



# SWADroid

Ampliación del cliente Android  
para la plataforma de educación SWAD



**José Antonio Guerrero Avilés**

Ingeniería Informática

**Curso 2013 / 2014**

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS  
INFORMÁTICA Y DE TELECOMUNICACIÓN**

**UNIVERSIDAD DE GRANADA**

**Proyecto Fin de Carrera de Ingeniería Informática:**

*“SWADroid: Ampliación del cliente Android para la plataforma de educación SWAD”*

**Autor:**

José Antonio Guerrero Avilés

**Director de proyecto:**

Antonio Cañas Vargas

*Departamento de Arquitectura y Tecnología de Computadores*

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS  
INFORMÁTICA Y DE TELECOMUNICACIÓN**

**UNIVERSIDAD DE GRANADA**

“*SWADroid: Ampliación del cliente Android para la plataforma de educación SWAD*”

**José Antonio Guerrero Avilés**

**Agradecimientos**

A *mi familia* por estar siempre ahí y por darme los medios y el apoyo necesarios para poder terminar mis estudios. A *Alejandro Alcalde Barros* por formar parte del equipo de desarrollo en el “VIII Concurso Universitario de Software Libre” ya que sin él no hubiese sido posible ganar el premio conseguido. A *Antonio Cañas Vargas* por su paciencia y dedicación con el proyecto. Y en especial a *Juan Miguel Boyero Corral* por haberme asesorado desde el primer momento y porque sin sus consejos no hubiese sido posible la realización de este proyecto.

A todos, de corazón, mil gracias.

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS  
INFORMÁTICA Y DE TELECOMUNICACIÓN**

**UNIVERSIDAD DE GRANADA**

*“SWADroid: Ampliación del cliente Android para la plataforma de educación SWAD”*

**José Antonio Guerrero Avilés**

**Resumen**

Ampliar las funcionalidades del cliente Android (SWADroid) para la plataforma de telecomunicación y gestión docente SWAD (<http://swad.ugr.es>).

El desarrollo de las nuevas funcionalidades se ha realizado en Java.

D. Antonio Cañas Vargas, profesor de la titulación de Ingeniería Informática, perteneciente al Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada, en calidad de director del proyecto fin de carrera de D. José Antonio Guerrero Avilés

INFORMA que el proyecto

***SWADroid: Ampliación del cliente Android para la plataforma de educación SWAD.***

Ha sido realizado y redactado por dicho alumno, y corresponde a la investigación realizada bajo su dirección, y con esta fecha autoriza su presentación.

Granada, Septiembre de 2014

**Director del proyecto**

**Firmado: Antonio Cañas Vargas.**

D. José Antonio Guerrero Avilés con DNI 75485683-M, autorizo a que una copia de este Proyecto Fin de Carrera, titulado "***SWADroid: Ampliación del cliente Android para la plataforma de educación SWAD***", se deposite en la biblioteca de la facultad de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación para que pueda ser libremente consultada.

Granada, Septiembre de 2014.

**Firmado: José Antonio Guerrero Avilés.**

## **Resumen.**

El presente proyecto ilustra el modo en que se ha realizado la implementación de las funciones que amplían la aplicación móvil SWADroid.

La aplicación móvil constituye un cliente para dispositivos basados en el sistema operativo Android para la plataforma de teleformación y gestión docente SWAD (<http://swad.ugr.es/>) de forma que sea fácilmente ampliable con nuevas funciones de acceso a SWAD. Es utilizado por alumnos y profesores para acceder a los servicios de SWAD que implemente el cliente. Consta de una parte cliente y otra que actúa como servidor. El servidor se ejecuta en el ordenador donde está alojado SWAD e implementa los servicios web requeridos para que el cliente pueda acceder a los servicios de SWAD. El cliente es un programa ejecutado por el profesor o alumno sobre un dispositivo basado en Android que implementa varios de los servicios de SWAD.

SWADroid está disponible para su instalación desde Android Market (tienda de aplicaciones de Android) de forma gratuita a través del enlace directo <https://market.android.com/details?id=es.ugr.swad.swadroid> y mediante el siguiente código QR:



El código fuente de SWADroid está disponible en <https://github.com/Amab/SWADroid> con licencia GPLv3.

## **Palabras clave**

Android, software libre, software colaborativo, aplicación móvil, SWAD, SWADroid ampliación SWADroid, aprendizaje electrónico móvil, e-learning, m-learning, mobile Learning.

## **Abstract.**

This project illustrates how has made the implementation of functions that extend the mobile application SWADroid.

The mobile application is a client for devices based on Android operating system for e-learning platform and educational administration SWAD (<http://swad.ugr.es/>) so as to be easily expandable with new access functions of SWAD. SWADroid is used by students and teachers to access to the SWAD services that implement the client. It consists of a client and a server. The server runs on the computer that hosts SWAD and implements web services required for the customer to access to the SWAD services. The client is a program executed by the teacher or student on an Android based device and initially deploy one or more of the SWAD services.

SWADroid is available for installation from the Android Market (Android app store) for free via the direct link <https://market.android.com/details?id=es.ugr.swad.swadroid> and by the following QR code:



SWADroid source code is available on <https://github.com/Amab/SWADroid> under GPLv3 license.

## **Keywords**

Android, software libre, software colaborativo, aplicación móvil, SWAD, SWADroid ampliación SWADroid, aprendizaje electrónico móvil, e-learning, m-learning, mobile Learning.

# Índice de contenido

1. Introducción.....	1
2. Definición de objetivos y especificaciones. .....	2
2.1. Estado del arte. ....	2
2.2. Motivación.....	4
2.3. Objetivos.....	8
2.4. Software y documentación libre. ....	10
2.5 Software colaborativo.....	11
3. Análisis y Diseño.....	12
3.1. Diagrama de casos de uso.....	12
3.2 Sistema operativo móvil. ....	13
3.3. Plataforma de desarrollo.....	14
3.4. Diagrama de funcionamiento de los servicios Web. ....	15
3.6. Diagramas de clase. ....	16
4. Módulos.....	17
4.1. Desarrollo del módulo informativo. ....	18
4.2. Desarrollo del módulo de mensajes.....	26
5. Internacionalización.....	41
6. Programación del proyecto.....	42
6.1 Introducción.....	42
6.2. Especificaciones técnicas. ....	45
6.3. Repositorio de SWADroid en GitHub.....	48
6.3.1. Instalar el plugin EGit. ....	49
6.3.2. Generar claves SSH.....	51
6.3.3. Configurar la información de desarrollador .....	53
6.3.4. Importar el proyecto SWADroid y subir nuestros cambios .....	54
6.4 API de Android.....	63
6.4.1 Activity .....	63
6.4.2 Intent.....	66
6.4.3 View.....	68
6.5. Temporización del Proyecto.....	70
7. Estadísticas. ....	78
8. Futuras mejoras. ....	84
9. Conclusiones.....	85
10. Bibliografía.....	87
11. Apéndices. ....	89
11.1. La nueva interfaz .....	89
11.2. Módulo informativo. La potencia del WebView.....	97
11.3. Módulo de mensajes. Manual de usuario. ....	102

## **1. Introducción.**

Este documento tiene como propósito detallar la labor realizada durante el periodo de ejecución del proyecto final de carrera, consistente en desarrollar nuevas funcionalidades para la aplicación para dispositivos móviles Android SWADroid.

