iteración 1	Registrar usuario Autenticar usuario Gestionar usuarios Iniciar sesión Cargar datos	2.6
Iteración 2	Gestionar ficheros Gestionar directorios	3
Iteración 3	Ejecutar fichero Guardar datos Mostrar ruta Gestionar contactos	2.8
Iteración 4	Gestionar mensajes Reiniciar aplicación Mostrar disponibilidad de memoria Gestionar creación de cuentas	1.8

	Cerrar sesión	

2.7.4 Plan de entregas

La fase de planificación concluye con el compromiso final del equipo de desarrollo con el cliente, generando el plan de entregas. Este plan retorna un cronograma donde se unen las funcionalidades referentes a un mismo tema o módulo, para lograr un mayor entendimiento en la implementación del sistema. A continuación, se muestra el plan de entregas elaborado para facilitar la implementación de la solución propuesta.

Tabla 7. Plan de entregas

Iteración	Fecha de entrega
1	16 de marzo de 2015
2	6 de abril de 2015
3	24 de abril de 2015
4	7 de mayo de 2015

2.8 Conclusiones del capítulo

A lo largo del capítulo se analizaron los requerimientos de la solución propuesta y en base a ellos, se generaron los artefactos: HU, plan de iteraciones, plan de duración de las iteraciones y plan de entregas. Estos artefactos constituyen los cimientos, ya que traducen las necesidades del cliente en funcionalidades, recogen las tareas que debe realizar cada desarrollador agrupadas por iteración y el tiempo máximo del que dispone para ejecutarlas.

Capítulo 3: Diseño del sistema

3.1 Introducción al capítulo

En el presente capítulo se elabora una propuesta del sistema a desarrollar mediante la creación de los artefactos correspondientes a la fase de diseño según plantea la metodología XP. Se muestra la estructura de la base de datos con la que cuenta el sistema, también se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización de la herramienta.

3.2 Diseño

Durante el diseño de la solución, la máxima simplicidad posible es la clave para el éxito de XP. Se debe tener en cuenta que un diseño complejo siempre tarda más en desarrollarse que uno simple y que siempre es más fácil añadir complejidad a un diseño simple que quitarla de uno complejo.

XP construye un proceso de diseño evolutivo que se basa en refactorizar un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, combina la disciplina con la adaptabilidad de una manera que indiscutiblemente la hace una de las más desarrolladas de entre todas las metodologías ágiles (Letelier, 2006).

3.3 Patrón arquitectónico

Un patrón arquitectónico define la estructura básica de una aplicación, provee un subconjunto de subsistemas predefinidos, incluyendo reglas, lineamientos para conectarlos, pautas para su organización y constituye una plantilla de construcción.

Entre las ventajas del uso de patrones, se pueden encontrar:

- Permiten la reutilización de soluciones arquitectónicas de calidad.
- Son de gran ayuda para controlar la complejidad de un diseño.
- Facilitan la documentación de diseños arquitectónicos.
- Proporcionan un vocabulario común que mejora la comunicación entre diseñadores.

Los patrones arquitectónicos expresan una organización estructural para un sistema, permiten estructurar los componentes y subsistemas de un sistema. La abstracción más alta en cuanto a

soluciones a través de patrones, se obtiene a través del uso de patrones de arquitectura (Reynoso, 2004).

3.3.1 Patrón arquitectónico N-Capas

Se propone la utilización del patrón arquitectónico N-Capas que tiene como objetivo principal separar los diferentes aspectos del desarrollo, tales como las cuestiones de lógica de negocio y los mecanismos de almacenamiento. Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidas, cada nivel agrega las responsabilidades y abstracciones del nivel inferior (Ramos, 2010).

El uso de este patrón permite que se puedan realizar actualizaciones en el interior de las capas sin que esto afecte el resto del sistema. Los cambios se realizan a alto nivel y se puede incrementar o reducir el nivel de abstracción que se usa en cada capa del modelo. Además aporta facilidad para las pruebas: Ya que cada capa tiene una interfaz bien definida sobre la que realizar las pruebas y la habilidad de cambiar entre diferentes implementaciones de una capa (Ramos, 2010).

3.3.1.3 Descomposición en capas propuesta

- **Capa de presentación:** Es la única que interactúa directamente con el usuario presentándole el sistema, permitiendo el intercambio de información entre ambos. Contiene una interfaz gráfica que posibilita mostrar a los usuarios los resultados de sus peticiones y estados de la aplicación. Esta capa solo interactúa con la capa de negocio, que es su inferior inmediata.
- **Capa de negocio:** Es la que recibe las peticiones de la capa de presentación y le envía las respuestas tras el proceso. Aquí se realiza la mayor parte del procesamiento de la información del dominio de la aplicación, donde se ejecutan los algoritmos específicos del programa. Esta capa interactúa con la capa de presentación, para recibir sus solicitudes y presentarle los resultados.
- **Capa de acceso a datos:** Es la que sirve como puente entre la capa lógica del negocio y el proveedor de datos. Esta capa pretende encapsular las especificidades del proveedor de datos a la siguiente capa.
- **Capa de lógica criptográfica:** Es la que se encarga de cifrar la información en claro, para mantenerla protegida.
- **Capa entidades:** Es la que posee las entidades del negocio y clases que utiliza el sistema para realizar validaciones.

Figura 2. Patrón arquitectónico N-Capas implementado

3.4 Patrones de diseño

Un patrón de diseño constituye un esquema para refinar subsistemas o componentes. Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades, además de que ayuda a construir clases y a estructurar sistemas de clases. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. (Gamma, 2003).

3.4.1 Patrones GRASP (General Responsibility Assignment Software Patterns)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos y constituyen la base del cómo se diseñará el sistema (Gamma, 2003). Los patrones GRASP usados son: Experto, Creador, Bajo Acoplamiento y Alta Cohesión.

Experto

Un Experto es una clase que tiene toda la información necesaria para implementar una responsabilidad. En la aplicación este patrón se evidencia en las clases de acceso a datos asignándoles las responsabilidades de ejecutar todas las funcionalidades comunes de las entidades que representan y de la cual poseen información.

Creador

El patrón creador permite identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que contenga la información necesaria para realizar la creación del objeto, o use directamente

las instancias creadas del objeto. En el sistema este patrón se ve reflejado en las clases de acceso a datos, las cuales cuentan con la información necesaria para la creación de instancias de las clases entidades que representan.

Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está relacionada a otras clases. Una clase con bajo o débil acoplamiento no depende de muchas otras. Es la idea de tener las clases lo menos atadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases. Ejemplo de ello lo constituyen las clases pertenecientes a la capa de cifrado que no presentan ninguna dependencia con las clases de la capa de negocio y poseen la menor dependencia posible con las clases de la capa de acceso a datos, propiciando la facilidad de que en caso de producirse un cambio tenga una repercusión mínima en el proyecto.

Alta Cohesión

Cada elemento del diseño debe realizar una labor única dentro del sistema, lo cual expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase. En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión tiene responsabilidades estrechamente relacionadas y poco complejas. Este patrón se refleja en la aplicación en clases como Explorardor.java, RegistrarUsuario.java y MensajesPrivadosdos.java pertenecientes a la capa de negocio.

3.4.2 Patrones GoF (Gang of Four)

Los patrones de diseño GoF se clasifican según el libro GOF en tres categorías: de creación, estructurales y de comportamiento. Los patrones de creación abstraen el proceso de creación de instancias, los estructurales se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño y los de comportamiento atañen a los algoritmos y a la asignación de responsabilidades entre objetos (Patrones, 2009).

Durante la implementación del sistema se hizo uso de los siguientes patrones:

***Singleton*:** Pertenece a la familia de los patrones creacionales, está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Este patrón se ve reflejado en la clase UsuarioInformacion.java, la cual permite la creación de una única instancia del objeto Usuario en el momento que el mismo se autentica.

***Composite*:** Este patrón de tipo estructural, permite a los desarrolladores tratar uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva. La utilización de este patrón se evidencia en las clases Archivo.java, Fichero.java y Directorio.java, ya que las clases Fichero.java y Directorio.java heredan directamente de la clase Archivo.java, pero a su vez la clase Directorio.java contiene implícita una lista de instancias de la clase Archivo.java. A continuación se muestra lo descrito en la Figura 3.

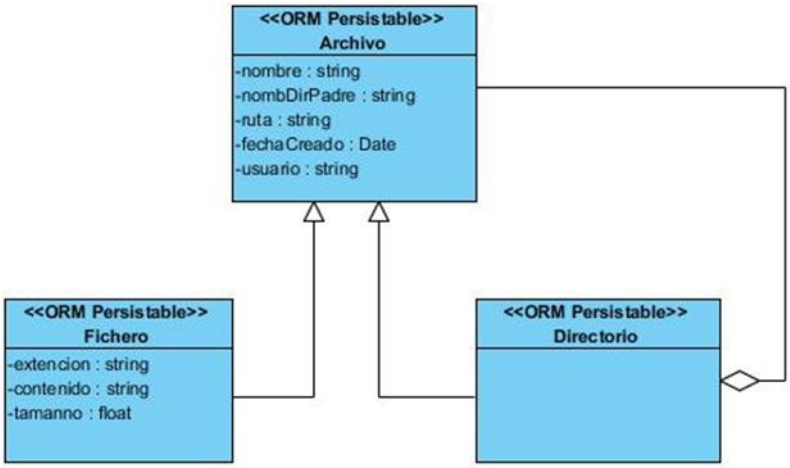


Figura 3. Uso del patrón Composite

***Facade* (Fachada):** Es un tipo de patrón estructural. Está motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre éstos. Un ejemplo claro del uso de este patrón en la aplicación se evidencia en la clase ConCriptografia.java, la cual actúa como punto de entrada en la capa Lógica de cifrado. A continuación se muestra lo descrito en la Figura 4.

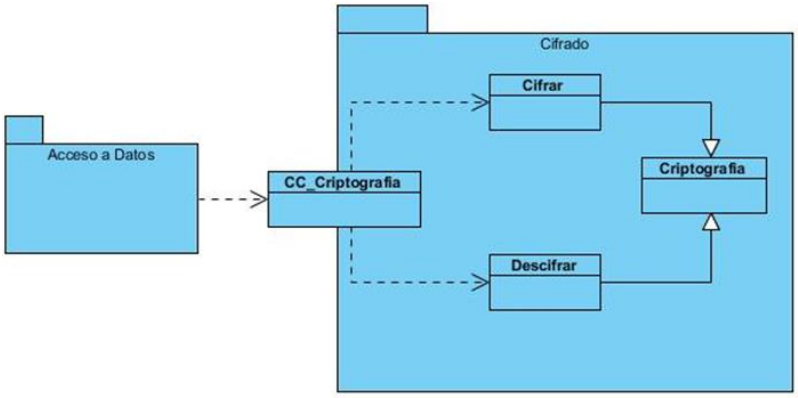


Figura 4. Uso del patrón Facade

3.5 Tarjetas CRC (Clases-Responsabilidad-Colaboración)

Diseñar un sistema utilizando la metodología XP solo es posible si se siguen tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Estos tres principios hacen posible que todo el equipo de desarrollo esté inmerso en la tarea de diseño.

Las tarjetas CRC representan objetos que se transforman en clases bien estructuradas con responsabilidades bien definidas. Son simples y adaptables, características que facilitan la interacción entre el cliente y el equipo de desarrollo (Tarjetas, 2015). Las tarjetas CRC generadas para el diseño de la solución propuesta se muestran a continuación:

Tabla 8. Tarjeta CRC Usuario

Tarjeta CRC	
Clase: Usuario	
Responsabilidades:	**Colaboradores:**
Gestiona la información de los usuarios	

Tabla 9. Tarjeta CRC Mensaje

Tarjeta CRC	
Clase: Mensaje	
Responsabilidades:	**Colaboradores:**
Gestiona la información de los mensajes	

Tabla 10. Tarjeta CRC Contacto

Tarjeta CRC	
Clase: Contacto	
Responsabilidades:	**Colaboradores:**
Gestiona la información de los contactos	

En el **Anexo IV** se muestran las restantes Tarjetas CRC.

3.6 Diseño de la base de datos

Para llevar a cabo el almacenamiento, organización de los datos y obtener acceso a la información con redundancia mínima, se hace necesaria una correcta estructuración del modelo de la base de datos. Ver Figura 5.

