UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Proyecto Final de Carrera

FotoCloud

David Gómez Torrecilla

07/09/2014

Contenido

Tabla de Ejemplos

Tabla de Ilustraciones:

Motivación

Según la página *web Internet World Stats[1]* en 2012 internet contó con 7.017 millones de usuarios, con una penetración mundial del 34,3%. Si nos centrásemos únicamente en los usuarios que acceden a la red de redes mediante un smarthphone nos encontramos con que las predicciones para 2014 son de nada menos que 1.750 millones de terminales portátiles (según eMarketer[2]).

Estos avances han propiciado la creación de un mercado que tiene como objetivo cubrir las necesidades de comunicación entre individuos conectados el cual ha dado como resultado la aparición de las "redes sociales" y "chats".

Existen infinidad de redes sociales, podríamos clasificarlas según su enfoque: cocina, mascotas, turismo, fotografía, etc. siendo la fotografía el enfoque que recoge más popularidad y usuarios actualmente.

Estas redes sociales de sobra conocidas, como Facebook o Picassa, podrían servir de herramienta para ser tratadas como albunes de fotos en el aspecto tradicional de forma tan sencilla que cualquiera pueda hacerlo sin tener que aprender sus extensas funcionalidades y de paso solucionar el problema de la pérdida de instantáneas cuando la tarjeta microSD o el móvil dejen de funcionar o lo perdamos.

Bajo estas circunstacias surge FotoCloud, aplicación presentada en este proyecto.

Objetivo:

FotoCloud tiene como objetivo ofrecer en una herramienta de almacenamiento de fotografías en la nube sin usar para ello un servicio de almacenamiento propio, sino aprovechandose de los que ofrecen gratuitamente redes sociales como "Picassa" o "Facebook" e integrarlos en una sencilla aplicación movil.

La aplicación además de la integración de varios servicios de almacenamiento de fotografías online también estará diseñada para que su utilización sea extremadamente sencilla y rápida de forma que el usuario pueda tomar y subir una fotografía de su galería o hecha en el momento lo más rápidamente posible en cualquiera de los servicios implementados.

También estará compuesta por un intuitivo visualizador de álbumes que te permitirá listar las fotos en forma de gridview o una por una en pantalla completa.

Para que todo esto sea posible, además estará pensado para su utilización en una plataforma portátil como un smartphone y en un sistema que llegue al mayor número de personas posible como Android.

Descripción de alto nivel

Fotocloud no es un sistema de almacenamiento de fotos tradicional, sino un servicio que interacciona con los las más populares redes socilales de fotografía como Picassa, Flickr o Facebook para simplificarte el uso en cuanto al manejo de fotografías.

Se trata de una aplicación Android compatible desde la version 2.0 hasta la 4.4.

Las distintas funcionalidades no relacionadas con la fotografía que disponen las redes sociales como el chat, el tablón, juegos etc. han sido eliminadas para alcanzar una simplicidad que permita realizar las acciones de forma intuitiva centrandonos en el objetivo de la aplicación..

La aplicación dispondra de 4 funciones principales:

- Pantalla de inicio de sesión: te permitirá iniciar sesión en cualquiera de las redes sociales soportadas
- Pantalla de lista de álbunes: mostrará todos tus álbunes por nombre en forma de list view.
- Pantalla de vista de fotos: mostrará las fotos de un álbum seleccionado en forma de grid view.
- Una vez que te encuentres identificado dentro de cualquier red social podrás subir fotos tanto recién tomadas como de la galería

Dada la envergadura del proyecto, la versión que se presenta en este proyecto implementa solo Facebook por ser la que disfruta de mayor número de usuarios.

Se trata de una aplicación Android, para la maquetación de lo bocetos se ha utilizado el programa de diseño de interfaces Balsamiq Mockup, realmente sencillo y útil a la hora de ordenar y plasmar las funcionalidades en forma de dibujo, posteriormente se ha utilizado la aplicaciçon UMLStar para diseñar el diagrama de clases, acompañando todo el desarrollo hemos utilizado como contenedor del proyecto al servicio GitHub con el sistema de control de versiones Git y como interface de desarrollo se ha elegido eclipse junto al SDK de android debido a la perfecta integracion de las diversas herramientas de programación que ofrece. Durante el desarrollo de la aplicacion he utilizado diversas librerias de terceros para mejorar destientos aspectos de la aplicación como SherlockBar y LazyLoad.

Estructura del documento:

En la siguiente sección definiremos el marco conceptual que rodea nuestra aplicación, definiendo los conceptos que rodean a las redes sociales y a las aplicaciones móviles.

En la sección 3 presentaremos las diferentes herramientas utilizadas en este proyecto, describiremos para que se han usado y sus características.

[...]

2.-Marco conceptual

2.1. Red Social

Un servicio de red social es un medio de comunicación social que se centra en encontrar gente para relacionarse en línea. Están formadas por personas que comparten alguna relación, principalmente de amistad, mantienen intereses y actividades en común, o están interesados en explorar los intereses y las actividades de otros.

En general, estos servicios de redes sociales permiten a los usuarios crear un perfil para ellos mismos, y se pueden dividir en dos grandes categorías: la creación de redes sociales internas (ISN) y privada que se compone de un grupo de personas dentro de una empresa, asociación, sociedad, el proveedor de educación y organización, o incluso una "invitación", creado por un grupo de usuarios en un ESN. El ESN es una red abierta y a disposición de todos los usuarios de la web para comunicarse; está diseñado para atraer a los anunciantes. Los usuarios pueden añadir una imagen de sí mismos y con frecuencia pueden ser "amigos" con otros usuarios. En la mayoría de los servicios de redes sociales, los usuarios deben confirmar que son amigos antes de que estén vinculados. Por ejemplo, si Camila pone a Daniel como un amigo, entonces Daniel tendría que aprobar la solicitud de amistad de Camila antes de que se colocaran como amigos. Algunos sitios de redes sociales tienen unos "favoritos" que no necesita la aprobación de los demás usuarios. Las redes sociales por lo general tienen controles de privacidad que permiten al usuario elegir quién puede ver su perfil o entrar en contacto con ellos, entre otras funciones.

2.2. Aplicación Movil

Una aplicación móvil o *app* es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Por lo general se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS,

BlackBerry OS, Windows Phone, entre otros. Existen aplicaciones móviles gratuitas u otras de pago, donde en promedio el 20-30% del costo de la aplicación se destina al distribuidor y el resto es para el desarrollador.[1] El término *app* se volvió popular rápidamente, tanto que en 2010 fue listada como *Word of the Year* (Palabra del Año) por la American Dialect Society.[2]

El desarrollo de aplicaciones para dispositivos móviles requiere tener en cuenta las limitaciones de estos dispositivos. Los dispositivos móviles funcionan con batería y tienen procesadores menos poderosos que los ordenadores personales. Los desarrollos de estas aplicaciones también tienen que considerar una gran variedad de tamaños de pantalla, datos específicos de software y configuraciones. El desarrollo de aplicaciones móviles requiere el uso de entorno de desarrollo integrados. Las aplicaciones móviles suelen ser probadas primero usando emuladores y más tarde se ponen en el mercado en periodo de prueba. Actualmente un gran número de empresas se dedica a la creación profesional de aplicaciones. Aún así, han surgido páginas web como Mobincube[3] donde un usuario común puede crear aplicaciones de manera gratuita y sin conocimiento de programación.

2.3. Servicio de alojamiento de archivos

Un servicio de alojamiento de archivos, servicio de almacenamiento de archivos *online*, o centro de medios *online* es un servicio de alojamiento de Internet diseñado específicamente para alojar contenido estático, mayormente archivos grandes que no son páginas web. En general estos servicios permiten acceso web y FTP. Pueden estar optimizados para servir a muchos usuarios (como se indica con el término "alojamiento") o estar optimizados para el almacenamiento de usuario único (como se indica con el término "almacenamiento"). Algunos servicios relacionados son el alojamiento de videos, alojamiento de imágenes, el almacenamiento virtual y el copiado de seguridad remoto.

2.4. Teléfono inteligente (SmarthPhone).

Un teléfono inteligente (*smartphone* en inglés) es un teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional. El término «inteligente», que se utiliza con fines comerciales, hace referencia a la capacidad de usarse como un ordenador de bolsillo, y llega incluso a reemplazar a un ordenador personal en algunos casos.

Generalmente, los teléfonos con pantallas táctiles son los llamados *teléfonos inteligentes*, pero el soporte completo al correo electrónico parece ser una característica

indispensable encontrada en todos los modelos existentes y anunciados desde 2007. Casi todos los teléfonos inteligentes también permiten al usuario instalar programas adicionales, habitualmente incluso desde terceros, hecho que dota a estos teléfonos de muchísimas aplicaciones en diferentes terrenos; sin embargo, algunos teléfonos son calificados como inteligentes aun cuando no tienen esa característica.

Entre otros rasgos comunes está la función multitarea, el acceso a Internet vía Wi-Fi o red 3G, función multimedia (cámara y reproductor de videos/mp3), a los programas de agenda, administración de contactos, acelerómetros,GPS y algunos programas de navegación, así como ocasionalmente la habilidad de leer documentos de negocios en variedad de formatos como PDF y Microsoft Office.

Los sistemas operativos móviles más frecuentes utilizados por los teléfonos inteligentes son Android (de Google), iOS (de Apple), Windows Phone (de Microsoft) y BlackBerry OS (de BlackBerry). Otros sistemas operativos de menor uso sonBada (de Samsung), Symbian (de Nokia), Firefox OS (de Mozilla), MeeGo (de Moblin y Maemo), webOS, Windows CE, etc. Desde 2012 se ha anunciado Ubuntu Touch como próximo contendiente en este segmento.

Según datos del tercer trimestre de 2013 en cuanto a uso de sistemas operativos móviles en teléfonos inteligentes en todo el mundo, estos fueron los resultados[1] :

- Android 81,9 %
- iOS 12,1 %
- Windows Phone 3,6 %
- BlackBerry OS 1,8 %
- Bada 0,3 %
- Symbian OS 0,2 %
- Otros 0,2 %

3. Contexto tecnológico

Para hacer posible el desarrollo del proyecto hemos tenido que decidir que tecnologías utilizaríamos para resolver o facilitarnos la labor. En este apartado analizaremos cada una de las herramientas utilizada o lenguaje de programación.

3.1. JAVA

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos

utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.[1] [2]

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Class path.

En este proyecto java es el lenguaje de programación en el que está escrita toda la lógica del programa, debido a que es el lenguaje que utiliza el kit de desarrollo de Android (Android SDK).

3.2 Eclipse

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. LaFree Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (*wizards*) para creación de proyectos, clases, tests, etc., y refactorización.

Asimismo, a través de "plugins" libremente disponibles es posible añadir control de versiones con Subversion. e integración con Hibernate.

Por último, el complemento ADT (Android Development Tools) ofrecido por Android SDK (Software Development Kit) nos ofrece una herramienta perfecta para desarrollar programas android junto con eclipse.

3.11 Android SDK.

El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen Linux (cualquier

distribución moderna), Mac OS X 10.4.9 o posterior, y Windows XP o posterior. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Eclipse junto con el complemento ADT (Android Development Tools plugin), aunque también puede utilizarse un editor de texto para escribir ficheros Java y Xml y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones. Además, pueden controlarse dispositivos Android que estén conectados (e.g. reiniciarlos, instalar aplicaciones en remoto).

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (este directorio necesita permisos de superusuario , root, por razones de seguridad). Un paquete APK incluye ficheros .dex (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

3.3 Android

Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, inicialmente desarrollado por Android, Inc. Google respaldó económicamente y más tarde compró esta empresa en 2005. Android fue presentado en 2007 junto la fundación del Open Handset Alliance: un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

El éxito del sistema operativo se ha convertido en objeto de litigios sobre patentes en el marco de las llamadas *Smartphone patent wars* entre las empresas de tecnología. Según documentos secretos filtrados en 2013 y 2014, el sistema operativo es uno de los objetivos de las agencias de inteligencia internacionales.

Los componentes principales del sistema operativo de Android (cada sección se describe en detalle):

- Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

- Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

- Runtime de Android: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx".

- Núcleo Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

3.4 XML

XML, siglas en inglés de *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. (Bases de datos Silberschatz).

XML no ha nacido sólo para su aplicación para Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan*bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.
- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

3.5 GitHub

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. [3] Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.

El diseño de Git se basó en BitKeeper y en Monotone.

El diseño de Git resulta de la experiencia del diseñador de Linux, Linus Torvalds, manteniendo una enorme cantidad de código distribuida y gestionada por mucha gente, que incide en numerosos detalles de rendimiento, y de la necesidad de rapidez en una primera implementación.

