UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

Gestión de documentos en entornos colaborativos mediante un cliente Android

(SITI_4/1314)

Ignacio del Pozo Martínez

Febrero 2014

Gestión de documentos en entornos colaborativos mediante un cliente Android

(SITI_4/1314)

AUTOR: Ignacio del Pozo Martínez

TUTOR: David Arroyo Guardeño

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Febrero 2014

Agradecimientos

En primer lugar me gustaría dar las gracias a todas las personas que han hecho posible que llegara hasta aquí con este proyecto

A mis profesores, por enseñarme a trabajar duro durante estos cuatro años y enseñarme que siempre hay algo que aprender

A mi tutor, por plantearme el trabajo y guiarme a través del mismo y de su realización, ayudándome a mirar en el sitio correcto y echándome una mano durante todo el proceso, sin él no habría sido posible

A todos mis amigos, los de verdad, los que siempre han estado ahí cuando parecía que prácticas, trabajos y exámenes eran lo único que había en mi vida, demostrándome que podía con todo eso y mucho más y que la ingeniería no era nada más que otra palabra

A Joserra, gracias por la ayuda con el diseño gráfico y el apoyo moral durante estos últimos meses de estrés

A Carmelo, gracias por revisar mi matemática criptográfica y estar ahí siempre para cualquier duda que tuviera

A Mara, por estar siempre y en todo momento, intentando sacarme una sonrisa y alegrarme, dándome confianza siempre que lo he necesitado

Y a mis padres, la mención más importante para mí. Gracias a los dos por estar siempre ahí, gracias por todo lo que me habéis dado y lo que siempre me dais, por vuestro apoyo incondicional, porque tengo claro que sin vosotros, nada de esto habría sido posible, no sería quién soy ahora y no habría sido capaz de llegar hasta aquí, al menos, seguro que no tan feliz de ser quien soy, ni tan agradecido por tener unos padres tan maravillosos como vosotros

A todos vosotros GRACIAS

Resumen

Este proyecto surge de la necesidad de crear un sistema de gestión documental adaptado a pequeñas y medianas empresas (PYMES), que les permita disponer de una alta competitividad y movilidad, sin invertir grandes cantidades de capital. Para ello haremos uso de la plataforma Alfresco, un servidor de gestión documental empresarial, como entorno de explotación, mientras utilizamos las capacidades en *la nube* de la plataforma Dropbox como entorno de producción, aprovechando su alta disponibilidad.

Teniendo en cuenta el panorama actual de la seguridad y privacidad en Internet, creemos que es imprescindible dotar a la aplicación desarrollada del conjunto de funcionalidades criptográficas necesarias para proteger los activos de información almacenados en servidores externos, que no están bajo el control de los potenciales clientes. Para protegerlos estipularemos un tratamiento de los mismos basado en la tríada confidencialidad-integridad-disponibilidad que establece la ISO-27001 [1], brindando así tranquilidad a las empresas para que puedan utilizar este tipo de servicios sin el miedo que produce que sus activos se encuentren siendo manejados por terceros.

Presentaremos también un cliente para el cual hemos elegido la plataforma Android debido a su elevada cuota de mercado (*alrededor del 90% en España* [2]) y la capacidad de poder ofrecer una gran movilidad a la hora de tener todos los archivos y documentos disponibles en cualquier momento y lugar.

Por último, aunque no menos importante, hemos decidido utilizar en TODO momento tecnologías libres *OpenSource*, para el desarrollo y la producción de este proyecto. Desde el entorno de desarrollo hasta el sistema operativo, la filosofía *GNU* ha guiado nuestro diseño, ya que creemos que es muy importante evitar todos aquellos procedimientos de tratamiento, codificación y protección de información basados en la ocultación de los principios de diseño y/o implementación. En resumidas cuentas la filosofía *GNU* permite evitar lo que en el argot criptográfico se denomina *Security Through Obscurity* [3].

Además pensamos que el software libre es un avance en el uso del software, especialmente en entornos educativos como es el ámbito de este proyecto, ya que permite a la comunidad crecer al poder disponer enteramente del código para aprender de él, además de ofrecer un exhaustivo nivel de evaluación, al aceptar las contribuciones de la misma, lo que nos permitirá asegurar niveles de protección que no podríamos alcanzar por nosotros mismos [4].

Palabras Clave

Gestión documental, Gestión de Repositorios, OpenSource, Java, Seguridad, Criptografía, AES 256, Multilenguaje, Android, Dropbox, Maven, Alfresco.

Summary

This project addresses the necessity of creating a document management system adapted to Small and Medium Enterprises (SMEs), being the main goal to provide a usable and low cost system. In order to do so, we use the Alfresco platform, a corporate document management server, as the exploitation environment, while we adopt the *cloud* capabilities of the Dropbox platform as production environment, leveraging the high availability that offers.

Due to the current panorama of security and privacy in the Internet, it is indispensable to endow the developed application with all the cryptographic functions needed to protect the information assets stored in external servers, since these servers are not under control of potential clients. In this regard, we considered as bottom line the Confidentiality-Integrity-Availability principles as defined in the ISO27001 standard [1]. By doing this we are providing a set of procedures to foster small business trust in this kind of technologies implying the management of information assets by a third party and the possibility of non-authorized interception. In addition, we have designed and implemented and Android client to access information stored in the Alfresco server and to encrypt and upload information to the Dropbox server. We have selected the Android platform to implement our client due to its high market share (*about 90% in Spain* [2]), which makes possible to offer great mobility opportunities, i.e., it enables files and information accessibility any time, any place .

Last, but not least, we decided to use Libre software, Open Source software, for EVERYTHING, i.e., for every single step in the production and development process of this project. From the development environment to the operating system, the *GNU* philosophy has guided our design, because we believe that it is important to avoid all the procedures for information treatment, coding and protection by means of concealing the design and/or implementation assumptions . Summing up, the GNU philosophy allow us to avoid what in cryptographic argot is coined as the *Security Trough Obscurity* principle [3].

We think also that Open Source software is a thrust for the software development community, especially in education environments such as the ambit of this project. Certainly, it paves the way for the community to grow up having complete access to the code and, eventually it improves the implemented software through programmers and users collaborative networks, achieving protection levels that we could never achieve by ourselves [4].

Key Words

Document Management, Repository Management, OpenSource, Java, Security, Cryptography, AES 256, Multilanguage, Android, Dropbox, Maven, Alfresco.

Índice de contenidos

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

A día de hoy las PYMES son un factor determinante en el sustrato económico de nuestra sociedad. Juegan un papel fundamental en el crecimiento y desarrollo de nuestro país, debido a que son muchas veces determinantes a la hora de encadenar nuestra actividad económica, ya sea produciendo y ofertando bienes y servicios, demandando y comprando productos, así como a través de la generación de empleo. Sin embargo, también son las que más problemas tienen a la hora de crecer, debido generalmente al tamaño y la limitación de recursos que las caracterizan.

Es por ello que en este clima de crisis económica, en el que las empresas deben alcanzar su máximo rendimiento para competir en el mercado, nos parecía adecuado desarrollar un proyecto de gran utilidad y atractivo para este tipo de negocios. Con este fin abordaremos un tema importante a la hora de gestionar una empresa, el tratamiento de toda la información importante que la misma genere, es decir, sus activos de información. Presentaremos nuestra propuesta basada en un cliente móvil que permite acceder a las distintas versiones de documentos en fase de explotación, mientras nos permitiría a su vez publicarlos de forma segura en una plataforma *cloud,* como entorno de producción, ofreciéndonos una gran movilidad y disponibilidad.

Debido al perfil de nuestros clientes potenciales (PYMES), hemos optado por soluciones vigentes en el mercado, ampliamente utilizadas y conocidas, con el fin de abaratar costes y lograr una alta aceptación del producto. Además hemos buscado añadir más atractivo al proyecto tomando como referencia la norma ISO-27001 [1] para los Sistemas Generales de Seguridad de la Información (SGSI), incluyendo como objetivo primordial la tríada Confidencialidad-Integridad-Disponibilidad (CID).

1.1 Gestión documental

La gestión documental es un proceso realmente importante y complejo en cualquier empresa. Ya sea digital o analógica, en servidores o archivadores, en nuestra empresa o en la de un tercero (*outsourcing*), debe ser cuidadosamente tratada, debido a que puede llegar a determinar el éxito o el fracaso de proyectos o incluso de la empresa misma. Por ello en muchos casos las empresas optan por contratar servicios de *outsourcing* (ver Glosario) a terceras empresas, exponiendo información sensible si no se siguen ciertas indicaciones de seguridad [5].

Según la definición de gestión documental (ver Glosario), observaremos que no especifica que en la mayoría de los casos necesitaremos tener un control de versiones de esos documentos, debido a la rapidez con la que las empresas pueden cambiar o evolucionar, modificando sus decisiones e incluso sus políticas para adaptarse a nuevos mercados y ser más competitivas. Por ello, debemos añadir que para administrar el flujo de información, la mayoría de las veces no tendremos que eliminar los documentos antiguos, sino guardar una copia de los mismos para su consulta, mientras conservamos las versiones actualizadas de éstos. Esto hace que la conservación de los documentos en papel, además de ser tremendamente cara, por el espacio que ocupan y su difícil mantenimiento, sea una opción muy ineficiente y poco viable.

Debido a eso, la mayoría de las empresas eligen guardar sus documentos electrónicamente. Una vez hemos tomado la decisión de tirar los archivadores y comprar servidores, llega el momento de desplegar la arquitectura, el sistema, un complejo problema para las PYMES que no tienen en plantilla a personas con conocimientos técnicos o simplemente no pueden permitírselo debido a la situación actual del mercado. Diseñar un proceso entero de gestión documental electrónico e implementarlo (instalando servidores, enseñando al personal, etc.), puede ser un auténtico problema. En efecto, a las dificultades inherentes a la instalación y despliegue de la infraestructura de gestión documental, habría que añadir su mantenimiento, elemento imprescindible para garantizar la protección de los activos de información de la PYME en cuestión.

La solución parece trivial, si no podemos gestionar estos sistemas por nosotros mismos con una inversión baja, se debe buscar a alguien que gestione este sistema por nosotros a través del *outsourcing*, o utilizar servicios de este tipo en internet nosotros mismos. Es ahí donde este proyecto gana todo su interés, tratando de aportar una solución atractiva para las PYMES, brindándoles la oportunidad de implementar su propio servicio de gestión documental competitivo y usable, sin invertir grandes cantidades de capital.

Para ello, nos basaremos en un principio fundamental de ingeniería que consiste en utilizar (siempre que sea factible) herramientas ya diseñadas, implementadas, verificadas y aceptadas por la comunidad. Además elegiremos en la medida de lo posible herramientas *OpenSource* basadas en estándares públicos, obteniendo dos grandes ventajas. Por un lado, ahorraremos tiempo abaratando los costes de desarrollo del proyecto, lo que puede ser determinante en el éxito o fracaso del mismo, pero aún más importante si cabe, estaremos incrementando enormemente la calidad del producto final, ya que en la mayoría de los casos estas herramientas han sido sometidas a un proceso

de prueba y mejora por la comunidad (en algunos casos dedicada por completo a ello [6]), ofreciéndonos un nivel de fiabilidad que no podríamos lograr por nosotros mismos. Por último, es importante considerar plataformas que sean conocidas en el mercado y tengan un gran número de usuarios, ya que a la hora de presentar nuestra *APP* la aceptación de los usuarios puede ser un factor clave en el éxito de la misma.

Dicho esto, para nuestro proyecto hemos propuesto el uso de un clásico esquema **cliente / servidor**, en el que haremos uso de los sistemas Dropbox y Alfresco, sistemas altamente utilizados y que nos permitirían aprovechar las ventajas de accesibilidad de la plataforma *cloud* Dropbox al mismo tiempo que mantenemos el control total sobre nuestros activos de información en virtud a la plataforma Alfresco. La combinación de estas dos plataformas conformaría nuestro **servidor,** mientras que utilizaremos el entorno Android, debido a su alta usabilidad y movilidad, como **cliente**.

1.2 Entornos *Cloud*

Un sistema de computación *cloud* o nube, es aquel sistema o arquitectura que permite ofrecer servicios digitales a través de internet, ya sea almacenamiento de datos, capacidad de procesamiento o similares.

Esto se consigue haciendo que el proceso de conexión/intercambio de información y la infraestructura que lo gestiona, sea completamente transparente al usuario, a través de internet, de tal forma que con replicación de datos, y el uso de una gran cantidad de servidores, se permite al cliente utilizar este sistema, sin preocuparse del mantenimiento del mismo, con una altísima disponibilidad y con acceso a través de internet desde cualquier punto del planeta.

Ahora bien, ¿qué diferencia a estos sistemas de otras arquitecturas basadas en *hosting* tradicional o *virtual hosting*? Existen unas condiciones que estos sistemas deben cumplir para ser llamados *Cloud*.

Configuración

El *cloud* utiliza varios recursos alojados en distintos servidores físicos, a diferencia del *hosting* tradicional, lo que permite una gran capacidad de replicación para que en caso de que un servidor falle, poder configurar otro casi al instante sin que el usuario lo note si quiera, de ahí su altísima disponibilidad.

Localización

Normalmente los servicios *cloud* se localizan en servidores de terceros, compañías externas a la nuestra que nos alquilan su infraestructura para nuestro uso, generalmente sin tener acceso físico a los terminales.

Escalabilidad

El *cloud* debe ser fácilmente ampliable y configurable a la hora de ofrecer nuevos servicios a los usuarios, por tanto tareas como crear una base de datos, o incluso un nuevo servidor virtual tienen que ser sencillas. Por ello decimos que tiene una gran escalabilidad, es decir, hacer que crezca o implementar nuevos servicios debería ser relativamente rápido.

Coste

Como comentamos anteriormente el coste de mantener tu propio servidor, mantenimiento, luz, acondicionamiento, reparaciones, etc. varía mucho respecto a tener un servicio *cloud* alquilado a un tercero, que puede suponer un coste al mes bastante inferior dependiendo del tipo de servicio que requiramos.

1.2.1 Tendencia actual

Actualmente los sistemas que llamamos *cloud* ofrecen, como comentamos, una gran cantidad de servicios. La tendencia actual está haciendo que utilicemos este tipo de arquitectura para TODOS los servicios digitales que se nos ocurran debido a su gran comodidad para el usuario no avanzado, que no tiene que configurar casi nada por su cuenta sino utilizar un servicio dedicado, con una alta disponibilidad. Ahora nombraremos los usos de los servidores *cloud* más comunes en el mercado.

Disco duro virtual

La mayoría de los servicios de almacenamiento online tienen esta arquitectura detrás, desde Dropbox, hasta Alfresco, que en principio será utilizado como servidor local, también tiene su implementación para este tipo de sistemas. También se encuentran en este sector la opción de Micorsoft "SkyDrive", la de Ubuntu "Ubuntu One", la de Apple "iCloud" o la de Google "Google Drive".

Hosting

Otro uso popular es el *Hosting* de páginas o servicios web, debido a que ningún servidor puede competir fácilmente con las arquitecturas *cloud* en relación precio/disponibilidad. Sin mencionar el mantenimiento del mismo y los problemas/costes que puede generar. El más popular en este sector es el *cloud* de Amazon, altamente flexible y personalizable.

Otros

Debido a las últimas tendencias y posibilidades, cada día se presentan nuevos usos de este herramienta. Uno de los ejemplos más llamativos es Gaikai [7], un sistema lanzado en PS4, que a través de una arquitectura *cloud*, nos permitiría jugar a videojuegos de cualquier plataforma, en cualquier plataforma, a través de *streaming*. Esto se debe a que el juego no se estaría ejecutando realmente en el terminal que poseemos, sino en un servidor externo al que accederemos a través de los servicios *cloud*, permitiéndonos ver la pantalla del mismo y controlarlo, sin notar la diferencia.

Casos como este suponen una serie de cambios en los paradigmas de los sistemas distribuidos a los que estamos acostumbrados, permitiéndonos crear un mundo nuevo de posibilidades y de servicios que ofrecer a los usuarios.

1.3 Software libre

Desde que nació, el software libre ha sido una cultura, una forma de vida para muchos desarrolladores que creemos que es una evolución para el software.

Desde sus inicios con Richard Stallman, donde la comunidad que desarrollaba software libre, era considerada por las grandes empresas como "-poco más o menos que un grupo de hippies que regalaban su software" [8], pasando

incluso por grandes ingenieros del siglo XX y XXI como Steve Wozniak, que como cuenta la biografía de Steve Jobs [8], creía que tenía que regalar literalmente sus diseños, tanto de hardware como de software, enseñarlos públicamente para que la comunidad pudiera aprender y progresar, aprovechándose de los conocimientos de los demás.

En la actualidad existen auténticos modelos de negocio basados en el software libre como por ejemplo, algunas de las distribuciones de Linux en las que se cobra por el soporte, no por las licencias ni el acceso al producto como Red Hat Linux, SUSE ó Elive [9] [10] [11]. No obstante, el software libre no ha dejado de ser una controversia, ya que por un lado las grandes empresas tecnológicas (como IBM, Microsoft o Apple) demostraban que ese estilo y esa forma de pensar no era la más rentable, pero por otro lado no paraban de crecer comunidades de programadores que compartían sus conocimientos para crecer juntos y desarrollar productos cada vez mejores, que competían perfectamente en funcionalidad y usabilidad con los de las grandes empresas.

Esto puede deberse también a que en la época en la que los primeros ordenadores personales entran en el mercado (el PC de IBM, el Apple I y II, el Altair y las primeras consolas, Atari, etc.) es en la que estas grandes compañías se afianzan, tomando una posición muy cómoda en el mercado debido a su gran capital y distanciándose de estas pequeñas comunidades.

En ese momento de monopolio del mercado (Al principio sólo IBM, más tarde Apple y luego Windows), nace GNU [12], un grupo de programadores liderados por Richard Stallman, que quieren crear un sistema operativo que sea totalmente libre, y casi lo consiguen, excepto porque necesitan un núcleo o *kernel* (ver Glosario).

Casualmente en esa época, Linus Torvalds [13], gran entusiasta del lenguaje de programación C, está terminando su versión de un sistema operativo basado en Minix [14], sistema operativo orientado al sector educativo de Andrew S. Tanenbaum, cuyas publicaciones [15] se han convertido en una cabecera para el estudio de los sistemas operativos, y por lo tanto el *kernel* del mismo. Esta alianza da como resultado el lanzamiento del sistema operativo GNU/Linux en 1991.

En 1992, el proyecto, que se conoce comúnmente como "Linux" adopta la licencia GPL, la licencia pública de GNU, abanderando el movimiento del software libre. Este hecho es más importante de lo que parece, ya que por aquel entonces fueron las primeras licencias de este tipo entre la comunidad software, que permitían licenciar software por el que no se cobrara, libre, pero a la vez impidieran el plagio y el robo de ideas. La última versión disponible de esta licencia es la GPLv3, disponible desde julio de 2007 para todos aquellos que deseen licenciar su trabajo a fin de evitar plagios, robo de ideas, etc. y conservar los derechos sobre el software desarrollado.

En la actualidad, un gran número de servidores en el mundo están basados en el sistema operativo Linux, incluso existe una pequeña cuota (en comparación con Windows y Mac, los sistemas más utilizados) de usuarios, que utilizan distribuciones Linux con interfaz gráfica de escritorio como Ubuntu, por ejemplo, en sus ordenadores personales, como podemos observar en las siguientes tablas (Tabla 1 y Tabla 2).

Source	Date	Microsoft Windows					
		8	7	Vista	XP	WP&RT	Other
Net Market Share[6][7][8]	Dec-13	▲7.60%	▼34.44%	▼2.62%	▼21.00%	▼0.16%	–0.00%
StatCounter Global Stats[9]	Dec-13	▲7.53%	▼39.72%	▼3.12%	▼15.62%	▲0.51%	▼0.14%
W3Counter[10]	Dec-13	▲7.20%	▼40.93%	▼3.26%	▼14.32%	–	–
AT Internet[11]	Nov-13	▲6.70%	▲44.70%	▼5.50%	▼16.70%	–	–
Wikimedia[12]	Dec-13	▼5.51%	▼28.89%	▼2.44%	▼8.70%	▲0.65%	▼0.09%

Tabla 1: Uso de terminales Microsoft

Source	Date	Apple		Linux kernel based			Other[a]
		OS X	iOS	Linux	Android	Other	
Net Market Share[6][7][8]	Dec-13	▼5.46%	▲14.94%	–1.25%	▲9.74%	▲0.30%	▲2.38%
StatCounter Global Stats[9]	Dec-13	▲5.68%	▲8.72%	▼1.14%	▲11.66%	▲1.33%	▲6.21%
W3Counter[10]	Dec-13	▲8.45%	▲9.41%	▲2.12%	▲6.46%	–	▲7.85%
AT Internet[11]	Nov-13	▼6.70%	–11.10%	–1.00%	▲6.50%	–	–0.30%
Wikimedia[12]	Dec-13	▼6.03%	▲31.49%	▼1.26%	▲9.92%	▲0.24%	▼0.27%

Tabla 2: Uso de terminales Unix y otros

No obstante, debido a la gran cantidad de distribuciones que existen, tanto para ordenadores como para *smartphones*, el término *software libre* se ha ido especificando. De hecho Richard Stallman, creador del movimiento, se encarga en la actualidad de dar conferencias por todo el mundo explicando lo que él considera realmente como software libre, y él no considera Android como un sistema de software libre, ya que aunque el código es abierto y está accesible a cualquiera que lo quiera (en palabras de Stallman, "Android no es un sistema que permite que los usuarios sean libres, ya que impone muchas restricciones de uso" [16]).

1.4 Nuestra propuesta: servidor + cliente (*OpenSource*)

Tras presentar la tecnología *cloud* y la filosofía del software libre, estamos preparados para presentar al lector nuestra propuesta *OpenSource* de una solución completa de gestión documental, con capacidades *cloud*, un control de versiones y un cliente móvil. Este punto articula toda nuestra propuesta según el esquema clásico cliente/servidor para una mejor comprensión de la misma.

1.4.1 Servidor: entorno *cloud* + repositorios locales

Empezaremos a describir nuestra propuesta con el servidor que deberá ser instalado en las instalaciones del cliente. Utilizaremos las plataformas Dropbox y Alfresco, plataformas comerciales con un gran número de usuarios y con un gran soporte detrás.

De Dropbox tomaremos sus capacidades *cloud* para ofrecer sobre todo disponibilidad. Nos centraremos en la alta disponibilidad de sus servidores para ofrecer la capacidad de poder acceder a los activos de información que se encuentren en su fase final de desarrollo, para su posterior consulta en cualquier momento y lugar. Éste será nuestro entorno de producción.

De Alfresco tomaremos su capacidad para implementar un control de versiones, implementado en un servidor en el cliente a través de repositorios locales, para poder manejar en todo momento el flujo de los activos de información sin exponerlos a terceros, poniendo así en peligro los activos de nuestros potenciales clientes en un proceso externo a la empresa. Éste será nuestro entorno de explotación, donde podremos trabajar con los documentos

mientras mantenemos un control de versiones de los mismos para incrementar la productividad a la hora de trabajar con ellos.

Con esta aproximación estaríamos ofreciendo las ventajas de cada plataforma, intentando mitigar los inconvenientes de la otra, como si fueran dos caras de una misma moneda. Presentamos un entorno atractivo y competitivo orientado al mercado actual, en el que las TIC son una base importante del mismo.

Por último cabe destacar que al ofrecer dos entornos de trabajo diferenciados, de explotación y de producción, podemos separar claramente los activos que se encuentran en su fase final de los que no lo están, permitiendo establecer niveles de permisos.

1.4.1.1 Entorno Cloud: Dropbox como entorno de producción

Dropbox surge como una plataforma *online*, donde guardar todos tus archivos en *la nube* para disponer de ellos en todo momento actualizados y en todos tus dispositivos. Asimismo, Dropbox permite compartir nuestros datos con otros usuarios de forma colaborativa, es decir, si alguien hace un cambio en un fichero compartido, todos los usuarios con ese fichero dispondrán de ese cambio. Dispone además de aplicaciones nativas en todas las plataformas del mercado que permiten abstraer al cliente de la sincronización de los archivos.

Con grandes características, una funcionalidad muy amplia y una interfaz amigable, Dropbox es uno de los sistemas líderes en almacenamiento de archivos en *la nube* con alrededor de 175 Millones de usuarios [17], compitiendo con Google Drive, Microsoft SkyDrive o Ubuntu One.

Si bien es cierto que otros sistemas de disco duro virtuales, como por ejemplo Skydrive de Microsoft, tienen más usuarios [250M] [18], habría que comprobar cuántos de estos usuarios son usuarios activos. En efecto, tal y como se explica en [18] 50 millones de usuarios se agregaron desde el debut de Windows 8, sistema que incorpora SkyDrive por defecto, por lo que aquí pueden contabilizarse usuarios que no son activos, sino que disponen del servicio por defecto.

No obstante, debido a su facilidad de uso (tan fácil como instalar el programa en tu ordenador, *smartphone*, *tablet,* etc.; copiar los archivos dentro de la carpeta que hayas elegido en la instalación; y dejar bajo el control de Dropbox la sincronización de datos a través de un proceso transparente para el usuario) es el sistema que muchos usuarios eligen para su uso personal.

Además, hay que decir que desde su inicio Dropbox fue incorporando nuevas características. Es de especial relevancia la inclusión de un control de versiones con funcionalidad limitada (en las versiones de pago funciona a la perfección, no siendo así en la versión gratuita) que permite recuperar archivos borrados por error y guardar diferentes versiones de un mismo archivo.

Su fácil uso e integración con sistemas operativos heterogéneos convierten a Dropbox en una herramienta que favorece la *ubicuidad* de nuestros activos de información, lo que explica su incorporación como elemento fundamental de las TIC más empleadas por el usuario medio. Si a la fidelización lograda en virtud de su fácil instalación y uso, unimos los competitivos planes para negocios que ofrece Dropbox (ver Tabla 3), resulta fácil comprender el motivo por el cual

Dropbox se constituye como una de las primera opciones contempladas para gestión documental en entornos empresariales.

	Dropbox	Dropbox Pro	Dropbox para empresas
Precio	Cuenta gratuita	$ 99/usuario/año	$ 795/año para 5 usuarios $ 125/usuario adicional/año
Almacenamiento	2 GB	100 GB	Todo lo que necesitas

Tabla 3: Precio de Dropbox para negocios - dic 2013

Estos planes además ofertan al usuario una gran cantidad de servicios adicionales como puede ser una seguridad mejorada, historial de versiones ilimitada (como comentábamos anteriormente), seguimiento de ubicaciones o soporte por email/teléfono. Algunas de estas características (Tabla 4).

	Dropbox	Dropbox Pro	Dropbox para empresas
Seguridad mejorada			
Recuperación de archivos ilimitada	×	$ 3,99/mes	✓
Historial de versiones ilimitado	×	$ 3,99/mes	✓
Impide que se compartan archivos con personas ajenas al equipo	×	×	✓
Administración del equipo			
Seguimiento de inicios de sesión, dispositivos y ubicaciones	×	×	✓
Facturación centralizada del equipo	×	×	✓
Permite agregar y quitar miembros con facilidad	×	×	✓
Inicio de sesión único y Active Directory	×	×	✓
Soporte			
Soporte prioritario por correo electrónico	×	✓	✓
Soporte telefónico	×	×	✓
Especialistas en implementación dedicados	×	×	✓

Tabla 4: Características de Dropbox - dic 2013

Como podemos observar en la tabla adjunta (Tabla 4) la funcionalidad cambia mucho de un servicio a otro. En este punto, cabe destacar que 'Dropbox Pro', está pensado para un uso más intensivo de los datos almacenados, mientras que la versión gratuita está orientada a una compartición de archivos de forma

ocasional. Por su parte, la distribución Dropbox para empresas sería el producto más interesante para el ámbito de estudio.