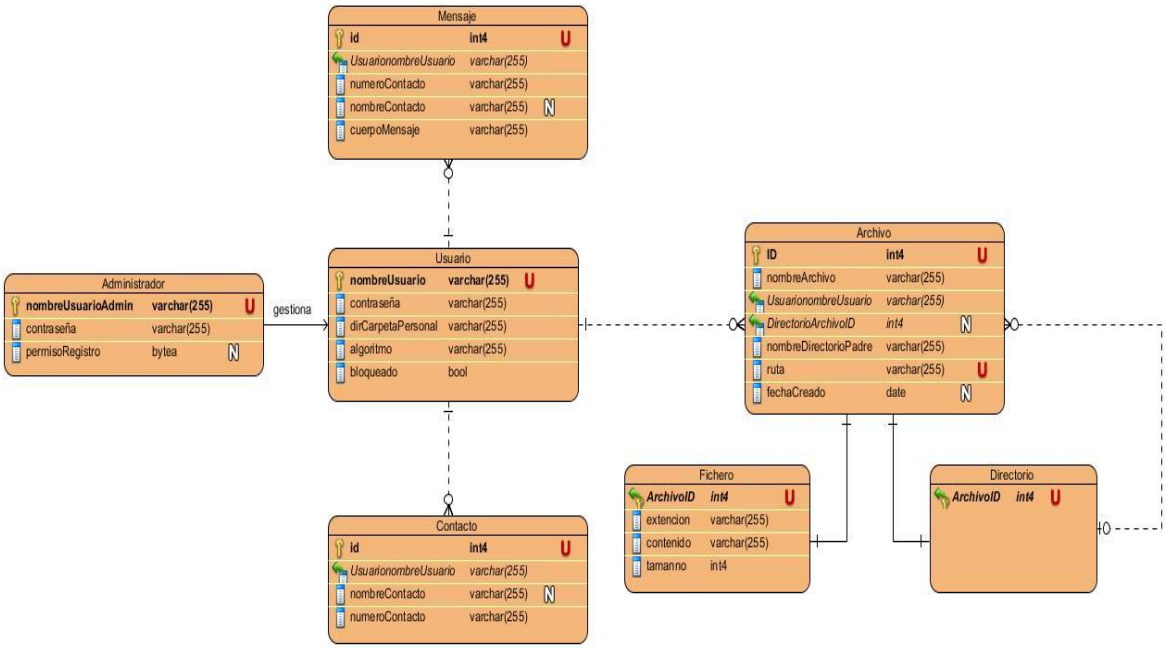


Figura 5. Diseño de la base de datos del sistema

3.8 Conclusiones del capítulo

La determinación de los patrones arquitectónicos y de diseño usados permitió lograr una mayor organización en los elementos que conforman la aplicación. La estructuración del modelo de la base de datos con la que cuenta el sistema facilita el almacenamiento y acceso rápido a los datos. La confección de las tarjetas CRC definidas para el desarrollo del sistema y los prototipos de interfaces con los que cuenta el sistema permitió conocer el comportamiento de cada una de las clases en un alto nivel.

Capítulo 4: Implementación y pruebas

4.1 Introducción al capítulo

En el presente capítulo se documentan las fases de implementación y prueba según lo propone la metodología de desarrollo XP. Se desglosan las historias de usuarios en tareas con el objetivo de facilitar el trabajo de los desarrolladores y se muestran los resultados de las pruebas unitarias y de aceptación realizadas a la aplicación.

4.2 Implementación

En esta fase XP propone tener en cuenta una serie de aspectos como son la disponibilidad del cliente y el desarrollo en pareja para lograr mayores resultados en la implementación del *software*.

Disponibilidad del cliente: El cliente formó parte del equipo de desarrollo, describió las HU, guío la toma de decisiones, aprobó las versiones del producto y verificó el cumplimiento de los objetivos trazados.

Desarrollo en pareja: Toda la implementación fue realizada por dos personas que trabajaron en forma conjunta.

4.3 Tareas de la ingeniería

Para llevar a cabo la correcta implementación de las HU se deben definir por parte del equipo de desarrollo las TI (Tareas de Ingeniería) que se realizarán en cada una de las iteraciones. Las TI también conocidas como tareas de implementación permiten a los desarrolladores obtener un nivel de detalle más avanzado que el que propicia las HU.

En el sistema propuesto se identificaron las siguientes TI:

Tabla 11. Tareas de ingeniería

Nro. HU	Nombre HU	Descripción de HU	Nro. TI	Tarea de la Ingeniería
1	Registrar usuario	Registra un nuevo usuario en la aplicación	1	Registrar usuario nuevo
2	Autenticar usuario	Permite la autenticación en la aplicación de los usuarios registrados a priori	1	Autenticar usuario
3	Gestionar usuarios	Gestiona los usuarios existentes en el sistema	1	Listar usuarios
			2	Eliminar usuario
			3	Bloquear usuario
			4	Desbloquear usuario
			5	Cambiar contraseña
			6	Mostrar información del usuario
4	Iniciar sección	Se inicia la sección de un usuario mostrando sus carpetas personales	1	Mostrar inicio sección
5	Cargar datos	Cada vez que el usuario accede a un directorio se cargan los archivos que este contiene	1	Descifrar datos
			2	Cargar datos
6	Gestionar ficheros	Permite gestionar los ficheros de los usuarios	1	Eliminar fichero
			2	Renombrar fichero
			3	Copiar fichero
			4	Cortar fichero
			5	Exportar a memoria del teléfono
			6	Importar desde memoria del teléfono
			7	Mostrar detalles de ficheros
7	Gestionar directorios	Permite gestionar las carpetas	1	Eliminar carpeta

		de los usuarios	2	Renombrar carpeta
			3	Copiar carpeta
			4	Cortar carpeta
			5	Mostrar detalles de carpetas
			6	Crear nueva carpeta
			7	Buscar
			8	Actualizar directorio
8	Ejecutar fichero	Se ejecuta el fichero y se abre con el programa correspondiente	1	Ejecutar fichero
9	Guardar datos	Se encriptan los datos del usuario y luego se guardan	1	Cifrar datos
			2	Guardar datos
10	Mostrar ruta	Muestra la ruta del directorio actual	1	Mostrar ruta
11	Gestionar contactos	Permite gestionar los contactos existentes en el sistema	1	Listar contactos del dispositivo
			2	Importar contacto del dispositivo
			3	Eliminar contacto del dispositivo
			4	Listar contactos privados
			5	Adicionar contacto privado
			6	Eliminar contacto privado
			7	Editar contacto privado
			8	Llamar a contacto privado
12	Gestionar mensajes	Permite gestionar los mensajes existentes en el sistema	1	Listar mensajes del dispositivo
			2	Importar mensaje del dispositivo
			3	Eliminar mensaje del dispositivo
			4	Listar mensajes privados