Entre las características más relevantes se encuentran:

- Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal. Una presunción fundamental en Git es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
- Gestión distribuida. Al igual que Darcs, BitKeeper, Mercurial, SVK, Bazaar y Monotone, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
- Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH. Git también puede emular servidores CVS, lo que habilita el uso de clientes CVS pre-existentes y módulos IDE para CVS pre-existentes en el acceso de repositorios Git.
- Los repositorios Subversion y svk se pueden usar directamente con git-svn.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial). Esto existía enMonotone.
- Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a CVS, que trabaja con base en cambios de fichero, pero mejora el trabajo con afectaciones de código que concurren en operaciones similares en varios archivos.

- Los renombrados se trabajan basándose en similitudes entre ficheros, aparte de nombres de ficheros, pero no se hacen marcas explícitas de cambios de nombre con base en supuestos nombres únicos de nodos de sistema de ficheros, lo que evita posibles, y posiblemente desastrosas, coincidencias de ficheros diferentes en un único nombre.
- Realmacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo.

GitHub es una forja para alojar proyectos utilizando el sistema decontrol de versiones Git. Utiliza el framework Ruby on Rails por*GitHub, Inc.* (anteriormente conocida como *Logical Awesome*).

Desde enero de 2010, GitHub opera bajo el nombre de *GitHub, Inc.*

El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

- Wiki para cada proyecto.
- Página web para cada proyecto.[1]
- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto.
- Funcionalidades como si se tratase de una red social, como por ejemplo: seguidores.

3.6 Balsamik Mockups

Balsamik Mockups es un software de diseño de borradores para interfaces web, escritorio y móvil.

Se trata de una aplicación extremadamente sencilla e intuitiva con la que hemos diseñado los bocetos de la aplicación.

3.7 Lenguaje Unificado de Modelado.

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir

un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no específica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

3.7 Uml Star

StarUML was an open source UML tool, licensed under a modified version of GNU GPL. After being abandoned for some time, the project had a last revival to move from Delphi to Java/Eclipse and then stopped again. However, the community is still active and many topics are discussed on the forums.[citation needed]

The stated goal of the project was to replace larger, commercial applications such as Rational Rose and Borland Together.

StarUML supports most of the diagram types specified in UML 2.0. It is currently missing object, package, timing and interaction overview diagrams (though the first two can be adequately modeled through the class diagrameditor).

StarUML was written in Delphi, which is one of the reasons[1] why it was abandoned for a long time.[citation needed] Since December 2005 StarUML was not updated anymore, although some external modules were updated[2]

At the end of 2011 StarUML was forked. Under the name WhiteStarUML it is now actively developed in Object Pascal. WhiteStarUML is based on the last version of StarUML (5.0) and started with that version number. The current version 5.5 was released on April 13th 2014. [3]

3.8 JSON

JSON, acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquiernavegador web.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o del rendimiento de los mismos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de eval() y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo, Google, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el no disponer de procesamiento XSLT para manipular los datos en el cliente.

Si bien es frecuente ver JSON posicionado *contra* XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP hacen necesario soportar ambos formatos.

Cada vez hay más soporte de JSON mediante el uso de paquetes escritos por terceras partes. La lista de lenguajes soportados incluye ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML,Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Lua y Visual FoxPro.

En diciembre de 2005 Yahoo! comenzó a dar soporte opcional de JSON en algunos de sus servicios web.[1]

El término JSON está altamente difundido en los medios de programación, sin embargo, es un término mal descrito ya que en realidad es solo una parte de la definición del estándar ECMA-262 en que está basado Javascript. De ahí que ni Yahoo, ni Google emplean JSON, sino LJS. Una de las cualidades intrínsecas de Javascript denominada LJS (Literal Javascript) facilita el flujo de datos e incluso de funciones, para la cual no requiere la función eval() si son datos los que se transfieren como en el caso de XML. Todo lo referente a transferencia de datos en todos sus tipos, incluyendo arrays, booleans, integers, etc. no requieren de la función eval(), y es precisamente en eso en donde supera por mucho JavaScript al XML, si se utiliza el LJS y no la incorrecta definición de JSON.

3.9 Facebook SDK para Android

Integrate with Facebook to help you build engaging social apps and get more installs.

What is in the Android SDK?

Login

Let people easily sign in to your app with their Facebook identity. If they have already signed in on the Facebook for Android app, they don't have to reenter their username and password.

Share

Allow people using your app to share or send a message from your app to Facebook. When people on Facebook engage with these posts they are directed to your app.

Custom Stories

Open Graph lets apps tell stories on Facebook through a structured, strongly typed API. This document shows you how to use the Object API to share stories from your app.

App Links

Link posts, stories and requests shared from your app back to your app. Handle these incoming links to direct people to relevant parts within your app.

App Events

App Events allow you to understand the actions taken by users within your app and to measure the effectiveness of your Mobile App Ads.

Ads

Reach the right people where they are most engaged, in News Feed. Drive installs with Mobile App Install Ads. Increase user engagement with Mobile App Engagement Ads.

Graph API

The Graph API allows you to get data in and out of Facebook's social graph. You can query data, post new stories, upload photos, and more.

Send Requests

Allow users to send requests to their friends from your app.

App Center

For the over one billion people that use Facebook, the App Center is the central place to find great social apps.

4. Descripción de la aplicación.

4.1. Descripción de la aplicación.

FotoCloud permitirá subir que se encuentren en tu galería o que vayas a tomar en ese momento, a cualquiera de las redes sociales o contenedor de fotografías que implemente la aplicación (Facebook en esta primera versión) y te encuentres identificado.

Así el usuario encontrará una forma ágil de conservar sus fotografías en caso de pérdida del terminal portátil o de avería.

Por otra parte también es una sencilla herramienta para compartir experiencias de forma instantánea con los contactos de esa red social o servicio evitándose la navegación por las aplicaciones oficiales las cuales incluyen un abanico mas amplio de funcionalidades, lo que las hace sensiblemente mas pesadas a la hora de tratar esa red social como un simple repositorio de instantáneas y poco mas.

Por último también ofrece la posibilidad de descargarte esas fotografías rápidamente al terminal donde se esté ejecutando la aplicación.

4.2. Perfiles de la aplicación

En nuestro caso, la aplicación cuenta con dos perfiles de usuario bien diferenciados "Usuario Identificado" y "Usuario No Identificado", esto no es más que la descripción del estado en el que se encuentra un usuario a lo largo de la sesión de uso, si es la primera vez que ejecutas la aplicación y además no te encuentras identificado en ninguna otra aplicación del mismo tipo, (como pudiera ser la oficial de Facebook) tu tocken de identificación se encontrará vacío y tu estado será de usuario no identificado, en cambio si se detecta algún tocken de identificación, o lo consigues autentificándote en la aplicación, pasarás a ser usuario identificado.

4.2.1 Usuario No Identificado

Los usuarios con esté perfil tienen un acceso muy limitado a las funcionalidades de la aplicación, debido a que la aplicación por si misma no te ofrece nada.

Los usuarios que se encuentran en esta situación es debido a que no disponen de ningún tipo de identificación válida para acceder a sus fotos de Facebook.

Si te encuentras identificado en la aplicación oficial de Facebook, Fotocloud detectará automáticamente el tocken de identificación y serás “Usuario Identificado” inmediatamente.

Por otra parte, también podrás conseguir tu identificación desde dentro de la aplicación haciendo uso de los botones de identificación.

4.2.2 Usuario Identificado

Una vez que te identificas como usuario del servicio de almacenamiento de fotos ya tienes acceso a todas las funcionalidades de la aplicación relacionadas con ese servicio,

4.3. Flujos principales de la aplicación.

La aplicación cuenta con cuatro escenas principales que conforman el flujo de la aplicación:

1. Pantalla de identificación: Incluye los accesos al formulario de identificación de cada uno de los servicios de almacenamiento de fotos implementados. (sólo Facebook en la primera versión)
2. Pantalla con el listado de álbumes: Contiene los álbumes residentes en la cuenta identificada.
3. Pantalla con el listado de fotos: Incluye las fotos que forman parte del álbum seleccionado.
4. Vista de fotos individual: Escena en la cual se muestran las fotos de un álbum uno por uno.

Además desde cualquier pantalla de la aplicación a excepción de la pantalla de identificación ofrecerá la opción de subir fotografía al álbum llamado fotoCloud.

A continuación se detallarán las funcionalidades de cada pantalla incluida.

4.3.1. Pantalla de identificación.

Esta pantalla es la primera que se muestra al iniciar la aplicación. En el caso de la primera versión se omitirá siempre que el sistema encuentre un tocken de identificación válido.

4.3.1.1. Restricciones.

Ninguna, todos los usuarios pueden accedes a la pantalla de identificación.

4.3.1.2 Funcionalidades.

Requerimiento	Autenticación
Descripción	Al iniciar la aplicación esta será la primera pantalla en aparecer, te dará la opción de identificarte en cualquiera de los servicios implementados, usando tu nombre de usuario y contraseña que deberás haber conseguido dándote de alta previamente en la página oficial del servicio.

4.3.2. Pantalla con la lista de álbumes.

En esta pantalla se lista el título de los álbumes pertenecientes a la cuenta del servicio en el que hayas iniciado sesión o que ya te encontraras identificado. Desde esta pantalla puedes accedes a la opción de menú para subir una foto al álbum fotoCloud y también en ese mismo menú tienes la opción de deslogearte.

4.3.1.1. Restricciones.

Solo los usuarios identificados pueden acceder a esta escena.

4.3.1.2 Funcionalidades.

Requerimiento	Lista álbumes
Descripción	Una vez te encuentras en la escena, automáticamente el sistema te muestra una lista con los títulos de los álbumes relacionados con la cuenta activa.

Requerimiento	Subir Foto
Descripción	Desde la identificación en adelante se añade la opción de subir foto en el menú de la aplicación, (situado en sitios distintos en función de tu versión de android) después de accionar esta funcionalidad tendrás que elegir entre importar la foto desde la galería android o tomar una foto usando la cámara del terminal.

Requerimiento	Finalizar sesión
Descripción	Desde la identificación en adelante se añade la opción de salir de la sesión en el menú de la aplicación. Esta funcionalidad finaliza la sesión con el servicio en el q te encuentres en este momento, expulsándote a la pantalla de inicio de sesión.

4.3.3. Pantalla con la lista de fotos.

A esta pantalla se accede tras seleccionar un álbum, desde la pantalla del listado de álbumes.

4.3.1.1. Restricciones.

Solo los usuarios identificados pueden acceder a esta escena.

4.3.1.2 Funcionalidades.

Requerimiento	Lista de fotos
Descripción	Una vez te encuentras en la escena, automáticamente el sistema te muestra una lista con las fotos pertenecientes al álbum seleccionado en forma de vista de rejilla, mediante un sistema de carga ágil llamado "lazy load" que consiste en cargar asíncronamente las imágenes, de forma que no da sensación de pesadez a la hora de desplazarte por el álbum.

Requerimiento	Subir Foto
Descripción	Desde la identificación en adelante se añade la opción de subir foto en el menú de la aplicación, (situado en sitios distintos en función de tu versión de android) después de accionar esta funcionalidad tendrás que elegir entre importar la foto desde la galería android o tomar una foto usando la cámara del terminal.

Requerimiento	Finalizar sesión
Descripción	Desde la identificación en adelante se añade la opción de salir de la sesión en el menú de la aplicación. Esta funcionalidad finaliza la sesión con el servicio en el q te encuentres en este momento, expulsándote a la pantalla de inicio de sesión.

Requerimiento	Descargar Foto
Descripción	En esta pantalla tienes la opción de descargarte la foto que quieras, solo tienes que hacer una pulsación larga sobre la foto que desees, poco después se mostrará un mensaje que te pedirá que confirmes la descarga.

4.3.4. Vista de fotos individual.

A esta pantalla se accede tras seleccionar una foto, desde la pantalla del listado de fotos.

4.3.4.1. Restricciones.

Solo los usuarios identificados pueden acceder a esta escena.

4.3.4.2. Funcionalidades.

Requerimiento	Foto individual a pantalla completa
Descripción	Esta escena tiene como única función mostrarte la foto seleccionada a tamaño de pantalla completa.

Requerimiento	Navegación por el álbum
Descripción	Una vez veas una sola foto a pantalla completa tienes la oportunidad de navegar a la foto anterior y a la siguiente mediante el gesto con el dedo típico para apartar la foto hacia la derecha o izquierda.

4.4. Diagrama de Clases.

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.).

Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño).

El diagrama de clases de más alto nivel, será lógicamente un dibujo de los paquetes que componen el sistema. Las clases se documentan con una descripción de lo que hacen, sus métodos y sus atributos.

Las relaciones entre clases se documentan con una descripción de su propósito, sus objetos que intervienen en la relación y su opcionalidad (cuando un objeto es opcional el que intervenga en una relación.