En ningún caso Dropbox ofrece una solución para utilizar o *alquilar* los servidores de la empresa en su totalidad. Es decir, no existe una opción donde tengamos controlado por completo el acceso a nuestros archivos, con lo que no existe garantía de que sólo nosotros tengamos acceso a los mismos.

A modo de resumen, la plataforma Dropbox ofrece una serie de ventajas e inconvenientes que debemos tener en cuenta a la hora de utilizarla:

Ventajas

- Archivos siempre sincronizados entre todos los dispositivos con una cuenta de Dropbox integrada.

- Alta disponibilidad, gracias al sistema *cloud*, la mayoría del tiempo no dependeremos de un servidor que pueda estar inactivo.

- Realmente fácil de usar y configurar debido a su interfaz sencilla.

- Su soporte para todas las plataformas y sus algoritmos de sincronización nos permiten disfrutar de nuestros archivos en distintas plataformas, con lo que los mantenemos actualizados de forma cómoda y transparente para el usuario.

Inconvenientes

- Nuestros archivos no tienen un lugar fijo en un servidor, están en *la nube* (consultar 1.2 Entornos *Cloud*) de Dropbox, es decir, es un *espacio virtual* externo a nuestra empresa que no controlamos directamente.

- En ningún momento puedes determinar el flujo que siguen tus archivos, por lo que en teoría podrían ser leídos o interceptados por terceras personas sin que lo percibiéramos.

- Cualquiera que pueda inferir tu contraseña de acceso será capaz de acceder desde cualquier ubicación geográfica a tus archivos, ya que no se encuentran dentro de una red empresarial ni nada por el estilo (aunque el seguimiento de ubicación y la seguridad mejorada de la que se habla en las versiones de pago puede mitigar este problema).

- La disponibilidad de Dropbox es muy alta en comparación con los servidores que podamos desplegar nosotros mismos, no es del 100%, aunque como podemos ver en la Tabla 3, existen otros servicios en los que sí lo es [19].

Plataforma	Tiempo de Inactividad	Disponibilidad
Box.com	NO	100.00%
Google Drive	1 minuto	100.00%
Dropbox	13 minutos	99.97%
SkyDrive	15 minutos	99.97%

Tabla 5: Disponibilidad de plataformas *cloud*

1.4.1.2 Repositorios Locales: Alfresco como entorno de explotación

En contraposición a Dropbox tenemos Alfresco. Alfresco es un sistema de gestión documental totalmente orientado a empresas (Figura 1), y que ofrece un sistema que debe instalarse y mantenerse por la propia empresa. Esto nos permite configurarlo a nuestro gusto, así como instalarlo en nuestros propios servidores, brindándonos un control absoluto. Mediante Alfresco es posible controlar en todo momento el lugar en el que se encuentran los archivos, así como la manera que tenemos de compartirlos con los demás.

Figura 1: Alfresco y las empresas

Alfresco ofrece soluciones basadas en el sistema que la empresa quiera implementar (ya sea repositorios locales o *cloud*) aunque como se ha hablado anteriormente, una PYME muchas veces no es capaz de dedicar recursos suficientes a gestionar su propio *cloud*, por lo que la opción preferente es utilizar repositorios en un servidor local. En efecto, si la empresa elige contratar un servicio de *cloud* a un tercero, de nuevo no gestionado por ella misma, volveríamos el problema de que esa tercera empresa pudiera interceptar los archivos.

A modo de resumen, la plataforma Alfresco ofrecería las siguientes ventajas e inconvenientes respecto a otros sistemas:

Ventajas

- Archivos en servidores controlados en todo momento por la empresa.

- Posibilidad de implementar nuestro propio *cloud* con la misma funcionalidad en el caso de disponer de recursos suficientes.

- Implementado sobre el estándar CMIS (*Content Management Interoperability Services*) estándar *OpenSource* que permite la integración de cualquier software que lo implemente [20].

Inconvenientes

- Documentación y solución compleja, lo que dificulta mucho empezar a trabajar con ello, así como implementar nueva funcionalidad.

- Soluciones de programación complejas, que fuerza el uso de *Maven*, debido a la forma en que la empresa entrega el código y sus dependencias.

- En todo momento el/los servidores deberán ser mantenidos por la empresa, lo que implica un coste de mantenimiento, tanto de luz como de posibles problemas de hardware, seguridad y mantenimiento en general.

1.4.2 Cliente: acceso desde dispositivo móvil

En el marco de competitividad y eficiencia en el que está basado el proyecto, creemos que era vital ofrecer un cliente que estuviera enfocado a la productividad y la movilidad. Además, el hecho de que cada vez más los dispositivos móviles se están convirtiendo en una herramienta de trabajo más en las empresas, apoya nuestra decisión.

Es por ello que decidimos investigar el mercado de los terminales móviles e intentar tomar una decisión sobre qué sistema operativo utilizar, decisión acotada debido al tiempo de desarrollo del proyecto, ya que lo ideal hubiera sido desarrollar un cliente en todas las plataformas.

1.4.2.1 Importancia de los dispositivos móviles en el contexto actual de las TIC

Cada día salen al mercado dispositivos móviles más avanzados, que nos permiten cambiar nuestra manera de ver el mundo. Las TIC, *Tecnologías de la Información y la Comunicación*, han pasado a formar parte de nuestra vida de formas que jamás hubiéramos imaginado, especialmente desde la llegada de los *smartphones*, que nos ponen Internet en la palma de la mano.

Desde las *APPs* de mensajería instantánea hasta las redes sociales, es un hecho que cada vez más todos disfrutamos inventando y utilizando nuevas formas de comunicarnos con los demás, generando una dependencia por estos dispositivos que nos hace dejar de lado otros menos llamativos. En este marco de crecimiento de los dispositivos móviles, donde las ventas de terminales de sobremesa se están desplomando frente a la venta de este tipo de terminales, se está creando un importante mercado de APPs de competitividad empresarial al que debemos acceder para poder ser atractivos a nuestros clientes.

Por ello, *APPs* de productividad que llevan el negocio hacia este tipo de terminales son actualmente muy populares en las empresas. El poder acceder a los documentos de la misma de forma segura, en cualquier momento y lugar, se ha convertido en algo básico a la hora de desarrollar nuestra actividad empresarial. Por ello creemos acertada la elección de un terminal móvil para nuestro cliente de acceso a los documentos.

1.4.2.2 Soluciones más populares: Android vs iOS

1.4.2.2.1 El sistema Android

El sistema operativo Android surge en 2005 desarrollado por una *startup* del mismo nombre que fue comprada por Google en ese mismo año. Es un sistema basado en Linux y de código abierto, por lo que cualquiera puede descargar su código y *cocinar* sus propias *ROMs*, término que se ha puesto de moda últimamente debido a que varios fabricantes de *smartphones* entregan versiones modificadas de Android que pueden contener software que el cliente no quiere utilizar, lo que ha llevado a algunos programadores a crear (o *cocinar)* sus propias *ROMs* alternativas que se pueden instalar en el dispositivo para poder cambiar desde el aspecto hasta el rendimiento, hecho que se ha vuelto realmente popular últimamente debido a que los usuarios crean comunidades para mejorar el software, llegando a superar el original en algunos aspectos.

Ejemplos de este tipo de *ROMs* pueden ser:

- CyanogenMod: http://www.cyanogenmod.org/
- OmniRom http://omnirom.org/

Figura 2: Logo CyanogenMod y OmniRom

Semejante grado de personalización, el fácil acceso a los terminales, tanto a nivel económico, donde la mayoría de los *smartphones* más baratos y competitivos del mercado tienen Android como OS [21], como a nivel de programación, donde basta con descargar el SDK que proporciona Google para empezar a programar [22] en cualquier sistema operativo y plataforma, no como el SDK de Apple por ejemplo, XCode, que sólo obtiene toda su funcionalidad en el OS de Mac, ha hecho a este robot verde líder del mercado con un 90% de cuota en España [2] y según los datos del 2012, un 70% de la cuota mundial.

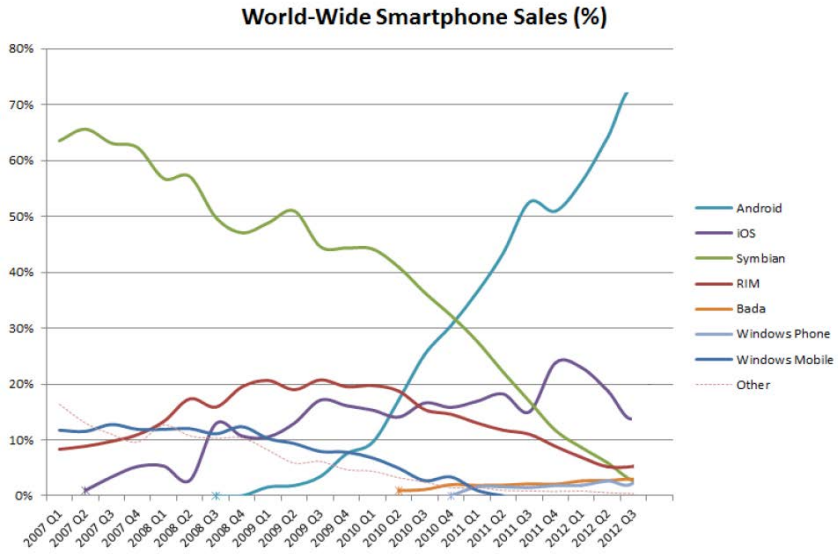

Figura 3: Cuota de mercado Android 2012

A día de hoy se pueden encontrar los siguientes datos relativos al 2013. Según la empresa Gartner, reflejan una subida de hasta el 79% en el 2º cuarto de 2013 [23]. Según IDC, la cuota alcanzaría el 81% en el tercer cuarto de este mismo año [24].

Figura 4: Cuota de mercado Android 2 cuarto de 2013 Gartner

También decir que las empresas líderes en la fabricación de *smartphones* con el OS Android serían Samsung, LG, Huawei y ZTE [25].

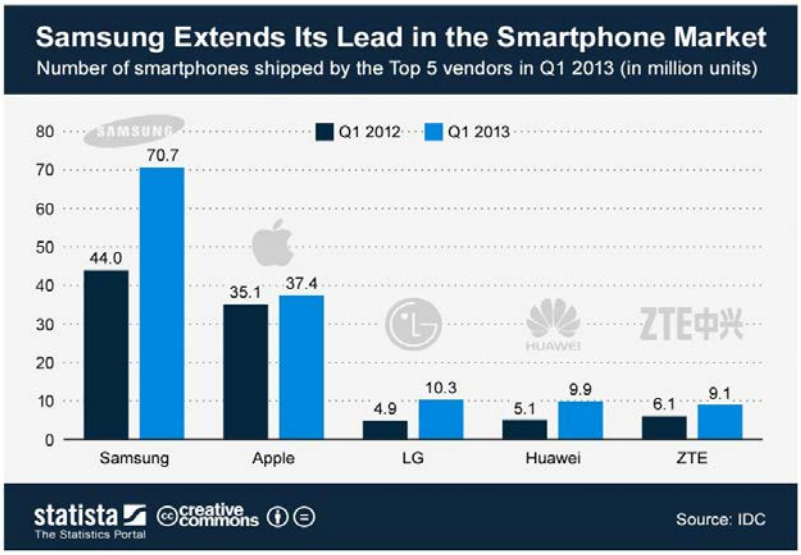

Figura 5: Empresas líderes en la fabricación de *smartphones*

Por último no podemos pasar por alto, como podemos observar en la Figura 5, el que sería el mayor rival de Android en el mercado, el sistema operativo iOS fabricado por Apple. Con su buque insignia, el iPhone. En cualquiera de sus versiones, desde el que muchas veces se considera el primer *smartphone*, el iPhone 3, debido a que fue la novedad en su fecha de lanzamiento, hasta el actual iPhone 5S (dic 2013, la gama más alta) presentan continuamente batalla a Android por atraer al mayor número de clientes para dominar el mercado, ya sea con el lanzamiento continuo de terminales cada vez más avanzados, o bien a través de batallas legales [26] [27] para controlar la oferta de terminales.

Como podemos ver, Android es el sistema operativo más popular en el mundo, pero ¿Por qué? ¿Qué es lo que realmente le hace tan popular? ¿Por qué esas cifras? Vamos a intentar analizar algunas de estas cuestiones para justificar nuestra decisión a la hora de elegir Android en nuestro proyecto.

1.4.2.2.2 Android y su usuarios (90% cuota de mercado España y 81% mundo)

Está claro que Android es el sistema que domina el mercado mundial ahora mismo pero, ¿por qué? Esto se debe a que existen innumerables distribuciones instaladas en todo tipo de terminales (desde la versión 2.3.3, las anteriores están prácticamente extintas, hasta la actual 4.4), mientras que por ejemplo Apple, sólo dispone de un terminal, de gama alta (aunque recientemente han lanzado lo que ellos consideran gama media, su precio dista mucho de ser de esta gama, ~500€) Android dispone de una enorme de gama de terminales entre los que se encuentran terminales de gama baja, como por ejemplo el Samsung *Galaxy Pocket* 79€, gama entrada como el *Motorola Moto G* 175€, gama media-alta como el *Nexus 5* 349€ y gama alta *Sony Xperia Z1* 699€. Con semejante abanico de teléfonos y precios, con aspectos y funciones completamente distintas ajustables a cualquier bolsillo, es sencillo encontrar terminales que ocupen todo el mercado más fácilmente que iOS.

En el mercado de las *tablets*, donde también se encuentra presente el sistema operativo Android, la diferencia no es tan significativa, como podemos observar en la figura adjunta, aunque estas cifras son recientes, del 2013, es el primer año en el que estas cifras se cumplen.

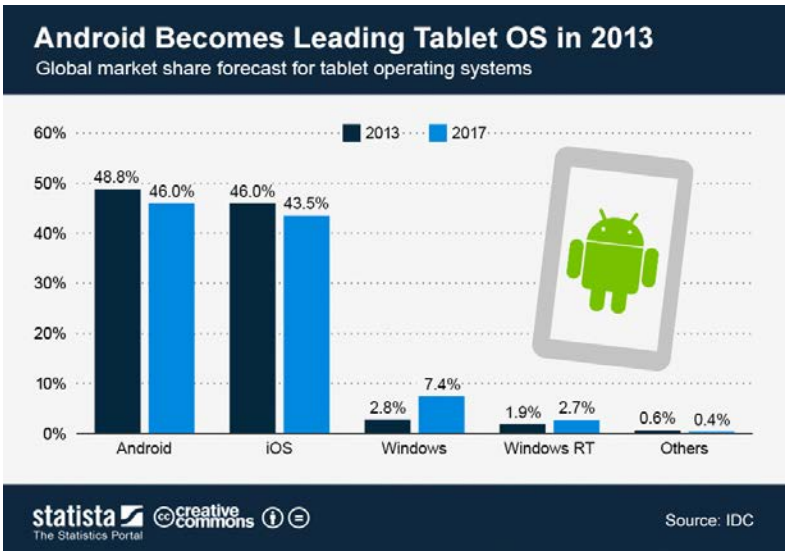

Figura 6: Cuota de Mercado OS *Tablets*

Android es el OS más usado, no obstante debemos fijarnos en los beneficios que produce y la pregunta que surge es: ¿Es un negocio rentable? Los datos demuestran que no.

[28] Como podemos observar de forma gráfica en la figura inferior y leer en este artículo de la prestigiosa revista digital Forbes, Apple obtiene más ingresos que sus competidores con sus terminales, pese a vender menos. Si el beneficio no se encentra en el hardware, la respuesta sólo puede estar en el software, el mercado de aplicaciones.

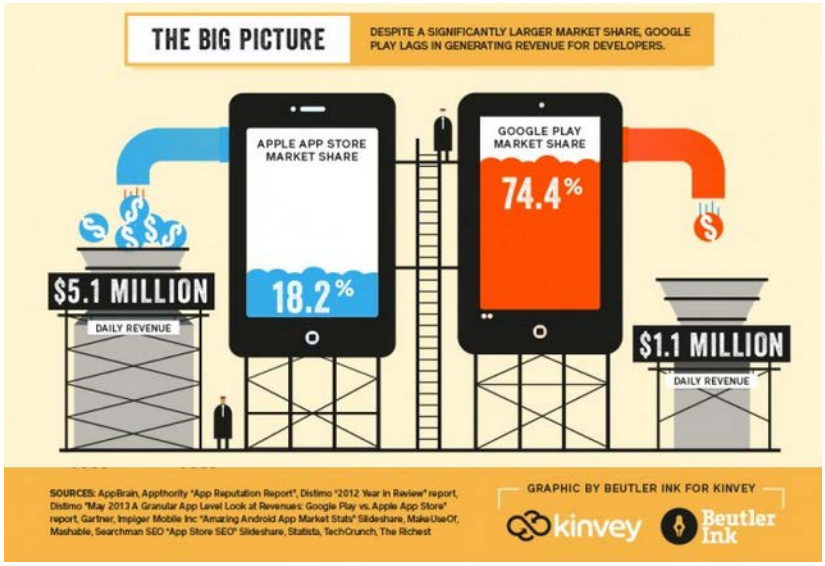

Figura 7: Beneficios de Google Play y Apple Store

1.4.2.2.3 Principales competidores, Google Play vs. Apple Store

En el ecosistema de Google, el código es abierto, accesible a cualquiera, como se comentaba al principio. Para programar en Android, sólo hace falta descargar el IDE, y los *plugins* (el oficial es Eclipse, aunque existen otras soluciones igualmente accesibles). En el caso de iOS sin embargo, es muy distinto, para programar con toda la funcionalidad se necesita el IDE Xcode, que únicamente funciona completamente bajo el sistema operativo Mac.

También la cultura de los usuarios es muy distinta. En el ecosistema Android, los usuarios no están acostumbrados a pagar por las *apps*. No es que no quieran pagar, sino que las APPs más utilizadas, quitando juegos de grandes firmas, son gratis. El modelo más común es lanzar una *app* con toda la funcionalidad apodada "Pro" o "Full" o algún nombre similar, mientras se lanza otra con la funcionalidad limitada llamada "Lite" o "Free" o similar, que contenga publicidad, lo que hace ganar dinero a los desarrolladores de forma continua, en vez de a través del pago inicial de la *app*.

Si bien es cierto que este modelo también existe en el Apple Store, los usuarios están más acostumbrados a pagar por las *apps* más utilizadas, por ejemplo, la popular *app* de mensajería instantánea, Whatsapp, en Apple Store cuesta 0,79€ mientras que en Google Play es gratis. Principales categorías de APPs:

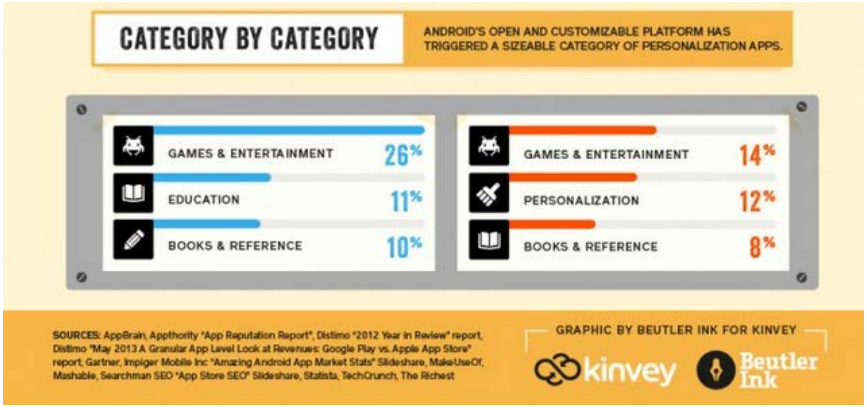

Figura 8: Categorías de APPs

Como podemos ver en la Figura 8, en Apple Store, de color azul, las *apps* de educación están por delante de las *apps* de personalización en Android. Esto se debe a dos motivos, las grandes ofertas que ofrece Apple en el sector educación, por un lado, y el altísimo grado de personalización que ofrece Android por otro, que permite a los usuarios configurar un *smartphone* totalmente personalizado y ajustado al uso que le dan.

Respecto a los beneficios y las licencias de desarrollador, también es distinto. Ambas empresas retienen el 30% de los beneficios de los desarrolladores [29] [30] por utilizar sus servicios de *hosting* de *apps*. En cambio, para publicar *APPs* en Google Play sólo necesitas una cuenta de desarrollador, para la que tienes que pagar 25$ una única vez, mientras que para *APPs* en Apple Store debes registrar un plan de desarrollador que cuesta 99$ anuales (estas costes para los desarrolladores pueden variar un poco si se requieren servicios externos como APIs de terceros o funcionalidad extra como *in-APP billing*, pasarelas de pago, etc.).

Por último decir que ambas establecen límites en la cuantía de una APP, en el caso de Google Play es de 200$ mientras que en Apple Store es de 999$. No obstante, aunque existen APPs de esta cuantía, exceptuando algunas que ***parecen*** ser de contenido serio, como *VIP Black* o *Nursing Central*, otras ***parecen*** meras trampas para clientes incautos que bien no se han percatado de la cuantía real, o han sido engañados para comprarlas (*Got Cash?* ó *Most Expensive Play APP*) ya que la media de la cuantía de las APPs en estos mercados ronda los 0.19$ y los 0.06$ en iOS y Android respectivamente (Fuente: Flurry Analytics [31]).

Aportando como argumentos los puntos anteriormente expuestos, su cuota de mercado, su alto número de usuarios, el fácil acceso a las herramientas de programación y su filosofía *OpenSource*, hemos decidido utilizar Android para nuestra propuesta de cliente móvil.

1.5 Organización del documento

Una vez hemos aclarado nuestras intenciones y objetivos al realizar este proyecto, así como presentado las tecnologías que vamos a utilizar y el esquema de arquitectura que vamos a seguir, analizaremos en profundidad todas las tecnologías utilizas en el mismo, explicando nuestra visión personal del proyecto, así como la solución presentada.

Empezaremos con el análisis del proyecto y su definición en el capítulo 2, a través de las herramientas que nos proporciona la ingeniería del software. Los problemas que planteamos en un primer momento, así como nuestra aproximación inicial y la planificación para llevarla a cabo. Además intentaremos transmitir al lector el sentido práctico que creemos tiene este trabajo, pudiéndose incluso enfocar como un proyecto de negocio, a través del plan presentado en el mismo capítulo.

A continuación en el capítulo 3, continuaremos mostrando el diseño propuesto para la solución que hemos planteado, desglosando de manera técnica y exhaustiva las tecnologías utilizadas en el desarrollo del proyecto, para llegar a la solución finalmente presentada, intentando esclarecer en todo momento cualquier duda que el lector pueda tener acerca de las mismas.

Por último, en el capítulo 4, mostraremos el producto final a través de una serie de capturas de pantalla, donde poder observar la APP con todo el detalle que el papel nos permite. Después discutiremos las posibles líneas de trabajo futuro e implicaciones que tendría este proyecto, para terminar de mostrar al lector nuestra visión sobre este producto. Para concluir, hablaremos de los conocimientos adquiridos y resaltaremos los aspectos que creemos más positivos de los mismos.

Capítulo 2

Análisis del problema

Tal y como se ha reseñado en la introducción, el objetivo principal de este proyecto es la implementación a bajo coste de una plataforma de gestión documental que permita diferenciar dos contextos: el contexto de producción y el contexto de explotación. Dado que se pretende que la propuesta sea de muy fácil uso, se ha optado por una de las alternativas *cloud* mejor conocida por los usuarios de las TIC: Dropbox

Ahora bien, el tránsito del dominio de producción al de explotación en el seno de una empresa involucra un riesgo, pues significa la publicación de parte de nuestros activos de información en un sitio o emplazamiento que no contralamos (esto es, se almacena en un servidor no confiable). En este línea, y teniendo en cuenta trabajos como el propuesto en [32] para el caso de *GoogleDocs,* un requisito esencial de nuestro trabajo viene constituido por la creación de una capa de seguridad para garantizar la confidencialidad de los documentos publicados en Dropbox.

Antes de continuar con la contextualización de nuestro proyecto en términos de seguridad informática, hemos de destacar la superioridad de nuestro diseño respecto al proyecto presentado en [32]. En efecto, en dicho trabajo el producto desarrollado sólo es válido para documentos almacenados *GoogleDocs* y no está enfocado a ser un producto final multiplataforma como el que proponemos, sino un *plug-in* para Firefox que busca añadir seguridad a *GoogleDocs*. Hecha esta aclaración procedemos con un descripción más en detalle de la matriz de amenazas del conjunto de tecnologías utilizadas como soporte de nuestro producto. En consecuencia, en las secciones siguientes se introducirán los principales problemas de seguridad que presentan los entornos *cloud* y las aplicaciones Android.

2.1 Seguridad Informática: conceptos fundamentales y principales desafíos

Hoy en día la seguridad y la protección de datos sensibles es primordial para una empresa tanto como para un usuario corriente, ya que puede dar lugar a situaciones muy desagradables, como por ejemplo el robo de nuestras credenciales bancarias para dejar vacía nuestra cuenta bancaria [33] o el robo de otro tipo de credenciales que pueden dar lugar a suplantaciones de identidad con fines lucrativos o dañinos [34].

Debido a los agujeros de seguridad que encontramos en Internet, explotados por diferentes perfiles con niveles de amenaza totalmente distintos, se puede conseguir información sensible de cualquiera a través de Internet. Desde los perfiles conocidos en la comunidad como *kiddies,* aficionado que se dedican a intentar robar información para molestar, principalmente haciendo cambios en páginas web públicas y comportamientos de este estilo, haciendo notar que han sido capaces de evitar la seguridad de los mismos, hasta auténticos profesionales del mercado negro que se dedican a conseguir toda la información posible de una víctima para intentar chantajearla, o vender su información para conseguir beneficios.

A pesar de este tipo de perfiles, últimamente la mayor parte de las noticias sobre seguridad informática, pasan por los gobiernos, que están en las portadas de prensa habitualmente desde el caso Assange y el caso Snowden. A raíz de estos casos se han hecho públicas varias noticias acerca del llamado hacking gubernamental [35] [36].

Ante la existencia de tal cantidad de amenazas surge la pregunta de si es posible a nivel de usuario, utilizar Internet de manera segura. La respuesta no siempre está en manos del desarrollador, pero lo que está claro es que como desarrolladores debemos, mejor dicho ***tenemos*** que ponérselo todo lo difícil que podamos a aquellos que quieran atacar a nuestros usuarios y ése es el punto de partida de nuestro proyecto.

Seguridad *Cloud*

El gran problema en el uso de esta tecnología reside en el hecho de que no tenemos control alguno sobre el tráfico y el mantenimiento de los datos, ya que están alojados en servidores de terceros a los cuales no tenemos un acceso completo, y nunca sabemos dónde, ya que no suelen estar en uno fijo. Por ello y debido a que configurar nuestra arquitectura *cloud* es verdaderamente complejo y costoso, muchas empresas se ven forzadas a utilizar otras alternativas menos competitivas, ya que no pueden asegurar con certeza que sus datos no se verán comprometidos (ver *outsourcing* y Sección 1.2).