			5	Eliminar mensaje privado
13	Reiniciar aplicación	Restaura el sistema a su estado inicial borrando sus archivos creados en el dispositivo y eliminando la base de datos	1	Reiniciar aplicación
14	Mostrar disponibilidad de memoria	Muestra el estado de la memoria interna y de la SD Card del dispositivo	1	Mostrar disponibilidad de memoria
15	Gestionar creación de cuentas	Permite habilitar o deshabilitar la opción de crear nuevas cuentas de usuarios	1	Gestionar creación de cuentas
16	Cerrar sesión	Posibilita a los usuarios autenticados salir de su sección	1	Cerrar sección

4.3.1 Desarrollo de las tareas de ingeniería

El desarrollo del *software* se planificó en cuatro iteraciones de trabajo. Para la implementación de las HU correspondientes a la primera iteración se definieron las siguientes TI:

Tabla 12. Tarea de ingeniería 1 de la HU 1

Tarea de ingeniería	
Número tarea: 1	**Número HU:** 1
Nombre tarea: Registrar usuario nuevo	
Tipo tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 25/2/2015	**Fecha fin:** 26/2/2015
Programadores responsables: Alberto Baxter Barrizonte y Osmanis del Risco Ferral	
Descripción: El usuario pulsa la opción “Crear cuenta”, luego se muestra una ventana en la que deberá insertar sus datos, los cuales son nombre de usuario, contraseña, luego deberá confirmar su contraseña, seleccionar el algoritmo a utilizar y posteriormente pulsar la opción “Registrar”.	

Tabla 13. Tarea de ingeniería 1 de la HU 2

Tarea de ingeniería	
Número tarea: 1	**Número HU:** 2
Nombre tarea: Autenticar usuario	
Tipo tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 26/2/2015	**Fecha fin:** 27/2/2015
Programadores responsables: Alberto Baxter Barrizonte y Osmanis del Risco Ferral	
Descripción: Para realizar esta operación se deberá insertar el nombre de usuario y la contraseña, luego pulsar la opción “Iniciar Sesión”.	

En el **Anexo III** se muestran las restantes Tareas de ingeniería.

4.4 Estándares de nomenclatura y codificación utilizados

Los estándares de codificación son un conjunto de directrices, normas y reglamentos enfocados a la especificación de cómo debe escribirse el código fuente de la aplicación. Estos incluyen pautas sobre la nomenclatura de las variables, clases y paquetes; la correcta identación del código, cómo escribir estructuras de control, entre otros aspectos. La correcta elaboración y utilización de los estándares de codificación permiten que todos los desarrolladores implementen siguiendo las mismas pautas y así puedan entender el código del resto como si estuvieran mirando el de ellos, así la aplicación será más legible, uniforme y fácil de mantener (Vermeulen, 2001).

4.4.1 Nomenclatura general

- El nombre de las clases, métodos y variables deben estar escritos en español.
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura *lowerCamelCase*, que dicta que para un nombre compuesto por varias palabras comenzará con minúscula pero todas las palabras internas que lo componen comienzan con mayúscula.

Paquetes

Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales. Los paquetes estarán definidos de la siguiente manera.

- **presentación:** Contiene todas las vistas con las que cuenta el sistema.

- **accesoDatos:** Contiene todas las clases modelos con las que cuenta el sistema.
- **negocio:** Contiene todas las clases que se encargan de realizar el proceso de negocio del sistema.
- **entidades:** Contiene todas las clases entidades con las que cuenta el sistema.
- **hilos:** Contiene todas las clases que se encargan de manejar los procesos de ejecución del sistema.
- **logicaCifrado:** Contiene todas las clases que se encargan de realizar el proceso de cifrar y descifrar la información, estas en conjunto conforman una librería.
- **libreriaSelectorArchivos:** Es una librería utilizada para realizar una exploración y desplazarse en los archivos del dispositivo.

A continuación se muestran las relaciones entre clases en un diagrama de paquetes, ver Figura 6.

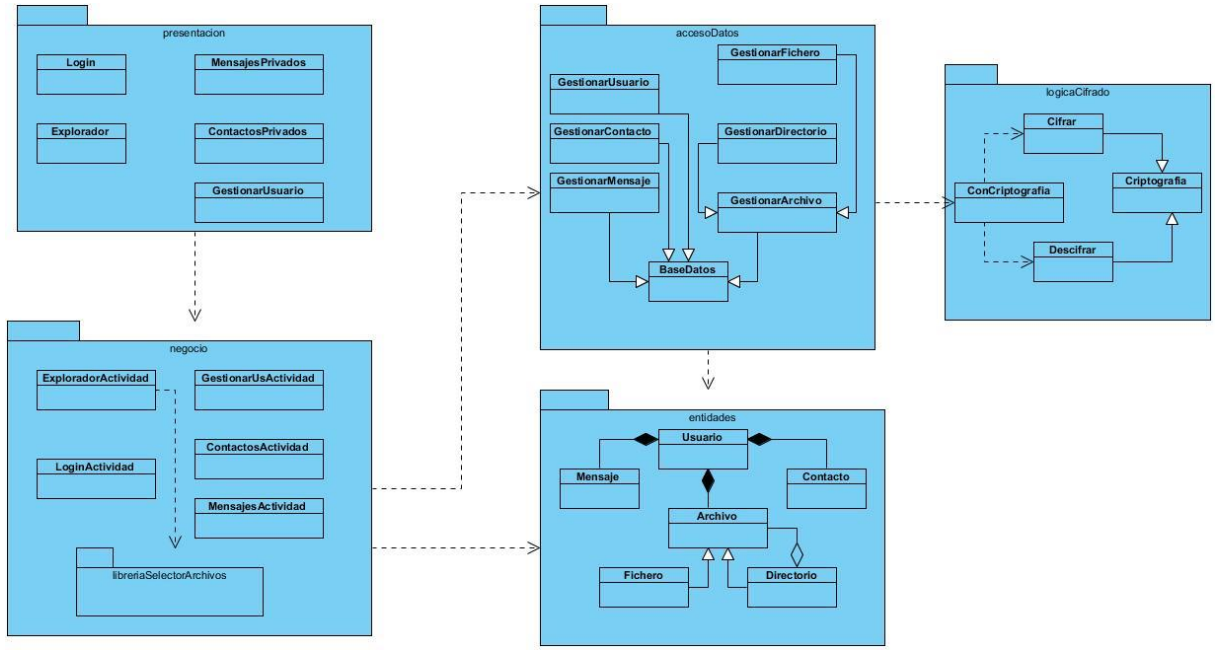


Figura 6. Diagrama de paquetes

Identación

En el contenido siempre se identará con tabulaciones, nunca utilizando espacios en blanco.