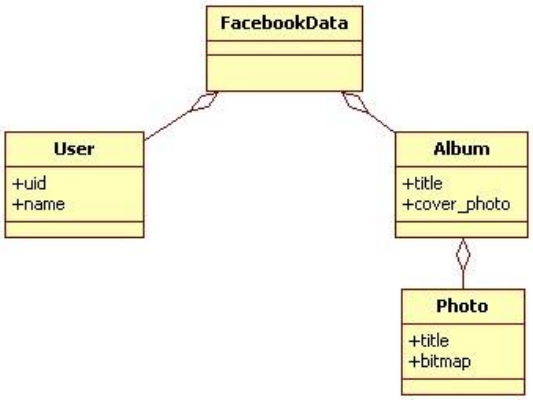

Ilustración 1 Diagrama de objetos

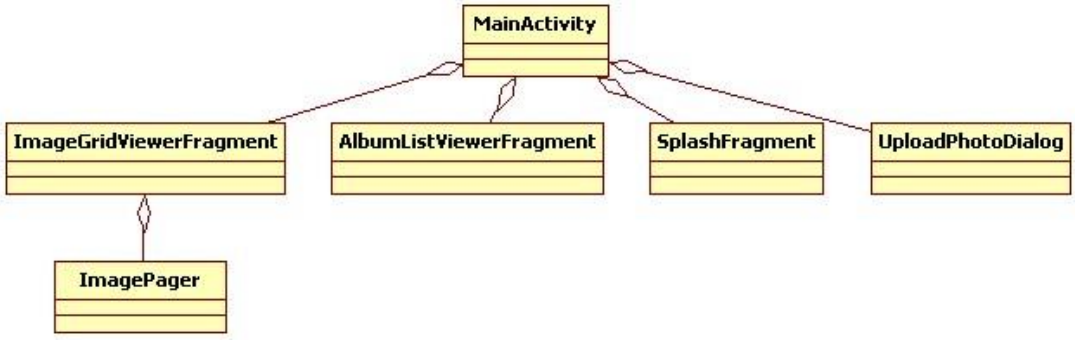

Ilustración 2 Diagrama de clases

5. Diseño

En este apartado describiremos la parte mas compleja e importante del proceso de desarrollo de una aplicación, el diseño.

La dividiremos en 3 subapartados:

- Diseño arquitectónico.
- Hilos y tareas asíncronas.
- Diseño de las escenas.
- Código de ejemplo.

5.1. Diseño arquitectónico.

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema.

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco

Una arquitectura de software se selecciona y diseña con base en objetivos (requerimientos) y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

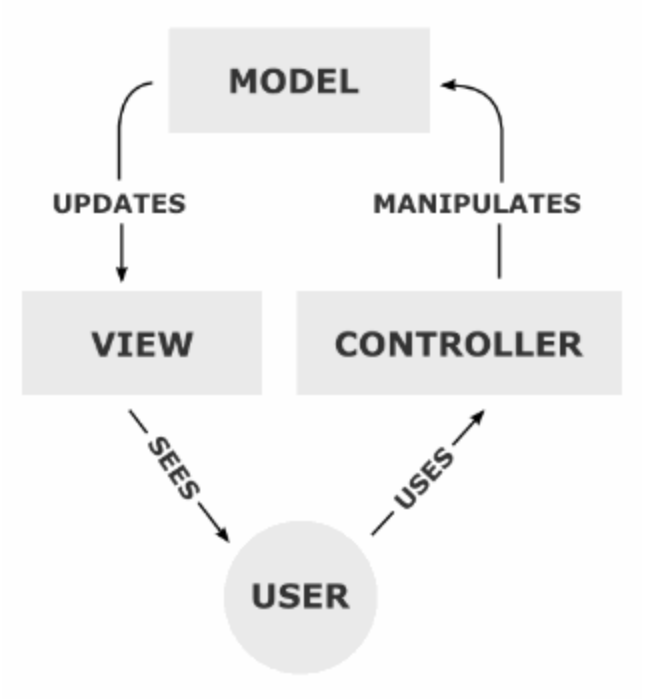

Ilustración 3 MVC

5.1.1. Modelo.

Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

Concretamente en el proyecto el modelo está compuesto por los diferentes objetos que forman la estructura interna del programa (usuario, álbum, foto) que tienen como raíz al objeto FacebookData (el cuál posteriormente será reemplazado por un objeto más genérico para que esta estructura sea reutilizada también en redes sociales como Picassa que serán implementadas en futuras versiones).

```java
package com.app.objects;

import java.util.ArrayList;
import java.util.List;

public class FacebookData {
	private User user;
	private List<Album> albums;

	public FacebookData(User user, ArrayList<Album> albums) {
		super();
		this.user = user;
		this.albums = albums;
	}
	public FacebookData(){this(null,new ArrayList<Album>());}

	public User getUser() {
		return user;
	}

	public void setUser(User user) {
		this.user = user;
	}

	public List<Album> getAlbums() {
		return albums;
	}

	public void setAlbums(List<Album> albums) {
		this.albums = albums;
	}

	public void addAlbum(Album album){
		this.albums.add(album);
	}


}
```

```java
package com.app.objects;

import java.util.ArrayList;
import java.util.List;

public class Album {
	private String title;
	private Photo cover_photo;
	private List<Photo> photos;

	public Album(){
		this.title="no title";
		this.cover_photo=null;
		this.photos=new ArrayList<Photo>();
	}

	public Album(String title,Photo cover_photo,ArrayList<Photo> photos){
		this.title=title;
		this.cover_photo=cover_photo;
		this.photos=photos;
	}
	public Album(String title,Photo cover_photo){
		this.title=title;
		this.cover_photo=cover_photo;
		this.photos=new ArrayList<Photo>();
	}

	public void addPhoto(Photo photo){
		this.photos.add(photo);
	}

	public String getTitle() {
		return title;
	}

	public void setTitle(String title) {
		this.title = title;
	}

	public Photo getCover_photo() {
		return cover_photo;
	}

	public void setCover_photo(Photo cover_photo) {
		this.cover_photo = cover_photo;
	}

	public List<Photo> getPhotos() {
		return photos;
	}

	public void setPhotos(List<Photo> photos) {
		this.photos = photos;
	}

}
```

5.1.2. Vista.

Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

La vista en este caso comprende los .xml. En estos archivos tenemos definida la estructura de toda la interface de usuario separada por escenas y nos permite una sincronización, flujo de información y eventos con el controlador.

Además en xml encontramos definidas las cadenas de texto que usaremos durante el programa en varios idiomas, lo que nos permite que la aplicación sea multilenguaje.

La misma estrategia de definición de objetos nos permite definir los estilos visuales con los que son dibujadas las líneas de texto, los botones o los menús.

```xml
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical"
              android:background="#000000"
              android:id="@+id/splashFragment">

    <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="30dp"
            android:gravity="center_horizontal"
            android:orientation="horizontal" >
        <ImageView
                android:id="@+id/splash_icon"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginLeft="10dp"
                android:gravity="center"
                android:src="@drawable/icon" />
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:gravity="center"
                android:textColor="#AFDEFE"
                android:textSize="28sp"
                android:typeface="serif"
                android:textStyle="italic"
                android:text="@string/app_name" />
    </LinearLayout>


    <TextView
            android:id="@+id/profile_name"
            android:layout_width="174dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="35dp"
            android:lines="2"
            android:textSize="17sp"
            android:text="@string/get_started"
            android:layout_gravity="center_horizontal"
            android:gravity="center_horizontal"/>

    <com.facebook.widget.LoginButton
            android:id="@+id/login_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="30dp"
            android:layout_marginBottom="30dp" />


      </LinearLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">FotoCloud</string>
    <string name="menu_settings">Settings</string>
    <string name="app_id">390843811014408</string>
    <string name="title_activity_splash_fragment">SplashFragment</string>
    <string name="get_started">To get started, log in using Facebook</string>
    <string name="title_activity_selection_fragment">SelectionFragment</string>
    <string name="settings">Settings</string>
    <string name="clearuser">LogOut</string>
    <string name="title_activity_photo_grid_viewer_fragment">PhotoGridViewerFragment</string>
    <string name="uploadPhoto">Upload Photo</string>
    <string name="title_activity_album_list_viewer_fragment">AlbumListViewerFragment</string>
    <string name="dialog_download_photo">Download Photo?</string>
    <string name="yes">Yes</string>
    <string name="no">No</string>
    <string name="camera"> Use Camera</string>
    <string name="disk">Select File</string>
    <string name="descr_image">Image</string>


</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">FotoCloud</string>
    <string name="menu_settings">Opciones</string>
    <string name="app_id">390843811014408</string>
    <string name="title_activity_splash_fragment">SplashFragment</string>
    <string name="get_started">Para empezar, Identificate usando Facebook</string>
    <string name="title_activity_selection_fragment">SelectionFragment</string>
    <string name="settings">Opciones</string>
    <string name="clearuser">Desconectar</string>
    <string name="title_activity_photo_grid_viewer_fragment">PhotoGridViewerFragment</string>
    <string name="uploadPhoto">Subir Foto</string>
    <string name="title_activity_album_list_viewer_fragment">AlbumListViewerFragment</string>
    <string name="dialog_download_photo">¿Descargar Foto?</string>
    <string name="yes">Si</string>
    <string name="no">No</string>
    <string name="camera">Usar Cámara</string>
    <string name="disk">Seleccione Archivo</string>
    <string name="descr_image">Imagen</string>


      </resources>
```

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_settings"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/menu_settings"/>


      </menu>
```

5.1.3. Controlador.

jResponde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

Esta función en el código la desempeñan clases .java que se encuentran escuchando los eventos que puedan ocurrir en la vista .xml y realizan las acciones que les hayan sido asigandas a dichos eventos, como puede ser la pulsación de un botón.

```java
package com.app.fotocloud.facebook;

import java.util.ArrayList;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.actionbarsherlock.app.ActionBar;
import com.actionbarsherlock.app.SherlockListFragment;
import com.app.fotocloud.MainActivity;
import com.app.fotocloud.R;
import com.app.objects.Album;
```

```
import com.app.objects.FacebookData;
import com.app.objects.User;
import com.facebook.Request;
import com.facebook.Response;
import com.facebook.Session;
import com.facebook.SessionState;
import com.facebook.UiLifecycleHelper;
import com.facebook.model.GraphUser;

public class AlbumListViewerFragment extends SherlockListFragment {

	private String[] albumArray;

	private FacebookData facebookData;
	private List<String> photoIdList;

	private UiLifecycleHelper uiHelper;
	private Session.StatusCallback callback;

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

		getActivity().setContentView(R.layout.list_view_fragment);

		facebookData=new FacebookData();
		photoIdList = new ArrayList<String>();

		uiHelper = new UiLifecycleHelper(getActivity(), callback);
	    uiHelper.onCreate(savedInstanceState);

		callback = new Session.StatusCallback() {
		    @Override
		    public void call(final Session session, final SessionState state, final
Exception exception) {
		        onSessionStateChange(session, state, exception);
		    }

			private void onSessionStateChange(Session session,
					SessionState state, Exception exception) {
				// TODO Auto-generated method stub

			}
		};
		//albumArray=((MainActivity)getSherlockActivity()).getAlbumNames();
		Session session = Session.getActiveSession();
	    if (session != null && session.isOpened()) {
		if(((MainActivity)getSherlockActivity()).isNetworkAvailable()){

			fillFacebookData(session);
		}
		else
			Toast.makeText(getSherlockActivity().getApplicationContext(), "Internet
Connection Unavailable", Toast.LENGTH_LONG).show();

		 if(facebookData.getAlbums().size()>0){
```

```
            }
        }

    }

    private void fillFacebookData(Session session) {
        makeMeRequest(session);
        //while(facebookData.getAlbums().size()==0){};
    }

    private void fillAlbums(final Session session) {
        Request request = Request.newGraphPathRequest(session,
""+facebookData.getUser().getUid()+"/albums", new Request.Callback() {
                JSONObject graphResposte=null;
                JSONArray jsonArray = null;
                @Override
                public void onCompleted(Response response) {
                    if(response.getGraphObject()!=null){
                        graphResposte = response.getGraphObject().getInnerJSONObject();
                        try {
                            jsonArray = graphResposte.getJSONArray("data");


                            for(int i = 0; i < jsonArray.length(); i++){

                                JSONObject c = jsonArray.getJSONObject(i);

                                String id = c.getString("id");
                                String name = c.getString("name");

                                photoIdList.add(id);
                                Album album=new Album();
                                album.setTitle(name);
                                facebookData.addAlbum(album);
                        }
                            albumArray=new String[facebookData.getAlbums().size()];
                        for(int i=0;i<facebookData.getAlbums().size();i++){
                            albumArray[i]=facebookData.getAlbums().get(i).getTitle();
                        }
                        if(albumArray!=null){
                            setListAdapter(new
ArrayAdapter<String>(getSherlockActivity().getApplicationContext(),

    android.R.layout.simple_list_item_1, albumArray));
                        }
                        getView().setBackgroundColor(Color.BLACK);

                        } catch (JSONException e1) {
                            // TODO Auto-generated catch block

    Toast.makeText(getSherlockActivity().getApplicationContext(), "JSON_EXCEPTION",
Toast.LENGTH_LONG).show();
                            e1.printStackTrace();
                        }