Además, esta disponibilidad y el acceso global a través de internet, puede ser sinónimo a veces de ataques, ya que en cierto modo, estamos facilitando el acceso a posibles atacantes para que puedan ejecutar su trabajo desde cualquier ubicación geográfica.

El *cloud* es un paradigma relativamente nuevo, que evoluciona día a día, ofreciendo una gran cantidad de posibilidades a usuarios con pocos recursos, permitiéndoles acceso a nuevas tecnologías para competir en nuevos mercados. El problema que surge es que debido a la complejidad de este tipo

de infraestructuras, a día de hoy no es posible instalar una propia, con características tan competitivas como las que ofertan las empresas dedicadas a ello, por lo que nos vemos forzados a ceder el control de nuestros activos de información a estas empresas si queremos utilizar sus servicios. De esta forma nos estaremos exponiendo en cierto modo. Por ello el usuario deberá valorar si quiere pagar este precio por utilizar este tipo de servicios.

Seguridad Android

Debido al incremento del número de de los dispositivos móviles, y al objetivo de los hackers de obtener información sensible de sus víctimas, los *smartphones* se han convertido en uno de los principales objetivos de los desarrolladores de *malware.* Dichos dispositivos albergan enormes cantidades de información sensible (como e-mails, cuentas de redes sociales, agendas, etc.), motivo por el que se convierten en objetivo prioritario para los atacantes.

Tal y como se subraya en [37], los terminales Android son cada vez más el objetivo principal de los desarrolladores de malware (ver Figura 9). En el artículo citado el autor nos desvela que el malware más popular son los troyanos (ver Glosario) (el 81% de los ejemplos de malware constatados.), que se centran el robo de información sensible [38].

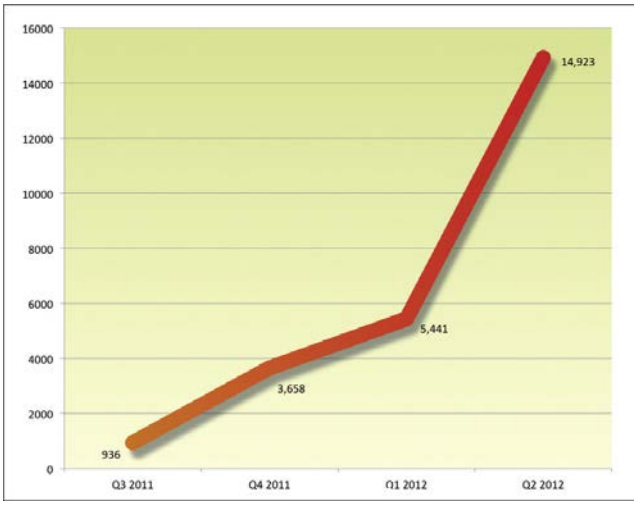

Figura 9: Evolución el número de programas malware según Kaspersky

Cabe destacar que en el informe de seguridad de Kaspersky para el año 2013, la cifra de programas de malware que apuntan a Android sigue aumentando debido a la creciente popularidad de estos dispositivos [39] haciendo que los usuarios se vean forzados incluso a modificar el uso de sus dispositivos.

Como podemos observar en el artículo, el malware se suele centrar en esta plataforma debido a la gran cuota de mercado, mostrada en los puntos anteriores. Al igual que ocurre con los equipos de sobremesa, por ejemplo, la cantidad de ataques que sufre Windows en comparación con los que sufre la distribución Ubuntu de Linux. Lo más curioso no obstante es el punto en el que habla de que todo el código malicioso está generado por la comunidad de investigadores de seguridad y no por la comunidad hacker, que según el escritor, "-No escribe su propio código malicioso y no ha demostrado la habilidad para hacerlo". Esto abre aún más el debate sobre quién está realmente detrás de la mayoría de los ataques para conseguir información de los usuarios.

En cualquier caso, centrándonos en el tema del proyecto, no gastaremos esfuerzos en averiguar quién es el culpable, sino en tratar de que los ataques no tengan éxito.

La cultura de los usuarios

A pesar de todo lo que hemos hablado anteriormente es importante destacar que la mayoría de los ataques que tienen éxito contra los usuarios no se deben muchas veces a que el software esté comprometido, sino al mal uso de las aplicaciones por parte de los usuarios.

Un buen ejemplo de este hecho es el reciente ataque que ha sufrido Adobe [40], en el que millones de cuentas de usuario han sido comprometidas, algunas con información sensible de los mismos como cuentas bancarias, direcciones de facturación, etc. Bien, al poco tiempo de producirse el ataque se publicaron las contraseñas más comunes de los usuarios [41] donde se podía comprobar efectivamente, que incluso en servicios donde muchos usuarios deben publicar cierta información sensible como sus cuentas bancarias, etc. la mayoría tenían contraseñas inseguras como '123456' (la más popular), '123456789' ó 'password', fácilmente atacables a través de fuerza bruta debido a que son las primeras que un hacker probará en un ataque así.

Esta situación manifiesta cómo la ingeniería social y el ataque al usuario mismo puede superar muchas veces toda clase de seguridad, por lo que es muy importante educar a los usuarios a utilizar las aplicaciones correctamente y en el caso de no poder hacerlo, porque sea costoso o imposible, intentar acotar programáticamente este tipo de comportamientos con comprobaciones a la hora de crear las contraseñas para las cuentas, etc.

2.2 Problemas planteados

Como hemos visto en la introducción de este capítulo, el principal problema que se plantea y al cual se quiere dar solución es que existen graves problemas de seguridad a la hora de utilizar ciertos servicios de terceros. Es por ello que muchos usuarios se ven forzados a evitar el uso de estos servicios debido a la inseguridad que producen. Aun así muchos otros no pueden sino utilizar este tipo de servicios para el desarrollo de sus proyectos empresariales. En la medida que hoy en día es inevitable utilizar software desarrollado por terceras partes, la debida protección de los usuarios sólo es plausible mediante la implantación de la transparencia en el desarrollo software y la correcta concienciación de los mismos en el uso de las TIC. Como hemos destacado previamente, el código abierto constituye una más que interesante posibilidad de hacer público todos los detalles de implementación, y sobre todo, evita cualquier tentación de desplegar falsas o no fundadas promesas de seguridad basadas en la ocultación de detalles cruciales de los diseños adoptados. Esta técnica de seguridad, basada en conseguir niveles de protección mediante implementaciones opacas, se denomina *security through obscurity.* Como se ha demostrado en repetidas ocasiones a lo largo de la historia de la criptografía [42], siempre con un tremendo fracaso.

Sin embargo, creemos que existe una solución. Ya que no podemos controlar el tráfico de nuestros datos, ni el lugar en el que están alojados, ni la forma en la que funcionan los servicios de forma interna. Por ello la solución viable y transparente al usuario que planteamos, consiste en el uso de la criptografía para cifrar los datos sensibles antes de que lleguen a parar a estos servicios de terceros, no controlados por nosotros, de tal forma que si llegaran a parar a manos de otros, no fueran capaces de leerlos ni utilizarlos de ningún modo.

Además mediante una declaración explícita de los objetivos que queremos alcanzar, así como la apertura del código, evitamos caer en la tentación de la opacidad a la hora de implementar nuestros servicios, excluyendo por tanto el ofrecer una falsa promesa de seguridad. En nuestro caso, el objetivo primordial del proyecto será garantizar la confidencialidad de los activos de información, y para conseguirlo utilizaremos el estándar criptográfico propuesto por el NIST, el AES con tamaño de clave 256 [43] y su modo de cifrado CBC [44].

No obstante existe un problema con esta aproximación. Debido a que los datos están cifrados, no podemos llevar un control de versiones de los documentos, ya que esta técnica consiste en guardar en una base de datos los bytes distintos de cada versión, para recuperarlos si fuera necesario, y ya que los bytes cifrados resultantes de cifrar con AES en modo CBC (el cifrado elegido) son completamente diferentes en cuanto ocurre el más mínimo cambio en el documento en claro, (de ahí su robustez, ver - AES 256, CBC *Cipher: Block Chaining mode*), el control de versiones no es de inmediata implementación, y excede los objetivos de este proyecto (definidos a través de los requisitos funcionales expuestos en el punto siguiente).

Por ello planteamos el uso del servidor Alfresco, controlado enteramente por nosotros, como entorno de explotación, donde poder llevar un control de versiones sobre los documentos. Una vez finalizado su trabajo en ellos, los subiremos a la plataforma Dropbox, elegida para su compartición y futuro acceso gracias a la tecnología *cloud*, cifrados para evitar el acceso de terceras personas no deseadas a nuestra información.

2.3 Requisitos funcionales

En este apartado especificaremos los requisitos que describen el comportamiento de nuestra aplicación, los requisitos funcionales.

- RF1: Crear un usuario en la aplicación
    1. En la pantalla de inicio se seleccionará *Nuevo Usuario*
    2. El sistema solicitará una contraseña de acceso
    3. Finalizado el proceso la aplicación lo guardará en la base de datos de la misma
- RF2: Crear repositorios anidados en el repositorio Alfresco
    1. En la pantalla de repositorios de Alfresco se seleccionará la opción de crear carpeta.
    2. La aplicación mostrará una pantalla donde rellenar los datos relevantes como el nombre de la misma.
    3. Una vez rellenado creará una nueva carpeta con los datos especificados.
- RF3: Subir archivos al repositorio local
    1. Desde la pantalla de Alfresco se realizará la navegación hasta el directorio deseado.
    2. Una vez allí, seleccionaremos la opción de subir fichero
    3. La aplicación mostrará una pantalla para elegir el fichero deseado en el dispositivo para seleccionarlo.
    4. Una vez seleccionado la aplicación lo subirá automáticamente, permitiéndonos verlo en el directorio una vez terminado el proceso

- RF4: Descargar archivos desde el repositorio local
    1. Desde la pantalla de Alfresco se realizará la navegación hasta el directorio donde se encuentre el archivo deseado.
    2. Una vez allí, seleccionaremos el archivo deseado desde la interfaz mostrada en pantalla.
    3. Una vez seleccionado la aplicación lo descargará y ofrecerá al usuario abrirlo con las aplicaciones nativas disponibles en el terminal.
- RF5: Subir archivos a la plataforma *cloud*
    1. Desde la pantalla de Dropbox podremos seleccionar la opción de subir archivos.
    2. La aplicación mostrará una pantalla para elegir el fichero deseado en el dispositivo para seleccionarlo.
    3. Una vez seleccionado la aplicación lo encriptará utilizando algoritmos seguros.
    4. Después lo subirá automáticamente en una subcarpeta que lo identifique como único, favoreciendo el orden en el entorno de producción.
- RF6: Descargar archivos desde la plataforma *cloud*
    1. Desde la pantalla de Dropbox podremos seleccionar la opción de descargar archivos.
    2. Una vez allí, la aplicación mostrará los directorios existentes en la cuenta de Dropbox.
    3. Podremos navegar por ellos hasta encontrar el archivo que queremos descargar.
    4. Una vez seleccionado el archivo la aplicación se encargará de descargarlo en la memoria local.
    5. Terminado este proceso, la aplicación descifrará el archivo y permitirá al usuario acceder a él con las aplicaciones nativas disponibles en el terminal.
- RF7: La APP será *multi-lenguaje* aprovechando las capacidades de Android para facilitar el crecimiento en distintos mercados
    1. En cualquier pantalla se podrá salir para dirigirnos al menú de ajustes del dispositivo
    2. En el menú se podrá seleccionar cualquier idioma del multi5 europeo (inglés, italiano, alemán, francés o español)
    3. De vuelta a la APP todas las cadenas se encontrarán traducidas, mostrando la capacidad de multi language
- RF8: La APP permitirá realizar control de versiones de los documentos en fase de explotación
    1. En la pantalla de detalles de documento de Alfresco se seleccionará el botón versiones
    2. La aplicación mostrará un histórico de versiones del documento seleccionado, permitiendo a acceder a los detalles de la versión actual

2.4 Requisitos no funcionales

Una vez especificado el comportamiento de la aplicación pasaremos a definir las características y la funcionalidad general de la misma.

- RNF1: La APP debe funcionar en pantallas de distinta resolución y distintos terminales Android para facilitar su implantación en el mercado.
- RNF2: Se requerirá un dispositivo con Android 3.0 *HoneyComb* o superior para poder ejecutar toda la funcionalidad debido a las restricciones de API implementadas en el proyecto, así como a las implementaciones de los procesos criptográficos.
- RNF3: La base de datos no debe exponer contraseñas ni otros datos sensibles de los usuarios en texto claro.
- RNF4: El cifrado de documentos debe ser suficientemente robusto como para que no se pueda romper en el tiempo de uso del documento.
- RNF5: La aplicación deberá utilizar en todo momento estándares de la industria para favorecer líneas de trabajo futuras que permitan la ampliación de la misma.
- RNF6: En línea con RNF5 se deberá tratar de seguir un diseño modular que permita la ampliación futura de la APP a través de nuevos módulos
- RNF7: Se requerirá un servidor dedicado, ya sea *hosting* tradicional o virtual o un entorno *cloud* para instalar y utilizar el sistema Alfresco.
- RNF8: Se requieren cuentas de usuario en los servicios que ofrece la APP para utilizar toda su funcionalidad.
- RNF9: Se requerirán las APPs nativas correspondientes instaladas en el dispositivo si se quiere trabajar en el mismo (procesadores de texto, lectores de pdf, APPs de tratamiento de imágenes, etc.).
- RNF10: Se requerirá conexión a internet en el dispositivo para poder utilizar toda la funcionalidad.

2.5 Planificación y presupuesto

2.5.1 Planificación del proyecto

A la hora de organizar el tiempo que le íbamos a dedicar a cada tarea, con el fin de terminar satisfactoriamente el proyecto decidimos confeccionar un diagrama de Gantt, donde observar claramente los plazos que pensábamos definir en el proyecto y facilitar el cumplimiento de los mismos. La planificación inicial fue la siguiente (ver Anexo I).

En todo momento intentamos cumplir con los plazos acordados, aunque no siempre nos fue posible, y algunas veces nos vimos obligados a ajustar la planificación. No obstante, en líneas generales acabamos satisfechos con esta planificación ya que se cumplió en la mayor parte del tiempo.

2.5.2 Plan de negocio

Para hacer más atractivo este proyecto se ha especificado un plan de negocio para sacar beneficio a nuestro software. Se propone el siguiente documento, orientado a una campaña tipo *KickStarter*, emulando una campaña real de este sitio. (*NOTA: el formato es el producido por el CSS de Github en ficheros markdown, para ilustrar el conocimiento adquirido. ver Anexo IV)

Con este plan de negocio se propone lanzar una campaña en KickStarter o una plataforma similar como IndieGoGo para poder lanzar un negocio a través de nuestro software, orientado a pequeñas y medianas empresas que quieran tener soluciones personalizadas a través de nuestros planes de desarrollo sin

invertir una gran cantidad de capital. En cualquier caso se entiende que la APP debe estar finalizada una vez se quiera empezar con este plan de negocio, para que, ya resulte el negocio rentable o no, se pueda contactar con el equipo de desarrollo para futuras oportunidades o demostrar el trabajo realizado. Para poder dar visibilidad a nuestro proyecto se ha incluido una página web elegida específicamente para el proyecto, diseñada en HTML5, con un diseño moderno y actual para poder captar futuros clientes (ver Anexo II).

2.5.3 Post-producción: monetización

A menudo uno de los aspectos más importantes para que una APP móvil, ya sea en Android o en iOS, funcione y genere beneficio, es la monetización de la misma, el seguimiento para comprobar si está generando beneficio o por el contrario está generando pérdidas y no está siendo rentable, y en este caso, intentar averiguar el momento en el que se produce para mitigar las mismas o incluso cambiar aquellas partes del modelo de negocio que hacen que pierda para que no sea así.

Para este proceso definiríamos un seguimiento de las palabras que hemos comprado en Google Adwords, para comprobar cuáles reciben más clicks útiles, es decir, cuáles conducen al público hacia nuestra web o nuestra APP a través de las mismas. Este parámetro se denomina CTR o *Click Through Rate.* Descartaríamos las que no fueran interesantes e intentaríamos revisar otras nuevas.

Por último recogeríamos todo el *feedback* obtenido de nuestros usuarios por parte de las redes sociales y comentarios en el mercado de aplicaciones para mejorar nuestra APP y hacerla más llamativa.

Una vez seguido este proceso, si vemos que tiene éxito, el plan sería empezar a ampliar la plantilla de personal para trabajar en nuevos proyectos, ya fuera ampliando la APP o lanzando proyectos nuevos relacionados con la imagen creada de la marca, que puedan ser rentables sobre esta empresa, con un nombre ya formado y conocido entre los usuarios.

2.6 Entorno de trabajo y estilo de programación

Es importante mencionar el entorno de trabajo con el que hemos abordado el proyecto para que el lector tenga claro en todo momento las tecnologías utilizadas para el desarrollo del mismo (se incluye una guía de instalación del entorno en Anexo I: Instalación del entorno).

2.6.1 Ubuntu + Android Developer Tools + Maven + Alfresco CE 4.2 + GitHub

Para este proyecto se ha utilizado el sistema operativo Ubuntu, en concreto la versión 12.04 LTS descargable de forma gratuita desde la web de ubuntu [45].

Para programar se ha utilizado el IDE proporcionado por Google, Android Developer Tools, que está basado en el popular IDE OpenSource Eclipse, y se entrega en el bundle del ADT al descargarlo, para empezar a programar rápidamente, ya que contiene todos los plugins para programar en Android.

Para las dependencias de Alfresco hemos instalado Maven, otro proyecto OpenSource de Apache que sirve para solucionar las dependencias de proyectos grandes.

El servidor Alfresco utilizado es el Alfresco Community Edition 4.2 para Linux.

Por último, pero no menos importante, para llevar un control de versiones del código y de los documentos relacionados con el proyecto, algo imprescindible en proyectos de una mínima envergadura, se ha utilizado GitHub, uno de los sistemas de control de código más populares entre los desarrolladores. Para poder utilizar la funcionalidad de repositorios privados, y que no existieran problemas con las licencias, de que alguien copiara código, etc. se ha utilizado una cuenta educativa, que permite crear repositorios privados.

En todo momento se ha optado siempre que ha sido posible, por utilizar software OpenSource, para ser consecuentes con nuestra forma de pensar y con las ideas del proyecto (ver - El software libre y Resumen).

2.6.2 Estilo en el código

En este proyecto hemos intentado en todo momento seguir un estilo definido tanto en el código como en la documentación.

Respecto al código hemos intentado respetar en todo momento el estilo y convencionalismos de Java. A la hora de escribir tanto los bucles como las sentencias condicionales hemos cerrado siempre las sentencias interiores entre corchetes, aunque fueran de una única línea, para tratar de aumentar en todo momento la legibilidad del código.

También se ha cumplido en todo el proyecto el hecho de no superar la línea que ofrece el IDE en el límite de 80 caracteres por pantalla, para favorecer la lectura del código en pantallas de cualquier resolución así como las convenciones en los nombres de los archivos Java, xml y las variables.

Para la documentación se ha aprendido el lenguaje Markdown, implementado de forma parecida a HTML (leer 2.6.3 Documentación del proyecto).

2.6.3 Documentación del proyecto

A la hora de documentar el proyecto hemos utilizado algunas herramientas que nos han ayudado enormemente a la hora de realizar esta tarea, principalmente dos, Markdown y Mendeley.

Markdown

Para la documentación utilizada durante la fase de desarrollo del proyecto se ha utilizado MarkDown, un lenguaje de etiquetado tipo HTML que se utiliza comúnmente en el portal GitHub por su comodidad al escribir y su claridad como texto plano.

Markdown es un lenguaje creado para facilitar la legibilidad del texto plano, a la vez que nos permite, al ser un lenguaje de etiquetas como html, utilizar distintos CSS para cambiar el estilo del documento según nos interese. La extensión de los ficheros es .md. La sintaxis de este lenguaje transforma los elementos clásicos de html en nuevas etiquetas más legibles en texto plano (ver Anexo VI: Sintaxis Markdown)

Para ver un ejemplo se puede consultar el output que produce el CSS de Github en los anexos (ver Anexo II: Plan de negocio y Anexo III: Instalación del entorno). Aunque sin duda, lo más útil a la hora de utilizar Markdown es su pequeño compilador, que permite escribir código en la documentación con un formato agradable y legible de forma sencilla (ver ejemplos en los anexos anteriormente citados)

Esta funcionalidad sumado a que el propio GitHub tiene un CSS claro y agradable para este tipo de ficheros, hace que sea ideal para generar documentación sobre la marcha de partes concretas del proyecto que puedan necesitar algún tipo de explicación para el proceso de desarrollo.

Mendeley

Otra herramienta que ha ayudado enormemente en la realización de este trabajo hasta volverse prácticamente imprescindible ha sido Mendeley [46], una herramienta para organizar toda la bibliografía del proyecto, permitiendo importar desde pdfs hasta enlaces web, organizando y sincronizado vía web. Además es capaz de generar citas en los procesadores de texto más conocidos, según sintaxis populares en artículos científicos de entidades reconocidas como IEEE o la *American Science Association*. Además es capaz de generar automáticamente toda la bibliografía utilizada para escribirlo a partir de las referencias incluidas en el documento.

Capítulo 3

Diseño de la solución

Partiendo de la arquitectura descrita anteriormente y resumida en la figura adjunta (Figura 10), presentaremos el diseño de nuestra aproximación a través del diagrama de clases de la misma.

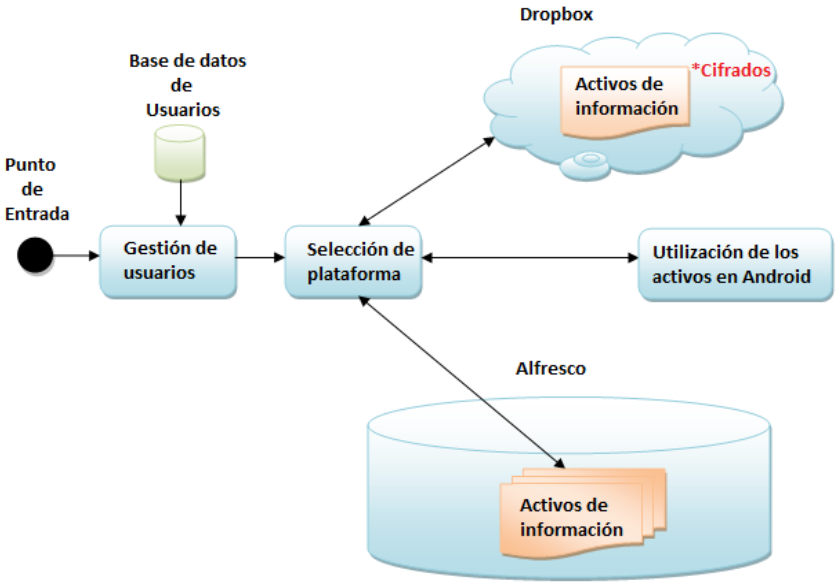

Figura 10: Arquitectura presentada

Como observamos en la figura adjunta, empezaremos nuestra aplicación autenticando al usuario que desea utilizarla, contra la base de datos de usuarios de la que dispondrá el dispositivo. Una vez autenticado podremos seleccionar la plataforma con la que se desea trabajar, ya sea Dropbox o Alfresco en un proceso transparente al usuario, donde podremos descargar y subir documentos según nuestras necesidades.

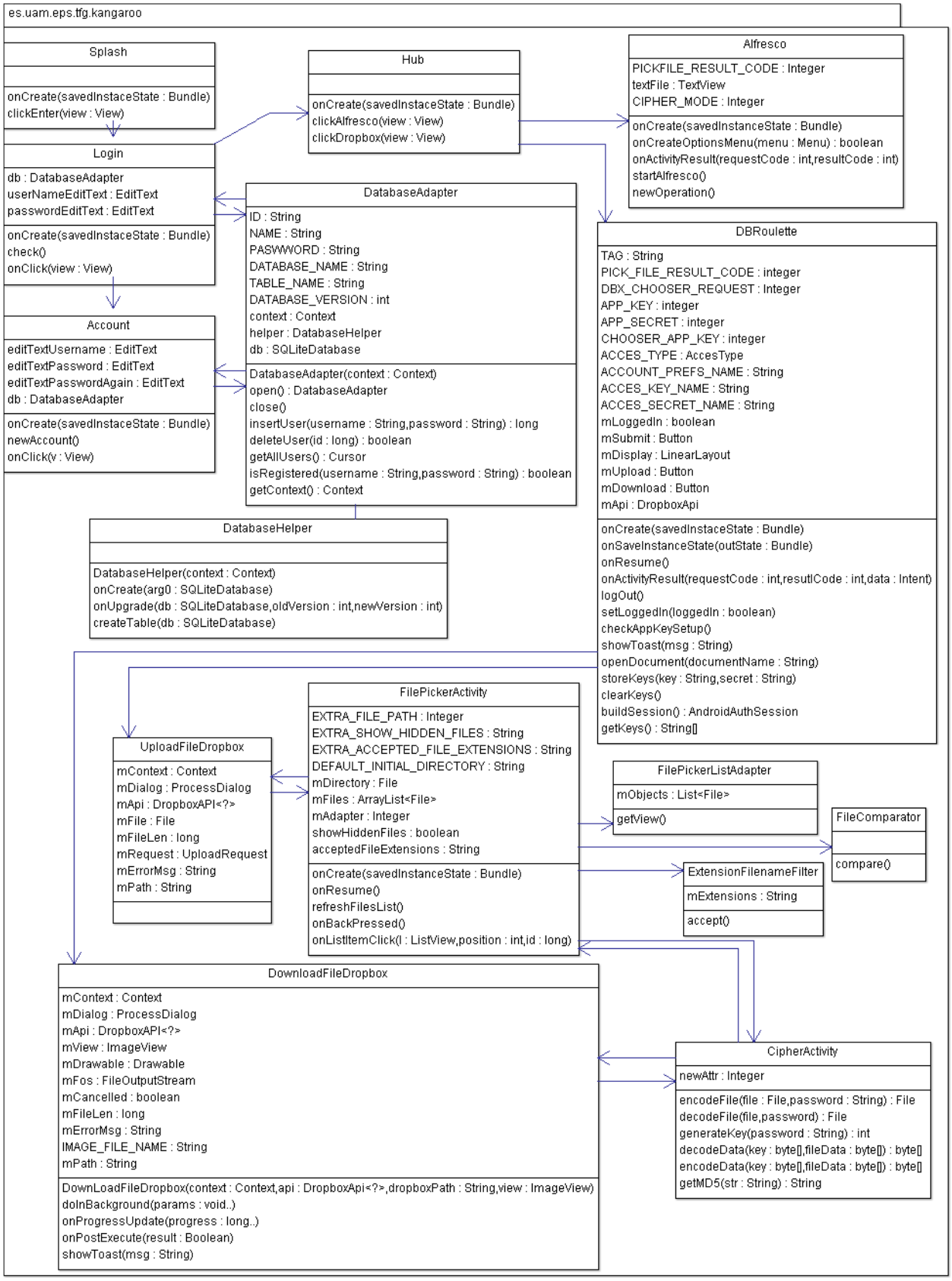

Figura 11: Diagrama de clases

Como podemos observar en el diagrama de clases adjunto (Figura 11) articularemos nuestro proyecto en base al paquete principal (es.uam.eps.tfg.kangaroo) que es el mostrado en el gráfico (los demás paquetes no se consideran relevantes ya que contienen código de los proveedores de servicios, principalmente Dropbox y Alfresco).