Clases

El nombre de las clases comenzará con mayúsculas y cada salto de palabras debe iniciar con mayúsculas. En el caso de las clases XML su nombre se escribirá con letra minúscula y para cada salto de palabra debe utilizarse guión bajo.

Métodos

Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas y con la primera letra de cada palabra interna en mayúsculas (*lowerCamelCase*).

No se permiten caracteres especiales.

Nombre de variables

El nombre de las variables debe empezar con letra minúsculas y de existir un salto de palabra comenzaría con mayúscula (Vermeulen, 2001).

4.4.2 Estilos de codificación

Comentarios

- Como norma es obligatorio proporcionar un comentario de documentación por cada clase y en el caso de los métodos solo se deberán comentar aquellos que posean cierto grado de dificultad.
- Comentario de la clase / método:
  - Prescripción genérica de la clase o método y su responsabilidad.
- Reglas generales a la hora de escribir comentarios de documentación.
  - Siempre se escribe en tercera persona.
  - Las descripciones siempre deberían empezar por un verbo.

Sentencias

- Una sentencia por línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de una condicional.

4.5 Pruebas al sistema

Uno de los pilares de la metodología utilizada es el proceso de pruebas. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar codificaciones y refactorizaciones (Malfará, 2006).

XP divide las pruebas del sistema en dos grupos:

- **Pruebas unitarias:** Encargadas de verificar el código y diseñada por los programadores.
- **Pruebas de aceptación:** Destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente final.

4.5.1 Pruebas unitarias

Uno de los métodos utilizados para realizar pruebas de *software* en la metodología XP son las pruebas unitarias. La base de este método es el hacer pruebas en pequeños fragmentos del código de la aplicación. Estos fragmentos deben ser unidades estructurales del sistema encargados de una tarea específica, en programación orientada a objetos se puede afirmar que estas unidades son los métodos o las funciones que se tienen definidos. El objetivo de estas pruebas es el aislamiento de partes del código y la demostración de que no contienen errores. Estas no generan artefactos y no son directamente palpables para el cliente (Malfará, 2006).

4.5.2 Resultados de las pruebas unitarias

Se utilizó la herramienta JUnit para la realización de las pruebas unitarias al sistema. Esta herramienta permite probar componentes específicos mediante clases de casos de pruebas. A continuación se muestran los resultados de las pruebas unitarias realizadas al paquete "**logicaCifrado**", ver Figura 7.

Figura 7. Resultados de las pruebas unitarias

4.5.3 Pruebas de aceptación

Su objetivo principal es verificar las características funcionales del sistema, las cuales son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada (Malfará, 2006).

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Esto significa que debe desarrollarse un nuevo *test* de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. La garantía de calidad es una parte esencial en el proceso de XP (Malfará, 2006).

Los casos de prueba (CP) elaborados comprenden las siguientes secciones:

- **Código:** Identificador del CP, es sugerente al nombre de la prueba a la que hace referencia.
- **Historia de Usuario:** HU a la que hace referencia.
- **Nombre:** Nombre que se le da a la prueba a realizar.
- **Descripción:** Se describe la funcionalidad que se desea probar.
- **Condiciones de Ejecución:** Condiciones que deben cumplirse para poder llevar a cabo el caso de prueba.
- **Entradas / Pasos de Ejecución:** Descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultado esperado:** Breve descripción del resultado que se espera obtener con la prueba.
- **Evaluación de la prueba:** Acorde al resultado de la prueba realizada se emitirá una evaluación.
- **Resultado obtenido:** Breve descripción del resultado obtenido con la prueba.

A continuación se presentan los CP elaborados.

Tabla 14. Prueba de aceptación Registrar Usuario

Prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** Registrar usuario
Nombre: Registrar usuario	
Descripción: Probar la funcionalidad **Registrar usuario**	

Condiciones de ejecución: El administrador del sistema debe activar la opción “Permitir Registrar Usuarios” en caso de que esta esté desactivada
Entradas / pasos de ejecución: Pulsar la opción “Crear Cuenta”, luego en el formulario que se muestra introducir datos en los campos: nombre de usuario, contraseña, confirmar contraseña y seleccionar el algoritmo de cifrado, posteriormente pulsar la opción “Registrar”
Resultado esperado: El sistema debe mostrar un mensaje con este texto: “El usuario ha sido registrado con éxito”. Esto indica que el usuario ha sido registrado satisfactoriamente
Evaluación de la prueba: Satisfactoria
Resultado obtenido: El sistema mostró un mensaje con este texto: “El usuario ha sido registrado con éxito”. Esto indica que el usuario ha sido registrado satisfactoriamente

Tabla 15. Prueba de aceptación Autenticar usuario

Prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** Autenticar usuario.
Nombre: Autenticar usuario	
Descripción: Probar la funcionalidad **Autenticar usuario**	
Condiciones de ejecución: El usuario que se desea autenticar debe tener una cuenta creada en el sistema	
Entradas / pasos de ejecución: Introducir los datos en los campos nombre de usuario y contraseña, luego pulsar la opción “Iniciar Sesión”	
Resultado esperado: Se muestra una vista con la sesión perteneciente al usuario autenticado, en esta sesión el usuario puede acceder a toda su información privada y realizar varias operaciones	
Evaluación de la prueba: Satisfactoria.	
Resultado obtenido: Se mostró una vista con la sesión perteneciente al usuario	

autenticado, en esta sesión el usuario puede acceder a toda su información privada y realizar varias operaciones

En el **Anexo V** se muestran las restantes Pruebas de aceptación.