                    }
                    else
                        Toast.makeText(getSherlockActivity().getApplicationContext(),
```

```
"NULL", Toast.LENGTH_LONG).show();
                    }

            });
            request.executeAsync();
        }

        private void makeMeRequest(final Session session) {
            // Make an API call to get user data and define a
            // new callback to handle the response.
            Request request = Request.newMeRequest(session, new Request.GraphUserCallback() {
                @Override
                public void onCompleted(GraphUser user, Response response) {
                    // If the response is successful
                    if (session == Session.getActiveSession()) {

                        if (user != null) {
                            //set userid and username
                           User userobj = new User(user.getId(),user.getName());
                           facebookData.setUser(userobj);
                           if(userobj.getName()!=null){
                                  final ActionBar actionBar =
getSherlockActivity().getSupportActionBar();
                                  actionBar.setDisplayHomeAsUpEnabled(true);
                                  actionBar.setTitle(userobj.getName());
                                  actionBar.setSubtitle("fotoCloud");
                           }
                           fillAlbums(session);
                        }
                    }
                    if (response.getError() != null) {
                        // Handle errors, will do so later.
                    }

                }
            });
            request.executeAsync();
        }


        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
              return super.onCreateView(inflater, container, savedInstanceState);
        }

        @Override
        public void onResume() {
            super.onResume();
            uiHelper.onResume();
        }

        @Override
        public void onAttach(Activity activity) {
              super.onAttach(activity);
              //Toast.makeText(getSherlockActivity().getApplicationContext(), "NAme:
"+facebookData.getUser().getName(), Toast.LENGTH_LONG).show();
        }
```

```
	@Override
	public void onActivityCreated(Bundle savedInstanceState) {
		super.onActivityCreated(savedInstanceState);
		if(albumArray!=null){
			setListAdapter(new
ArrayAdapter<String>(getSherlockActivity().getApplicationContext(),
                                                    android.R.layout.simple_list_item_1,
albumArray));
		}
		getView().setBackgroundColor(Color.BLACK);
	}

	@Override
	public void onListItemClick(ListView l, View v, int position, long id) {
		((MainActivity)getSherlockActivity()).callGridView(photoIdList.get(position));

	}



}
```

5.2. Hilos y tareas asíncronas.

Todos los componentes de una aplicación Android, tanto las actividades, los servicios, o los broadcast receivers se ejecutan en el mismo hilo de ejecución, el llamado hilo principal, main thread o GUI thread, que como éste último nombre indica también es el hilo donde se ejecutan todas las operaciones que gestionan la interfaz de usuario de la aplicación. Es por ello, que cualquier operación larga o costosa que realicemos en este hilo va a bloquear la ejecución del resto de componentes de la aplicación y por supuesto también la interfaz, produciendo al usuario un efecto evidente de lentitud, bloqueo, o mal funcionamiento en general, algo que deberíamos evitar a toda costa. Incluso puede ser peor, dado que Android monitoriza las operaciones realizadas en el hilo principal y detecta aquellas que superen los 5 segundos, en cuyo caso se muestra el famoso mensaje de "Application Not Responding" (ANR) y el usuario debe decidir entre forzar el cierre de la aplicación o esperar a que termine.

En esta aplicación no nos encontramos con la necesidad de realizar operaciones largas a lo largo de la ejecución del programa, las cuales deberíamos de realizar mediante tareas asíncronas para no dar sensación de bloqueo mientras se ejecuta la operación. En cambio sí que realizamos numerosas peticiones por internet que por su naturaleza es frecuente que se interrumpan o se ralenticen, por este motivo a partir de la versión de Android 4.0 no se permite realizar este tipo de actividades de forma síncrona

y por este motivo FaceBook SDK incorpora una forma de realizar las peticiones asíncronamente como se muesta en el siguiente código de ejemplo:

```
private void fillAlbums(final Session session) {
                Request request = Request.newGraphPathRequest(session,
""+facebookData.getUser().getUid()+"/albums", new Request.Callback() {
                        JSONObject graphResposte=null;
                        JSONArray jsonArray = null;
                        @Override
                        public void onCompleted(Response response) {
                                if(response.getGraphObject()!=null){
                                        graphResposte = response.getGraphObject().getInnerJSONObject();
                                        try {
                                                jsonArray = graphResposte.getJSONArray("data");


                                                for(int i = 0; i < jsonArray.length(); i++){

                                                        JSONObject c = jsonArray.getJSONObject(i);

                                                        String id = c.getString("id");
                                                        String name = c.getString("name");

                                                        photoIdList.add(id);
                                                        Album album=new Album();
                                                        album.setTitle(name);
                                                        facebookData.addAlbum(album);
                                    }
                                                albumArray=new String[facebookData.getAlbums().size()];
                                        for(int i=0;i<facebookData.getAlbums().size();i++){
                                                albumArray[i]=facebookData.getAlbums().get(i).getTitle();
                                        }
                                        if(albumArray!=null){
                                                setListAdapter(new
ArrayAdapter<String>(getSherlockActivity().getApplicationContext(),

        android.R.layout.simple_list_item_1, albumArray));
                                        }
                                        getView().setBackgroundColor(Color.BLACK);

                                        } catch (JSONException e1) {
                                                // TODO Auto-generated catch block
                                                Toast.makeText(getSherlockActivity().getApplicationContext(),
"JSON_EXCEPTION", Toast.LENGTH_LONG).show();
                                                e1.printStackTrace();
                                        }

                                }
                                else
                                        Toast.makeText(getSherlockActivity().getApplicationContext(), "NULL",
Toast.LENGTH_LONG).show();
                        }

                });
                request.executeAsync();


                }
```

5.3. Diseño de las pantallas.

En este apartado encontraremos en diseño en formato de boceto de las pantallas que comprenden la aplicación.

5.3.1. Pantalla de inicio.

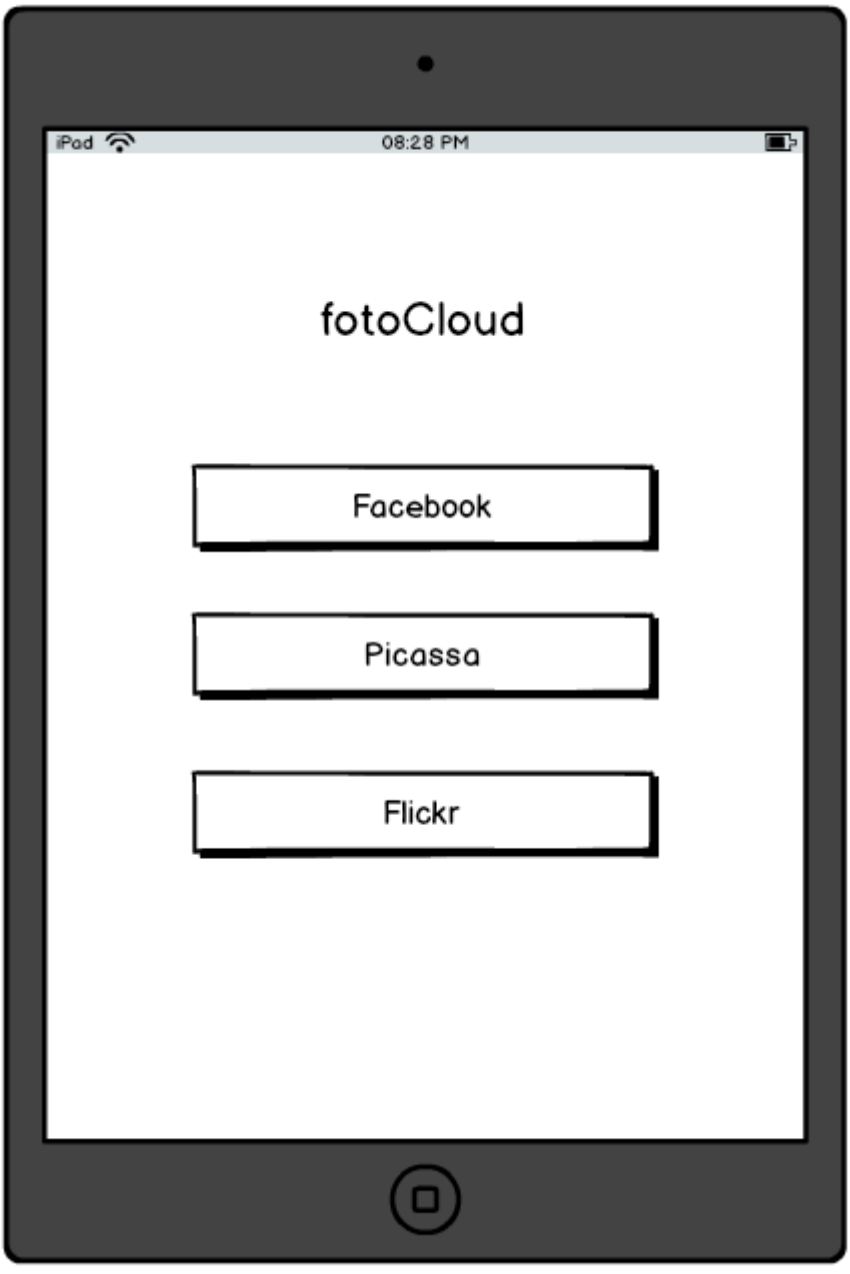

Ilustración 4 Boceto principal

5.3.2. Pantalla de inicio de sesión.

Ilustración 5 Boceto inicia sesión en FaceBook

5.3.3. Pantalla de listado de álbumes.

Ilustración 6 Lista álbumes

5.3.4. Pantalla de listado de fotos.

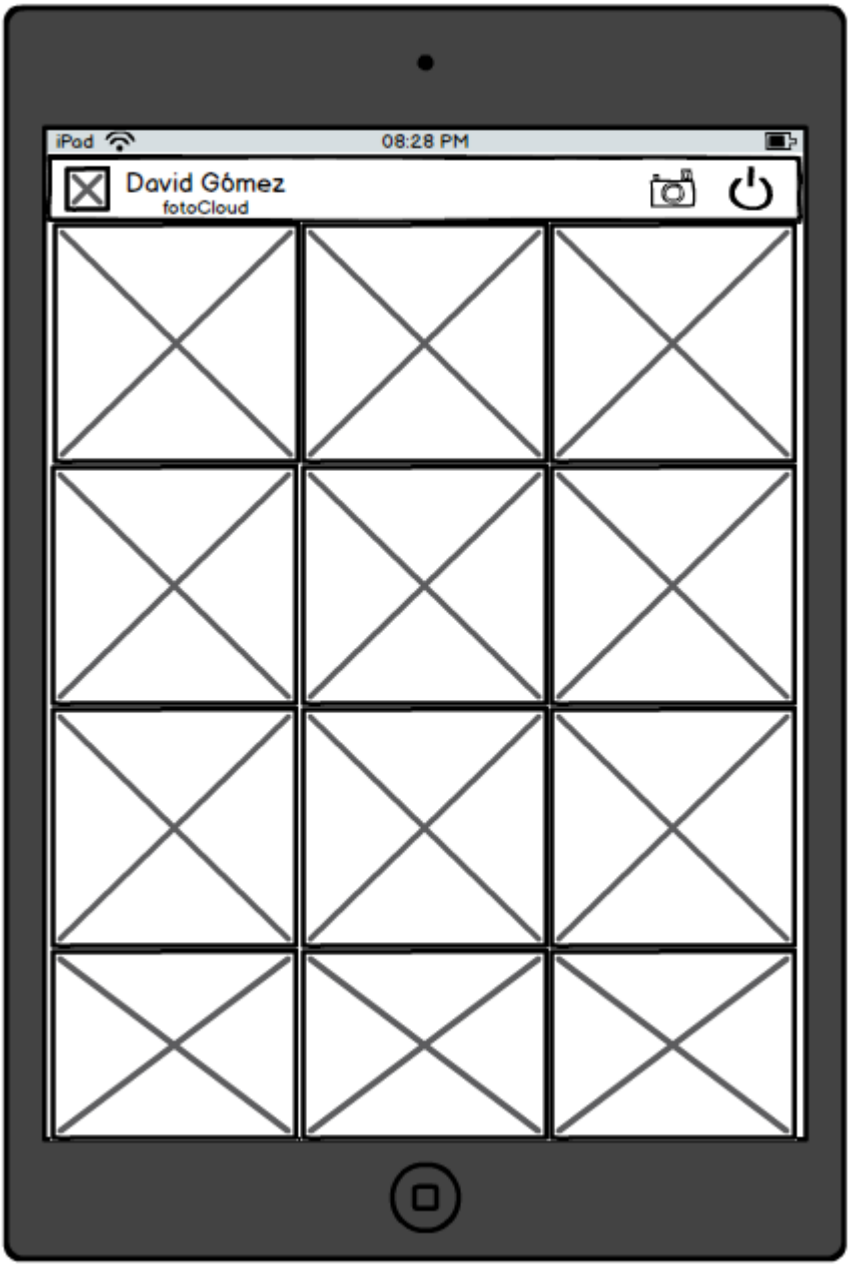

Ilustración 7 Lista Fotos

5.3.5. Pantalla de subir foto.

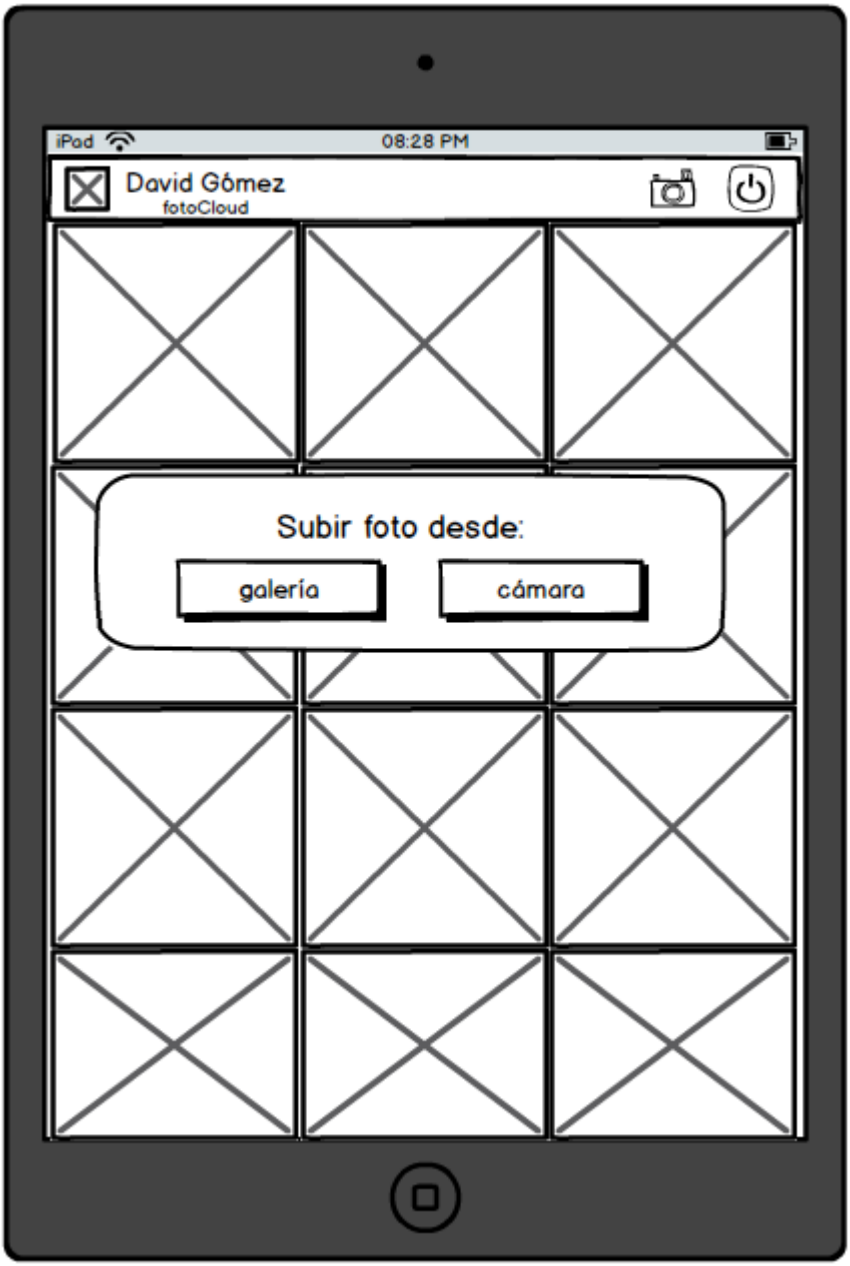

Ilustración 8 Subir foto

5.4. Código de ejemplo.

En esta sección mostraremos el flujo de funcionamiento de la aplicación a nivel alto.

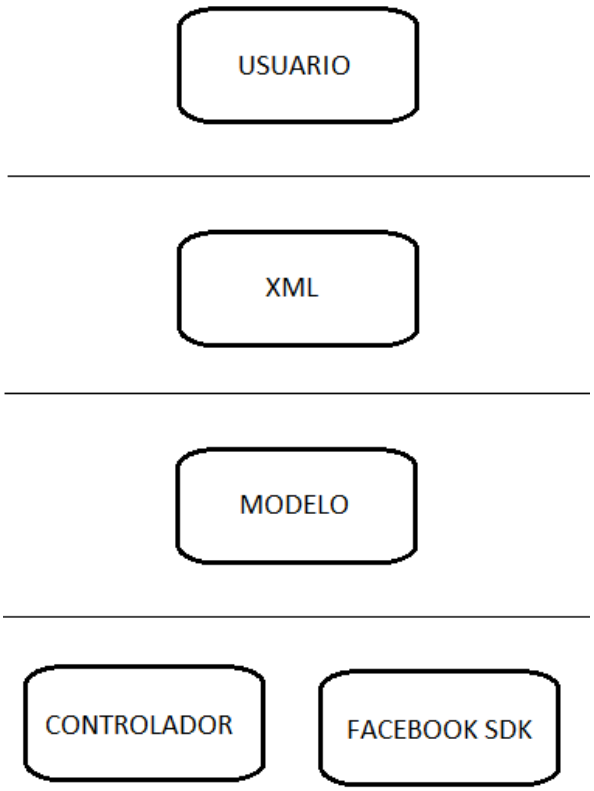

Ilustración 9 Flujo aplicación

5.4.1. XML

Código de ejemplo de “ac_image_grid.xml”:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/gridViewGroup" >

<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:horizontalSpacing="4dip"
    android:numColumns="3"
    android:stretchMode="columnWidth"
    android:verticalSpacing="4dip"
    android:padding="4dip"
/>
</LinearLayout>
```