A continuación se resaltarán las clases más importantes:

- La clase Splash gestionará la creación de una pantalla de inicio/carga con el logo de la aplicación.

- La clase Login será la encargada de gestionar la pantalla de acceso a la aplicación a través las credenciales existentes en base de datos. Ayudado por la clase Account, que implementará la creación de una pantalla donde dar a la aplicación los datos necesarios para crear una cuenta de usuario.

- La clase DatabaseAdapter se encargará de implementar las funciones necesarias para tratar con la base de datos, abstrayendo al programador de utilizar *queries*. Para ello utilizará la clase privada DatabaseHelper.

- La clase Hub permitirá al usuario elegir entre la plataforma que desea utilizar, Dropbox o Alfresco.

- La clase Alfresco será la encargada de lanzar la actividad que controla este servicio (no se incluye dado que la mayoría del código es código ofrecido por Alfresco, por lo que no lo consideraremos en este estudio).

- La clase DBRoulette implementa el control sobre la plataforma *cloud* permitiendo al usuario decidir entre subir o descargar archivos de la misma. Para empezar a utilizar el servicio, primero nos obligará a enlazar la APP con Dropbox si es la primera vez que la utilizamos.

- Dependiendo de la opción elegida anteriormente haremos uso de la clase UploadFileDropbox, en el caso de querer subir un archivo, o de la clase DownloadFileDropbox, en el caso de querer descargarlo. En el primer caso, a la hora de subir archivos, haremos uso de la clase FilePickerActivity, permitiéndonos elegir un archivo en el dispositivo. A su vez esta clase se poyará en otras tres, FilePickerListAdapter, FileComparator y ExtensionFilenameFilter, clases que permiten abstraer al programador ciertos aspectos de la programación y que no consideramos importantes en el estudio que nos ocupa. En el caso de haber elegido la segunda opción, encontraremos la implementación de la descarga de archivos en la clase DownloadFileDropbox.

- Como podemos ver, ambas clases se apoyarán en la funcionalidad criptográfica ofrecida en CipherActivity, ya sea a la hora de cifrar los archivos (al subirlos) o de descifrarlos (al descargarlos).

3.1 El entorno de explotación, Alfresco

3.1.1 Servidor Alfresco

A la hora de instalar el servidor, nos dirigiremos a la página de desarrolladores de Alfresco, donde podremos descargar todos los archivos para levantar nuestro propio servidor. Por temas de licencias, se ha escogido Alfresco Community Server, ya que en principio no tiene coste para desarrolladores.

Tras hacer esto, y para continuar con los estándares de seguridad que se están intentando seguir durante todo el trabajo, hemos levantado el servidor con una configuración que no es la estándar, es decir, hemos cambiado los puertos de acceso por defecto y en todo momento hemos evitado utilizar combinación de usuario/contraseña de tipo 'admin'/'admin' ó 'admin'/'1234' por tentadoras que parezcan a la hora de desarrollar.

Se puede ver un vídeo de la instalación del mismo aquí

http://youtu.be/DebOnHo81yM

[47]

3.1.2 La API de Alfresco

Para programar con el entorno Alfresco hemos necesitado primero importar el SDK que hemos descargado de su página de GitHub:

(https://github.com/Alfresco)

Una vez descargado, nos hemos visto obligados a instalar Maven en nuestro entorno de desarrollo. Maven es una herramienta comúnmente utilizada en entornos con proyectos de gran dimensión y complicadas dependencias, que se encarga de forma transparente de las mismas para que el usuario no tenga que hacerlo. Hemos elegido el plugin m2ee, *OpenSource*, acorde con la filosofía del proyecto para nuestro entorno basado en eclipse. Una vez instalado, hemos importado los proyectos del sdk como proyectos Maven, lo que nos permite a través de los ficheros de dependencias 'pom.xml' de cada proyecto, adquirir las dependencias concretas de los mismos y empezar a programar en ellos. Tras este paso, se ha configurado la dependencia del SDK en el proyecto (*Propiedades* > *Android* > *Library* > *Add* - select alfresco-android-sdk).

Terminado este proceso ya estamos listos para empezar a utilizar la API de Alfresco. Al igual que con Dropbox, lo que se ha hecho es importar el ejemplo que se proporciona con la API y modificarlo hasta conseguir el resultado deseado. Para ello hemos importado los códigos fuente que se proporcionan y los hemos modificado para lograr este fin. No se incluye código de esta parte del proyecto porque no hemos alterado la funcionalidad de manera significativa, ya que el esfuerzo importante ha sido configurar correctamente el servidor y las dependencias de Maven.

Para generar el fichero .apk con Maven desde consola escribiremos

```
$ mvn clean install -DskipTests=true
```

Para instalar la APK utilizando ADB (ver Glosario):

```
$ ./adb install /home/path_to_file/alfrescoAPP.apk
```

3.2 El entorno de producción, Dropbox

A la hora de realizar nuestro proyecto necesitábamos una plataforma *cloud* en la que compartir todos los ficheros que se iban a generar en el entorno de explotación Alfresco, para no comprometerlo exponiéndolo a otras personas, ya fueran de la misma empresa o de otra. Por ello, hicimos una pequeña investigación, en la que analizamos la popularidad y la facilidad a la hora de programar de estas plataformas *cloud*, y llegamos a la conclusión de que la mejor plataforma para esto era Dropbox. Debido a su cómoda API de programación, su popularidad entre los usuarios y que últimamente está empezando a venir pre-instalada en muchos dispositivos Android, Dropbox se muestra como el candidato ideal para nuestro proyecto.

3.2.1 La API de Dropbox

Para agilizar la programación se ha utilizado las dos APIs de Dropbox, tanto la que llaman 'Core API' que es la API que nos permite interactuar plenamente con Dropbox, subiendo y bajando archivos de sus servidores, como el 'Chooser Drop-in ' que es un componente multiplataforma que nos permite abstenernos de implementar un gestor de ficheros para descargarlos a nuestro dispositivo.

Para utilizar estos componentes, tanto la API como el DropIn necesitamos seguir una serie de pasos, con el fin de que nuestro entorno de programación sea capaz de reconocer el código proporcionado por Dropbox y nos permita utilizarlo.

Dropbox API

La API de Dropbox se ha utilizado para controlar la subida de archivos a la cuenta de Dropbox. Para utilizarla lo primero que hay que hacer es enlazar la APP con una cuenta de Dropbox. Una vez hecho esto, la APP Kangaroo se encarga de cifrar el archivo con AES 256 (ver - AES 256) de forma transparente al usuario. Para solucionar el desorden de muchos usuarios, por defecto el archivo cifrado se subirá dentro de un directorio en Dropbox llamado Kangaroo, en el que creará (si no existe) una carpeta con nombre, la fecha y la hora de la subida, intentando solucionar este problema de algunos usuarios. Para poder utilizar la API necesitaremos registrarnos como desarrolladores en [48], acceder a la consola de la API y crear una APP con los permisos de "Full Dropbox".

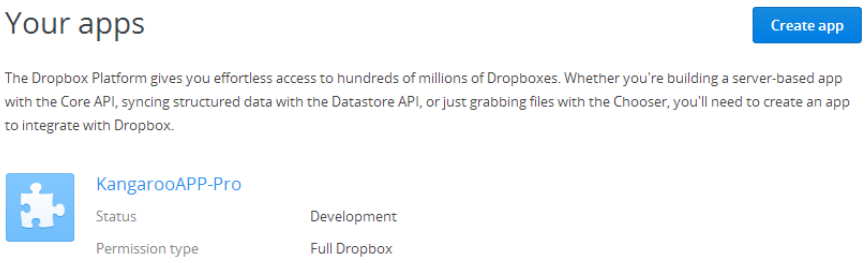

Figura 12: Creación de una clave para la API de Dropbox

Después de hacer esto, necesitamos añadir los permisos pertinentes en el fichero AndroidManifest.xml, tanto para la actividad de Dropbox, como para que establezca la conexión a Internet para funcionar.

```
<activity
  android:name="com.dropbox.client2.android.AuthActivity"
  android:launchMode="singleTask"
  android:configChanges="orientation|keyboard">
  <intent-filter>
    <data android:scheme="**********" />
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.BROWSABLE"/>
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>

<uses-permission
android:name="android.permission.INTERNET"></uses-permission>
```

Una vez hecho esto, cambiando los asteriscos por la clave de uso real obtenida de la consola de desarrolladores (no incluida por razones de seguridad) sólo tenemos que añadir los JARs que nos proporciona Dropbox con el código fuente. Para obtenerlos nos dirigimos al apartado Android en su página de desarrolladores y los descargamos (dropbox-android-sdk-1.5.4.zip).

Después los adjuntamos (*Propiedades* > *Java Build Path* > *Add External Jars* > Encontrar dropbox-android-sdk JAR en Referenced Libraries > *Propiedades* > *Java Source Attachment* > *External File* > Seleccionar el archivo anterior)

Sólo queda guardar el proyecto y ya estamos listos para interactuar con Dropbox. Para ello, se descargó el ejemplo obtenido de la página, siguiendo el tutorial y se modificó para adaptarlo a nuestras necesidades. Las funciones más interesantes a resaltar serían la clase que controla la subida de archivos a Dropbox, la clase que controla la bajada de archivos y por último la clase que controla el cifrado y maneja las demás (ver Anexo II: Código de interés).

Chooser DropIn

Para utilizar esta funcionalidad se debe descargar el fichero .zip con el código fuente de la página de desarrolladores de Dropbox [49]. Una vez obtenido esto, debe crearse una segunda clave en la consola de desarrolladores, con un nombre diferente.

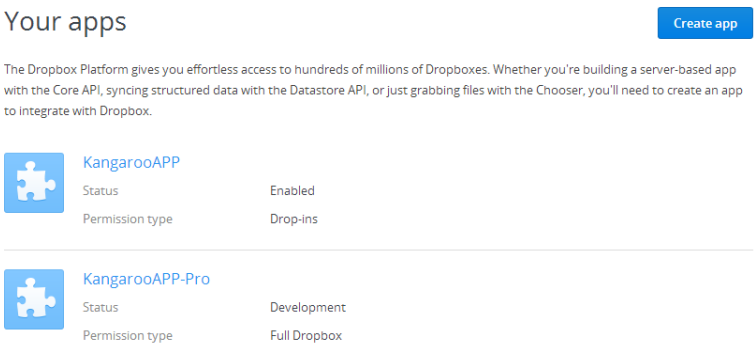

Figura 13: Creación de una clave para el DropIn de Dropbox

Tras hacer esto deben enlazarse las librerías pertinentes. Primero debe importarse el SDK en el espacio de trabajo de Eclipse. Una vez hecho esto,

referenciaremos nuestro proyecto al igual que hemos hecho con la API de Dropbox (*Propiedades* > *Android* > *Library* > *Add*-select DropboxChooserSDK).

Habiendo terminado todos los pasos previos, solamente tenemos que asociar la actividad del chooser a un botón de nuestro proyecto. Para ello, importaremos las librerías necesarias

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import com.dropbox.chooser.android.DbxChooser;
```

Después debemos configurar el tipo de link que queremos que devuelva nuestro chooser. Puede ser cualquiera de estos tres tipos

- **ResultType.PREVIEW_LINK** link que apunta a un enlace web generado por Dropbox donde descargar el archivo
- **ResultType.DIRECT_LINK** link que apunta al contenido del fichero para descargarlo directamente
- **ResultType.FILE_CONTENT** link válido para Android que apunta al fichero en la caché de descargas del teléfono.

En este caso, el link que queremos es la URI del fichero en el teléfono, que se encuentra en una cache temporal donde Dropbox descarga localmente los ficheros a los que queremos acceder. Con esta URI podremos abrir el fichero, leer sus bytes, descifrarlos y guardarlos en un nuevo fichero. Debemos hacerlo así en vez de cifrar el fichero directamente, pasándoselo a la función de cifrado, ya que la caché de archivos es controlada por el sistema operativo, sobre el cual no tenemos control alguno, lo que puede dar lugar a pérdida de datos inesperada al trabajar con el fichero directamente si el sistema operativo necesitase espacio o recursos.

Una vez configurado el chooser, lo asociamos a un botón de nuestra APP y ésta lanzará la actividad correspondiente, de la cual se encarga Dropbox, hasta el momento en que termina, cuando el usuario elige el fichero. Es en ese momento en el cual recogeremos el código con el que termina la actividad y si es correcto, pasaremos a tratar de descifrar el fichero como se ha descrito en el punto anterior (ver Anexo II: Código de interés).

El proceso termina con el fichero ya descifrado en la carpeta de descargas del teléfono, ofreciendo al usuario que lo abra con las APPs que tenga instaladas previamente en el terminal, a través de un INTENT genérico (ver Anexo II: Código de interés).

3.3 Capa de Seguridad, Android como cliente seguro

3.3.1 Java Cryptography Architecture

Antes de continuar es importante recalcar la estructura que existe a la hora de trabajar con funciones criptográficas y protocolos de seguridad en Java. Para esto hay que tener en cuenta tres elementos, la JCA (*Java Cryptography Architecture* [50], no confundir con *Java EE Connector Architecture*), la JCE (*Java Cryprography Extension*) y los proveedores de seguridad.

JCA Es la Arquitectura sobre la que se construyen las funciones criptográficas y de seguridad de nuestra aplicación, pero no es donde encontraremos la implementación de las mismas. Aquí entran en juego la JCE y los *Security Providers,* que sí definen esta implementación. Para aclarar estos conceptos utilizaremos la siguiente imagen:

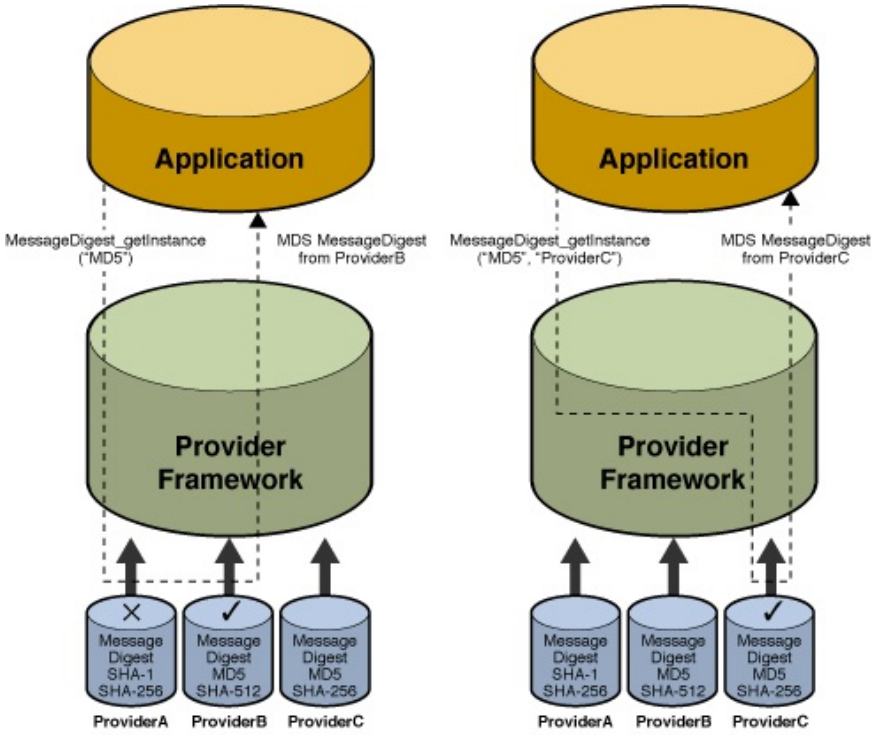

Figura 14: Java JCA

Como podemos observar en la Figura 14, una descripción gráfica de la JCA, las funciones Java son especificadas mediante protoripos por la arquitectura, mientras son los *Security Providers* y la JCE, los que realmente proporcionan la implementación de los mismos a través de APIs. Esta arquitectura permite una gran escalabilidad así como el uso de diferentes implementaciones de servicios criptográficos según convenga al desarrollador.

En nuestro caso elegimos al proveedor de seguridad más popular entre la comunidad, Bouncycastle [6]. Pero existe un problema, dado que Android no es una plataforma Java tan completa como la que podemos encontrar en un ordenador, no pudimos utilizar los servicios directamente, tuvimos que investigar hasta encontrar la existencia de su equivalente en Android, Spongycastle [51].

Como comentamos, debido a las restricciones de Android no todos los servicios ni algoritmos de cifrado, están disponibles para su uso en todas las versiones del sistema operativo. Aunque puede solucionarse con la propiedad del fichero AndroidManifest.xml en la que especificamos la versión mínima de Android para la que es compatible la APP, solucionando así el problema de compatibilidad.

Para este proyecto se ha decidido utilizar AES-256, el cifrado más utilizado en la actualidad y estándar en la industria, aconsejado por expertos como Jeff Six en su libro "*Application Security for the Android Platform*" [52] o Sheran Gunasekera, en "Android Apps Security" [53].

3.3.2 AES 256, *CBC Cipher: Block Chaining mode*

Para cifrar nuestros documentos utilizaremos el cifrado AES, específicamente su versión de tamaño de clave de 256 bits. Como comentamos en el punto anterior, este cifrado en concreto no está disponible en todos los terminales por los recursos que necesita para funcionar correctamente. Por lo tanto, para no generar incompatibilidades nos veremos forzados a cambiar la propiedad del fichero AndroidManifest.xml