La aplicación SWADroid constituye un cliente Android capaz de acceder a la plataforma de teleformación SWAD (<http://swad.ugr.es>) de la Universidad de Granada mediante servicios web basados en el protocolo SOAP (Simple Object Access Protocol). Esto conlleva ventajas tanto para los usuarios de la plataforma como para el propio SWAD.

En primer lugar, SWAD amplía su presencia en el ámbito académico ofreciendo a sus usuarios una nueva forma de acceder a sus contenidos de forma rápida y sencilla.

En segundo lugar, los usuarios de SWAD pueden acceder al mismo en cualquier momento y lugar e incluso podrán acceder a ciertos servicios de SWAD sin necesidad de una conexión a Internet.

## **2. Definición de objetivos y especificaciones.**

### **2.1. Estado del arte.**

Se denomina aprendizaje electrónico móvil, en inglés, m-learning, a una metodología de enseñanza y aprendizaje valiéndose del uso de pequeños y maniobrables dispositivos móviles, tales como teléfonos móviles, celulares, agendas electrónicas, tablets PC, pocket pc, i-pods y todo dispositivo de mano que tenga alguna forma de conectividad inalámbrica.

La educación va incorporando intensivamente las nuevas tecnologías de la comunicación, pasando por varias etapas. Diversos conceptos describen ese fenómeno, según avanza la tecnología: EAO (Enseñanza apoyada por el ordenador), multimedia educativo, tele-educación, enseñanza basada en web (web-based teaching), aprendizaje electrónico (e-learning), etc.

Tanto desde el simple uso de la computadora y los soportes multimedia, como el advenimiento de Internet y las redes en general, todo ha servido para apoyar el proceso de enseñanza-aprendizaje en sus diferentes modalidades y aspectos.

De un tiempo a esta parte, se vienen incorporando a nuestras vidas, cada vez con más fuerza, las tecnologías móviles, y por lo tanto, está surgiendo lo que denominamos mobile learning o mlearning y que consiste en usar estos aparatos electrónicos para aprender.

Esto está generando gran expectativa en el sistema educativo, sobre el que se están realizando interesantes iniciativas empresariales y proyectos de investigación.

El "**Mobile Learning Engine**" (MLE) es una aplicación integral de aprendizaje. Transfiere el aprendizaje asistido por ordenador y multimedia (conocidos como e-learning) para un entorno móvil (con un teléfono móvil). Este tipo especial de aprendizaje asistido por ordenador y multimedia se conoce comúnmente como "mLearning".

El MLE permite al estudiante a utilizar su teléfono móvil como un medio para el aprendizaje. Como consecuencia, es posible utilizar cualquier tipo de tiempo de espera para el aprendizaje, no importa dónde se encuentre.

Actualmente se puede acceder a la plataforma SWAD desde un dispositivo móvil gracias a la aplicación SWADroid, lo que hace que se pueda aprovechar todo el potencial de los dispositivos móviles, además de no requerir una conexión a Internet permanente para su uso. Gracias a esto, dispositivos móviles como los dispositivos con Android 2.1 o inferior que tienen problemas para acceder a páginas web mediante el protocolo HTTPS (requerido por SWAD) pueden acceder a dicha plataforma.

SWADroid fue liberado bajo licencia GPLv3, esto significa que cualquiera puede usarlo para cualquier propósito, compartirlo, estudiar su funcionamiento, modificarlo y compartir esas modificaciones, es decir, SWADroid es lo que se denomina “**software libre**”.

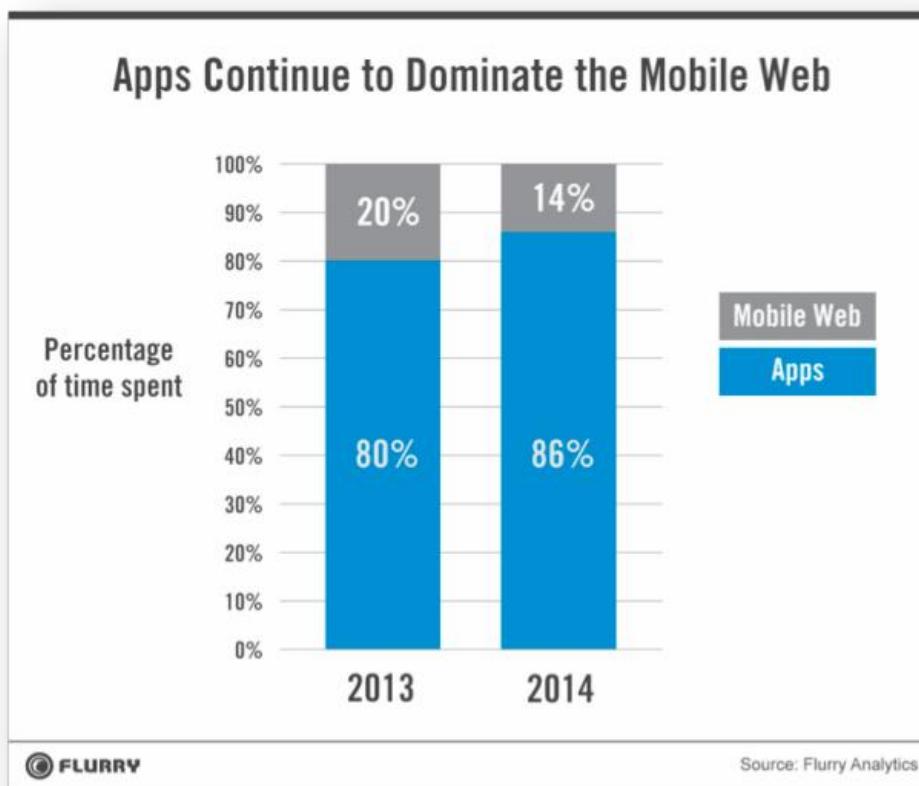
El software libre no se traduce únicamente en una serie de libertades para el usuario sino que también es beneficioso para el propio proyecto: recibe visibilidad (publicidad), logra mejoras gracias a la retroalimentación de los usuarios y recibe la colaboración de otros usuarios.