4.5.4 Resultados de las pruebas de aceptación

Las pruebas aplicadas contribuyeron a mejorar la calidad y el funcionamiento del sistema, detectándose en la primera iteración un total de seis no conformidades significativas, las cuales fueron resueltas y cuatro recomendaciones. En la segunda iteración se detectaron tres no conformidades las cuales fueron resueltas y dos recomendaciones. En la tercera iteración no se detectaron no conformidades, sin embargo se encontró una recomendación, ver Figura 8.

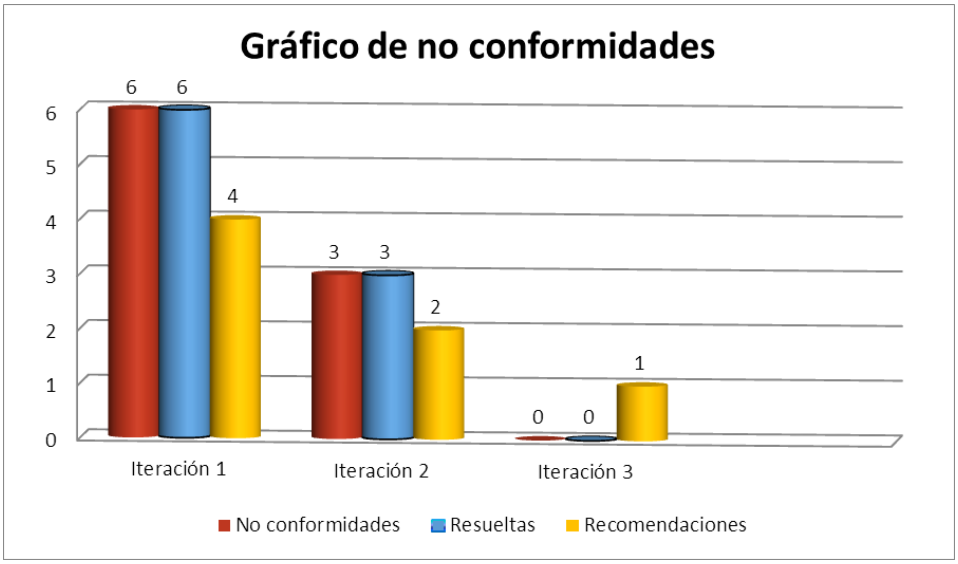


Figura 8. Resultado de las pruebas de aceptación

4.6 Conclusiones del capítulo

En este capítulo con la realización de la fase de implementación y prueba planteada por la metodología XP, se logró comprobar que el sistema creado cumpliera con los requerimientos planteados. La realización de las tareas de ingeniería proporcionó solución a todas las HU logrando una mayor organización y rapidez en el desarrollo del sistema. El uso de un estándar de codificación permitió una mejor legibilidad de la aplicación y que esta fuera de fácil entendimiento para posteriores desarrolladores de este sistema en versiones futuras. La culminación del desarrollo de las pruebas unitarias y de aceptación con resultados satisfactorios garantizó una buena calidad del sistema.

Conclusiones

El aporte principal de este trabajo consiste en el diseño e implementación de una aplicación que sea capaz de brindar confidencialidad a los datos personales de múltiples usuarios.

Después de concluida la investigación y ser analizados los resultados obtenidos, las conclusiones a las que se arribaron son las siguientes:

- Las herramientas estudiadas que presentaban un comportamiento similar al deseado por el cliente, no eran consideradas adecuadas para dar solución al problema de la investigación, por lo que resultó necesario la creación de una nueva herramienta.
- La selección de la metodología de desarrollo ágil XP permitió un fácil entendimiento entre los distintos participantes en la producción de *software* a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad.
- Las herramientas, los lenguajes de programación y la tecnología utilizada facilitaron el trabajo realizado.
- La utilización de los algoritmos AES e IDEA para la lógica de cifrado es una solución factible que provee una mayor seguridad a los datos personales ante la piratería informática.
- El uso de patrones arquitectónicos y de diseño propició tener estructuras de código más completas y mejor elaboradas, permitiendo mejorar en gran medida los resultados esperados por el cliente.
- Las pruebas realizadas comprobaron el grado de cumplimiento respecto a las especificaciones iniciales del sistema, garantizando la calidad del *software*.

Recomendaciones

Una vez concluida la investigación se proponen las siguientes recomendaciones:

- Para nuevas versiones del sistema incluir una funcionalidad que permita cifrar y descifrar todo el contenido dentro de un directorio.
- Internacionalizar el *software* para que su uso sea más frecuente y por un mayor número de usuarios.

Bibliografía referenciada

Android. 2015. sevilla.abc.es. *sevilla.abc.es.* [En línea] 2015. [Citado el: 15 de Marzo de 2015.] http://sevilla.abc.es/mobility/las_mejores_app/android/las-mejores-app-android/protege-tu-android-con-estas-aplicaciones-para-encriptar-tu-smartphone/.

Confidencialidad. 2015. Confidencialidad de la información. *Confidencialidad de la información.* [En línea] 2015. [Citado el: 9 de 5 de 2015.] http://www.coreoneit.com/confidencialidad-de-la-informacion/.

Cormen. 2011. *Introduction to Algorithms.* 2011.

Eclipse. 2015. eclipse.org. [En línea] 2015. [Citado el: 10 de Marzo de 2015.] https://eclipse.org/.

Fabri. 2013. ¿Qué es el cifrado? *¿Qué es el cifrado?* [En línea] 23 de Julio de 2013. [Citado el: 2 de Marzo de 2015.] http://articulos.softonic.com/que-es-el-cifrado-encriptar.

Gamma, Erich. 2003. *Patrones de Diseño.Elementos de software orientado a objetos reutilizables.* España : s.n., 2003.

Hüseyin, Demirc. 2003. *A New Meet in the Middle Attack on The IDEA Block Cipher.* s.l. : 10th Annual Workshop on Selected Areas in Cryptography, 2003.

Información, Seguridad en los Sistemas de. 2015. Seguridad en los Sistemas de Información. *Seguridad en los Sistemas de Información.* 2015.

Inteco. 2012. *Estudio sobre seguridad en dispositivos móviles y smartphones.* Barcelona : s.n., 2012.

Interface, User. 2015. User Interface Android. *User Interface Android.* [En línea] 7 de 1 de 2015. [Citado el: 12 de 4 de 2015.] https://developer.android.com/guide/topics/ui/index.html.

Java. 2015. Java. *Java.* [En línea] 2015. [Citado el: 8 de Marzo de 2015.] http://www.java.com.