5.4.2. Modelo

Código de ejemplo de “ImageGridFragment.java”

```java
package com.app.fotocloud.facebook;

import java.util.ArrayList;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;

import com.actionbarsherlock.app.SherlockFragment;
import com.app.fotocloud.MainActivity;
import com.app.fotocloud.R;
import com.app.fotocloud.common.ImagePagerActivity;
import com.facebook.Request;
import com.facebook.Response;
import com.facebook.Session;
import com.nostra13.universalimageloader.cache.disc.naming.Md5FileNameGenerator;
import com.nostra13.universalimageloader.core.DisplayImageOptions;
import com.nostra13.universalimageloader.core.ImageLoader;
import com.nostra13.universalimageloader.core.ImageLoaderConfiguration;
import com.nostra13.universalimageloader.core.assist.QueueProcessingType;

public class ImageGridFragment extends SherlockFragment{

	private ImageLoader imageLoader;
	private AbsListView listView;

	List<String> imageUrlsList;
	String[] imageUrls;
	String albumId;

	DisplayImageOptions options;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		getActivity().setContentView(R.layout.ac_image_grid);

		imageLoader =ImageLoader.getInstance();

	imageLoader.init(ImageLoaderConfiguration.createDefault(getActivity()));

		//Fill Image URLs
		imageUrlsList = new ArrayList<String>();
		albumId=getArguments().getString("albumId");
```

```
            getPhotoUrls(albumId);


      }
      /*public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
          View view = inflater.inflate(R.layout.ac_image_grid, container, false);
          ///((Button)view.findViewById(R.id.button)).setOnClickListener(this);

          return view;
      }*/

      private void getPhotoUrls(String albumId2) {
            Session session = Session.getActiveSession();
            if(albumId!="null" && albumId!=null){
                if (session != null && session.isOpened()) {
                    // Get the user's data
                  if(((MainActivity)getSherlockActivity()).isNetworkAvailable()){
                        fillUrls(session);
                  }
                  else
                        Toast.makeText(getSherlockActivity().getApplicationContext(),
"No Network Available", Toast.LENGTH_LONG).show();

                }
          }

      }

      private void fillUrls(final Session session) {
            imageUrlsList.clear();
            imageUrlsList.removeAll(imageUrlsList);
                  Request request = Request.newGraphPathRequest(session,
""+albumId+"/photos", new Request.Callback() {
                        JSONObject graphResposte=null;
                        JSONArray jsonArray = null;
                        @Override
                        public void onCompleted(Response response) {
                              if(response.getGraphObject()!=null){
                                    graphResposte =
response.getGraphObject().getInnerJSONObject();
                                    try {
                                          jsonArray =
graphResposte.getJSONArray("data");

                                          for(int i = 0; i < jsonArray.length();
i++){
                                                JSONObject c =
jsonArray.getJSONObject(i);
                                                JSONArray a =
c.getJSONArray("images");
                                                JSONObject c2=
a.getJSONObject(0);


      imageUrlsList.add(c2.getString("source"));

                                    }
                                          if(imageUrlsList.size()>0){
                                                imageUrls = new
```

```
String[imageUrlsList.size()];
                                                imageUrlsList.toArray(imageUrls);
                                        }
                                        else{
                                                imageUrls= new String[0];

        Toast.makeText(getSherlockActivity().getApplicationContext(), "This Album is
empty", Toast.LENGTH_LONG).show();
                                        }

                                        options = new
DisplayImageOptions.Builder()
                                                .showStubImage(R.drawable.ic_stub)

        .showImageForEmptyUri(R.drawable.ic_empty)

        .showImageOnFail(R.drawable.ic_error)
                                                .cacheInMemory()
                                                .cacheOnDisc()


        .bitmapConfig(Bitmap.Config.RGB_565)
                                                .build();

                                        if(imageUrls.length!=0){
                                                listView = (GridView)
getActivity().findViewById(R.id.gridView);
                                                ((GridView)
listView).setAdapter(new ImageAdapter());

        listView.setOnItemClickListener(new OnItemClickListener() {
                                                        @Override
                                                        public void
onItemClick(AdapterView<?> parent, View view, int position, long id) {

        startImagePagerActivity(position);
                                                        }
                                                });

        listView.setOnItemLongClickListener(new OnItemLongClickListener() {

                                                        @Override
                                                        public boolean
onItemLongClick(AdapterView<?> arg0, View arg1, int arg2,
                                                                        long arg3) {

        ((MainActivity)getSherlockActivity()).showDownloadPhotoDialog(imageUrls[arg2]);
                                                                return false;
                                                        }

                                                });
                                        }

                                } catch (JSONException e1) {
                                        // TODO Auto-generated catch block

        Toast.makeText(getSherlockActivity().getApplicationContext(), "JSON_EXCEPTION",
Toast.LENGTH_LONG).show();
                                        e1.printStackTrace();
```

```
                                        }

      //imageView.setImageBitmap(albums.get(0).getCover_photo().getPhoto());

                                  }
                                  else

      Toast.makeText(getSherlockActivity().getApplicationContext(), "NULL",
Toast.LENGTH_LONG).show();

                              }
                        });
                        request.executeAsync();
      }

      private void startImagePagerActivity(int position) {
            Intent intent = new Intent(getSherlockActivity(),
ImagePagerActivity.class);
            intent.putExtra("imagesUrl", imageUrls);
            intent.putExtra("position", position);
            startActivity(intent);
      }
      public static ImageLoaderConfiguration initImageLoader(Context context) {
            // This configuration tuning is custom. You can tune every option,
you may tune some of them,
            // or you can create default configuration by
            //  ImageLoaderConfiguration.createDefault(this);
            // method.
            ImageLoaderConfiguration config = new
ImageLoaderConfiguration.Builder(context)
                        .threadPriority(Thread.NORM_PRIORITY - 2)
                        .denyCacheImageMultipleSizesInMemory()
                        .discCacheFileNameGenerator(new Md5FileNameGenerator())
                        .tasksProcessingOrder(QueueProcessingType.LIFO)
                        .enableLogging() // Not necessary in common
                        .build();
            return config;
            // Initialize ImageLoader with configuration.
            //ImageLoader.getInstance().init(config);
      }

      public class ImageAdapter extends BaseAdapter {
            @Override
            public int getCount() {
                  return imageUrls.length;
            }

            @Override
            public Object getItem(int position) {
                  return null;
            }

            @Override
            public long getItemId(int position) {
                  return position;
            }

            @Override
            public View getView(int position, View convertView, ViewGroup parent)
{
```

```
                    final ImageView imageView;
                    if (convertView == null) {
                          imageView = (ImageView)
getActivity().getLayoutInflater().inflate(R.layout.item_grid_image, parent, false);
                    } else {
                          imageView = (ImageView) convertView;
                    }

                    imageLoader.displayImage(imageUrls[position], imageView,
options);

                    return imageView;
              }

      }
}
```

5.4.3. Controlador.

```
package com.app.objects;

import android.graphics.Bitmap;

public class Photo {
      private String title;
      private Bitmap photo;

      public Photo(){
            this.title="no_title";
            this.photo=null;
      }
      public Photo(String title,Bitmap bitmap){
            this.title=title;
            this.photo=bitmap;
      }

      public String getTitle() {
            return title;
      }

      public void setTitle(String title) {
            this.title = title;
      }

      public Bitmap getPhoto() {
            return photo;
      }

      public void setPhoto(Bitmap photo) {
            this.photo = photo;
      }
}
```

6. Resultados.

La aplicación se ejecuta como cualquier otra aplicación instalada en el dispositivo Android, solo tienes que ejecutarla para encontrarte con la pantalla de bienvenida.