## 2.2. Motivación.

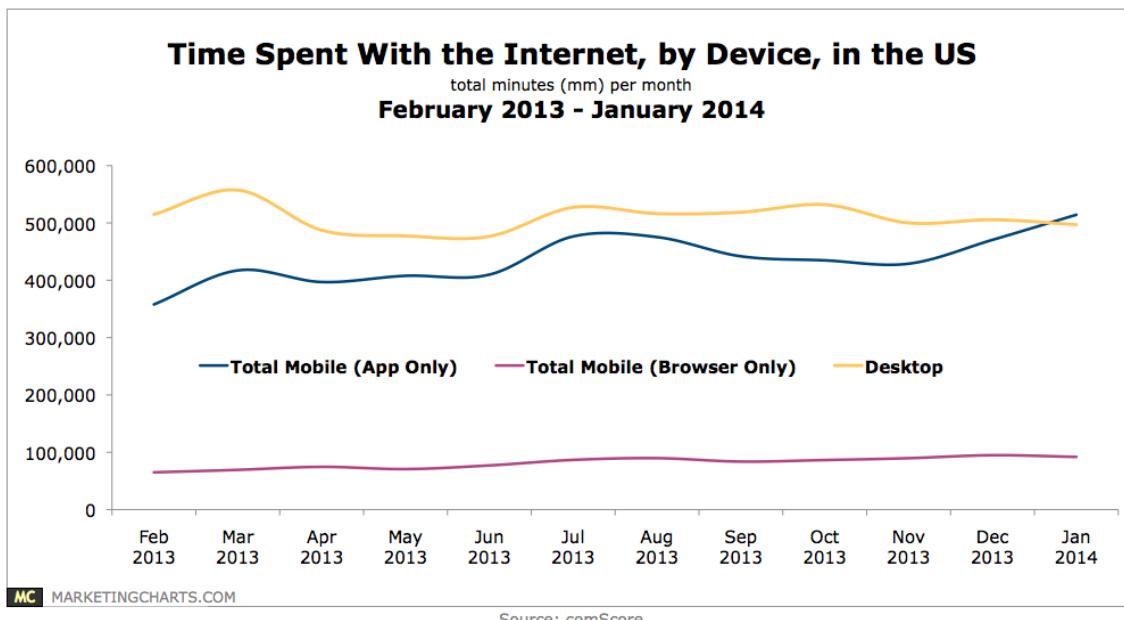
La finalidad de este proyecto es ofrecer a los usuarios ya existentes de SWADroid nuevas funcionalidades. Por una parte, esto aporta beneficios a la plataforma SWAD puesto que continúa su ampliación con las nuevas tecnologías. Por otra parte, los usuarios ven cómo la plataforma evoluciona y les proporciona nuevas funcionalidades y formas de acceso a la misma; esto proporcionará confianza y satisfacción a los usuarios actuales.

Al mismo tiempo, SWADroid puede ayudar a la captación de nuevos usuarios, bien sea por descubrimiento de la existencia de la plataforma al acceder a SWAD desde su dispositivo Android, o usuarios que utilizaban otras plataformas similares a SWAD y que pueden sentirse atraídos por las nuevas posibilidades que este proyecto les ofrece, ya que es posible que las herramientas que estén utilizando no se hayan adaptado a las nuevas tecnologías.

Los últimos estudios que analizan la procedencia de los accesos a la información por parte de los usuarios muestran claramente un continuo crecimiento del acceso a la información a través de las aplicaciones móviles respecto al acceso vía web.



Además, en 2014, el tiempo que pasan los usuarios en Internet en los teléfonos móviles ha superado al tiempo que pasan en Internet con sus ordenadores personales.

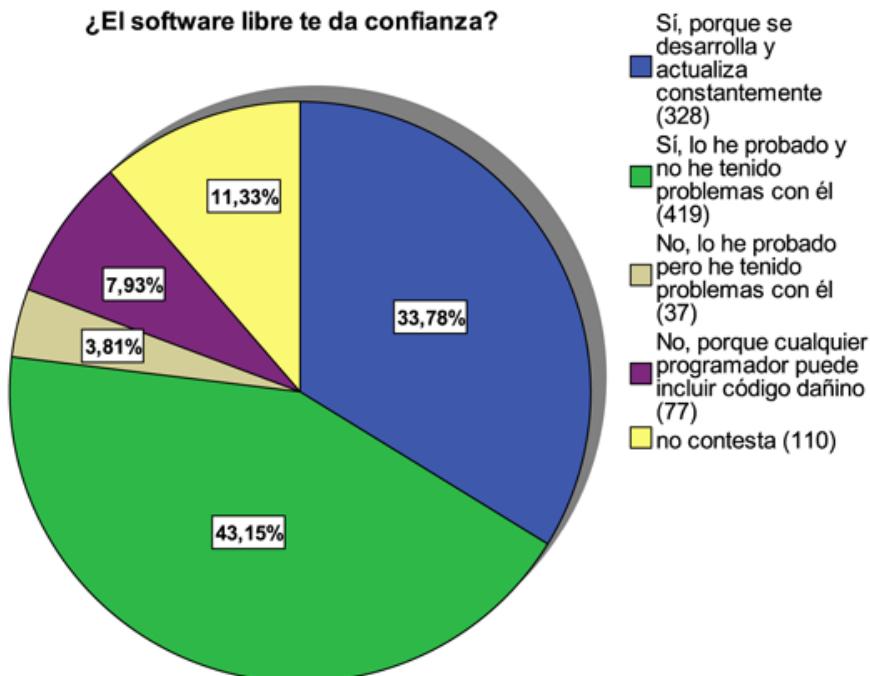


Por otro lado, según las últimas encuestas, el software libre se va abriendo paso con fuerza entre la sociedad y se usa cada vez más tanto por empresas como por usuarios.



Fuente: <http://www.portalprogramas.com/descargas/estudio-valoracion-software-libre.pdf>

Además, cada vez hay más usuarios que confían en el software libre, ya sea por buenas experiencias como por buena información.



Fuente: <http://www.portalprogramas.com/descargas/estudio-valoracion-software-libre.pdf>

Las razones que justifican la existencia de una aplicación nativa para acceder a SWAD son:

- Menor funcionalidad, pero más inmediata y sencilla.
- Un solo toque para activarla.
- Usuario-contraseña almacenado en configuración (no requiere introducirlos cada vez que se accede a SWAD).
- Menos información en pantalla (evita saturar al usuario con el exceso de información).
- Mayor claridad.
- Menor consumo de ancho de banda.
- Funcionalidades off-line.

Y las razones para continuar con el desarrollo de la aplicación y ampliar sus funcionalidades son las siguientes:

- Aumento del número de accesos a SWAD desde SWADroid respecto de los accesos Webs.
- Peticiones expresas de los usuarios que consumen la aplicación de nuevas funcionalidades.
- Hacer que las funcionalidades que ofrece SWAD sean accesibles desde cualquier parte con el simple hecho de tener tu teléfono móvil a mano.

Se han implementado dos funcionalidades muy usadas en SWAD (como muestran las estadísticas proporcionadas por la plataforma) que no se usaban a penas en SWADroid:

- Acceso a la información de las asignaturas:
  - Introducción.
  - Programa de teoría.
  - Programa de prácticas.
  - Guía docente.
  - Bibliografía.
  - FAQ.
  - Enlaces.
  - Sistema de evaluación.

Esta funcionalidad no estaba disponible para SWADroid y en el caso de la Web, el acceso a los apartados anteriormente mencionados es de unos 1500 clics de media y se dispara hasta casi 18.000 en el caso de la Introducción de la asignatura (Estadísticas a 07/09/2014).