JUnit. 2015. JUnit About. *JUnit About.* [En línea] 2015. [Citado el: 10 de 5 de 2015.] http://www.junit.org/.

Letelier, Patricio . 2006. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia : s.n., 2006.

Malfará, Dayvis . 2006. *Testing en XP.* 2006.

Móviles, SO. 2015. Ranking de los SO móviles más usados. *Ranking de los SO móviles más usados.* [En línea] 25 de 2 de 2015. [Citado el: 8 de 4 de 2015.] https://www.android.com/.

Paar, Christof . 2009. *The Data Encryption Standard (DES) and Alternatives.* s.l. : Springer, 2009.

Paradigm. 2015. Visual Paradigm Empresa. *Visual Paradigm Empresa.* [En línea] 2015. [Citado el: 4 de 3 de 2015.] http://www.visual-paradigm.com/aboutus/.

Pastor Franco, José. 1998. *Criptografía digital: fundamentos y aplicaciones.* Universitarias de Zaragoza : s.n., 1998.

Patrones. 2009. *Patrones del "Gang of Four".* Madrid : s.n., 2009.

Pooley. 2002. *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* 2002.

Ramos, Miguel Angel. 2010. *Guía de Arquitectura N-Capas.* España : s.n., 2010.

Reynoso, Carlos. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Rijmen, Vincent. 2002. *The Advanced Encryption Standard.* 2002.

Salama, Diaa. 2010. *Evaluating The Performance of Symmetric.* Egipto : s.n., 2010.

SDK, Android. 2015. androidcentral. *androidcentral.* [En línea] 2015. [Citado el: 10 de Marzo de 2015.] http://www.androidcentral.com/tag/android-sdk.

SQLite. 2015. SQLite. *SQLite.* [En línea] 2015. [Citado el: 13 de 2 de 2015.] http://es.scribd.com/doc/52882068/SQLite.

Tardáguila, César. 2006. *Dispositivos Móviles y Multimedia.* Primera. 2006. pág. 29.

Tarjetas. 2015. Actividades de Implementacion Tarjetas CRC. [En línea] 2015. [Citado el: 20 de Marzo de 2015.] http://www.inf.utfsm.cl/~visconti/xp/Tarjetas_CRC_2.doc..

Tools, Android Developer. 2015. Android Developer Tools. *Android Developer Tools.* [En línea] 2015. [Citado el: 23 de 2 de 2015.] https://developer.android.com/tools/help/adt.html.

Vanstone, Menezes. 1997. *Handbook of Applied Cryptography.* 1997.

Vermeulen, Alan. 2001. *The Elements of Java(TM) Style.* New York : Cambridge University Press : s.n., 2001.

XML. 2015. xml.com. [En línea] 2015. [Citado el: 13 de marzo de 2015.] http://www.xml.com.

Bibliografía

Android. 2015. sevilla.abc.es. *sevilla.abc.es.* [En línea] 2015. [Citado el: 15 de Marzo de 2015.] http://sevilla.abc.es/mobility/las_mejores_app/android/las-mejores-app-android/protege-tu-android-con-estas-aplicaciones-para-encriptar-tu-smartphone/.

Arenas, Stella. 2014. *El Resumen.* 2014.

Arquitectura. 2011. Arquitectura de Android. *Arquitectura de Android.* [En línea] 4 de 7 de 2011. [Citado el: 23 de 11 de 2014.] http://androideity.com/2011/07/04/arquitectura-de-android/.

Conclusiones. *Concluisones elaboradas de una tesis.*

Confidencialidad. 2015. Confidencialidad de la información. *Confidencialidad de la información.* [En línea] 2015. [Citado el: 9 de 5 de 2015.] http://www.coreoneit.com/confidencialidad-de-la-informacion/.

Cormen. 2011. *Introduction to Algorithms.* 2011.

Eclipse. 2015. eclipse.org. [En línea] 2015. [Citado el: 10 de Marzo de 2015.] https://eclipse.org/.

Fabri. 2013. ¿Qué es el cifrado? *¿Qué es el cifrado?* [En línea] 23 de Julio de 2013. [Citado el: 2 de Marzo de 2015.] http://articulos.softonic.com/que-es-el-cifrado-encriptar.

Gamma, Erich. 2003. *Patrones de Diseño.Elementos de software orientado a objetos reutilizables.* España : s.n., 2003.

Hüseyin, Demirc. 2003. *A New Meet in the Middle Attack on The IDEA Block Cipher.* s.l. : 10th Annual Workshop on Selected Areas in Cryptography, 2003.

Información, Seguridad en los Sistemas de. 2015. Seguridad en los Sistemas de Información. *Seguridad en los Sistemas de Información.* 2015.

Inteco. 2012. *Estudio sobre seguridad en dispositivos móviles y smartphones.* Barcelona : s.n., 2012.

Interface, User. 2015. User Interface Android. *User Interface Android.* [En línea] 7 de 1 de 2015. [Citado el: 12 de 4 de 2015.] https://developer.android.com/guide/topics/ui/index.html.

Java. 2015. Java. *Java.* [En línea] 2015. [Citado el: 8 de Marzo de 2015.] http://www.java.com.

JUnit. 2015. JUnit About. *JUnit About.* [En línea] 2015. [Citado el: 10 de 5 de 2015.] http://www.junit.org/.

Letelier, Patricio . 2006. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia : s.n., 2006.

Malfará, Dayvis . 2006. *Testing en XP.* 2006.

Móviles, SO. 2015. Ranking de los SO móviles más usados. *Ranking de los SO móviles más usados.* [En línea] 25 de 2 de 2015. [Citado el: 8 de 4 de 2015.] https://www.android.com/.

Paar, Christof . 2009. *The Data Encryption Standard (DES) and Alternatives.* s.l. : Springer, 2009.

Paradigm. 2015. Visual Paradigm Empresa. *Visual Paradigm Empresa.* [En línea] 2015. [Citado el: 4 de 3 de 2015.] http://www.visual-paradigm.com/aboutus/.

Pastor Franco, José. 1998. *Criptografía digital: fundamentos y aplicaciones.* Universitarias de Zaragoza : s.n., 1998.

Patrones. 2009. *Patrones del "Gang of Four".* Madrid : s.n., 2009.

Pooley. 2002. *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* 2002.