Ilustración 10 Pantalla principal

Una vez en la pantalla de bienvenida podrás logearte en cualquiera de los servicios implementados, en el caso de la primera versión solo es posible hacerlo con Facebook.

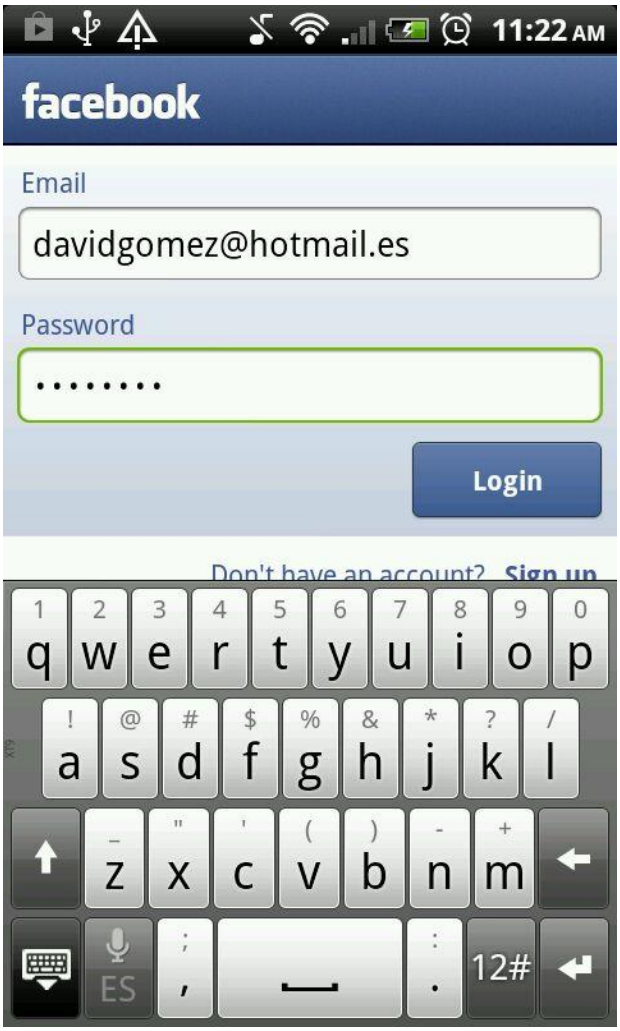

Ilustración 11 Identificación FaceBook

Esta actividad se realiza directamente usando los servicios que Facebook ofrece a los desarrolladores.

Una vez identificado y aceptados los términos y condiciones que requiere la aplicación serás llevado al listado de álbumes.

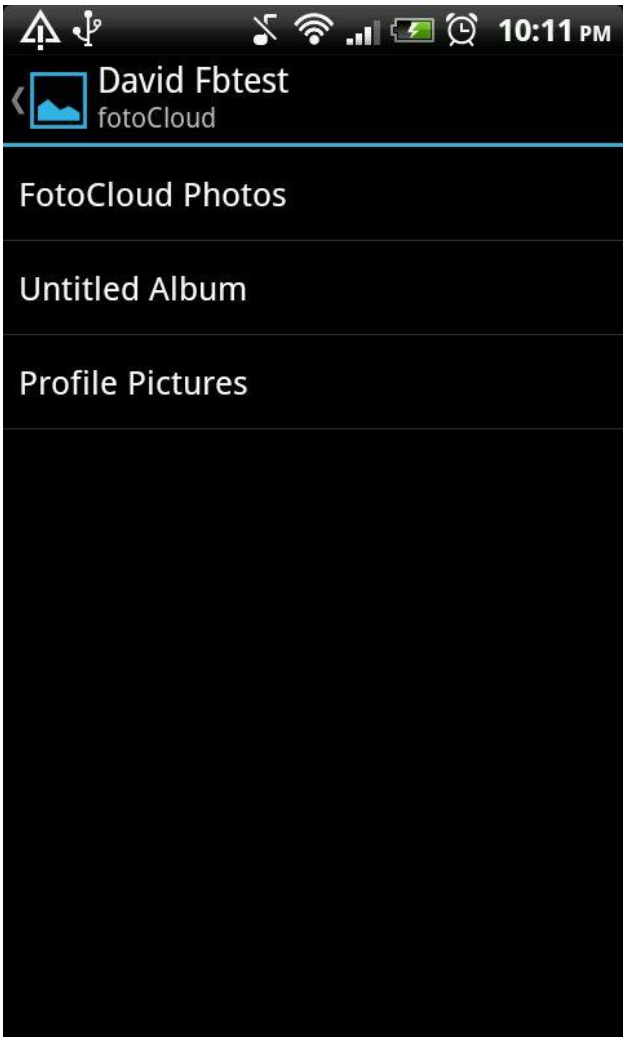

Ilustración 12 Lista álbumes

Desde esta escena podrás acceder a las fotos contenidas en cada álbum y a las acciones que ofrece el menú.

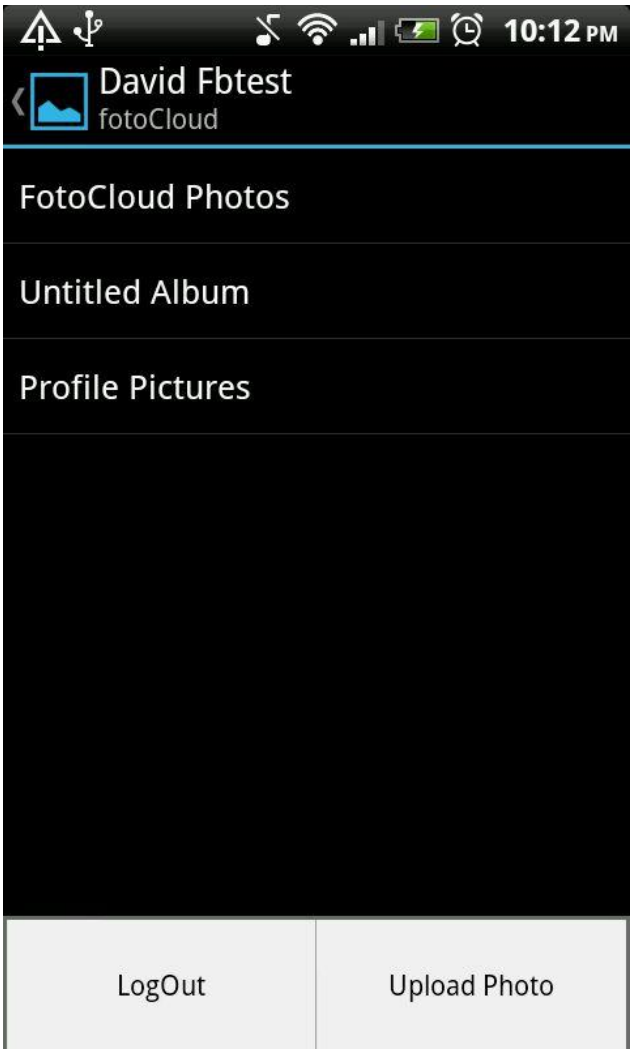

Ilustración 13 Menu opciones

Desde cualquier parte de la aplicación, siempre y cuando te encuentres logeado tendrás las opciones de cerrar sesión y de subir una fotografía.

Ilustración 14 Subir foto

Si pruebas a subir una foto te preguntará desde donde quieres obtener esa foto, podrías subirla desde tu galería de fotos o haciendo una foto directamente con la cámara del smarthphone.

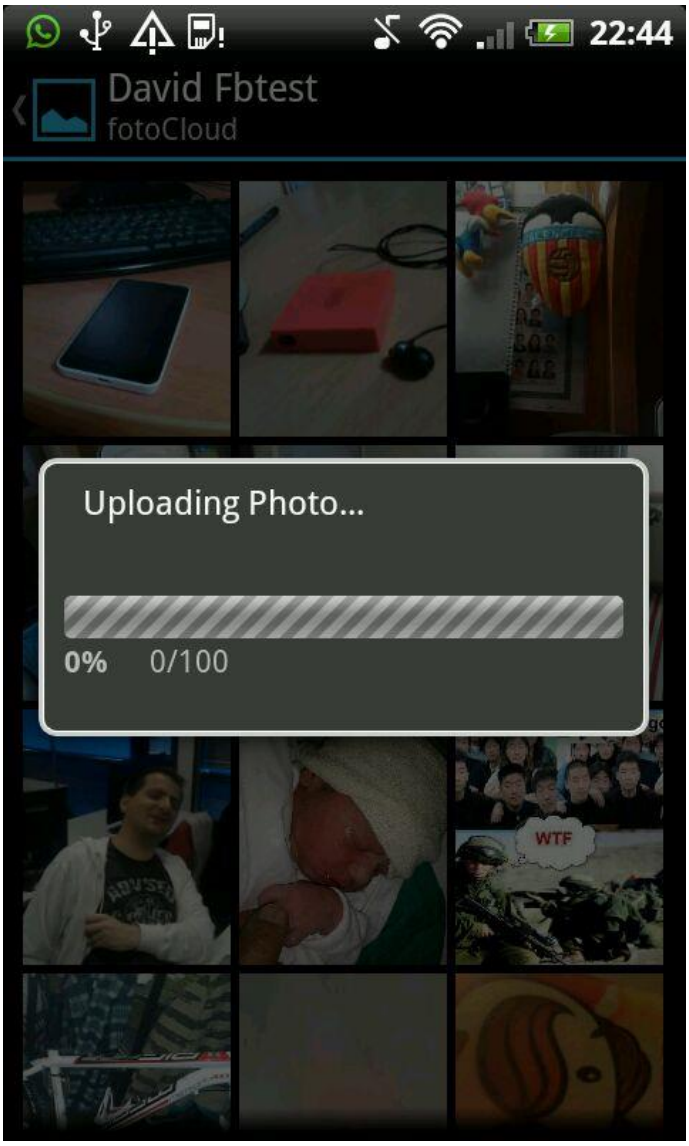

Ilustración 15 Barra de progreso subiendo foto

La foto se subirá acompañada de esta barra de progreso y aparecerá en el álbum llamado FotoCloud Photos.

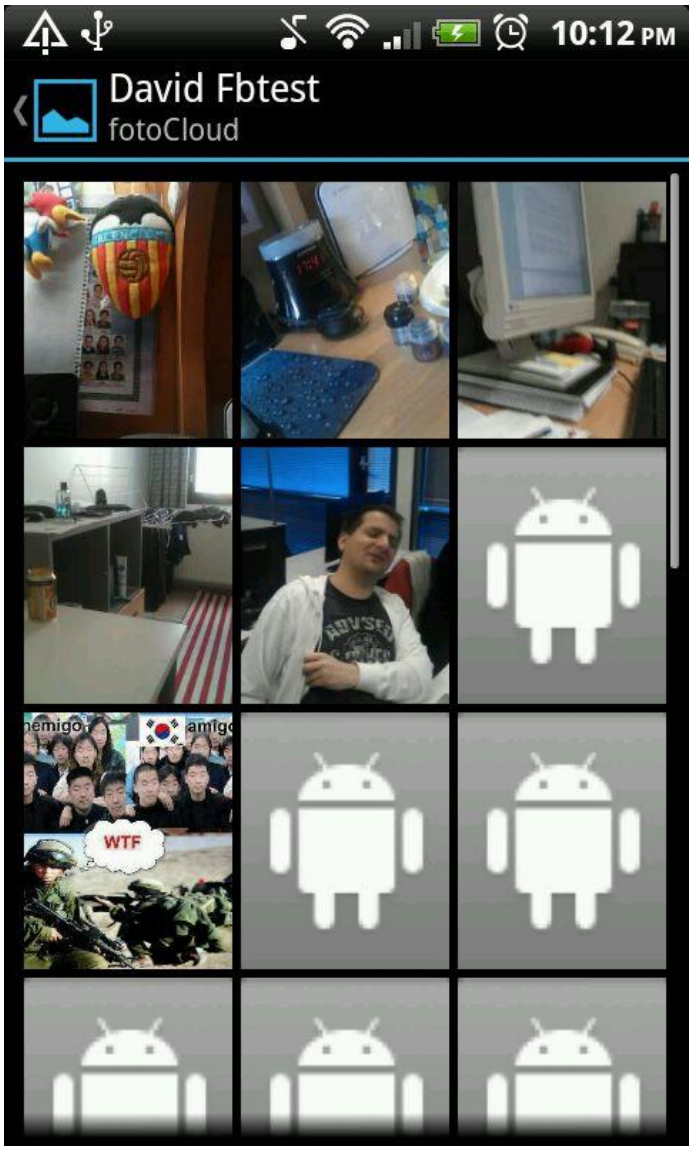

Ilustración 16 Lista fotos

Cuando seleccionas un álbum accedes a su contenido. Las fotos se cargan de forma ágil y asíncrona usando una técnica llamada “lazy load”.