- Envío de mensajes:

Si bien es cierto que esta funcionalidad está disponible, una minoría de usuarios la usan (aproximadamente 1.721 envíos desde la Web en una semana según las estadísticas a 07/09/14 respecto a solo 39 desde la aplicación) debido a que, entre otras cosas, nos obliga a sabernos el alias o DNI de la persona a la que queremos enviarle el mensaje, cosa que, normalmente, no se suele conocer. Por estos motivos, se ha optado por rediseñar el módulo por completo de la siguiente manera:

- Nueva interfaz.
- Posibilidad de visualizar los usuarios de la asignatura con nombre y foto y actualizar dicho listado.
- Posibilidad de marcar usuarios y añadirlos al campo destinatarios desde la lista de usuarios.

## **2.3. Objetivos.**

Hoy en día es cada vez más habitual el uso de Internet en un dispositivo móvil. En el caso de Android, y más teniendo en cuenta que el principal promotor de este sistema operativo es Google, los dispositivos móviles están orientados a un uso optimizado de Internet.

Teniendo en cuenta las nuevas tendencias del mercado, el principal objetivo de este proyecto es intentar posibilitar en mayor o menor medida el acceso a los usuarios de SWAD a la mayor parte posible de sus funcionalidades desde cualquier lugar sin necesidad de un ordenador personal ni de una potente conexión a Internet.

El objetivo principal de la aplicación se materializará en el desarrollo de una nueva funcionalidad y la modificación completa de otra por lo que podría considerarse realmente que se van a implementar dos nuevas funcionalidades:

- Acceso a la información de las asignaturas: Esta funcionalidad permitirá que un usuario pueda consultar la siguiente información de una asignatura:
  - Introducción.
  - Programa de teoría.
  - Programa de prácticas.
  - Guía docente.
  - Bibliografía.
  - FAQ.
  - Enlaces.
  - Sistema de evaluación.
- Envío de mensajes: Ya se permite enviar un mensaje (aunque con las dificultades ya comentadas). En este caso lo que se pretende para el módulo de mensajes es :
  - Implementar una nueva interfaz.
  - Posibilidad de visualizar los usuarios de la asignatura con nombre y foto y actualizar dicho listado.
  - Posibilidad de marcar usuarios y añadirlos a los destinatarios de un mensaje desde el listado de usuarios que se podrá visualizar.

Estos cambios lograrán ampliar altamente las funcionalidades de SWADroid y los accesos a SWAD desde la aplicación móvil, lo que logrará que un usuario pueda acceder e interactuar con sus datos de una manera rápida y sencilla en cualquier momento y lugar.

Respecto a los objetivos personales, teniendo en cuenta que el mundo de la informática está en continua evolución, se busca aprender a utilizar las nuevas tecnologías puesto que aunque se termine la etapa de estudiante, un informático va a estar aprendiendo a usar tanto nuevos lenguajes de programación como nuevas tecnologías durante toda su vida laboral.

En este caso, el objetivo personal es aprender a programar para una plataforma en la que no se ha trabajado durante la carrera, dado que Android es un sistema operativo muy reciente en el que aún no se ha profundizado a nivel académico.

Por otro lado, también está presente el objetivo de profundizar en el tema del software libre, colaborando en un proyecto para intentar hacerlo más grande y mejorarlo aportando mi esfuerzo y conocimientos al igual que tanto el autor principal como los diferentes colaboradores del proyecto hicieron en su momento.

## **2.4. Software y documentación libre.**

SWADroid fue liberado bajo la licencia GPLv3, al igual que la documentación que se creó en sus inicios como proyecto fin de carrera de Juan Miguel Boyero Corral. Esto significa que cualquiera puede usarlo para cualquier propósito, compartirlo, estudiar su funcionamiento, modificarlo y compartir esas modificaciones. Pero el software libre no significa únicamente una serie de libertades para el usuario, también es beneficioso para el propio proyecto: recibe visibilidad (publicidad), logra mejoras gracias a la retroalimentación de los usuarios y recibe la colaboración de otros usuarios.

La liberación del software y la documentación ha permitido la transferencia de conocimientos y la innovación tecnológica, haciendo que el proyecto no haya quedado estancado tras su finalización, sino todo lo contrario, han ido surgiendo otros proyectos fin de carrera a partir de este que han servido como ampliación de la aplicación como es el caso de este proyecto.

Este proyecto, además, se basa completamente en estándares abiertos y herramientas libres, por lo que es también una obligación moral devolver a la comunidad lo recibido, además de que algunas licencias obligan a liberar los desarrollos derivados.

El primer proyecto y los sucesivos, no solo fueron liberados, sino que han sido desarrollados en un proceso completamente abierto, siendo accesibles todos los avances del desarrollo en una forja (<https://github.com/Amab/SWADroid>) y publicando información sobre ellos en las redes sociales. El desarrollo ha permitido también la colaboración de los usuarios mediante el envío de sugerencias de mejoras y errores.

Además, este documento ha sido liberado bajo licencia FDL.

Copyright (C) 08 de sept. de 2014, José Antonio Guerrero Avilés

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available in <http://www.gnu.org/copyleft/fdl.html>

## **2.5 Software colaborativo.**

Hay que tener en cuenta que SWADroid no es un proyecto individual, es un proyecto de software libre relativamente maduro que cuenta ya con más de 3 años de desarrollo y en el que han participado más de 6 personas, de las cuales 4 lo han tenido como proyecto fin de carrera. Aunque SWADroid fue diseñado por Juan Miguel Boyero Corral para que se pudiesen añadir nuevos módulos y nuevas funcionalidades de una manera sencilla, debido al número de personas que han implementado código es obvio que la más mínima modificación puede afectar a lo que ya hay implementado. Y no sólo es eso, si no que Android, al ser relativamente nuevo, se va actualizando constantemente, aportando nuevas maneras más sencillas (o más versátiles) de hacer las cosas y hay que adaptar el código a ellas y a las nuevas versiones del sistema operativo.

Este es el caso de este proyecto. Aunque añadir el módulo informativo, a priori, resulta sencillo, veremos más adelante que tiene una serie de dificultades. Respecto al nuevo módulo de mensajes, resulta mucho más difícil de lo esperado ya que, por los motivos mencionados anteriormente, se tienen que modificar, por ejemplo, algunas clases o la base de datos.