```
<uses-sdk android:minSdkVersion="8"/>
```

El cifrado AES es un cifrado muy robusto, pero ¿por qué? esto se debe a que cumple los principios de difusión y confusión, es decir, CADA bit de salida depende de TODOS los bits de entrada

$$y_i = f(x_1, x_2, \dots, x_{256}) \ \forall i$$

Esto provoca que un mínimo cambio en los bits de entrada, supone un cambio total en los bits de salida, lo que anula los llamados *ataques por texto claro conocido*, como por ejemplo el ataque de cumpleaños (en el que se intercambian palabras conocidas de un texto para comparar el resultado del cifrado con las nuevas palabras e inferir la clave a través de un estudio estadístico [54])

La confusión se manifiesta haciendo que los bits de la entrada cambien en función de la clave en cada ronda (etapa AddRoundKey del cifrado). De esta manera la clave se "mezcla" con los resultados intermedios de cada ronda, lo que hace que no queden trazas de la clave en el texto, previniendo ataques diferenciales y análisis estadísticos del texto que permitan al atacante obtener la clave o reducir el espacio de búsqueda de claves de manera significativa para poder realizar un ataque por fuerza bruta.

Por último, las cajas de sustitución del AES están diseñadas para cumplir los criterios de BIC y SAC, "*Bit Independence Criterion*" y "*Strict Avalanche Criterion*", que especifican que un cambio en un bit de la entrada, hace que la probabilidad de que un bit de la salida cambie es igual a 1/2, es decir:

$$P\left(y_j = 1 \middle| \overline{x_i}\right) = P\left(y_j = 0 \middle| \overline{x_i}\right) = \frac{1}{2} \ \forall i \forall j$$

Y por otra parte, el criterio SAC, reflejado en la siguiente propiedad:

$$P\left(y_i, y_j \middle| \overline{x_k}\right) = P(y_i | \overline{x_k}) \cdot P\left(y_j \middle| \overline{x_k}\right) \ \forall i \forall j \forall k$$

Esta se traduce en que dado un cambio en un bit de la entrada, el hecho de que se produzca un cambio en dos bits de la salida cualesquiera, son sucesos independientes probabilísticamente.

El cifrado AES surge en 2001 tras la necesidad de encontrar un reemplazo para el algoritmo 3DES popular en aquella época (aunque se sigue utilizando hoy en día en algunos casos). El cifrado AES es un cifrado por bloques de tamaño fijo, que pasan por una serie de rondas, diseñadas para cumplir los criterios anteriormente explicados y obtener así el texto cifrado.

Por dentro sigue el siguiente algoritmo:

- Expansión de la clave
- Etapa inicial

  1. AddRoundKey — se combina cada byte con la clave con operación XOR

- Rondas:

  1. SubBytes — se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla.*
  2. ShiftRows — se realiza una transposición donde cada fila es rotada de manera cíclica un número determinado de veces.**
  3. MixColumns — operación de mezclado que consiste en combinar los cuatro bytes en cada columna usando una transformación lineal.
  4. AddRoundKey — cada byte es combinado con la clave «round»; cada clave «round» se deriva de la clave de cifrado usando una iteración.***

- Etapa final:

1. SubBytes*
2. ShiftRows**
3. AddRoundKey***

A día de hoy no se conocen ataques contra este cifrado, exceptuando el ataque teórico "ataque XSL", de Nicolás Courtois y Josef Pieprzyk, hoy, puramente teórico [55]. No obstante, los expertos opinan que la estructura matemática tan ordenada de este cifrado, diferente de otros, puede originar futuras vulnerabilidades [56]–[58].

La implementación que se ha seguido en Java se puede ver en Anexo II: Código de interés. Como puede observarse en el código, el modo de cifrado que hemos elegido es CBC, *Cipher Block Chaining*, que no es el modo por defecto (éste es el ECB, *Electronic Code Book*), ya que como puede observarse en la figura adjunta, no es un modo del todo seguro a la hora de cifrar, ya que permite ataques estadísticos y diferenciales.

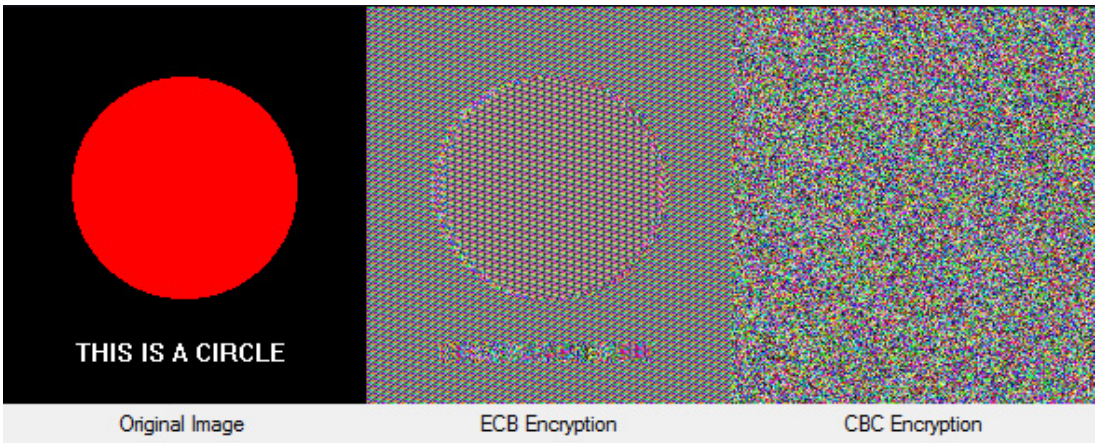

Figura 15: Modo de cifrado ECB vs CBC

Este modo funciona de la siguiente manera:

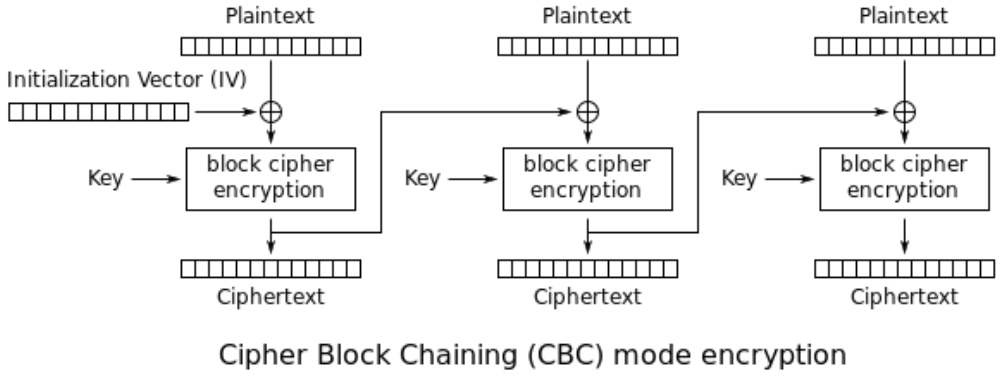

Figura 16: Cifrado con el modo CBC

Como puede verse en la figura superior, en este modo de cifrado, a la salida de cada bloque cifrado, se le hace un XOR con la entrada del siguiente, lo que produce un cifrado robusto, nuestro objetivo prioritario.

La operación comienza con la elección de un vector de inicialización (IV) que debe ser elegido de forma aleatoria, ya que si esto no es así, se pueden dar de nuevo ataques de texto claro conocido contra el texto cifrado (éstos se producen al determinar una o varias palabras frecuentes en el idioma del texto cifrado, haciendo un análisis estadístico, por ejemplo, en castellano tendríamos palabras como de, que, el, la, etc. Una vez hecho esto, las cifraríamos con nuestro algoritmo, y si generaran coincidencias con el texto cifrado, descifraríamos el resto del texto con la clave hallada, si este tuviera sentido, ya habríamos conseguido atacarlo, en caso contrario, se proseguiría con la siguiente palabra, hasta dar con la clave adecuada).

Para el descifrado se sigue el siguiente proceso [44], [58]:

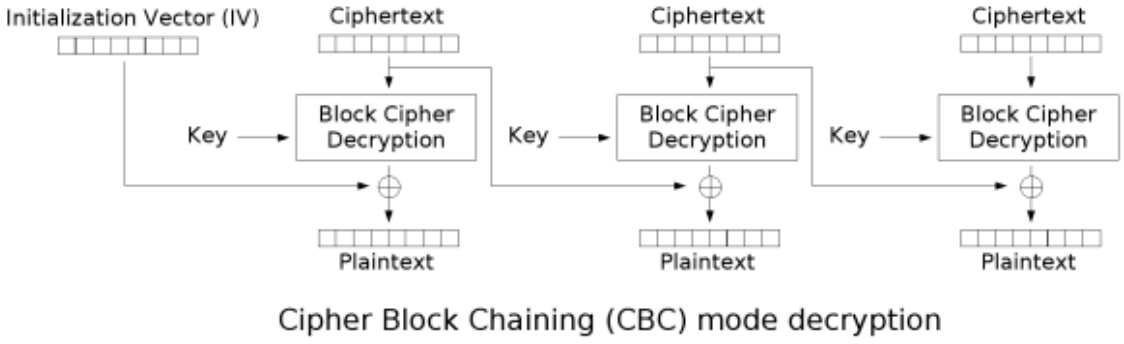

Figura 17: Descifrado con el modo CBC

3.3.3 Login

Como comentamos anteriormente es necesario guardar los datos sensibles del usuario de forma cifrada, por lo que hemos incluido una función de HASH en el login, tanto a la hora de crear nuevos usuarios como a la hora de hacer la comprobación en el login mismo (ver Anexo II: Código de interés).

3.3.4 SSL

Otra medida de seguridad utilizada en el proyecto es SSL. En las comunicaciones entre la APP y Dropbox, por defecto en los servicios que ofrece Dropbox, todas las comunicaciones funcionan sobre la capa de conexión segura SSL, que implementan una serie de protocolos criptográficos para establecer comunicaciones seguras a través de internet. Toda la conexión es controlada por Dropbox, de forma transparente para el desarrollador.

3.3.5 Servidor de distribución de claves

En este trabajo, y dado que no era competencia del mismo, tal cual está configurado el cifrado no somos capaces de descifrar los documentos que subimos sin nuestra clave de usuario, por lo que seríamos incapaces de compartir estos documentos con otras personas sin distribuir nuestra clave. Por ello, y al no ser prioritario en este trabajo, damos por hecho que para que el sistema permitiera compartición de documentos debe existir la implementación de un servidor de gestión de claves tipo Kerberos o un sistema parecido, que permita autenticar externamente a los usuarios de nuestra aplicación a través de tokens de duración limitada, y les permitiera acceder a una base de datos externa donde se encontrarían todas las claves de sesión, cifradas para que no puedan obtenerse fácilmente, para poder descifrar estos documentos sin distribuir nuestra clave. Para no implementar este tipo de solución podemos ver otra solución en el punto 4.2.2 Firma de documentos.

Capítulo 4

Conclusiones y trabajo futuro

A continuación presentaremos el resultado de nuestro proyecto a través de una serie de capturas de pantalla y terminaremos con las conclusiones del mismo.

4.1 Demo de la APP

El usuario abre la APP por primera vez. Al no tener un usuario asignado, se dirige a la pantalla de creación de usuarios a través del botón *New User* (Figura 18).

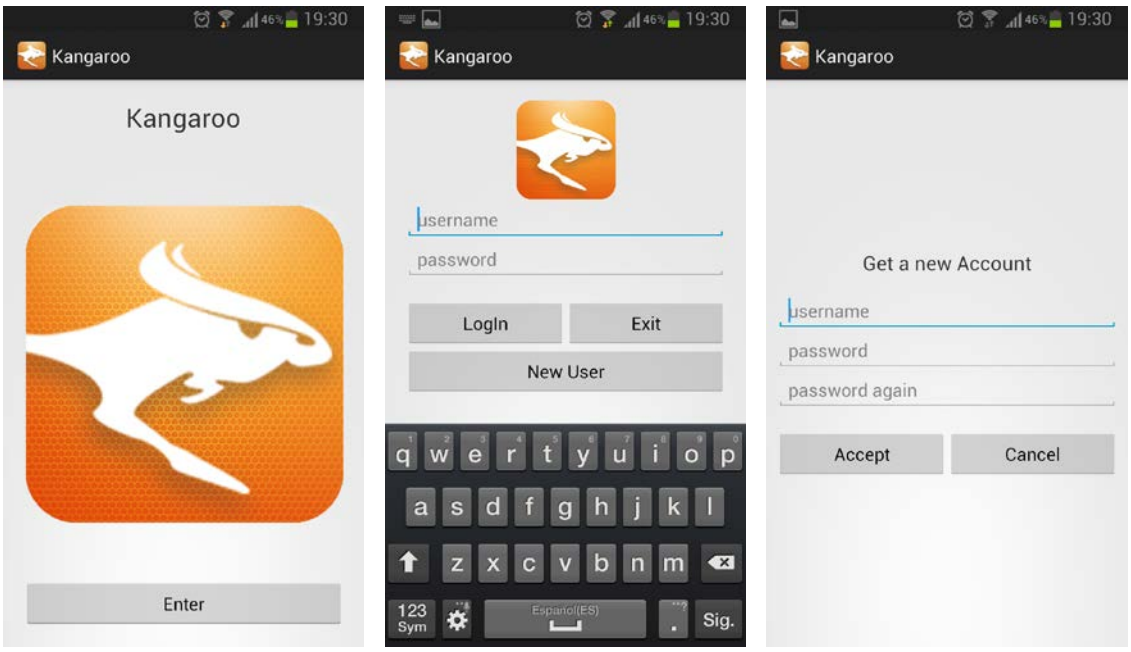

Figura 18: Creación de una cuenta de usuario

Se comprobará que escribe una *password* correcta, la misma en ambos campos y se procederá a agregar el usuario en nuestra base de datos.

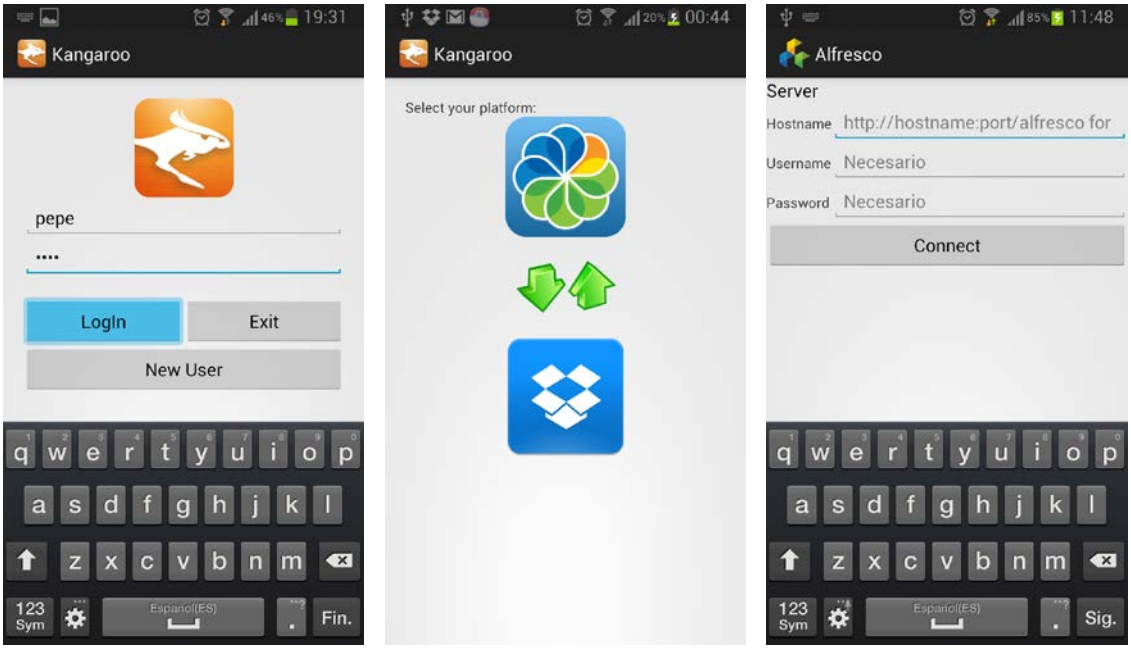

Figura 19: Pantalla de elección de plataforma

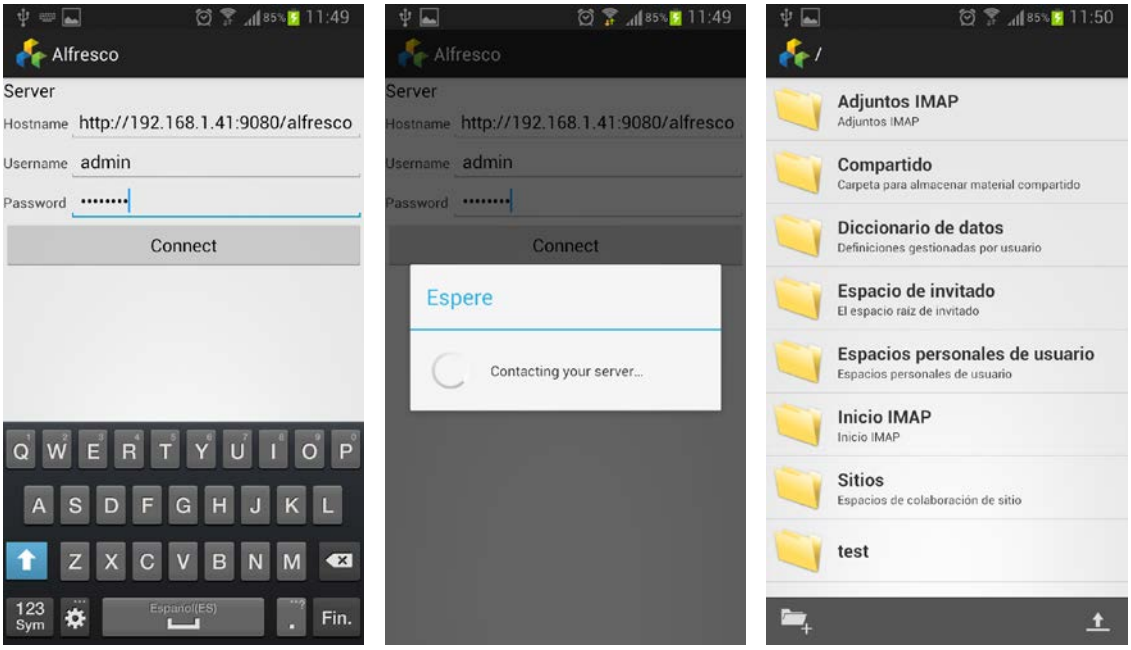

Figura 20: Pantalla de acceso a Alfresco

Una vez creado el usuario, podremos acceder desde la pantalla de *login*. Esto nos lleva a una pantalla de selección de plataforma donde podremos elegir entre el entorno de explotación o el de producción (Figura 19). Eligiendo el entorno de explotación, podremos acceder a una pantalla donde escribiremos los datos de nuestro servidor Alfresco para acceder al mismo (Figura 20). Si la autenticación es correcta, accederemos al árbol de directorios. Para esta prueba se ha creado la carpeta 'test' desde el icono que se encuentra en la parte inferior izquierda de la pantalla, la carpeta con el símbolo '+'.

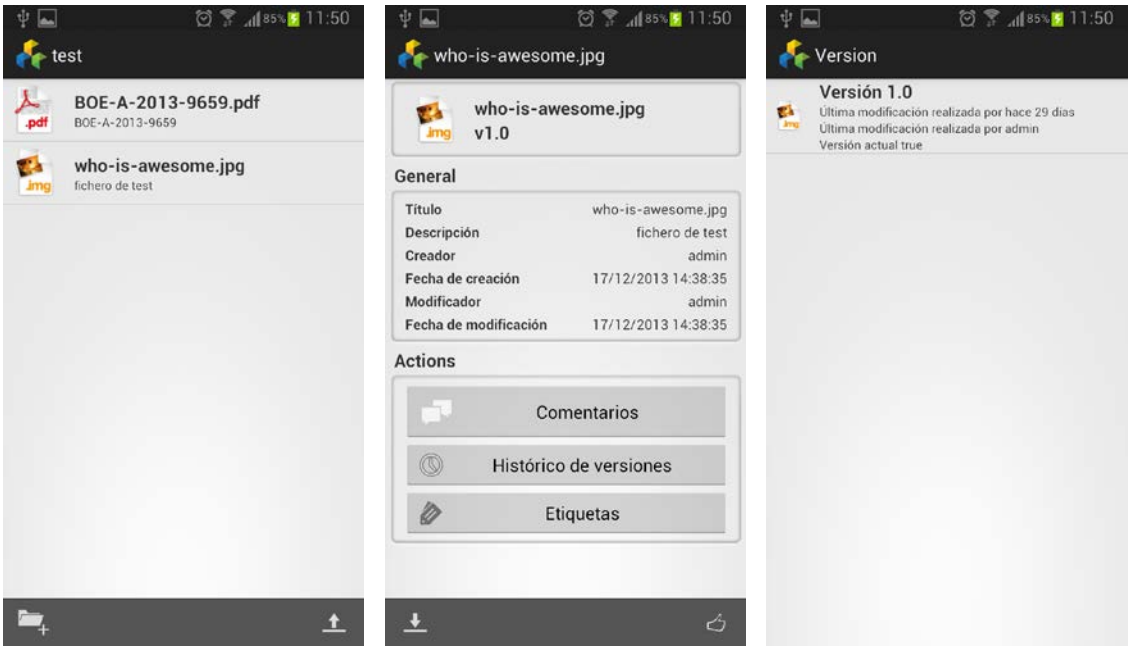

Figura 21: Pantalla de control de versiones en Alfresco

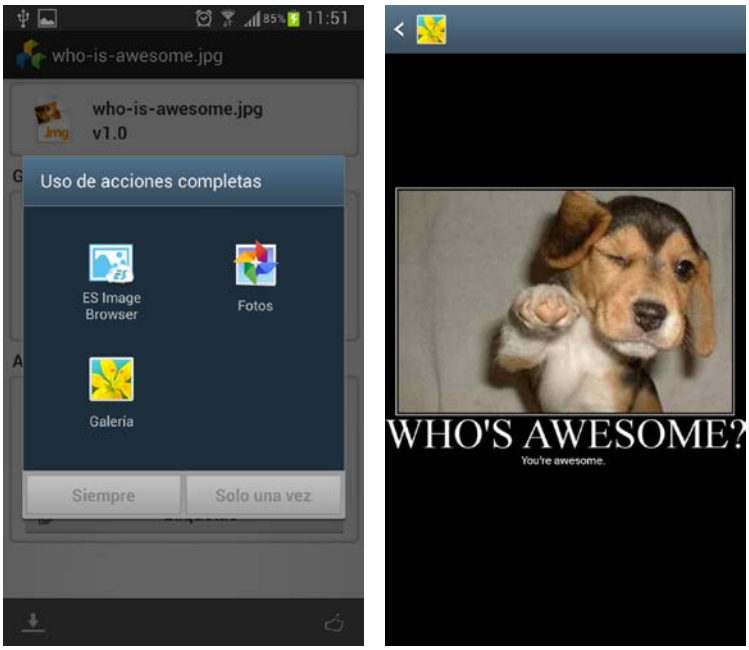

Figura 22: Pantalla descarga/apertura de archivos en Alfresco

Una vez creada esta carpeta probaremos a subir un par de archivos a través del icono de subida, la flecha apuntando hacia arriba de la esquina inferior derecha de la pantalla. Si accedemos a cualquiera de ellos, podremos ver las versiones de cada archivo (Figura 21). Para acceder al archivo, sólo hace falta pinchar en el icono de bajada, la flecha apuntando hacia abajo de la esquina inferior izquierda de la pantalla. Una vez terminada la descarga, se nos ofrecerá la posibilidad de abrir el archivo con los programas disponibles en el terminal, para este caso hemos elegido la foto de prueba que podemos ver en la última captura de pantalla (Figura 22).

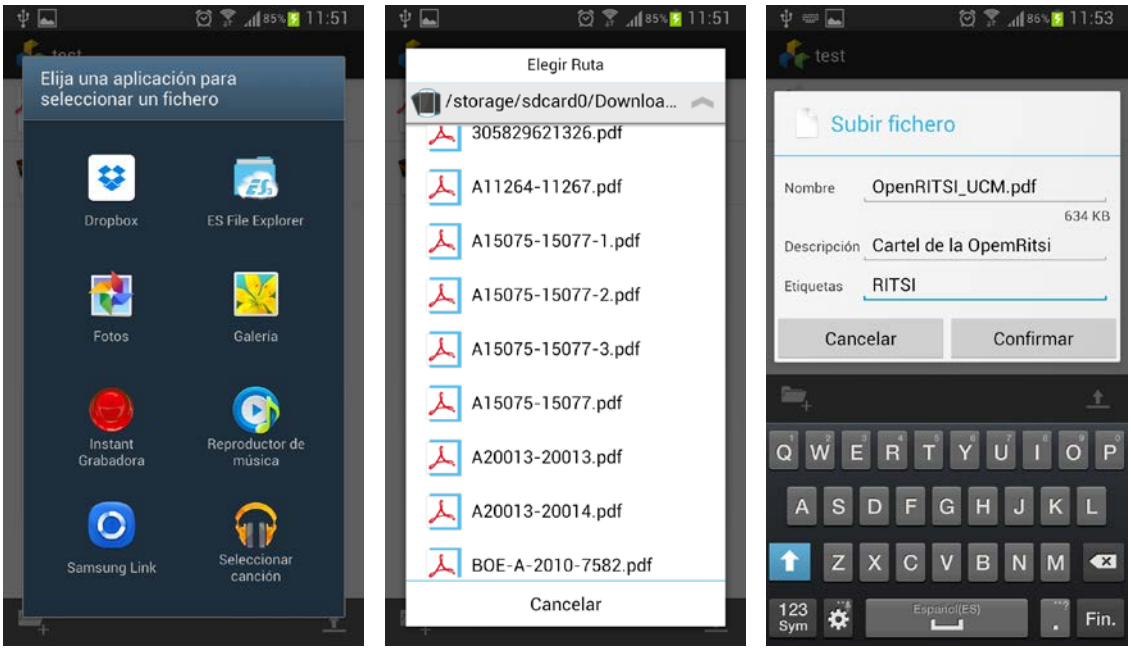

Figura 23: Pantalla de subida de documentos en Alfresco

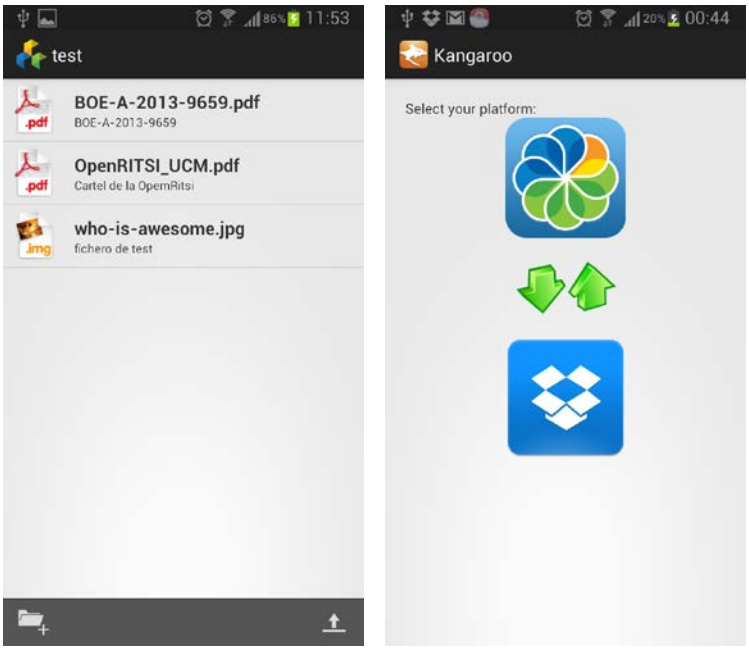

Figura 24: Pantalla de archivos una vez terminada la subida

Esta vez subiremos un archivo. Para ello, pulsaremos en la tecla inferior de subida (la flecha que apunta hacia arriba en la esquina inferior derecha, ver Figura 21, primera imagen). Un menú para seleccionar el archivo con una APP pre-instalada en el terminal aparecerá para que seleccionemos el que más nos convenga. Para este caso utilizaremos el gestor de ficheros. A través de él seleccionaremos un pdf, por ejemplo. Revisaremos los detalles y si son correctos terminaremos el proceso (Figura 23). Si volvemos a la carpeta de test podemos comprobar que efectivamente el archivo se ha subido correctamente. Volvemos a la pantalla de decisión para el siguiente escenario (Figura 24).

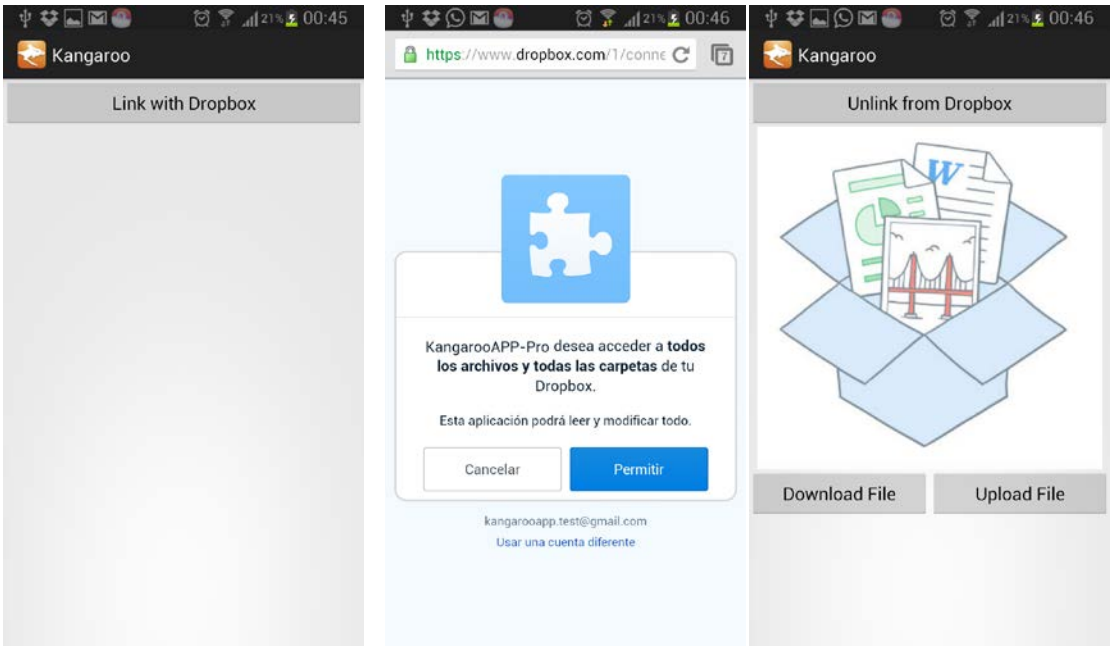

Figura 25: Pantalla de enlace a Dropbox

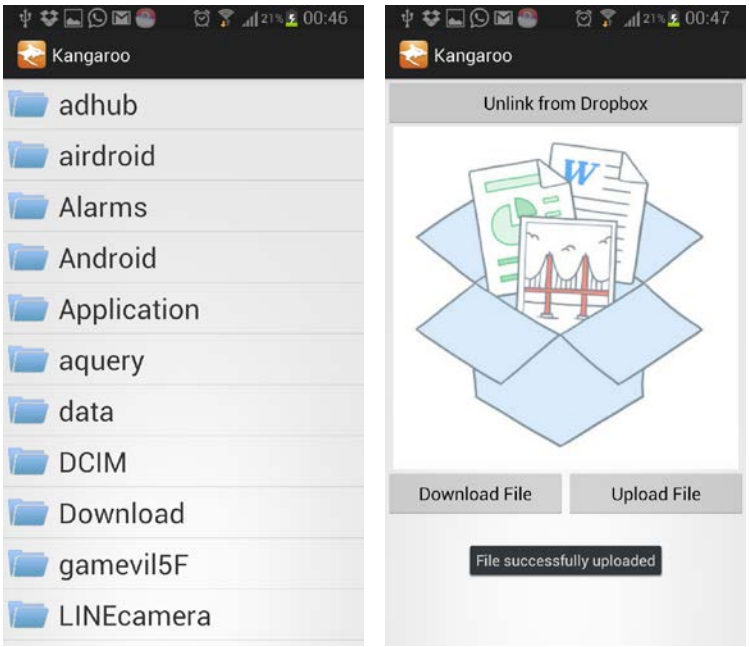

Figura 26: Pantalla de subida de archivos a Dropbox

Esta vez elegiremos el entorno de producción. La primera vez que accedamos al servicio, nos pedirá que enlacemos nuestra cuenta de usuario con una cuenta de Dropbox (Figura 25). Terminado este proceso, podremos empezar a utilizar la APP. Para ello empezaremos con la subida de archivos. Al pulsar el botón 'Upload File', un gestor de ficheros nos ofrecerá la posibilidad de elegir el fichero deseado (Figura 26). Tras hacer esto, la APP cifrará el archivo y lo subirá a la cuenta de Dropbox de forma transparente, como se describe en la memoria.

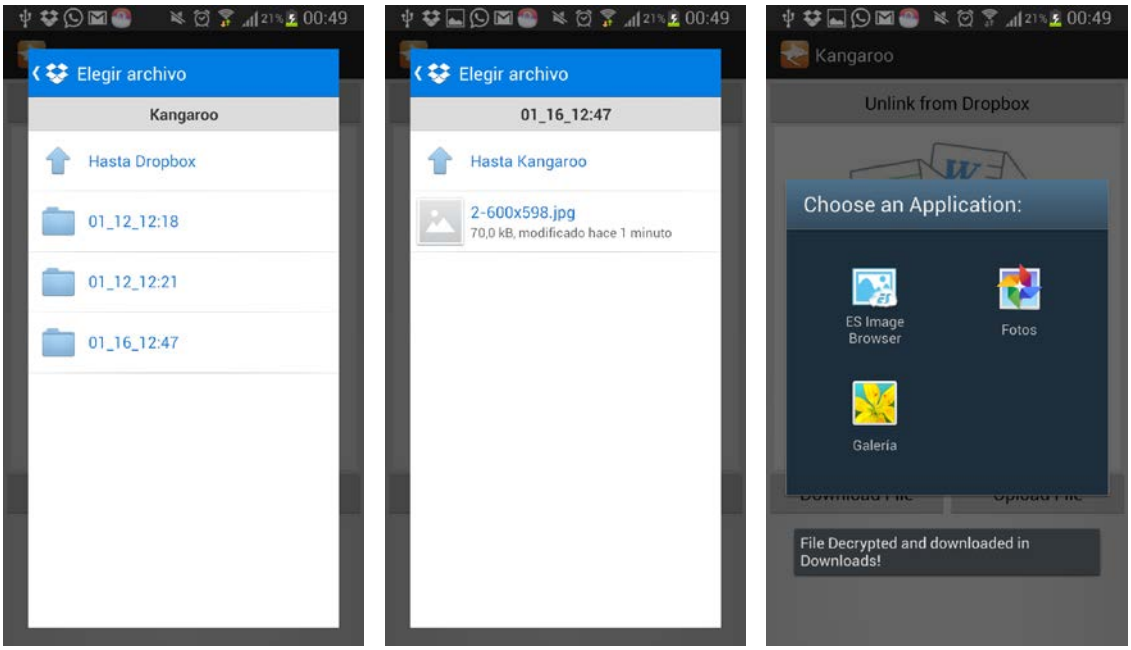

Figura 27: Pantalla de descarga de archivos de Dropbox

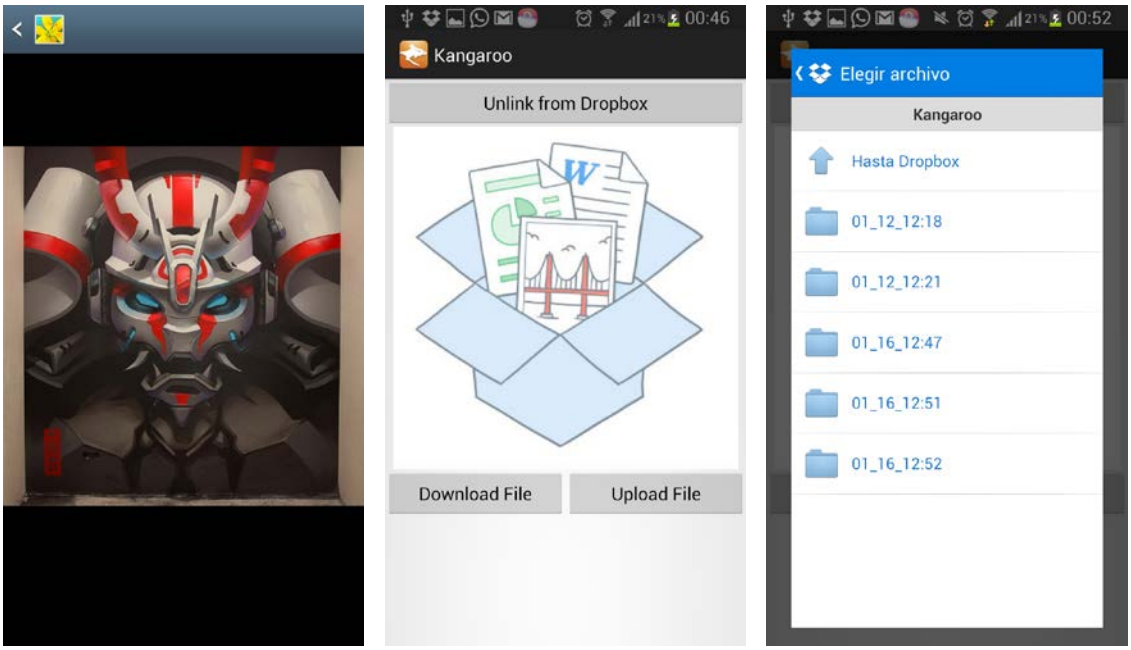

Figura 28: Pantalla de descarga de archivos II

Ahora vamos a tratar de descargar el archivo. Para ello, pulsaremos el botón 'Download File' que nos mostrará un gestor de ficheros de nuestra cuenta de Dropbox (Figura 27). Tras localizar el fichero a través del nombre de la carpeta (fecha y hora de la subida del mismo, como se puede observar en la hora de las capturas de pantalla anteriores), podremos descargarlo pulsando sobre él. Una vez finalizada la descarga, nos dará la opción de nuevo de abrirlo. Por último mostramos una bajada de otro tipo de archivo que no sea una imagen, en el ejemplo un documento en formato .docx (Figura 28).

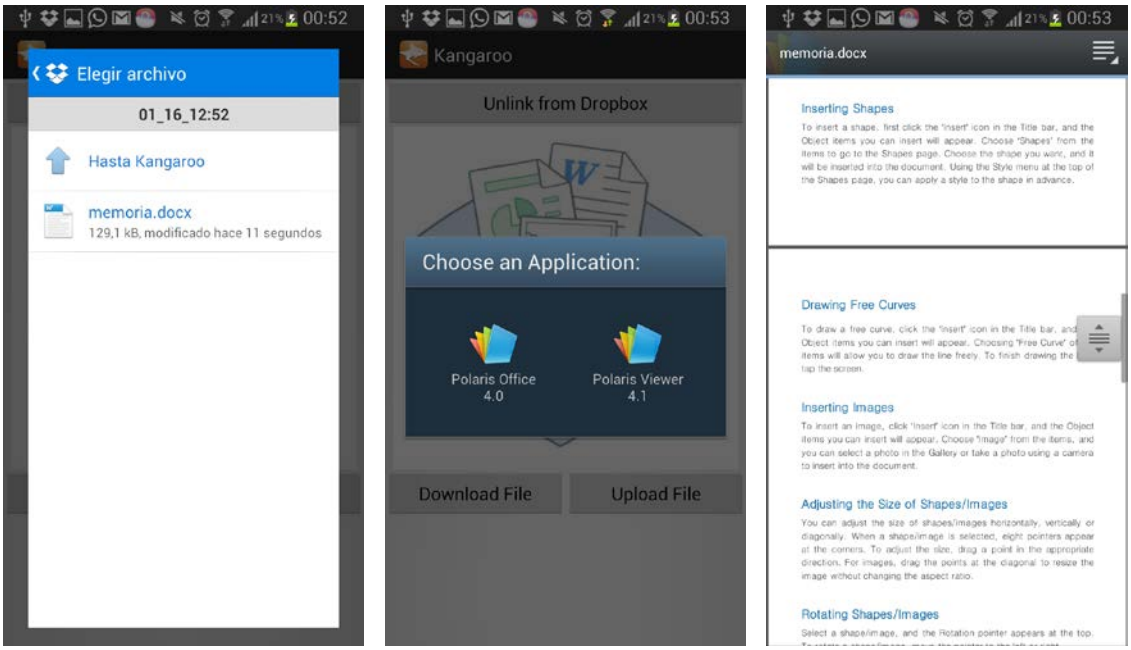

Figura 29: Pantalla de apertura de archivos

Como podemos observar el proceso es el mismo, únicamente cambia la manera de abrir el fichero en el dispositivo (Figura 29).

4.2 Trabajo futuro

La seguridad es algo que crece y se actualiza día a día por lo que la continuación y el mantenimiento de este proyecto debe continuarse a lo largo de la vida del proyecto. No obstante, planteamos unas líneas de trabajo futuro que consideramos realmente importantes y útiles para el mismo.

4.2.1 The Guardian Poject's SQLCipher: Encrypted Database

Consideramos realmente interesante la propuesta de "The Guardian Project" [59] entusiastas de la seguridad en Android que han puesto al servicio de los usuarios y desarrolladores toda una suite de software de seguridad realmente útil como por ejemplo una API para cifrar discos duros o el tráfico de red o una APP que consiste en un navegador privado sobre Tor.

A pesar del enorme interés de los recursos que pone a nuestra disposición, sin duda el más interesante para este proyecto sería su SQLCipher: Encrypted Database, con el que podríamos cifrar toda nuestra base de datos, haciéndola aún más segura, permitiéndonos incluso desarrollar nueva funcionalidad [60].

4.2.2 Envoltura digital

Otro punto importante en el trabajo es que, como hablamos anteriormente, no somos capaces de distribuir documentos sin distribuir la clave con la que se han cifrado, nuestra clave privada. Para mitigar este problema podríamos utilizar la criptografía asimétrica, con algoritmos como el RSA.

Basándonos en este tipo de algoritmos y utilizando de nuevo como base la norma ISO-27001, podemos a través de la tríada Confidencialidad-Integridad-Disponibilidad (CID) plantear el uso de la técnica de envoltura digital para

abordar este problema. Dotaremos así al proyecto de aún más ventajas. Por un lado estaríamos cerrando el triángulo CID, ya que a la confidencialidad y disponibilidad ya ofertadas, añadiríamos la integridad de los activos, debido a la posibilidad de comprobar la integridad de los mismos gracias a los mecanismos de resumen que esta técnica implementa, permitiéndonos detectar errores en la transmisión de los mismos o pérdidas de datos. Por otro lado, ofreceríamos otra funcionalidad, la posibilidad de averiguar sin error quién es la persona autora del documento, así como si éste ha sido modificado desde que se firmó.

Con esta solución utilizaríamos la tecnología de empaquetado PKCS7, que consiste en tomar el activo correspondiente y una vez cifrado, cifrar la clave de sesión con la que se ha cifrado el documento, con la clave pública de la persona o personas a las que va dirigida, e incluirlo justo a continuación. Tras terminar este proceso, firmaríamos el mensaje completo para demostrar que lo enviamos nosotros, cifrando con nuestra clave privada el resumen del texto.

A la hora de distribuir el activo, si no tuviéramos este implementación de envoltura digital nos encontraríamos con el siguiente problema:

Figura 30: Distribución de mensajes cifrados sin envoltura digital*

Tendríamos que replicar el mensaje una vez por cada destinatario, dado que tendríamos que firmar un mensaje con cada clave pública de cada usuario al que va dirigido. No obstante con la implementación de la envoltura digital:

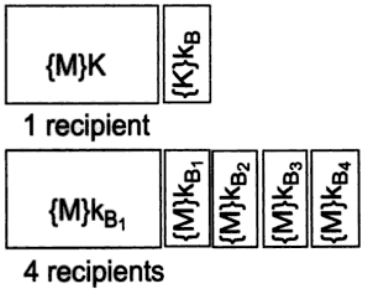

Figura 31: Distribución de mensajes cifrados con envoltura digital*

Añadiríamos la sección con la clave cifrada tantas veces como destinatarios haya, sin tener que replicar el mensaje. ***(Figuras extraídas de** [61]**)**.

De esta forma, a partir de la metodología de la criptografía asimétrica y del empaquetado PKCS7, que nos permitiría separar el texto cifrado, de la clave en sí, podríamos ser capaces de compartir, a través de certificados digitales que respaldarían nuestras identidades, los documentos con otras personas.

Además estaríamos ofrecemos otra funcionalidad, asegurar que estos documentos provienen de nosotros, y que no han sido modificados desde que se firmaron, gracias a la inclusión de la firma del resumen del texto al final del mensaje, imposible de modificar ya que está firmado con nuestra clave privada.

Para generar los certificados con las firmas convenientemente utilizaríamos la herramienta openssl [62] a través de la terminal de linux. El proceso sería el siguiente:

1. Generar nuestro par de clave pública y privada