Ilustración 17 Foto

Al seleccionar una foto del álbum podrás verla en pantalla completa y navegar por el resto de fotos con gestos.

7. Usos Futuros

Este proyecto nació con el objetivo de integrar el servicio que ofrecen las redes sociales a la hora de almacenar fotos en una sola aplicación sencilla, que resumiera las extensas funcionalidades sociales que estas permiten, en únicamente una colección de álbumes de fotos que son almacenados en la nube y garantizarnos la perpetuidad de estos archivos frente a imprevistos con el teléfono móvil o la facilidad de encontrarlos en la web de estas redes sociales desde cualquier dispositivo conectado de forma habitual y ser capaces de utilizar las funciones que la red social ofrezca.

Al verse reducido la funcionalidad con varias redes sociales a solamente una de ellas (Facebook). En el futuro se priorizará la integración de al menos dos redes sociales más.

Además se implementarán algunas de las funcionalidades sociales más cercanas al ámbito fotográfico como pueden ser los comentarios.

Por último, para darle más atractivo fotográfico a la aplicación, se le añadirá la posibilidad de aplicar varios filtros a la fotografía (blanco/negro, sepia, etc.).

8. Conclusiones.

Este proyecto surgió de la propuesta de un profesor de la universidad Metropolia en Finlandia al verme interesado en la realización del proyecto final de carrera en la plataforma Android.

Ha resultado una experiencia muy buena académicamente porque empecé desde cero en cuanto a lo que conocimientos de desarrollo de aplicaciones Android se refiere, viéndome obligado a participar en la asignatura de desarrollo Android nivel intermedio sin haber cursado el nivel básico previamente.

Al encontrarme inmerso en el proyecto más serio de todos los que había hecho previamente, encontrarme en un país diferente en muchos aspectos a España y desconociendo la tecnología con la que tuve que desarrollar la aplicación, considero que hubiera resultado bastante positivo haber realizado el proyecto conjuntamente con algún otro estudiante de la universidad.

9. Referencias

- Estadísticas del uso de internet:

  http://www.internetworldstats.com/stats.htm

- Uso de los smarthphones

  http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536

- Tareas de segundo plano (async task)

  http://www.sgoliver.net/blog/?p=3099

- Red Social

  http://es.wikipedia.org/wiki/Red_social

- Aplicación Movil

  http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_m%C3%B3vil

- Servicio de alojamiento

  http://es.wikipedia.org/wiki/Servicio_de_alojamiento_de_archivos

- Teléfono inteligente

  http://es.wikipedia.org/wiki/Tel%C3%A9fono_inteligente

- JAVA

  http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

- Eclipse

  http://es.wikipedia.org/wiki/Eclipse_(software)

- Android SDK

  http://developer.android.com/sdk/index.html

- Android

http://es.wikipedia.org/wiki/Android

- XML

  http://es.wikipedia.org/wiki/Extensible_Markup_Language

- GitHub

  https://github.com/

- Balsamik Mockups

  http://balsamiq.com/products/mockups/

- Lenguaje unificado de modelado

  http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

- Uml Star

  http://staruml.io/

- JSON

  http://es.wikipedia.org/wiki/JSON

- Facebook SDK

  https://developers.facebook.com/docs/android

A. Anexo I: Configuración Eclipse.

1. Requisitos previos.

Descargar Eclipse ADT de la página http://developer.android.com/sdk/index.html que contiene todos lo que necesitas para desarrollar aplicaciones Android:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- A version of the Android platform
- A version of the Android system image for the emulator

Descargar Java JDK 6 o superior, el JRE no es suficiente.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Configuración

Una vez que ejecutes Eclipse ADT por primera vez detectará automáticamente tu compilador java y solo tendrás que comprobar que tienes todas las herramientas android que necesitas instaladas en el "Android SDK Manager"

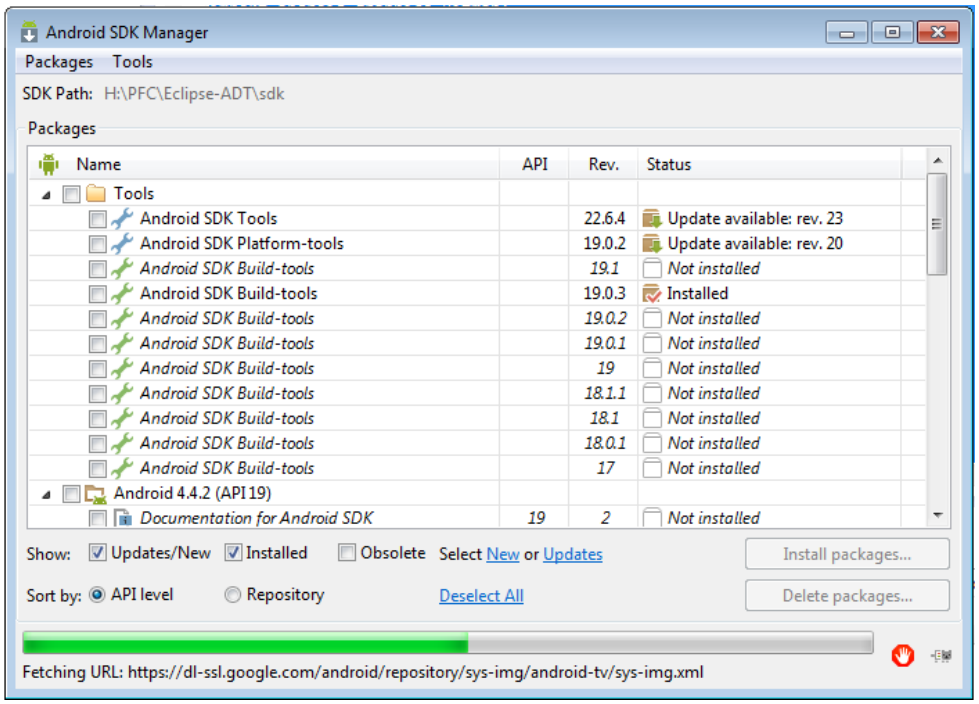

Ilustración 18 Android SDK Manager

B. Anexo II: Ejemplo Compilación.

Lo primero que debes hacer es clonarte el proyecto desde GitHub a tu ordenor: https://github.com/davidgt/FotoCloud y bajarte la aplicación Eclipse ADT como indica el anexo anterior.

Una vez tienes el entorno preparado y el proyecto clonado en tu disco duro, lo primero que debes hacer es importar el proyecto como "Existing Projects into Workspace".

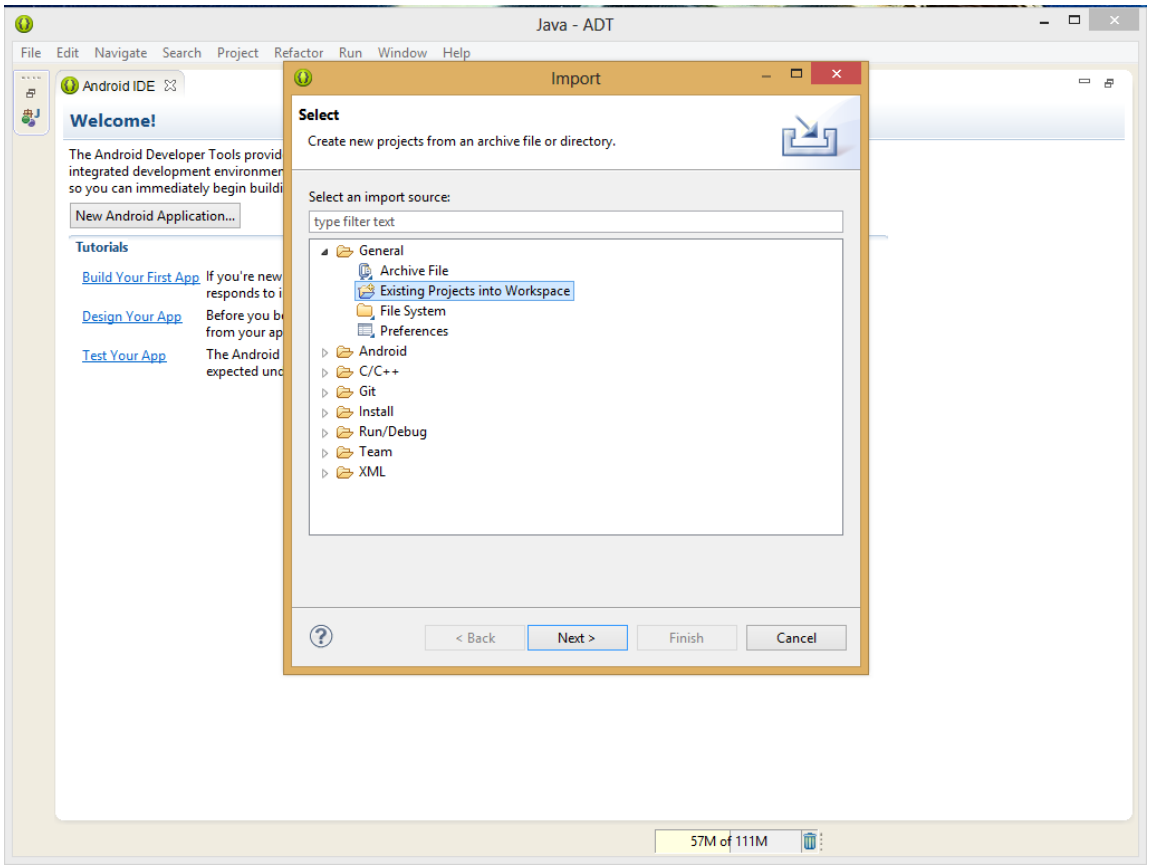

Ilustración 19 Importar proyecto

En el menú de importar selecciona la carpeta raíz donde hayas clonado el repositorio del proyecto.

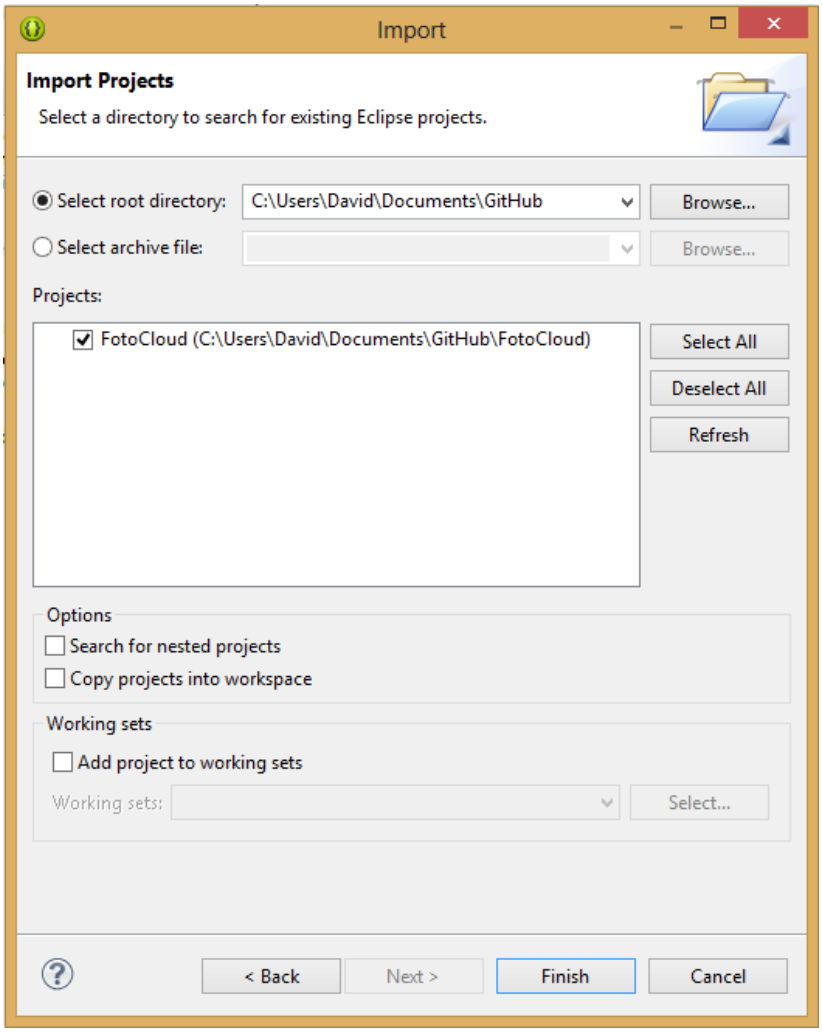

Ilustración 20 Importar proyecto 2

Una vez importado será necesario asegurarse de que las dos librerías que usa el proyecto (FaceBook SDK y actionbarsherlock) están correctamente añadidas o añadirlas si no lo están, las encontrarás en la carpeta /externals desde la raíz del repositorio.