La inclusión de los nuevos módulos y el tener que adaptar lo implementado a las nuevas versiones de Android y a lo que los usuarios van demandando ha dado lugar a muchas modificaciones durante el desarrollo del proyecto. Estas han sido:

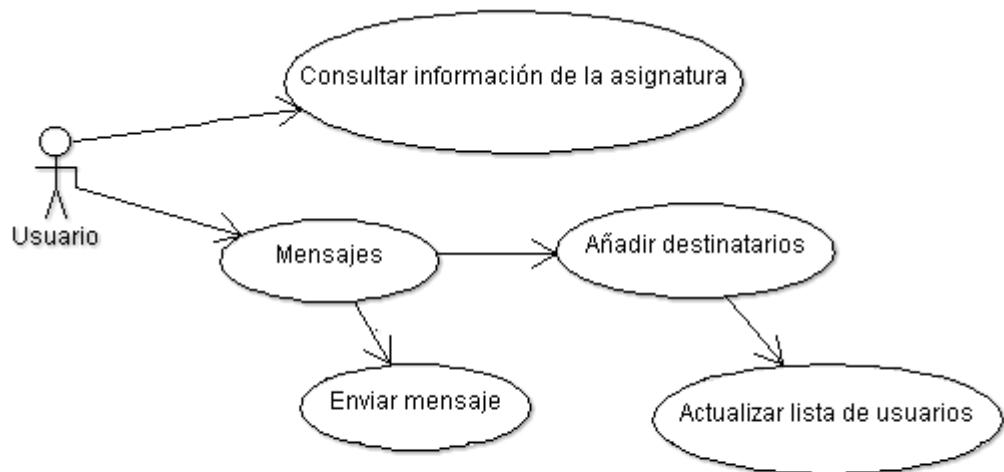
- Nueva interfaz gráfica estilo Holo.
- Añadida función de recuperación de contraseña.
- Añadido cierre de sesión en SWADroid.
- Reorganizado el menú principal para alineararlo con SWAD.
- Ahora las alertas de notificaciones solo aparecen cuando se reciben notificaciones no abiertas.
- Ahora el botón atrás lleva al directorio superior en el módulo de Documentos.
- Deshabilitado el módulo de pasar lista.
- Eliminados los idiomas no utilizados.

Aquí nos vamos a centrar solo en el módulo informativo y en el de mensajes, pero todas estas modificaciones mencionadas han dado lugar a que el código de diferentes funcionalidades haya tenido que ser adaptado o incluso a que se hayan creado otras nuevas.

SWADroid requiere un constante feedback entre todos los colaboradores y una comunicación fluida y frecuente para que los cambios se acoplen perfectamente a lo que ya hay implementado. Por otra parte, la implementación de una nueva funcionalidad puede requerir de la ayuda o el consejo del algún integrante del equipo de desarrollo bien sea porque la nueva funcionalidad es demasiado extensa o compleja o bien porque vaya a afectar a algo que ya hay implementado.

### **3. Análisis y Diseño.**

#### **3.1. Diagrama de casos de uso.**



### **3.2 Sistema operativo móvil.**

Los dos principales sistemas operativos para dispositivos móviles que existen en la actualidad son iOS de Apple para iPhone/iPad, y Android de Google.

Los motivos por los que se ha decidido continuar con la ampliación de SWADroid y no realizar la implementación de un cliente para iOS son principalmente los siguientes:

- iOS:

El desarrollo para iOS resulta demasiado caro y problemático para este proyecto. Para poder obtener el entorno de desarrollo es necesario darse de alta como desarrollador en la página web de Apple con un coste de 99\$/mes, una tarifa demasiado alta para las características y los objetivos de este proyecto.

Por otra parte, la licencia GPL que tiene este proyecto es incompatible con la tienda de aplicaciones de Apple (App Store), único modo de distribuir las aplicaciones de iOS, así que no podría ser distribuido en la misma y, por tanto, no estaría disponible para ningún dispositivo que utilizara el sistema iOS.

Por otra parte, habría que comenzar el proyecto desde el principio y seguramente se daría el caso de tener dos aplicaciones (una para iOS y otra para Android) con falta de funcionalidad en ambos casos e incompletas.

Esto provoca que desarrollar el proyecto como una aplicación de iOS sea una tarea absurda y totalmente improductiva, ya que se crearía una aplicación que nadie podría instalar ni utilizar en su dispositivo móvil.

- Android:

Es la plataforma de desarrollo inicial del proyecto. El desarrollo para Android no requiere pago alguno para obtener el entorno de desarrollo, que se puede descargar de forma gratuita desde <http://developer.android.com/sdk/index.html>.

Aunque la aplicación se puede distribuir por muchos medios, es aconsejable publicarla en la tienda de aplicaciones de Android (Android Market, tal y como está hecho), lo que requiere un único pago de 25\$ en el momento de registrarse en la misma como desarrollador. A partir de ese momento se pueden publicar cuantas aplicaciones se deseen sin coste adicional. Estas aplicaciones estarán disponibles para multitud de dispositivos móviles que utilicen el sistema Android.

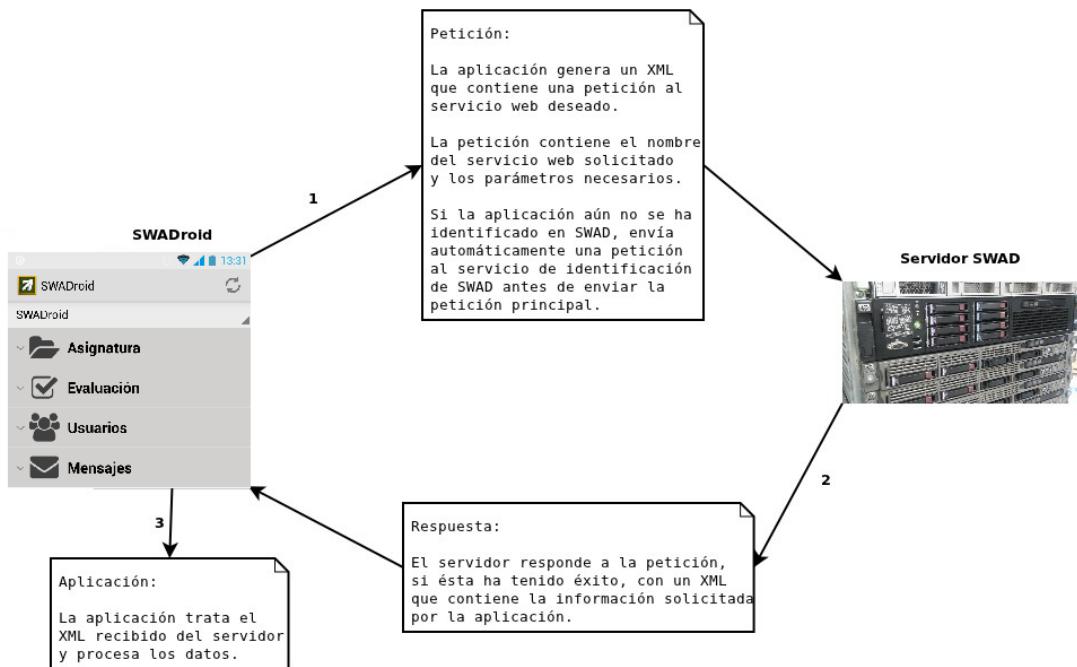
### **3.3. Plataforma de desarrollo.**

Como plataforma de desarrollo para el proyecto se ha utilizado el IDE Eclipse junto con el plugin ADT (Android Development Tools) sobre Windows 7.

Además, se ha añadido el plugin EGit (Git version control with Eclipse) para poder mantener el proyecto actualizado en todo momento con los cambios que puedan realizar los diferentes colaboradores en el desarrollo de SWADroid

Estas opciones eran las que mejor resultado me han dado y es por eso que las he elegido. Con ellas se centraliza todo el trabajo en Eclipse y se trabaja más fácilmente.