```
$ openssl genrsa -aes256 -out kangaroo.key
```

2. Generar el certificado x509 para demostrar nuestra identidad

```
$ openssl req -x509 -sha256 -new -key kangaroo.key -out kangaroo.csr
```

3. Generar un certificado firmado por nosotros útil durante largo tiempo

```
$ openssl x509 -sha256 -days 3652 -in kangaroo.csr -signkey kangaroo.key -out selfsigned.crt
```

4. Crear un almacén de claves PKCS12 para importar utilizar desde Java

```
$ openssl pkcs12 -export -name myKangaroocert -in selfsigned.crt -inkey kangaroo.key -out keystore.p12
```

5. Convertir el almacén de claves en un formato válido para la JCE (JKS)

```
$ keytool -importkeystore -destkeystore mykeystore.jks -srckeystore keystore.p12 -srcstoretype pkcs12 -alias myservercert
```

Al terminar el último pasado obtendríamos un almacén de claves con nuestra identidad digital, así como nuestro par de claves públicas y privadas, lista para implementar la tecnología de envoltura digital PKCS7 en nuestro código.

4.3 Adquisición de conocimientos y valor añadido

Este trabajo ha sido un punto de inflexión para mí. Debido a su envergadura y su complejidad. Esto sumado al hecho de que he tenido que compaginar la realización de las prácticas en empresa con el desarrollo del mismo, ha supuesto para mí un importante esfuerzo que me ha hecho madurar en muchos sentidos.

Me ha servido por un lado, para afianzar varios conceptos de la carrera y utilizarlos en un caso práctico, en un proyecto de una envergadura mayor a la acostumbrada durante las prácticas desarrolladas a lo largo de estos años, un proyecto en el que he tenido que sacar a la luz todo mi potencial para poder hacer de él algo que valga la pena, siempre con la ayuda de mi tutor, que ha sabido guiarme en la realización del mismo.

Respecto a la adquisición de conocimientos me gustaría resaltar los siguientes puntos:

- Búsqueda y filtrado de información relevante para el proyecto
- Profundización en la tecnología Android
- Profundización en el uso de la criptografía y las buenas praxis
- Gestión de servidores Alfresco, muy demandados en algunas empresas
- Programación de funcionalidad a través de APIs complejas
- Git como tecnología de control de versiones.

Eso respecto a las tecnologías, no siendo esto lo único que creo que he aprendido. He aprendido también a organizar mi tiempo y a desenvolverme por mí mismo, siendo capaz de presentar un proyecto actual y útil, que espero que el tribunal considere igual de interesante de lo que lo considero yo, permitiéndome terminar mi carrera y obtener mi título universitario.

Probablemente, si tengo que quedarme con lo más importante que he aprendido de este proyecto, sería el conocimiento que he obtenido a la hora de redactar textos científicos serios de mano de mi tutor, sin el que no habría sido capaz de llegar a presentar una memoria tan rica en contenido. Él me ha enseñado las herramientas que podía utilizar, así como me ha sugerido la estructura que podía seguir, guiándome a través de la realización del mismo, esperando pacientemente a que lo realizara por mí mismo para que aprendiera. Valoro enormemente todo esto.

Como valor añadido al proyecto me gustaría mucho resaltar tres aspectos fundamentales:

- *OpenSource* Software

A lo largo de todo el proyecto, las tecnologías *OpenSource* han sentado las bases del mismo. Como ya hemos comentado con anterioridad, es fundamental que en un proyecto realizado en la comunidad educativa, todos podamos aprender de él, por lo que no tendría sentido ocultar absolutamente nada del mismo.

- Valor práctico y actual

Hemos intentado en todo momento dotar al proyecto de un valor práctico, utilizando tecnologías en auge o muy novedosas, así como temas y un enfoque del proyecto actuales, que haga de este proyecto algo más que un simple *trabajo universitario* y no deje indiferente al lector.

- Utilidad

Por último expresar nuestra idea de hacer de este proyecto un producto final, que permitiera, hipotéticamente como se expone, desarrollar incluso una idea de negocio basada en él. Un producto útil que pudiera encontrar un hueco en el mercado actual.

Glosario

ADB: *Android Debug Bridge*, utilidad entregada por Google para programar en Android. Básica para cualquier desarrollador de esta plataforma, permite comunicarse plenamente con un dispositivo, o una máquina virtual del mismo, ofreciendo funcionalidad como la instalación de paquetes software, la lectura de bases de datos o la extracción de los mismos [63].

AES: *Advanced Encription Standard,* algoritmo de cifrado comúnmente utilizado en la industria debido a su fortaleza contra ataques criptográficos (a día de hoy no se conocen ataques con éxito hacia el mismo). Basado en el algoritmo Rijndael, es un cifrado por bloques de 128 bits, que utiliza claves de tamaño 128, 192 ó 256 bits. Se convierte en el cifrado estándar del gobierno de los Estados Unidos en 2001 [43].

APK: *Aplication Package File,* extensión de los paquetes de software destinados al sistema operativo Android. Equivalente a la extensión JAR de Java, se utiliza para instalar y distribuir software en Android [64].

Ataque Man in the middle: traducido literalmente *ataque del hombre en el medio*, describe un tipo de ataque de seguridad en el que un intermediario se coloca entre medias de otras dos, con el fin de interceptar sus comunicaciones para utilizarlas a su favor [65].

Crowdfunding: modelo de financiación muy popular en el mundo emprendedor en estos momentos. Consiste en plantear de forma generalizada, sin ninguna limitación a ningún tipo de inversor, sea grande o pequeño, un proyecto que le pueda parecer interesante, normalmente a través de internet para que lo apoyen económicamente. De esta forma se puede conseguir una gran cantidad de capital, ya que en vez de buscar un gran inversor se buscan muchos más pequeños, consiguiendo sacar adelante proyectos arriesgados, en los que un sólo inversor jamás habría participado.

Gestión documental: conjunto de normas técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permitir la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que ya no sirven y asegurar la conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía.

Hash: función matemática de un único sentido, que suele tener un bajo coste computacional y que se utiliza para a partir de una entrada, generar una salida única, que represente un resumen de la entrada. En criptografía permiten la implementación de funcionalidad tal como garantizar si un mensaje ha sido modificado o no desde que se envió o autenticar sin error quién ha sido el emisor de un mensaje concreto [66].

KickStart/KickStarter: [67] portal de internet que se ha convertido en referencia para todos los emprendedores que deseen lanzar al mercado un proyecto a través del Crowdfunding, hasta el punto de acuñar este término. En este portal pueden encontrarse campañas para financiar desde videojuegos

hasta películas, novelas gráficas, etc. Actualmente por temas legislativos sólo se pueden lanzar campañas desde Norteamérica, por lo que poco a poco surgen otras alternativas como IndieGoGo [68].

OpenSource: término que define software que puede ser libremente utilizado, cambiado, compartido por cualquiera. Es software hecho por la comunidad y que se distribuye bajo licencias acordes a esta filosofía [69].

Outsourcing: proceso económico en el cual una empresa destina los recursos orientados a cumplir tareas hacia una empresa externa por medio de un contrato [70].

Plugin: una aplicación, normalmente pequeña, que se relaciona con otra, normalmente más grande, para aportarle una funcionalidad nueva y/o muy específica. También se conoce como *plug-in*, *add-on*, conector o extensión. Viene del verbo transitivo inglés *plug in,* que significa conectar, haciendo ilusión a este hecho.

Repositorio: (del latín *repositorĭum,* armario, alacena) se define por la RAE como el lugar donde se guarda algo. En nuestro caso, utilizaremos la palabra repositorio para definir los lugares o carpetas donde guardaremos nuestros documentos [71].

Sistema Operativo: (abreviado SO/SSOO en español y *OS* en inglés) programa o conjunto de programas utilizados para gestionar los recursos hardware de un dispositivo.

Software Libre: es el software que respeta la libertad de los usuarios y la comunidad. En grandes líneas, significa que **los usuarios tienen la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software**. Es decir, el software libre es una cuestión de libertad, no de precio. Para entender el concepto, piense en libre como en libre expresión, no como en barra libre [72].

Streaming: técnica de distribución de contenidos multimedia bajo demanda a través de la red.

Troyano: (proveniente del caballo de madera que utilizaron los griegos para infiltrarse en Troya y capturarla) software malicioso que busca ser instalado en el dispositivo objetivo sin que su usuario sea consciente, ya sea en secreto o camuflado en otro tipo de actividades, con aspecto lícito. Este tipo de malware no es auto-propagable, fundamental diferencia entre ellos y los virus o gusanos. Generalmente se utilizan en el robo de información sensible, ya sean credenciales bancarias, cuentas de redes sociales, etc [73].

Bibliografía

[1] “ISO27000.es - El portal de ISO 27001 en español. Gestión de Seguridad de la Información,” 2013. [Online]. Available: http://www.iso27000.es/enlaces.html.

[2] J. Pastor, “Android, absoluto protagonista en España con un 90% de cuota de mercado,” *Xataka*, 2013. [Online]. Available: http://www.xatakamovil.com/mercado/android-absoluto-protagonista-en-espana-con-un-90-de-cuota-de-mercado.

[3] “The Non-Security of Secrecy.” [Online]. Available: https://www.schneier.com/essay-056.html.

[4] “Open Source Cryptography,” 1999. [Online]. Available: https://www.schneier.com/crypto-gram-9909.html#OpenSourceandSecurity.

[5] S. Foresti, “Preserving Privacy in Data Outsourcing,” 2007.

[6] “bouncycastle.org,” 2013. [Online]. Available: http://www.bouncycastle.org/.

[7] “Gaikai.com :: About,” 2013. [Online]. Available: http://www.gaikai.com/about.

[8] W. Isaacson, *Steve Jobs*. Simon & Schuster, 2011, p. 656.

[9] “Red Hat | The World’s Open Source Leader,” 2013. [Online]. Available: http://es.redhat.com/.

[10] “SO Linux | SUSE Linux Enterprise,” 2013. [Online]. Available: https://www.suse.com/es-es/.

[11] “Elive - Intuitive and Powerful Operating System,” 2013. [Online]. Available: http://www.elivecd.org/.

[12] “GNUes,” 2013. [Online]. Available: http://www.es.gnu.org/Página_Principal.

[13] “Linus Torvalds biography by The Linux Information Project,” 2013. [Online]. Available: http://www.linfo.org/linus.html.

[14] “Minix3,” 2013. [Online]. Available: http://www.minix3.org/index.html.

[15] “Andrew S. Tanenbaum, Professor at the Vrije Universiteit,” 2013. [Online]. Available: http://www.cs.vu.nl/~ast/.

[16] “Stallman: ‘Android no es realmente libre’,” 2011. [Online]. Available: http://www.muylinux.com/2011/09/20/stallman-android-no-es-realmente-libre.

[17] J. Constine, “Dropbox Now Has 175 Million Users, Up From 100M In November 2012,” *TechCruch*, 2013. [Online]. Available: http://techcrunch.com/2013/07/09/dropbox-dbx-conference/.

[18] J. M. Revilla, “SkyDrive alcanza los 250 millones de usuarios,” *itespresso*, 2013. [Online]. Available: http://www.itespresso.es/skydrive-250-millones-usuarios-110920.html.

[19] “Cloud storage shoot-out: Google Drive vs. Dropbox vs. SkyDrive vs. Box,” 2013. [Online]. Available: http://royal.pingdom.com/2012/06/21/cloud-storage-shoot-out-google-drive-vs-dropbox-vs-skydrive-vs-box-com/.

[20] Oasis, *The CMIS v1.0 OASIS Standard Specification*. 2010.

[21] M. Sawh, “Best cheap mobile phones to buy 2013 - Motorola Moto G - Trusted Reviews,” *trustedreviews*, 2013. [Online]. Available: http://www.trustedreviews.com/best-cheap-mobile-phones_round-up_Page-5.

[22] “Android SDK | Android Developers,” *developer.android*, 2013. [Online]. Available: http://developer.android.com/sdk/index.html.

[23] Gartner, “Gartner Says Smartphone Sales Grew 46.5 Percent in Second Quarter of 2013 and Exceeded Feature Phone Sales for First Time,” *Gartner*, 2013. [Online]. Available: http://www.gartner.com/newsroom/id/2573415.

[24] “Q3 2013: Android Hit 81.0% Smartphone Share, iOS Fell to 12.9%,” 2013. [Online]. Available: http://thenextweb.com/insider/2013/11/12/idc-android-hit-81-0-smartphone-share-q3-2013-ios-fell-12-9-windows-phone-took-3-6-blackberry-1-7/#!qhnrk.

[25] Statista, “Samsung blew Apple away in Q1 smartphone sales - Smarter Investing,” *Smarter Investing*, 2013. [Online]. Available: http://investing.covestor.com/2013/04/samsung-blew-apple-away-in-q1-smartphone-sales.

[26] “Samsung ha copiado a Apple, el jurado condena a los coreanos en la guerra de patentes,” *Xataka*, 2012. [Online]. Available: http://www.xataka.com/moviles/samsung-ha-copiado-a-apple-el-jurado-condena-a-los-coreanos-en-la-guerra-de-patentes.

[27] “Samsung consigue bloquear la venta de varios modelos de iPhone e iPad en EEUU,” *Xataka*, 2013. [Online]. Available:

http://www.xataka.com/moviles/samsung-consigue-bloquear-la-venta-de-varios-modelos-de-iphone-e-ipad-en-eeuu.

[28] “Android Dominates Market Share, But Apple Makes All The Money - Forbes,” *Forbes*, 2013. [Online]. Available: http://www.forbes.com/sites/tonybradley/2013/11/15/android-dominates-market-share-but-apple-makes-all-the-money/.

[29] “Tarifas de transacciones - Ayuda de Android Developer,” 2013. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/112622?hl=es.

[30] “Distribute your App - iOS Developer Program - Apple Developer,” 2013. [Online]. Available: https://developer.apple.com/programs/ios/distribute.html.

[31] “Flurry,” 2013. [Online]. Available: http://www.flurry.com/flurry-analytics.html#.

[32] L. Adkinson-orellana, D. A. Rodríguez-silva, F. J. González-castaño, and D. González-martínez, “SHARING SECURE DOCUMENTS IN THE CLOUD A Secure Layer for Google Docs,” pp. 439–444, 2010.

[33] “The Big Four Banking Trojans Kaspersky,” 2013. [Online]. Available: http://blog.kaspersky.com/the-big-four-banking-trojans/.

[34] “The Biggest Identity Theft Case in U.S. History,” 2013. [Online]. Available: http://www.trutv.com/library/crime/gangsters_outlaws/outlaws/major_heists/5.html.

[35] J. Pastor, “Hacking gubernamental: La NSA y el CGHQ descifran protocolos seguros para espiarlo todo en Internet,” *Xataka*, 2013. [Online]. Available: http://www.xataka.com/otros/hacking-gubernamental-la-nsa-y-el-cghq-descifran-protocolos-seguros-para-espiarlo-todo-en-internet.

[36] J. Pastor, “NSA: un compendio del escándalo,” *Xataka*, 2013. [Online]. Available: http://www.xataka.com/otros/nsa-un-compendio-del-escandalo.

[37] S. Mansfield-Devine, “Paranoid Android: just how insecure is the most popular mobile platform?,” *Netw. Secur.*, vol. 2012, no. 9, pp. 5–10, Sep. 2012.

[38] “An Android Trojan swindles banking credentials Kaspersky,” 2013. [Online]. Available: http://blog.kaspersky.com/an-android-trojan-swindles-banking-credentials/.

[39] “99% of all mobile threats target Android devices,” 2013. [Online]. Available:

http://www.kaspersky.com/about/news/virus/2013/99_of_all_mobile_threats_target_Android_devices.

[40] “Customer security alert,” 2013. [Online]. Available: http://helpx.adobe.com/x-productkb/policy-pricing/customer-alert.html.

[41] Stricture Group, “adobePasswords.” stricture-group, 2013.

[42] S. Sigh, *Los códigos Secretos*. Debate, 2000, p. 352.

[43] NIST, “ADVANCED ENCRYPTION STANDARD (AES),” 2001.

[44] NIST Computer Security Division’s (CSD) Security Technology Group (STG) and N. C. S. D. (CSD) S. T. G. (STG), “Block cipher modes,” *Cryptogr. Toolkit*, 2013.

[45] “Download Ubuntu Desktop | Ubuntu,” 2013. [Online]. Available: http://www.ubuntu.com/download/desktop.

[46] “Dashboard | Mendeley,” 2013. [Online]. Available: http://www.mendeley.com/.

[47] I. del Pozo, “Alfresco Server Installation,” 2013. [Online]. Available: http://www.youtube.com/watch?v=DebOnHo81yM&feature=youtu.be.

[48] “Dropbox - Developers,” 2013. [Online]. Available: https://www.dropbox.com/developers.

[49] “Android Chooser API - Dropbox,” 2013. [Online]. Available: https://www.dropbox.com/developers/dropins/chooser/android.

[50] “Java Cryptography Architecture (JCA),” 2013. [Online]. Available: http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html.

[51] “Spongy Castle by rtyley,” 2013. [Online]. Available: http://rtyley.github.io/spongycastle/.

[52] J. Six, *Application Security for the Android Platform*. Sebastopol: O’Reilly, 2011, p. 152.

[53] S. Gunasekera, *Android Apps Security*. Berkeley, CA: Apress, 2012, p. 236.

[54] J. R. Aguirre, “Ataque cumpleaños | Criptored UPM,” 2013. [Online]. Available: http://www.criptored.upm.es/crypt4you/temas/RSA/leccion10/leccion10.html#apartado10-1.

[55] N. T. Courtois and J. Pieprzyk, “Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,” pp. 267–287, 2002.

[56] W. Stallings, *Cryptography And Network Security*, 4th ed. 2005, p. 744.

[57] D. Stinson, *Cryptography Theory And Practice*, 3rd ed. CRC Press, 2003, p. 593.

[58] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Massachusetts Institute of Technology, 1996, p. 755.

[59] “The Guardian Project | Secure Mobile Apps and Open-Source Code for a Better Tomorrow,” 2013. [Online]. Available: https://guardianproject.info/.

[60] “SQLCipher: Encrypted Database | The Guardian Project,” 2013. [Online]. Available: https://guardianproject.info/code/sqlcipher/.

[61] R. Oppliger, *Internet and Intranet Security (Google eBook)*. Artech House, 2001, p. 403.

[62] “OpenSSL: The Open Source toolkit for SSL/TLS,” 2013. [Online]. Available: http://www.openssl.org/.

[63] “Android Debug Bridge | Android Developers,” 2013. [Online]. Available: http://developer.android.com/tools/help/adb.html.

[64] “Application Fundamentals | Android Developers,” 2013. [Online]. Available: http://developer.android.com/guide/components/fundamentals.html.

[65] “Ataque Man in The Middle Kaspersky,” 2013. [Online]. Available: http://blog.kaspersky.es/que-es-un-ataque-man-in-the-middle/.

[66] “¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales,” 2013. [Online]. Available: http://www.genbetadev.com/seguridad-informatica/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales.

[67] “Kickstarter,” 2013. [Online]. Available: http://www.kickstarter.com/.

[68] “Indiegogo: una plataforma de crowdfunding para recaudar dinero,” 2013. [Online]. Available: http://www.indiegogo.com/?locale=es.

[69] “The Open Source Initiative | Open Source Initiative,” 2013. [Online]. Available: http://opensource.org/.

[70] “Outsourcing | Diccionario de e-conomic,” 2013. [Online]. Available: http://www.e-conomic.es/programa/glosario/definicion-outsourcing.

[71] “Real Academia Española - Repositorio,” 2013. [Online]. Available: http://buscon.rae.es/drae/srv/search?val=repositorio.

[72] “¿Qué es el software libre? | gnu.org,” 2013.

[73] “Seguridad informática: Preguntas y respuestas de Kaspersky Lab,” 2013. [Online]. Available: http://www.kaspersky.es/threats_faq#trojan.

Anexo I: Diagrama de Gantt

Nombre de tarea	Duración	Comienzo	Fin
Lectura de Documentación	15 días	lun 16/09/13	vie 04/10/13
Instalación del entorno Alfresco	4 días	lun 23/09/13	jue 26/09/13
Creación de cuentas necesarias	1 día	vie 27/09/13	vie 27/09/13
Lectura de las APIs	5 días	lun 30/09/13	vie 04/10/13
Aprendizaje de las herramientas	5 días	lun 30/09/13	vie 04/10/13
Análisis de requisistos	1 día	lun 07/10/13	lun 07/10/13
Esbozo del diseño	2 días	mar 08/10/13	mié 09/10/13
Codificación del esqueleto de la APP	10 días	jue 10/10/13	mié 23/10/13
Integración de los módulos Dropbox	7 días	jue 24/10/13	vie 01/11/13
Adaptación de los módulo de Dropbox	5 días	lun 04/11/13	vie 08/11/13
Integración de los módulos Alfresco	10 días	lun 11/11/13	vie 22/11/13
Adaptación de los módulo de Alfresco	5 días	lun 25/11/13	vie 29/11/13
Codificación de la capa de seguridad	10 días	lun 02/12/13	vie 13/12/13
Pruebas	3 días	lun 16/12/13	mié 18/12/13
Redacción de la memoria	13 días	jue 19/12/13	lun 06/01/14
Revisón de la memoria	10 días	mar 07/01/14	lun 20/01/14

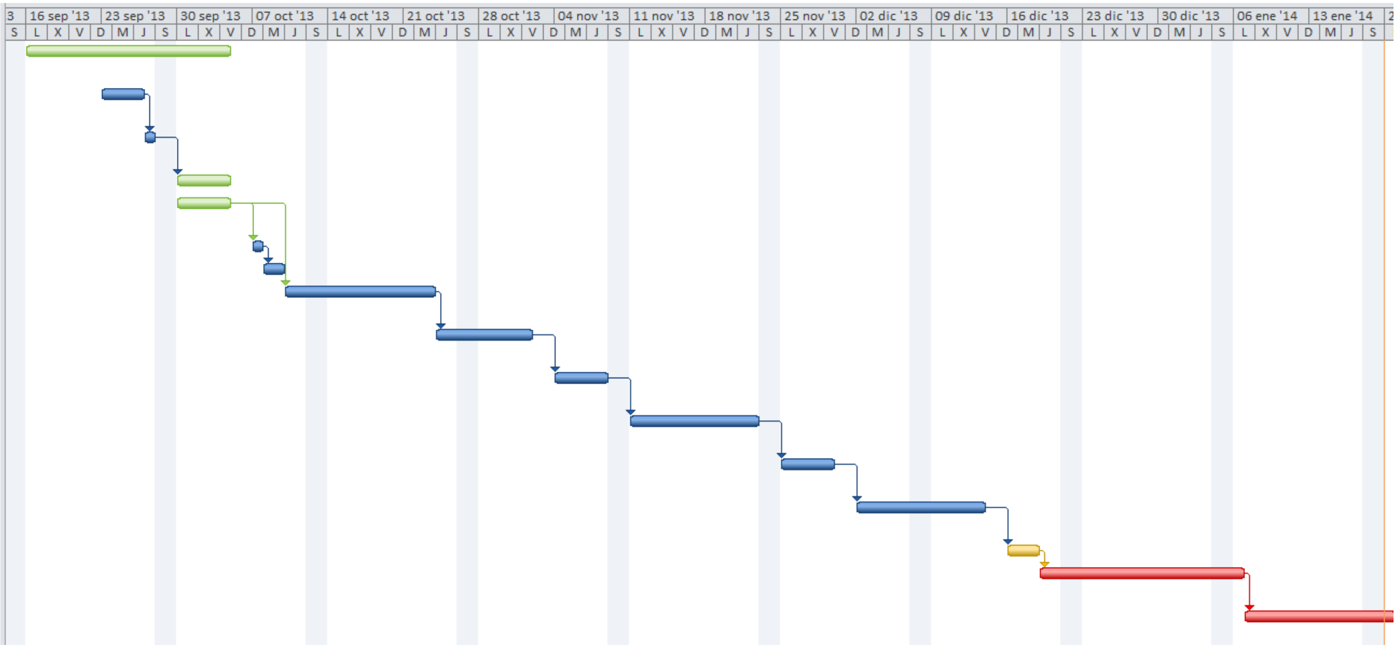

Figura 32: Diagrama de Gantt

Anexo II: Plan de negocio

(**formato producido como output de ficheros markdown en github)

Plan de Negocio (2 años)

Inversión Inicial - 10.000$ / 7.445€

- Activos Físicos **NO**
- Desglose
  - Plantilla Web --- **14$** - [themeForest | Porto template]
  - Hosting --- **5$/mes** - [HostGator | (100$ en GoogleAdWords)]
  - Dominio --- **10$/año** - [CheapNames | considerar otros además del .com]
  - Mantenimiento futuras versiones de Android, etc.
    - 6 MESES + X
  - Desarrollo Futuras conexiones con G. Drive, etc.
    - 6 MESES + X ~ *2 años*
  - Personal Coste Empresa:1.500€ - (Impuestos + Salario)
  - Marketing --- **800$**
    - GoogleAdWords
      - File | File Manager | Repository
      - Security | Secure
      - Encripted | Encryption, etc.
    - Agencia de Publicidad
  - Gastos --- **3000$** [constitución compañía, gastos jurídicos, etc.]

Total

14$ + 120$* + 20$** + 4.500€~6044,85$*** + 800$ + 3.000$

= 10.000$

* 120$ = 5$*24meses [HOSTING]

** 20$ = 10$*2años [DOMINIO]

*** 4.500€ = 1.500€*2 [SALARIO, 2 meses comienzo y beneficio]

Modelo de Negocio

1. Lanzar el producto en redes sociales, internet y quizás prensa, a través de la campaña de GoogleAdwords y la agencia digital.

2. Una vez lanzada la campaña, lanzar el producto estándar en los mercados digitales (Google Play) y la página web donde contratar los distintos planes.

3. Empezar un plan de trabajo en esos 2 primeros meses para arreglar bugs reportados e intentar desarrollar funcionalidad (app para PC por ejemplo)

4. Medir el impacto de la APP en el público y si es bueno, buscar socios para invertir más capital para seguir con el futuro de la APP

Para obtener beneficios con la APP contamos con cuatro planes de pago:

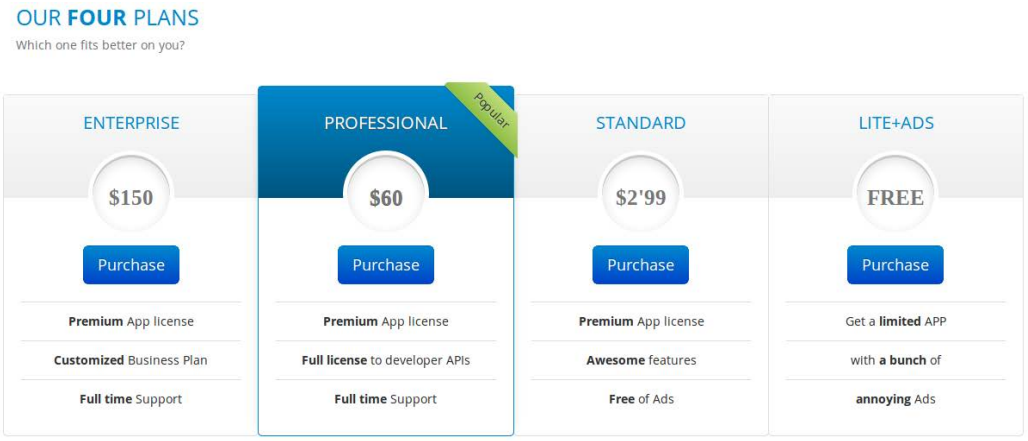

Tabla 6: Tabla de precios de nuestro plan de negocio

que se podrían obtener desde la página web a través de una pasarela de pago.

Futuro de la APP

- Suite de APPs para conectar con otras plataformas para mayor comodidad como por ejemplo PC, Unix, iOS, Web, etc.
- Nuevos cifrados y mecanismos de seguridad como firma grupal, modelo de suscriptores para los repositorios, etc. para hacer la funcionalidad de la misma adecuada a un mayor número de usuarios.
- Mejora de rendimiento e interfaz en la APP para que funcione de forma más rápida y elegante y atraiga de esta forma a más público.

Anexo III: Instalación del entorno

(se entrega en inglés ya que está pensado para ser colgado en GitHub, accesible a todo el mundo, por lo que el idioma no debería ser un problema. Es el output de un fichero Markdown de nuevo, a través del css de GitHub)

Kangaroo APP

Git Cheet Sheet

Sublime commands