Ramos, Miguel Angel. 2010. *Guía de Arquitectura N-Capas.* España : s.n., 2010.

Reynoso, Carlos. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Rijmen, Vincent. 2002. *The Advanced Encryption Standard.* 2002.

Sabino, Carlos. 2012. *Como hacer una tesis.* Caracas : Panapo, 2012.

Salama, Diaa. 2010. *Evaluating The Performance of Symmetric.* Egipto : s.n., 2010.

SDK, Android. 2015. androidcentral. *androidcentral.* [En línea] 2015. [Citado el: 10 de Marzo de 2015.] http://www.androidcentral.com/tag/android-sdk.

Segjava. Seguridad en Java. *Seguridad en Java.* [En línea] [Citado el: 1 de 4 de 2015.] http://www.uv.es/~sto/cursos/seguridad.java/html/sjava-12.html.

Sentencias. Sentencias SQLite. *Sentencias SQLite.* [En línea] [Citado el: 23 de 1 de 2015.] http://www.aprendeandroid.com/l5/sql3.htm.

SQLite. 2015. SQLite. *SQLite.* [En línea] 2015. [Citado el: 13 de 2 de 2015.] http://es.scribd.com/doc/52882068/SQLite.

Tardáguila, César. 2006. *Dispositivos Móviles y Multimedia.* Primera. 2006. pág. 29.

Tarjetas. 2015. Actividades de Implementacion Tarjetas CRC. [En línea] 2015. [Citado el: 20 de Marzo de 2015.] http://www.inf.utfsm.cl/~visconti/xp/Tarjetas_CRC_2.doc..

Tools, Android Developer. 2015. Android Developer Tools. *Android Developer Tools.* [En línea] 2015. [Citado el: 23 de 2 de 2015.] https://developer.android.com/tools/help/adt.html.

Vanstone, Menezes. 1997. *Handbook of Applied Cryptography.* 1997.

Vermeulen, Alan. 2001. *The Elements of Java(TM) Style.* New York : Cambridge University Press : s.n., 2001.

XML. 2015. xml.com. [En línea] 2015. [Citado el: 13 de marzo de 2015.] http://www.xml.com.

Glosario de términos

Internet: Conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, lo cual garantiza que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.

Sistema de Posicionamiento Global o **GPS:** Objeto que permite a una persona determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión.

Sistema Operativo: Programa o conjunto de programas de un sistema informático que gestiona los recursos de *hardware* y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes (aunque puede que parte de él se ejecute en espacio de usuario).

Malware: *Software* malicioso o *software* malintencionado, es un tipo de *software* que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario.

Bluetooth: Es una especificación que define redes de área personal inalámbricas (*wireless personal area network*, WPAN).

Número de Identificación Personal o **PIN:** Es una clave de cuatro cifras que se utiliza en combinación con su número de Seguro Social, nombre y apellido y fecha de nacimiento, para determinar si alguien puede acceder a datos personales.

Interfaz de Programación de Aplicaciones o **API:** Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción. Son usados generalmente en las bibliotecas.

Hipertext Markup Language o **HTML:** Es el lenguaje con el que se escriben las páginas web. Es un lenguaje de hipertexto, es decir, un lenguaje que permite escribir texto de forma estructurada y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento.

Cumputer Aided Software Engineering o **CASE:** Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software*. Este puede ser generalmente aplicado a cualquier

sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de *software*.

Framework: En el desarrollo de *software*, un *framework* o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Entorno de Desarrollo Integrado o **IDE:** Un Entorno de Desarrollo Integrado (IDE) es un conjunto de herramientas utilizadas por los programadores, que incluye por lo general, un buen editor de código, administrador de proyectos y archivos, enlace a compiladores e integración con sistemas controladores de versiones o repositorios, además de brindar facilidades para la construcción de interfaces gráficas de usuario.

Lenguajes de programación: Se consideran lenguajes de programación a los lenguajes que pueden ser utilizados para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Dispositivos PDA: PDA significa ("*Personal Digital Assistant*") o asistente personal digital, por lo general muchas personas lo llaman Palm®, debido a que es una marca popular de PDA, pero no la única. Se trata de pequeñas computadoras, con funciones similares a las de una convencional, más las de funciones de una agenda electrónica. Estos dispositivos son totalmente portátiles ya que son del tamaño de la mano y muy delgados, utilizan un sistema operativo y tienen aplicaciones específicas a nivel usuario como aplicaciones ofimáticas (procesadores de palabras, hojas electrónicas), juegos, interfaz para conexión a redes, etc.

ACID: En base de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Así pues, si un sistema de gestión de base de datos es ACID *compliant* quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID. En concreto ACID es un acrónimo de *Atomicity, Consistency, Isolation and Durability*: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.

Secure Sockets Layer o **SSL:** Es un protocolo diseñado para permitir que las aplicaciones para transmitir información de ida y de manera segura hacia atrás. Las aplicaciones que utilizan el protocolo *Secure Sockets Layer* sí saben cómo dar y recibir claves de cifrado con otras aplicaciones, así como la manera de cifrar y descifrar los datos enviados entre los dos.

Wired Equivalent Privacy o **WEB:** En su traducción al español significa "Privacidad Equivalente a Cableado", es el sistema de cifrado incluido en el estándar IEEE 802.11 como protocolo para redes *Wireless* que permite cifrar la información que se transmite.

Graphical User Interface o **GUI:** La interfaz gráfica de usuario, conocida también como GUI es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Android: Es un sistema operativo inicialmente pensado para teléfonos móviles que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma**.** El sistema permite programar aplicaciones en una variación de Java llamada *Dalvik*.

Pretty Good Privacy o **PGP:** Es un programa desarrollado por Phil Zimmermann y cuya finalidad es proteger la información distribuida a través de *Internet* mediante el uso de criptografía de clave pública, así como facilitar la autenticación de documentos gracias a firmas digitales.

Java Runtime Environment o **JRE**: Es un conjunto de utilidades que permite la ejecución de programas Java. El JRE actúa como intermediario entre el sistema operativo del ordenador y Java.

Layouts: En *Android* los *layout* permiten posicionar cada objeto gráfico en el lugar que queramos de la pantalla, es decir, permite diseñar el aspecto gráfico que va a tener nuestra pantalla.

Interfaz de Usuario o UI: Es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, la UI comprende todos los puntos de contacto entre el usuario y el equipo.