### 3.4. Diagrama de funcionamiento de los servicios Web.

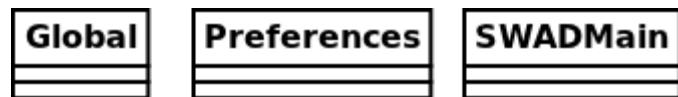


SWADroid se comunica con SWAD mediante un conjunto de servicios web proporcionados por la plataforma y que han sido implementados por Antonio Cañas utilizando el protocolo SOAP. El código fuente de los servicios web está implementado en C y ha sido generado con la ayuda de la herramienta gSOAP.

Durante la etapa de identificación, SWADroid envía a SWAD una petición de identificación que contiene el usuario y la contraseña del usuario, y la AppKey de la aplicación.

La AppKey, o clave de aplicación, es una cadena alfanumérica proporcionada por Antonio Cañas (creador de SWAD) con el fin de que SWADroid demuestre durante la etapa de identificación que está autorizada a utilizar los servicios web de SWAD. Si SWADroid intenta acceder a SWAD con datos de usuario correctos y una AppKey errónea, la petición será rechazada por SWAD. Si la petición ha tenido éxito, SWAD devuelve los datos del usuario identificado y un código de sesión que será utilizado para acceder a los servicios web de SWAD. Si la petición es errónea, SWAD devolverá un mensaje de error y SWADroid informará al usuario de que los datos de identificación proporcionados en la configuración de la aplicación son incorrectos.

### 3.6. Diagramas de clase.



El núcleo de SWADroid, exceptuando algunas clases de apoyo para construir la ventana principal y realizar operaciones menores, lo constituyen tres clases principales:

- **Global**

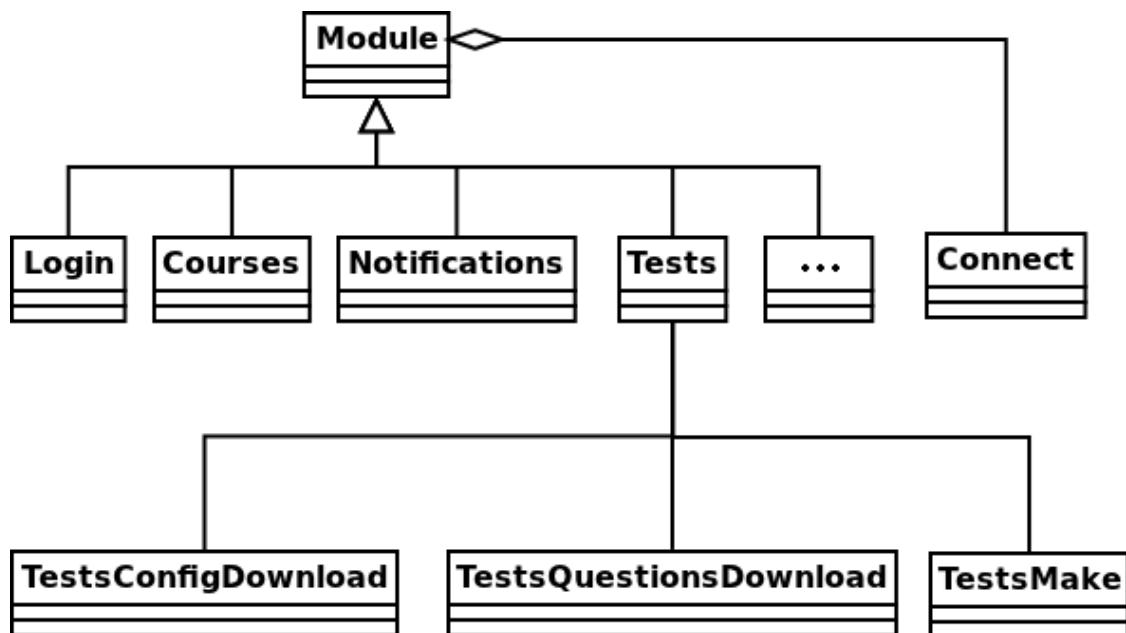
La clase Global define las constantes que se utilizarán a nivel global en toda la aplicación, como las constantes que definen los nombres de las tablas de la base de datos, y la AppKey. También define algunos métodos auxiliares para realizar tareas menores, como la conversión de valores de tipo booleano a tipo entero o cadena y viceversa.

- **Preferences**

La clase Preferences se encarga de construir la pantalla de configuración y gestionar la configuración de SWADroid permitiendo que sea accedida y modificada por el resto de la aplicación.

- **SWADMain**

La clase SWADMain es la clase principal y el punto de entrada de SWADroid. Se encarga de construir la pantalla principal y lanzar todas las funciones que implementa la aplicación a petición del usuario.

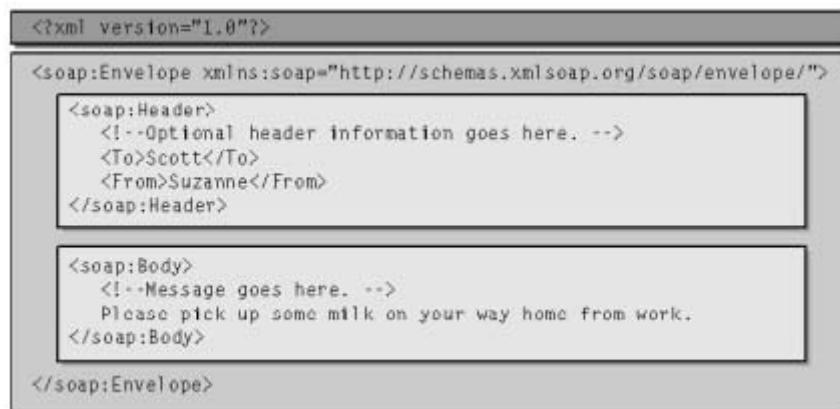


## 4. Módulos.

SWADroid sigue una arquitectura modular que permite añadir funcionalidad en forma de nuevos módulos realizando modificaciones mínimas en el resto de la aplicación. Estas modificaciones vienen impuestas principalmente por la arquitectura de Android, que obliga a que las aplicaciones sigan un esquema determinado y a que ciertas acciones se realicen de una forma preestablecida por la arquitectura del sistema.

La arquitectura modular de SWADroid se basa en la clase Module, de la que heredan todos los módulos de la aplicación y que implementa las estructuras de datos y las operaciones comunes a todos los módulos, tanto presentes como futuros. Esta clase implementa, entre otras funciones, el mecanismo de comunicación mediante el acceso a los servicios web que implementa SWAD proporcionando métodos para construir las peticiones que se realizan a dichos servicios web y recibir y procesar las respuestas recibidas de los mismos.

Los métodos de la clase Module generan las peticiones en ficheros XML que envían a SWAD, que a su vez envía las respuestas en otro fichero XML. Estos ficheros se estructuran con el formato especificado por el protocolo SOAP, utilizado en las comunicaciones con SWAD.