```
$ git clone /URL  /* download everything**              */

$ git status      /* show changes, etc.                 */

$ git add .       /* upload everything and refresh      */

$ git commit      /* upload to local repository         */

$ git push        /* upload to master repository        */

$ git rm file.txt /* delete file.txt                    */
```

**NOTE: the good way to do it if you are collaborating with more people would be to create a local branch and merge changes with the master.

Environment installation

Please note that I used 64-bit packages every time because my computer uses a 64 bit OS, so the names of the packages may change depending on your OS. If you are not sure, you can always choose the 32 bit package.

- Install Ubuntu (64/32bits depending on your computer)

  i. Download the OS from http://www.ubuntu.com/download/desktop (LTS is a long-term support release, Hard Disk partition, etc.)

ii. I decided to create a bootable USB stick (no need to buy a CD) from http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-windows

iii. Follow Ubuntu installation

iv. Upgrade packages once Ubuntu is installed

- Install Git

i. Install it from command-line

```
$ sudo apt-get install git
```

ii. Create an account (I chose EDU account because I can create private repo) https://github.com/edu

iii. Set up git https://help.github.com/articles/set-up-git

```
$ git config --global user.name "Your Name Here"

$ git config --global user.email "your_email@example.com"

$ git config --global credential.helper 'cache --timeout=3600'
```

NOTE: the last line will cache your repository user & password for an hour, so you don't need to type it before every commit

- Install Sublime Text Editor

i. Download Sublime Text Editor http://www.sublimetext.com/2

ii. Follow http://www.technoreply.com/how-to-install-sublime-text-2-on-ubuntu-12-04-unity/ to an appropriate installation

iii. Extract the files

```
$ tar xf Sublime\ Text\ 2.0.1\ x64.tar.bz2
```

iv. Move Sublime folder to an appropriate directory

```
$ sudo mv Sublime\ Text\ 2 /opt/
```

v. Create a symbolic link, to execute from terminal

```
$ sudo ln -s /opt/Sublime\ Text\ 2/sublime_text /usr/bin/sublime
```

vi. Create a Unity launcher, to add it to side-bar, etc.

```
$ sudo sublime /usr/share/applications/sublime.desktop
```

and paste:

```
[Desktop Entry]

Version=1.0

Name=Sublime Text 2

# Only KDE 4 seems to use GenericName, so we reuse the KDE strings.

# From Ubuntu's language-pack-kde-XX-base packages, version 9.04-
20090413.

GenericName=Text Editor



Exec=sublime

Terminal=false

Icon=/opt/Sublime Text 2/Icon/48x48/sublime_text.png

Type=Application

Categories=TextEditor;IDE;Development

X-Ayatana-Desktop-Shortcuts=NewWindow



[NewWindow Shortcut Group]

Name=New Window
```

```
Exec=sublime -n

TargetEnvironment=Unity
```

vii. Install Package Control https://sublime.wbond.net/installation

Open console (View>Show Console) and paste the code in the website

```
import urllib2,os; pf='Package Control.sublime-package'; ipp =
sublime.installed_packages_path(); os.makedirs( ipp ) if not
os.path.exists(ipp) else None; urllib2.install_opener(
urllib2.build_opener( urllib2.ProxyHandler( ))); open( os.path.join(
ipp, pf), 'wb' ).write( urllib2.urlopen( 'http://sublime.wbond.net/'
+pf.replace( ' ','%20' )).read()); print( 'Please restart Sublime Text
to finish installation')
```

- Install Android Developer Tools (based in sublime installation)

i. Download the bundle from http://developer.android.com/sdk/index.html

ii. Extract files

```
$ unzip adt-bundle-linux-x86_64-20130911
```

iii. Move the folder to an appropiate directory

```
$ sudo mv adt-bundle-linux-x86_64-20130911 /opt/
```

iv. Create a symbolic link, to execute from terminal

```
$ sudo ln -s /opt/adt-bundle-linux-x86_64-20130911
/usr/bin/androidDeveloperTools
```

v. Create a Unity launcher, to add it to side-bar, etc.

```
$ sudo sublime /usr/share/applications/androidDeveloperTools.desktop
```

and paste:

```
[Desktop Entry]

Version=1.0

Name=Android Developer Tools

# Only KDE 4 seems to use GenericName, so we reuse the KDE strings.

# From Ubuntu's language-pack-kde-XX-base packages, version 9.04-
20090413.

GenericName=Integrated Development Environment



Exec=androidDeveloperTools

Terminal=false

Icon=/opt/adt-bundle-linux-x86_64-20130911/eclipse/icon.png

Type=Application

Categories=TextEditor;IDE;Development

X-Ayatana-Desktop-Shortcuts=NewWindow



[NewWindow Shortcut Group]

Name=New Window

Exec=androidDevelopmentTools

TargetEnvironment=Unity
```

vi. Download SDK upgrades

```
$ cd /opt/adt-bundle-linux-x86_64-20130911/sdk/tools

$ ./android
```

Download the last version and download the API you will use

- Install Alfresco Community Edition

    i. Download packages from http://www.alfresco.com/products/community

    ii. Check the video for used configuration and installation https://github.com/Hashael/Kangaroo/blob/master/AlfrescoInstallation.mp4 or http://youtu.be/DebOnHo81yM

Anexo IV: Código de interés

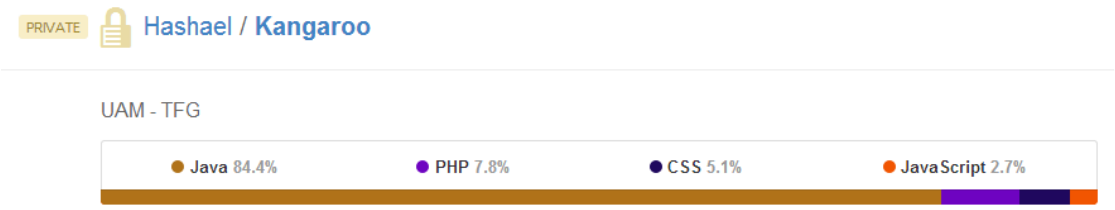

Figura 33: Lenguajes de programación utilizados en el proyecto

Dropbox UploadFile

```
/*
 * Copyright (c) 2010-11 Dropbox, Inc.
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use,
 * copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 */
package es.uam.eps.tfg.kangaroo;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.os.AsyncTask;
import android.util.Log;
```

```
import android.widget.Toast;
import com.dropbox.client2.DropboxAPI;
import com.dropbox.client2.DropboxAPI.UploadRequest;
import com.dropbox.client2.ProgressListener;
import com.dropbox.client2.exception.DropboxException;
import com.dropbox.client2.exception.DropboxFileSizeException;
import com.dropbox.client2.exception.DropboxIOException;
import com.dropbox.client2.exception.DropboxParseException;
import com.dropbox.client2.exception.DropboxPartialFileException;
import com.dropbox.client2.exception.DropboxServerException;
import com.dropbox.client2.exception.DropboxUnlinkedException;

@SuppressLint("SimpleDateFormat")
public class UploadFileDropbox extends AsyncTask<Void, Long, Boolean> {
...
    private DropboxAPI<?> mApi;
    private String mPath;
    private File mFile;

    private long mFileLen;
    private UploadRequest mRequest;
    private Context mContext;
    private final ProgressDialog mDialog;

    private String mErrorMsg;

    @SuppressWarnings("deprecation")
        public UploadFileDropbox(Context context, DropboxAPI<?> api, String
                        dropboxPath, File file) {
        // We set the context this way so we don't accidentally leak activities
        mContext = context.getApplicationContext();

        mFileLen = file.length();
        mApi = api;
        mPath = dropboxPath;
        mFile = file;

        mDialog = new ProgressDialog(context);
        mDialog.setMax(100);
        mDialog.setMessage("Uploading " + file.getName());
        mDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        mDialog.setProgress(0);
        mDialog.setButton("Cancel", new OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                // This will cancel the putFile operation
                mRequest.abort();
            }
        });
        mDialog.show();
    }
```

```
@Override
protected Boolean doInBackground(Void... params) {
    try {
        // By creating a request, we get a handle to the putFile operation,
        // so we can cancel it later if we want to
        FileInputStream fis = new FileInputStream(mFile);
        String path = mPath + mFile.getName();
        mRequest = mApi.putFileOverwriteRequest(path, fis, mFile.length(),
                new ProgressListener() {
            @Override
            public long progressInterval() {
                // Update the progress bar every half-second or so
                return 500;
            }

            @Override
            public void onProgress(long bytes, long total) {
                publishProgress(bytes);
            }
        });

        if (mRequest != null) {
            mRequest.upload();
            return true;
        }

    } catch (DropboxUnlinkedException e) {
        // This session wasn't authenticated properly or user unlinked
        mErrorMsg = "This app wasn't authenticated properly.";
    } catch (DropboxFileSizeException e) {
        // File size too big to upload via the API
        mErrorMsg = "This file is too big to upload";
    } catch (DropboxPartialFileException e) {
        // We canceled the operation
        mErrorMsg = "Upload canceled";
    } catch (DropboxServerException e) {
        // Server-side exception.  These are examples of what could happen,
        // but we don't do anything special with them here.
        if (e.error == DropboxServerException._401_UNAUTHORIZED) {
                // Unauthorized
                Log.e("ERROR", "Unauthorized user");
        } else if (e.error == DropboxServerException._403_FORBIDDEN) {
                // Not allowed to access this
                Log.e("ERROR", "Forbidden access");
        } else if (e.error == DropboxServerException._404_NOT_FOUND) {
                // path not found
                Log.e("ERROR", "File not Found");
        } else if (e.error ==
                        DropboxServerException._507_INSUFFICIENT_STORAGE) {
```

```
                    // user is over quota
                    Log.e("ERROR", "Insufficnet storage");
                } else {
                    // Something else
                    Log.e("ERROR", "Everything failed, now this APP will" +
                                   "be auto-destroyed in 5 seconds");
                }
                // This gets the Dropbox error, translated into the user's language
                mErrorMsg = e.body.userError;
                if (mErrorMsg == null) {
                    mErrorMsg = e.body.error;
                }
            } catch (DropboxIOException e) {
                // Happens all the time, probably want to retry automatically.
                mErrorMsg = "Network error.  Try again.";
            } catch (DropboxParseException e) {
                // Probably due to Dropbox server restarting, should retry
                mErrorMsg = "Dropbox error.  Try again.";
            } catch (DropboxException e) {
                // Unknown error
                mErrorMsg = "Unknown error.  Try again.";
            } catch (FileNotFoundException e) {
                    // Something else
                Log.e("ERROR", "File not Found");
            }
            return false;
        }

        @Override
        protected void onProgressUpdate(Long... progress) {
            int percent = (int)(100.0*(double)progress[0]/mFileLen + 0.5);
            mDialog.setProgress(percent);
        }

        @Override
        protected void onPostExecute(Boolean result) {
            mDialog.dismiss();
            if (result) {
                showToast("File successfully uploaded");
                mFile.delete();
            } else {
                showToast(mErrorMsg);
            }
        }

        private void showToast(String msg) {
            Toast error = Toast.makeText(mContext, msg, Toast.LENGTH_LONG);
            error.show();
        }
...
}
```

Dropbox DownloadFile

```
/*
 * Copyright (c) 2010-11 Dropbox, Inc.
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use,
 * copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 */
package es.uam.eps.tfg.kangaroo;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.graphics.drawable.Drawable;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.ImageView;
import android.widget.Toast;

import com.dropbox.client2.DropboxAPI;
import com.dropbox.client2.DropboxAPI.Entry;
import com.dropbox.client2.DropboxAPI.ThumbFormat;
import com.dropbox.client2.DropboxAPI.ThumbSize;
import com.dropbox.client2.exception.DropboxException;
import com.dropbox.client2.exception.DropboxIOException;
import com.dropbox.client2.exception.DropboxParseException;
import com.dropbox.client2.exception.DropboxPartialFileException;
import com.dropbox.client2.exception.DropboxServerException;
import com.dropbox.client2.exception.DropboxUnlinkedException;
```

```
public class DownloadFileDropbox extends AsyncTask<Void, Long, Boolean> {
...
    private Context mContext;
    private final ProgressDialog mDialog;
    private DropboxAPI<?> mApi;
    private String mPath;
    private ImageView mView;
    private Drawable mDrawable;

    private FileOutputStream mFos;

    private boolean mCanceled;
    private Long mFileLen;
    private String mErrorMsg;

    // Note that, since we use a single file name here for simplicity, you
    // won't be able to use this code for two simultaneous downloads.
    private final static String IMAGE_FILE_NAME = "dbroulette.png";

    @SuppressWarnings("deprecation")
        public DownloadFileDropbox(Context context, DropboxAPI<?> api,
            String dropboxPath, ImageView view) {
        // We set the context this way so we don't accidentally leak activities
        mContext = context.getApplicationContext();

        mApi = api;
        mPath = dropboxPath;
        mView = view;

        mDialog = new ProgressDialog(context);
        mDialog.setMessage("Downloading Image");
        mDialog.setButton("Cancel", new OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                mCanceled = true;
                mErrorMsg = "Canceled";

                // This will cancel the getThumbnail operation by closing
                // its stream
                if (mFos != null) {
                    try {
                        mFos.close();
                    } catch (IOException e) {
                            // Failed or was cancelled by the user.
                                Log.e("ERROR", "Operation Cancelled");
                    }
                }
            }
        });

        mDialog.show();
    }
```

```
@Override
protected Boolean doInBackground(Void... params) {
    try {
        if (mCanceled) {
            return false;
        }

        // Get the metadata for a directory
        Entry dirent = mApi.metadata(mPath, 1000, null, true, null);

        if (!dirent.isDir || dirent.contents == null) {
            // It's not a directory, or there's nothing in it
            mErrorMsg = "File or empty directory";
            return false;
        }

        // Make a list of everything in it that we can get a thumbnail for
        ArrayList<Entry> thumbs = new ArrayList<Entry>();
        for (Entry ent: dirent.contents) {
            if (ent.thumbExists) {
                // Add it to the list of thumbs we can choose from
                thumbs.add(ent);
            }
        }

        if (mCanceled) {
            return false;
        }

        if (thumbs.size() == 0) {
            // No thumbs in that directory
            mErrorMsg = "No pictures in that directory";
            return false;
        }

        // Now pick a random one
        int index = (int)(Math.random() * thumbs.size());
        Entry ent = thumbs.get(index);
        String path = ent.path;
        mFileLen = ent.bytes;


        String cachePath = mContext.getCacheDir().getAbsolutePath() + "/" +
                        IMAGE_FILE_NAME;
        try {
            mFos = new FileOutputStream(cachePath);
        } catch (FileNotFoundException e) {
            mErrorMsg = "Couldn't create a local file to store the image";
            return false;
        }
```

```
            // This downloads a smaller, thumbnail version of the file.  The
            // API to download the actual file is roughly the same.
            mApi.getThumbnail(path, mFos, ThumbSize.BESTFIT_960x640,
                    ThumbFormat.JPEG, null);
            if (mCanceled) {
                return false;
            }

            mDrawable = Drawable.createFromPath(cachePath);
            // We must have a legitimate picture
            return true;

        } catch (DropboxUnlinkedException e) {
            // The AuthSession wasn't properly authenticated or user unlinked.
                mErrorMsg = "This app wasn't authenticated properly.";
        } catch (DropboxPartialFileException e) {
            // We canceled the operation
            mErrorMsg = "Download canceled";
        } catch (DropboxServerException e) {
            // Server-side exception.  These are examples of what could happen,
            // but we don't do anything special with them here.
            if (e.error == DropboxServerException._304_NOT_MODIFIED) {
                // won't happen since we don't pass in revision with metadata
                    Log.e("ERROR", "Not modified file");
            } else if (e.error == DropboxServerException._401_UNAUTHORIZED) {
                // Unauthorized
                    Log.e("ERROR", "Unauthorized user");
            } else if (e.error == DropboxServerException._403_FORBIDDEN) {
                // Not allowed to access this
                    Log.e("ERROR", "Forbidden access");
            } else if (e.error == DropboxServerException._404_NOT_FOUND) {
                // path not found
                    Log.e("ERROR", "File not Found");
            } else if (e.error == DropboxServerException._406_NOT_ACCEPTABLE) {
                // too many entries to return
                    Log.e("ERROR", "Too many entries");
            } else if (e.error ==
                        DropboxServerException._415_UNSUPPORTED_MEDIA) {
                // can't be thumbnailed
                    Log.e("ERROR", "The file can't be thumbnailed");
            } else if (e.error ==
                        DropboxServerException._507_INSUFFICIENT_STORAGE) {
                // user is over quota
                    Log.e("ERROR", "Insufficnet storage");
            } else {
                    // Something else
                Log.e("ERROR", "Everything failed, now this APP will" +
                                "be auto-destroyed in 5 seconds");
            }
```

```
            // This gets the Dropbox error, translated into the user's language
            mErrorMsg = e.body.userError;
            if (mErrorMsg == null) {
                mErrorMsg = e.body.error;
            }
        } catch (DropboxIOException e) {
            // Happens all the time, probably want to retry automatically.
            mErrorMsg = "Network error.  Try again.";
        } catch (DropboxParseException e) {
            // Probably due to Dropbox server restarting, should retry
            mErrorMsg = "Dropbox error.  Try again.";
        } catch (DropboxException e) {
            // Unknown error
            mErrorMsg = "Unknown error.  Try again.";
        }
        return false;
    }

    @Override
    protected void onProgressUpdate(Long... progress) {
        int percent = (int)(100.0*(double)progress[0]/mFileLen + 0.5);
        mDialog.setProgress(percent);
    }

    @Override
    protected void onPostExecute(Boolean result) {
        mDialog.dismiss();
        if (result) {
            // Set the image now that we have it
            mView.setImageDrawable(mDrawable);
        } else {
            // Couldn't download it, so show an error
            showToast(mErrorMsg);
        }
    }

    private void showToast(String msg) {
        Toast error = Toast.makeText(mContext, msg, Toast.LENGTH_LONG);
        error.show();
    }

...
}
```

Dropbox Manager (DBRoulette)

```
/*
 * Copyright (c) 2010-11 Dropbox, Inc.
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use,
 * copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 */
package es.uam.eps.tfg.kangaroo;

import java.io.File;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.dropbox.chooser.android.DbxChooser;
import com.dropbox.client2.DropboxAPI;
import com.dropbox.client2.android.AndroidAuthSession;
import com.dropbox.client2.android.AuthActivity;
```

```
import com.dropbox.client2.session.AccessTokenPair;
import com.dropbox.client2.session.AppKeyPair;
import com.dropbox.client2.session.Session.AccessType;
import com.dropbox.client2.session.TokenPair;

@SuppressLint("SimpleDateFormat")
public class DBRoulette extends Activity {
...
    private static final String TAG = "DBRoulette";
    private static final int PICKFILE_RESULT_CODE = 1;
    static final int DBX_CHOOSER_REQUEST = 2;