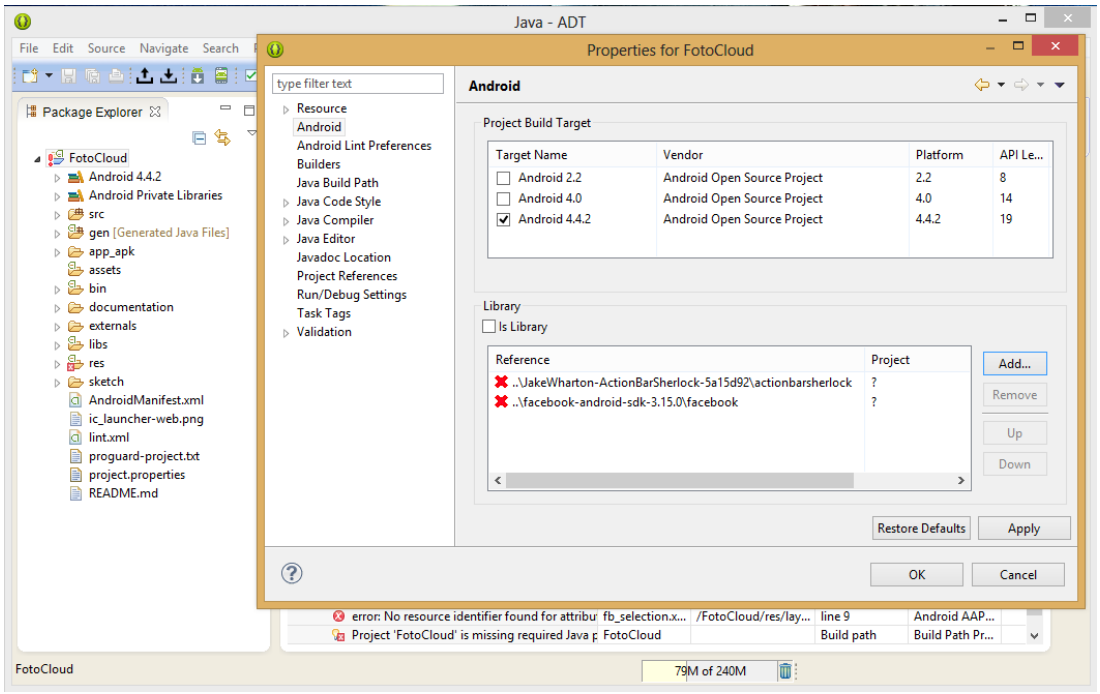

Ilustración 21 Propiedades proyecto

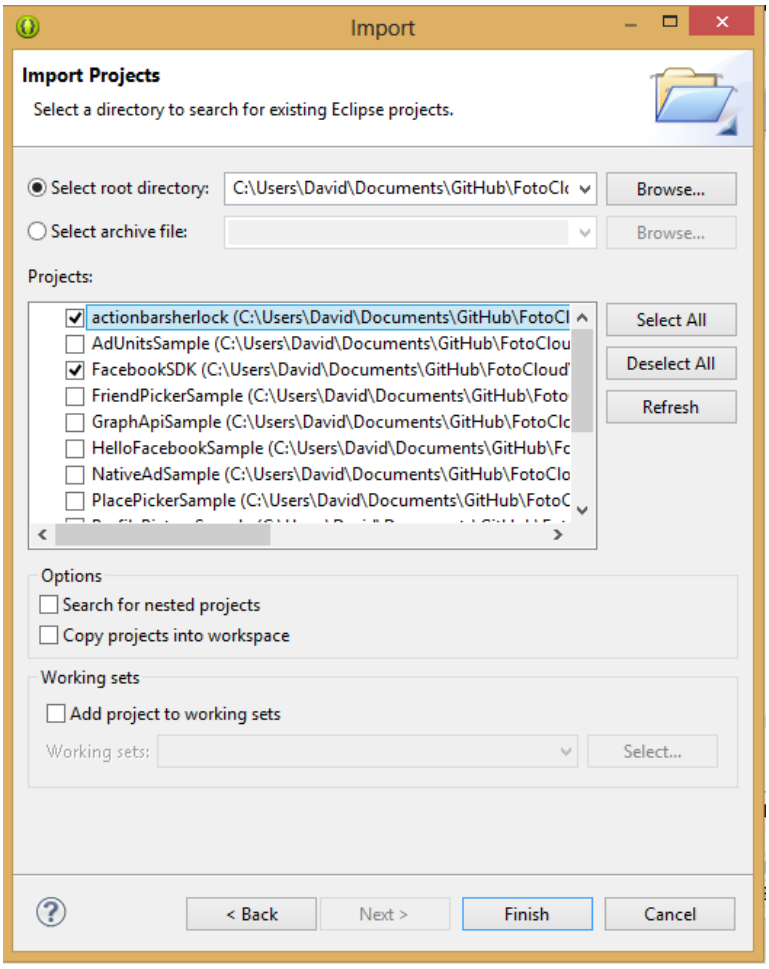

Ilustración 22 Insertar librerias

Una vez importadas todas las librerías hoy que volver a las propiedades del proyecto y añadirlas.

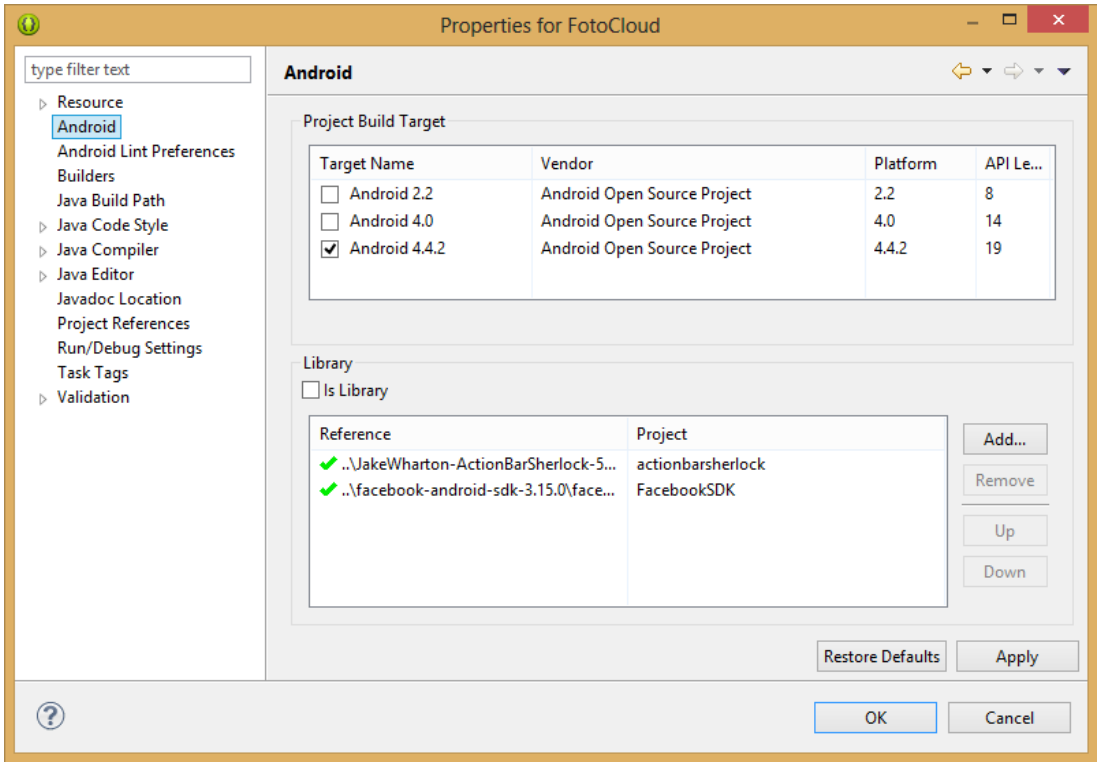

Ilustración 23 Propiedades proyecto 2

Ahora deberías encontrarte con el proyecto listo para compilar.

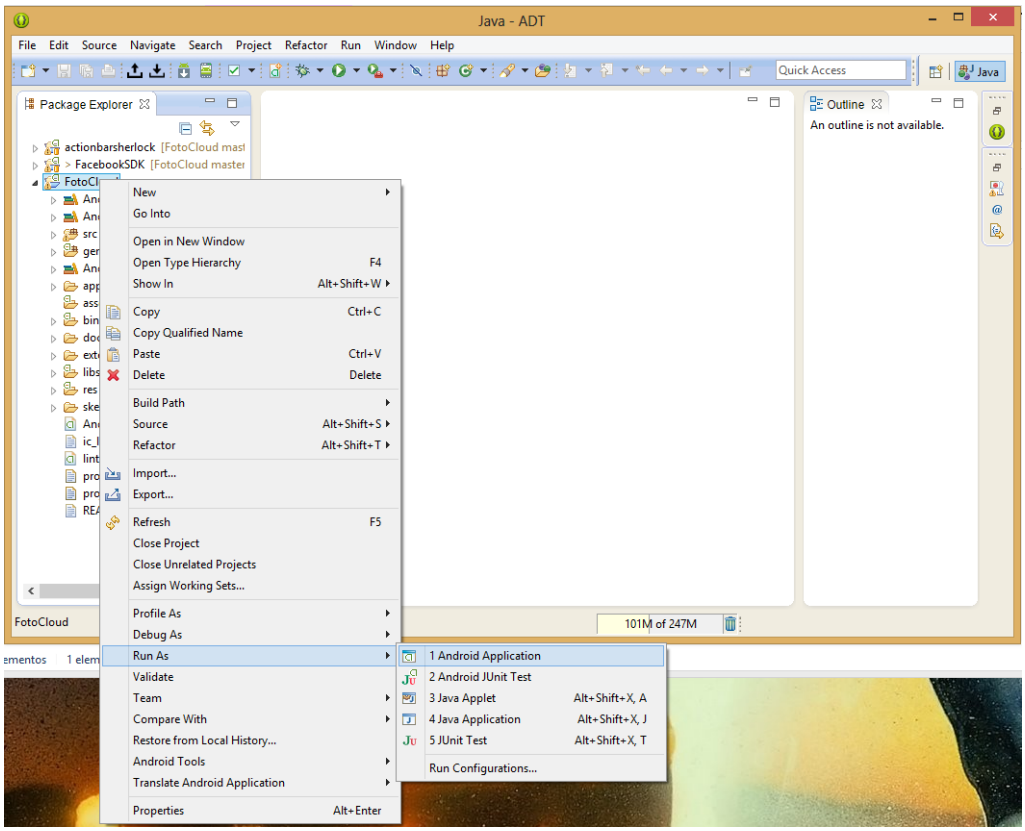

Ilustración 24 Compilar proyecto

Una vez compilado correctamente, si no tienes ningún dispositivo predefinido para la ejecución del programa te mostrará la siguiente ventana, que te permite crear una máquina virtual android de la versión que elijas.

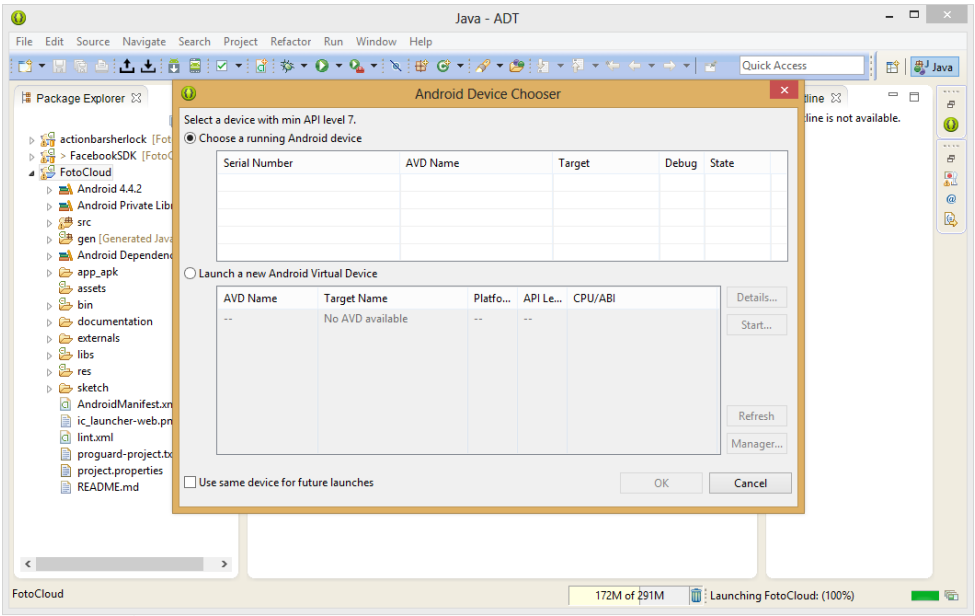

Ilustración 25 Seleccionar dispositivo android