En sus inicios SWADroid contaba únicamente con 4 módulos:

- Login.
- Courses.
- Notifications.
- Tests.

En la actualidad, a fecha 12/09/2014, SWADroid cuenta con más de 15 módulos entre los que se pueden destacar, por ejemplo:

- Documents.
- Messages.
- Groups.
- Information.

## 4.1. Desarrollo del módulo informativo.

Este módulo es el encargado de recoger el contenido de los apartados informativos que aparecen en SWAD:

- Introducción.
- Programa de teoría.
- Programa de prácticas.
- Guía docente.
- Bibliografía.
- FAQ.
- Enlaces.

Para la creación de este módulo se han seguido los siguientes pasos:

1. Crear el paquete correspondiente:

/SWADMain/src/es/ugr/swad/swadroid/modules/information

2. Crear el archivo .java correspondiente y su clase dentro de él:

Archivo → Information.java

```
public class Information extends Module { ... }
```

3. Declarar el módulo en el archivo *AndroidManifest.xml*.

```
<activity
    android:name="es.ugr.swad.swadroid.modules.information.Information"
    android:label="@string/informationModuleLabel"
    android:theme="@@android:style/Theme.NoTitleBar" >
</activity>
```

4. Añadir en */res/values/strings.xml* las etiquetas correspondientes:

Por ejemplo:

```
<string name="informationModuleLabel">Information</string>
<string name="introductionModuleLabel">Introduction</string>
```

5. Añadir una llamada al módulo desde la actividad que debe llamarlo:

Por ejemplo:

```
} else if (keyword.equals(getString(R.string.introductionModuleLabel))) {  
  
activity = new Intent(this, Information.class);  
activity.putExtra("requestCode", Constants.INTRODUCTION_REQUEST_CODE);  
startActivityForResult(activity, Constants.INTRODUCTION_REQUEST_CODE);  
  
}
```

6. Añadir las diferentes opciones al apartado correspondiente del menú principal

En este caso, se añaden a Asignatura (o en Evaluación en el caso de Sistema de Evaluación), por lo que no hay que crear un nuevo apartado en el menú, sino añadirlas dentro de Asignatura (o en Evaluación en el caso de Sistema de Evaluación).

Por ejemplo:

```
//Introduction category  
  
map = new HashMap<String, Object>();  
map.put(NAME, getString(R.string.introductionModuleLabel));  
map.put(IMAGE, getResources().getDrawable(R.drawable.notif));  
courseData.add(map);
```

En lugar de hacer:

```
final ArrayList<HashMap<String, Object>> headerData = new  
ArrayList<HashMap<String, Object>>();  
final HashMap<String, Object> introduction = new HashMap<String,  
Object>();  
introduction.put(NAME, getString(R.string.introduction));
```

Que es lo que se haría si quisiéramos añadir una nueva opción al menú en lugar de que esta opción aparezca dentro de otra.

## 7. Implementar el interfaz gráfico:

Que dependiendo del módulo que estemos implementando contendrá más o menos elementos. En el caso del módulo informativo el interfaz gráfico va a ser el siguiente:

```
<?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:background="@color/background"
    android:orientation="vertical">

    <include layout="@layout/action_bar"/>
    <include android:id="@+id/courseNameLayout"
        layout="@Layout/course_or_group_name"/>

    <WebView
        android:id="@+id/webview_dialog"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </WebView>
</LinearLayout>
```

El motivo de usar un WebView en lugar de, por ejemplo, un TextView lo veremos más adelante.

## 8. Implementar el módulo:

Todos los módulos de SWADroid deben heredar de la clase Module, que implementa una serie de métodos abstractos que todos los módulos están obligados a implementar para mantener el flujo de ejecución de SWADroid. Estos métodos son:

- *connect*

Este método se encarga de mostrar un cuadro de dialogo de progreso, si es necesario, y de lanzar la ejecución de la tarea implementada por el modulo en una hebra en segundo plano. En el caso de este módulo sería:

```
StringprogressDescription = getString(
    R.string.informationProgressDescription);
int progressTitle = R.string.informationProgressTitle;
startConnection(true, progressDescription, progressTitle);
```

- *postConnect*

Este método se encarga de realizar las acciones que deban llevarse a cabo después de haber ejecutado la tarea principal del módulo, como por ejemplo el refresco de la pantalla para actualizar la información mostrada.

Para el módulo informativo:

```
WebView webview = (WebView) this.findViewById(R.id.contentView);

if (infoSrc.equals("none")) {

    webview.loadData(getString(R.string.emptyInformation),
                    "text/html; charset=UTF-8", null);

} else if (infoSrc.equals("URL")) {

    webview.loadUrl(infoTxt);

} else {

    webview.loadData(infoTxt, "text/html; charset=UTF-8", null);

}

finish();

}
```

- *requestService*

Este método implementa la tarea principal que llevará a cabo el módulo, que normalmente consiste en construir y enviar una petición a un servicio Web de SWAD y procesar la respuesta devuelta por el mismo.

El flujo de trabajo de este método es el siguiente:

- Configurar el nombre del servicio Web al que se enviara la petición (está configurado durante la ejecución del método `onCreate`, que se ejecuta la primera vez que se lanza el módulo durante la ejecución de SWADroid, pero también puede hacerse en este método).

```
setMETHOD_NAME ("getCourseInfo");
```

b) Inicializar la petición, añadir a esta los parámetros requeridos por el servicio Web y enviarla a dicho servicio:

```
createRequest(SOAPClient.CLIENT_TYPE);
addParam("wsKey", Constants.getLoggedUser().getWsKey());
addParam("courseCode", Constants.getSelectedCourseCode());
addParam("infoType", infoTypeToAdd);
sendRequest(User.class, false);
```

c) Procesar la respuesta del servicio Web:

```
if (result != null) {
    SoapObject soap = (SoapObject) result;
    infoSrc = soap.getProperty(infoSrc).toString();
    infoTxt = soap.getPrimitiveProperty(infoTxt).toString();

    // Request finalized without errors
    setResult(RESULT_OK);
} else {
    infoTxt = getString(R.string.connectionRequired);
}
```

Como ya he dicho, estos métodos son los que TODOS los módulos implementan, pero dependiendo de qué modulo estemos implementando, necesitaremos añadir otros.

Cabe destacar el método ***onStart***, que es el encargado de iniciar el flujo de ejecución del módulo informativo y cuyo código es el siguiente:

```
super.onStart();
try {
    runConnection();
} catch (Exception e) {
    String errorMsg = getString(R.string.errorServerResponseMsg);
    error(TAG, errorMsg, e, true);
}
```

El resto de métodos que se han usado para implementar el módulo informativo pueden verse en la forja del proyecto. (<https://github.com/Amab/SWADroid>).

### **Problema:**

En la implementación de este módulo ha surgido el siguiente problema: cada profesor que gestiona una asignatura en SWAD, sube la información pertinente en el formato que a ellos les viene mejor, está dentro de estos formatos HTML, enlaces a otras páginas, texto plano, etc., y no sigue ningún tipo de estándar por lo que a la hora de mostrar la información en SWADroid hay que tener todos esos formatos en cuenta.