    ///////////////////////////////////////////////////////////////////////////
    //                      Your app-specific settings.                      //
    ///////////////////////////////////////////////////////////////////////////

    // Replace this with your app key and secret assigned by Dropbox.
    // Note that this is a really insecure way to do this, and you shouldn't
    // ship code which contains your key & secret in such an obvious way.
    // Obfuscation is good.
    final static private String APP_KEY = "twk7t2lq42tdtul";
    final static private String APP_SECRET = "g0swvxtp93wxy22";
    final static private String CHOOSER_APP_KEY = "d6w7qpagjn3zvhr";

    // If you'd like to change the access type to the full Dropbox instead of
    // an app folder, change this value.
    final static private AccessType ACCESS_TYPE = AccessType.DROPBOX;

    ///////////////////////////////////////////////////////////////////////////
    //                      End app-specific settings.                       //
    ///////////////////////////////////////////////////////////////////////////

    // You don't need to change these, leave them alone.
    final static private String ACCOUNT_PREFS_NAME = "prefs";
    final static private String ACCESS_KEY_NAME = "ACCESS_KEY";
    final static private String ACCESS_SECRET_NAME = "ACCESS_SECRET";


    DropboxAPI<AndroidAuthSession> mApi;

    private boolean mLoggedIn;

    // Android widgets
    private Button mSubmit;
    private LinearLayout mDisplay;
    private Button mUpload;
    private Button mDownload;

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

```
        // We create a new AuthSession so that we can use the Dropbox API.
        AndroidAuthSession session = buildSession();
        mApi = new DropboxAPI<AndroidAuthSession>(session);

        setContentView(R.layout.activity_dropbox);

        checkAppKeySetup();

        mSubmit = (Button)findViewById(R.id.auth_button);

        mSubmit.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // This logs you out if you're logged in, or vice versa
                if (mLoggedIn) {
                    logOut();
                } else {
                    // Start the remote authentication
                    mApi.getSession().startAuthentication(DBRoulette.this);
                }
            }
        });

        mDisplay = (LinearLayout)findViewById(R.id.logged_in_display);

        // This is the button to take a photo
        // THIS HAS TO THROW THE FILE PICKER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        mUpload = (Button)findViewById(R.id.photo_button);

        mUpload.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                        Intent intent = new Intent(mUpload.getContext(),
                                        FilePickerActivity.class);
                        startActivityForResult(intent, PICKFILE_RESULT_CODE);
            }
        });

        // This is the button that throws the chooser
        final DbxChooser mChooser = new DbxChooser(CHOOSER_APP_KEY);

        mDownload = (Button)findViewById(R.id.roulette_button);
        mDownload.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                    mChooser.forResultType(DbxChooser.ResultType.FILE_CONTENT)
                .launch(DBRoulette.this, DBX_CHOOSER_REQUEST);
            }
        });

        // Display the proper UI state if logged in or not
        setLoggedIn(mApi.getSession().isLinked());

    }
```

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
}

@Override
protected void onResume() {
    super.onResume();
    AndroidAuthSession session = mApi.getSession();

    // The next part must be inserted in the onResume() method of the
    // activity from which session.startAuthentication() was called, so
    // that Dropbox authentication completes properly.
    if (session.authenticationSuccessful()) {
        try {
            // Mandatory call to complete the auth
            session.finishAuthentication();

            // Store it locally in our app for later use
            TokenPair tokens = session.getAccessTokenPair();
            storeKeys(tokens.key, tokens.secret);
            setLoggedIn(true);
        } catch (IllegalStateException e) {
            showToast("Couldn't authenticate with Dropbox:" +
                            e.getLocalizedMessage());
            Log.i(TAG, "Error authenticating", e);
        }
    }
}

// This is what gets called on finishing a media piece to import
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(resultCode == RESULT_OK) {

                // Get the user's password
                    SharedPreferences settings = PreferenceManager
                                    .getDefaultSharedPreferences(this);
                    String pass = settings.getString("password", null);

                    switch(requestCode) {
                    case PICKFILE_RESULT_CODE:
                            if(data.hasExtra(FilePickerActivity.EXTRA_FILE_PATH){
                                    // Get the file path
                                    File f = new File(data.getStringExtra(
                                            FilePickerActivity.EXTRA_FILE_PATH));

                                /***  ENCRYPT MODE  ***/
                                    File encodedFile =
                                          CipherActivity.encondeFile(f, pass);
```

```
                    if (encodedFile == null){
                          Toast.makeText(getApplicationContext(),
                                      "Error Encrypting!",
                                      Toast.LENGTH_SHORT).show();
                    } else {
                          Toast.makeText(getApplicationContext(),
                                      "File Encrypted!",
                                      Toast.LENGTH_SHORT).show();

                          Calendar cal = Calendar.getInstance();
                          cal.getTime();
                          SimpleDateFormat sdf =
                             new SimpleDateFormat("MM_dd_hh:mm");

                          String path = "/Kangaroo/" +
                                    sdf.format(cal.getTime());

                          UploadFileDropbox upload = new
                                    UploadFileDropbox(this,
                                        mApi, path, encodedFile);
                          upload.execute();
                    }
              }
              break;
        case DBX_CHOOSER_REQUEST:
              if (resultCode == Activity.RESULT_OK) {
        DbxChooser.Result result = new DbxChooser.Result(data);

        // Get the file path
        File f2 = new File(result.getLink().getEncodedPath());

                    /***  DECRYPT MODE  ***/
                    int decodeResult =
                          CipherActivity.decodeFile(f2, pass);
                    if (decodeResult == 0){
                          Toast.makeText(getApplicationContext(),
                                "File Decrypted and downloaded in
                                   Downloads!",
                                Toast.LENGTH_SHORT).show();

                            String docPath = Environment.
                                getExternalStorageDirectory().
                                    getPath() + File.separator +
                                    "Download" + File.separator +
                                    result.getName();
                            openDocument(docPath);
                    } else {
                          Toast.makeText(getApplicationContext(),
                                   "Error Decrypting!",
                                   Toast.LENGTH_SHORT).show();
                    }
```

```
                    } else {
                        // Failed or was cancelled by the user.
                            Log.e("ERROR", "Failed or cancelled by the user");
                    }
                            break;
                        default:
                               super.onActivityResult(requestCode, resultCode, data);
                            break;
                        }
                }
    }

    private void logOut() {
        // Remove credentials from the session
        mApi.getSession().unlink();
        // Clear our stored keys
        clearKeys();
        // Change UI state to display logged out version
        setLoggedIn(false);
    }

    /**
     * Convenience function to change UI state based on being logged in
     */
    private void setLoggedIn(boolean loggedIn) {
            mLoggedIn = loggedIn;
            if (loggedIn) {
                    mSubmit.setText("Unlink from Dropbox");
            mDisplay.setVisibility(View.VISIBLE);
            } else {
                    mSubmit.setText("Link with Dropbox");
            mDisplay.setVisibility(View.GONE);
            }
    }

    private void checkAppKeySetup() {
        // Check to make sure that we have a valid app key
        if (APP_KEY.startsWith("CHANGE") ||
                APP_SECRET.startsWith("CHANGE")) {
            showToast("You must apply for an app key and secret from " +
                            "developers.dropbox.com, and add them to the DBRoulette"
+
                            "app before trying it.");
            finish();
            return;
        }

        // Check if the app has set up its manifest properly.
        Intent testIntent = new Intent(Intent.ACTION_VIEW);
        String scheme = "db-" + APP_KEY;
        String uri = scheme + "://" + AuthActivity.AUTH_VERSION + "/test";
```

```
        testIntent.setData(Uri.parse(uri));
        PackageManager pm = getPackageManager();
        if (0 == pm.queryIntentActivities(testIntent, 0).size()) {
            showToast("URL scheme in your app's " +
                    "manifest is not set up correctly. You should have a " +
                    "com.dropbox.client2.android.AuthActivity with the " +
                    "scheme: " + scheme);
            finish();
        }
    }

    private void showToast(String msg) {
        Toast error = Toast.makeText(this, msg, Toast.LENGTH_LONG);
        error.show();
    }

    /**
     * Shows keeping the access keys returned from Trusted Authenticator in a
     * local store, rather than storing user name & password, and
     * re-authenticating each time (which is not to be done, ever).
     *
     * @return Array of [access_key, access_secret], or null if none stored
     */
    private String[] getKeys() {
        SharedPreferences prefs = getSharedPreferences(ACCOUNT_PREFS_NAME, 0);
        String key = prefs.getString(ACCESS_KEY_NAME, null);
        String secret = prefs.getString(ACCESS_SECRET_NAME, null);
        if (key != null && secret != null) {
                String[] ret = new String[2];
                ret[0] = key;
                ret[1] = secret;
                return ret;
        } else {
                return null;
        }
    }

    /**
     * Shows keeping the access keys returned from Trusted Authenticator in a
     * local store, rather than storing user name & password, and
     * re-authenticating each time (which is not to be done, ever).
     */
    private void storeKeys(String key, String secret) {
        // Save the access key for later
        SharedPreferences prefs = getSharedPreferences(ACCOUNT_PREFS_NAME, 0);
        Editor edit = prefs.edit();
        edit.putString(ACCESS_KEY_NAME, key);
        edit.putString(ACCESS_SECRET_NAME, secret);
        edit.commit();
    }
```

```
    private void clearKeys() {
        SharedPreferences prefs = getSharedPreferences(ACCOUNT_PREFS_NAME, 0);
        Editor edit = prefs.edit();
        edit.clear();
        edit.commit();
    }

    private AndroidAuthSession buildSession() {
        AppKeyPair appKeyPair = new AppKeyPair(APP_KEY, APP_SECRET);
        AndroidAuthSession session;

        String[] stored = getKeys();
        if (stored != null) {
            AccessTokenPair accessToken = new AccessTokenPair(stored[0],
                            stored[1]);
            session = new AndroidAuthSession(appKeyPair, ACCESS_TYPE,
                            accessToken);
        } else {
            session = new AndroidAuthSession(appKeyPair, ACCESS_TYPE);
        }

        return session;
    }

    public void openDocument(String documentName) {
        Intent intent = new Intent(android.content.Intent.ACTION_VIEW);
        File file = new File(documentName);
        String extension = android.webkit.MimeTypeMap.getFileExtensionFromUrl
                        (Uri.fromFile(file).toString());
        String mimetype = android.webkit.MimeTypeMap.getSingleton().
                        getMimeTypeFromExtension(extension);
        if (extension.equalsIgnoreCase("") || mimetype == null)
        {
                // if there is no extension or there is no definite mimetype,
                // still try to open the file
            intent.setDataAndType(Uri.fromFile(file), "text/*");
        }
        else
        {
            intent.setDataAndType(Uri.fromFile(file), mimetype);
        }
        // custom message for the intent
        startActivity(Intent.createChooser(intent, "Choose an Application:"));
    }
...
}
```

Cryptographic Suite

```
package es.uam.eps.tfg.kangaroo;

import java.io.*;
import java.security.MessageDigest;
import java.security.SecureRandom;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

import android.os.Environment;

public class CipherActivity {
        ...
        // Encrypt a file with AES
        public static File encondeFile(File file, String password) {
                // Open file for cipher operations
                FileInputStream fin = null;
                File encriptedFile = null;

                try {
                        // Create FileInputStream object to read the file
                        fin = new FileInputStream(file);
                        byte fileContent[] = new byte[(int)file.length()];

                        // Reads up to certain bytes of data from this input
                        // stream into an array of bytes.
                        fin.read(fileContent);

                        // Create the encrypted file
                        encriptedFile = new File(
                        Environment.getExternalStorageDirectory().getPath() +
                        File.separator + file.getName());

                        // Write the encrypted data in the new file
                        BufferedOutputStream bos = new BufferedOutputStream(
                                        new FileOutputStream(encriptedFile));

                        // Generate the AES key and encodeData
                        byte[] yourKey = generateKey(password);
                        byte[] fileBytes = encodeData(yourKey, fileContent);

                        // Write in the new file
                        bos.write(fileBytes);
                        bos.flush();
                        bos.close();
                }
```

```
            catch (FileNotFoundException e) {
                    System.out.println("File not found" + e);
                    return null;
            }
            catch (IOException ioe) {
                    System.out.println("Exception while reading the file "+ ioe);
                    return null;
            }
            catch (Exception e) {
                    System.out.println("Exception while ciphering the file "+e);
                    return null;
            }
            finally {
                    // Close the streams
                    try {
                            if (fin != null) {
                                    fin.close();
                            }
                    }
                    catch (IOException ioe) {
                            System.out.println("Error while closing stream: " +   ioe);
                            return null;
                    }
            }
            return encriptedFile;
    }

    // Decode a file with AES
    public static int decodeFile(File file, String password) {
            // Open file for cipher operations
            FileInputStream fin = null;

            try {
                    // Create FileInputStream object to read the file
                    fin = new FileInputStream(file);
                    byte fileContent[] = new byte[(int)file.length()];

                    // Reads up to certain bytes of data from this input
                    // stream into an array of bytes.
                    fin.read(fileContent);

                    // Create the decoded file
                    File decripted = new File(
                    Environment.getExternalStorageDirectory().getPath() +
                    File.separator + "Download" + File.separator +
                    file.getName());

                    // Write the decoded data in the new file
                    BufferedOutputStream bos = new BufferedOutputStream(
                                    new FileOutputStream(decripted));
```

```
                // Generate the AES key and decodeData
                byte[] yourKey = generateKey(password);
                byte[] fileBytes = decodeData(yourKey, fileContent);

                // Write in the new file
                bos.write(fileBytes);
                bos.flush();
                bos.close();
        }
        catch (FileNotFoundException e) {
                System.out.println("File not found" + e);
                return -1;
        }
        catch (IOException ioe) {
                System.out.println("Exception while reading the file " +ioe);
                return -1;
        }
        catch (Exception e) {
                System.out.println("Exception while ciphering the file " +e);
                return -1;
        }
        finally {
                // Close the streams
                try {
                        if (fin != null) {
                                fin.close();
                        }
                }
                catch (IOException ioe) {
                        System.out.println("Error while closing stream: " +   ioe);
                        return -1;
                }
        }
        return 0;
}

// Generate a private key for AES cipher
public static byte[] generateKey(String password) throws Exception {
        byte[] keyStart = password.getBytes("UTF-8");

        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(keyStart);
        kgen.init(256, sr);
        SecretKey skey = kgen.generateKey();
        return skey.getEncoded();
}


 // Encode data in fileData with AES
public static byte[] encodeData(byte[] key, byte[] fileData)
```

```
                throws Exception {
        // Initialization Vector Required for CBC
        byte[] iv =
          {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00};
        IvParameterSpec ips = new IvParameterSpec(iv);

        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ips);

        byte[] encrypted = cipher.doFinal(fileData);
        return encrypted;
    }

    // Decode data in fileData with AES
    public static byte[] decodeData(byte[] key, byte[] fileData)
                throws Exception {
        // Initialization Vector Required for CBC
        byte[] iv =
          {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00};
        IvParameterSpec ips = new IvParameterSpec(iv);

        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, ips);

        byte[] decrypted = cipher.doFinal(fileData);
        return decrypted;
    }

    // Function to encode in MD5
    public static String getMD5(String str) throws Exception {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] b = md.digest(str.getBytes());

        int size = b.length;
        StringBuilder h = new StringBuilder(size);
        for (int i = 0; i < size; i++) {
            int u = b[i] & 255;
            if (u < 16) {
                h.append("0").append(Integer.toHexString(u));
            } else {
                h.append(Integer.toHexString(u));
            }
        }
        return h.toString();
     }
...
}
```

Hash - New Account

```
package es.uam.eps.tfg.kangaroo;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Account extends Activity implements OnClickListener {
     ...
        private void newAccount() {
                String name = editTextUsername.getText().toString();
                String pass1 = editTextPassword.getText().toString();
                String pass2 = editTextPasswordAgain.getText().toString();
                String passHashed = null;
                if (!pass1.equals("") && !name.equals("") && pass1.equals(pass2)) {
                        try {
                                passHashed =
                            es.uam.eps.tfg.kangaroo.CipherActivity.getMD5(pass1);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                        db = new DatabaseAdapter(this);
                        db.open();
                        db.insertUser(name, passHashed);
                        db.close();
                        String newUser =
                                  new String(getString(R.string.new_user_added));
                        Toast.makeText(Account.this, newUser,
                                        Toast.LENGTH_SHORT).show();
                        finish();

                } else if (pass1.equals("") || pass2.equals("") || name.equals("")) {
                        String missingData =
                                   new String(getString(R.string.missing_data));
                        Toast.makeText(Account.this, missingData, Toast.LENGTH_SHORT)
                                        .show();
                } else if (!pass1.equals(pass2)) {
                        String passwordMissmatch =
                                   new String(getString(R.string.password_missmatch));
                        Toast.makeText(Account.this, passwordMissmatch,
                                        Toast.LENGTH_SHORT).show();
                }
        }
    ...
}
```

Hash - Login

```
package es.uam.eps.tfg.kangaroo;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class Login extends Activity implements OnClickListener {
   ...
        private void check() {
                String username = userNameEditText.getText().toString();
                String password = passwordEditText.getText().toString();

                try {
                        // HASH time!
                        password =
                              es.uam.eps.tfg.kangaroo.CipherActivity.getMD5(password);
                        SharedPreferences settings = PreferenceManager
                                        .getDefaultSharedPreferences(this);
                        SharedPreferences.Editor editor = settings.edit();
                        editor.putString("password", password);
                        editor.commit();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                /* TOAST Strings */
                String title = new String(getString(R.string.error));
                String message = new String(getString(R.string.login_failed));
                String button = new String(getString(R.string.try_again));

                db = new DatabaseAdapter(this);
                db.open();
                boolean firstIn = db.isRegistered(username, password);
                db.close();
                if (firstIn) {
                        SharedPreferences settings = PreferenceManager
                                        .getDefaultSharedPreferences(this);
                        SharedPreferences.Editor editor = settings.edit();
                        editor.putString("firstplayername", username);
                        editor.commit();
                        startActivity(new Intent(this, Hub.class));
                        finish();
```

```
                } else {
                        new AlertDialog.Builder(this)
                                        .setTitle(title)
                                        .setMessage(message)
                                        .setNeutralButton(button,
                                              new DialogInterface.OnClickListener() {
                                                  public void onClick(DialogInterface
                                                       dialog, int which) {
                                                  }
                                              }).show();
                }
        }
   ...
}
```

Android generic Open File - Intent (en DbRoulette)

```
public void openDocument(String documentName) {
        Intent intent = new Intent(android.content.Intent.ACTION_VIEW);
        File file = new File(documentName);
        String extension = android.webkit.MimeTypeMap.getFileExtensionFromUrl
                        (Uri.fromFile(file).toString());
        String mimetype = android.webkit.MimeTypeMap.getSingleton().
                        getMimeTypeFromExtension(extension);
        if (extension.equalsIgnoreCase("") || mimetype == null)
        {
            // if there is no extension or there is no definite mimetype,
            // still try to open the file
            intent.setDataAndType(Uri.fromFile(file), "text/*");
        }
        else
        {
            intent.setDataAndType(Uri.fromFile(file), mimetype);
        }
        // custom message for the intent
        startActivity(Intent.createChooser(intent, "Choose an Application:"));
 }
```

**NOTA: se incluye únicamente el código de interés del proyecto, el que ha sido desarrollado o modificado por nosotros, no así el código fuente adquirido de los proveedores que no aporta interés extra.

Anexo V: Evolución del logo

Para este proyecto se ha diseñado un logo corporativo para la APP y su lanzamiento al mercado. Se ha elegido un canguro por la simbología de estar "saltando" entre plataformas.

Respecto al color se ha elegido el naranja por sus tonos vivos que sugieren agilidad y en cierto modo confianza, al alejarnos del azul en una gama cromática totalmente diferente, que sugiere todo aquello que tiene que ver con aspectos sociales y comunicación con los demás, justo lo que se quiere evitar cuando hablamos de una APP que sirve para cifrar archivos para que otros no puedan acceder a ellos.

Es curioso como esta elección de color en concreto ha sido modificada por Dropbox cuyo logo inicial era

Figura 34: Evolución del logo de Dropbox

Del primer al logo al segundo, el cambio en el diseño se debe al cambio de paradigma, que cambio de los iconos realistas, casi siempre caracterizados por el efecto "Gloss" al icono plano, que se pone realmente de moda con el lanzamiento de iOS7, su principal defensor, y es la tendencia de diseño actual, diseños simples y sin 3D que trasmitan claramente la imagen de la empresa. El último paso y el logo actual, es el cambio en el color, que en mi opinión se debe a que no se quiere asociar Dropbox como una APP social, ya que cada vez más los usuarios la utilizan como disco duro virtual para guardar información personal de la que quieren disponer online, en muchos casos archivos que no quieren compartir, por lo que se ha cambiado el color del logo para evitar esta asociación en los consumidores.

Respecto a nuestro logo realizamos un trabajo de investigación para averiguar qué queríamos transmitir. Partimos de la siguiente fotografía, queríamos algo parecido a esto, ágil y dinámico, que transmita cierta sensación de movimiento.

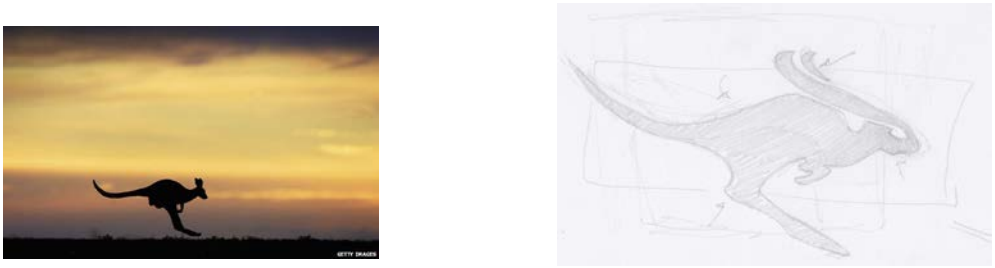

Figura 35: Concepto del Logo de KangarooAPP

En nuestro caso el logo empezó con el concept que se puede ver primero, queríamos un canguro un poco fantástico, con algún rasgo que lo caracterizase, de ahí las orejas.

La primera prueba en Android no nos gustó demasiado, debido a que no se distinguía correctamente y se confundía con el fondo. Por ello optamos por el siguiente diseño. Este nos gustaba mucho, pero quedaba extraño, no resaltaba como los demás y pensamos en añadir una textura, obteniendo el diseño final.

Figura 36: Evolución del logo Kangaroo

Anexo VI: Web Corporativa

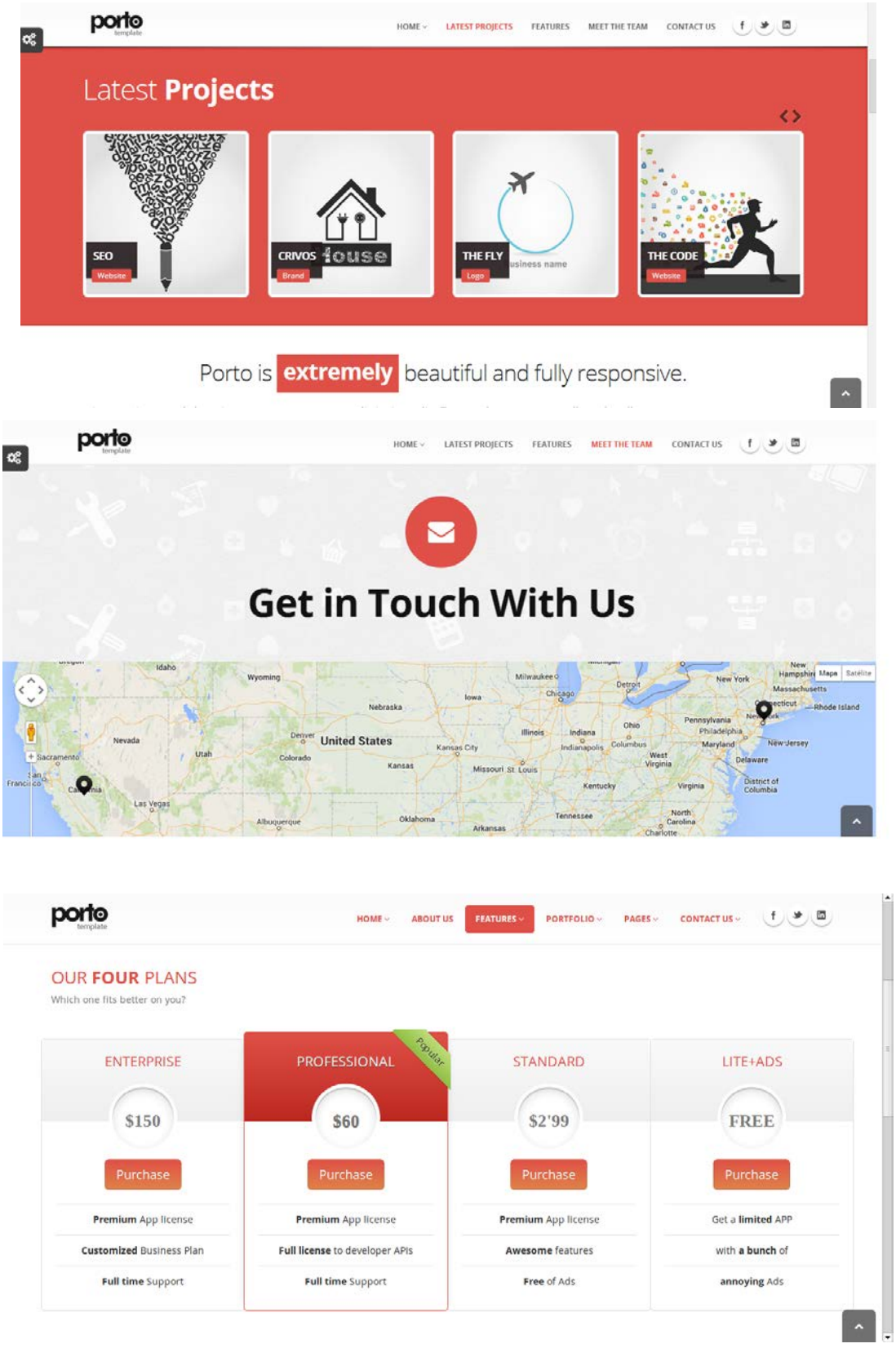

Figura 37: Capturas de pantalla de la web corporativa

Anexo VII: Sintaxis Markdown

````
Plan de Negocio (2 años)     (h1)
=========================

Inversión Inicial    -    10.000$ / 7.445€     (h2)
-----------------

- Desglose        (listas)
    - Plantilla Web --- **14$** - [themeForest | Porto template]
    - Hosting --- **5$/mes** - [HostGator | (100$ en GoogleAdWords)]

![pricingTable]         (imágenes)

[pricingTable]:
file:///home/nacho/workspace/TFG/Kangaroo/resources/pricingTable.png
 width=580px height=250px

```sh             (cajas de código)
$ git clone /URL  /* download everything**                */
$ git status      /* show changes, etc.                   */
$ git add .       /* upload everything and refresh      */
$ git commit      /* upload to local repository                */
$ git push        /* upload to master repository          */
$ git rm file.txt /* delete file.txt                            */
```
**negrita**
*cursiva*

- elemento1       (ul lista sin orden)
- elemento2
- elemento3

1 elemento1       (ol lista ordenada)
2 elemento2
3 elemento3
````