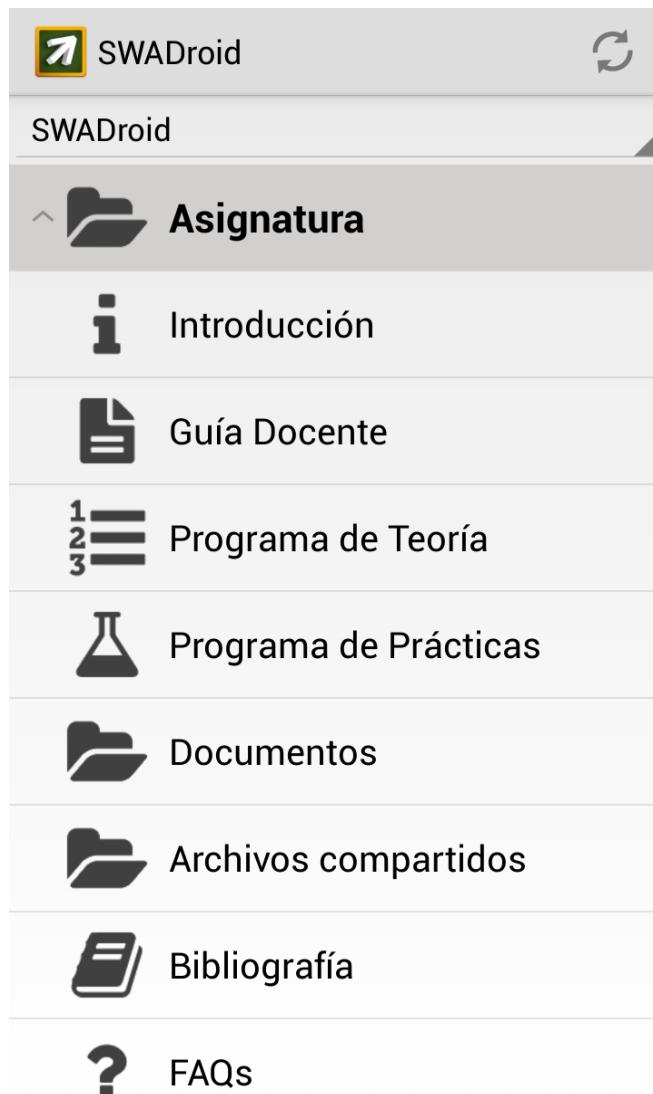
### *¿Cómo se ha solucionado este problema?*

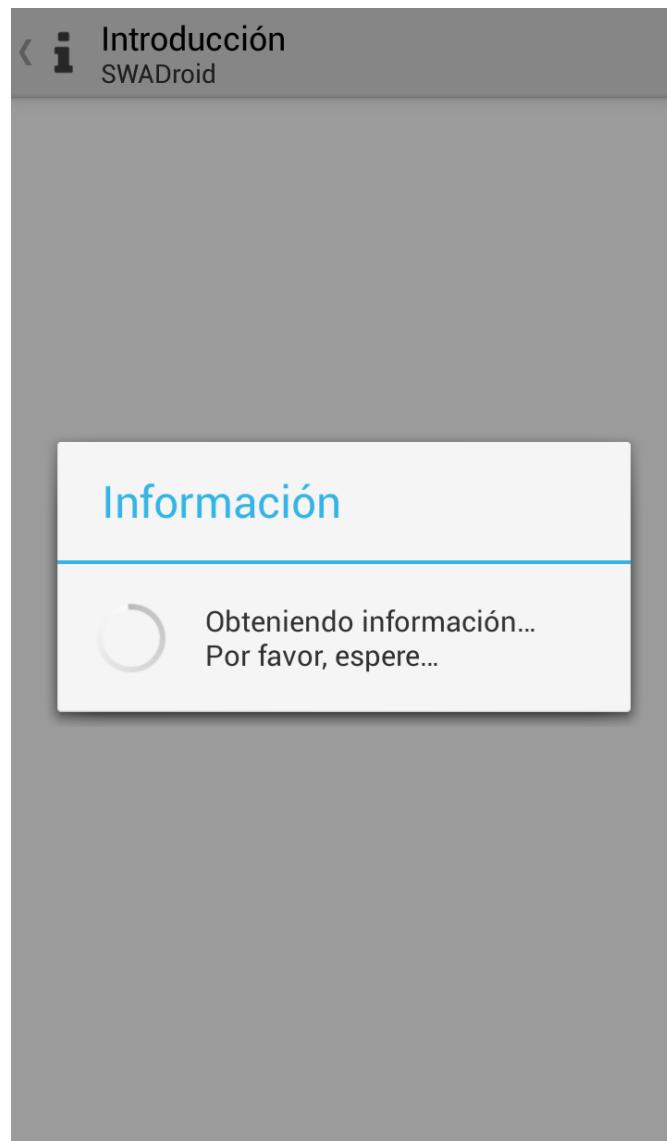
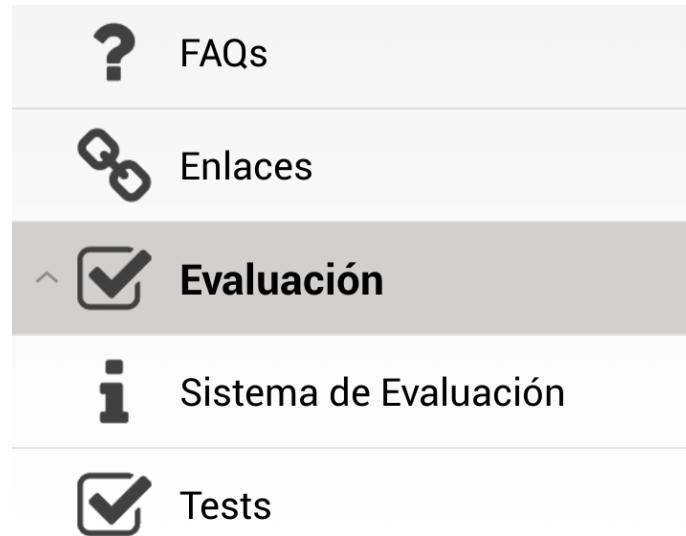
En un principio, esta información iba a cargarse en SWADroid en un TextView pero de haber sido así, no se podrían haber visualizado todos los formatos correctamente.

Visto esto, la solución fue usar un WebView (como ya se ha comentado anteriormente), el cual es capaz de cargar texto formateado (HTML por ejemplo) e incluso abrir una página Web incrustándola en nuestra app sin necesidad de abrir el propio navegador.

Tomando como solución final el WebView, lo único que hay que hacer es comprobar la información que nos devuelve el Web Service e invocar a la función de carga correspondiente del WebView tal y como se ha mostrado anteriormente en el método postConnect de este módulo.

El aspecto final de SWADroid tras la implementación de este módulo es el siguiente:





La inclusión del WebView hace posible que se pueda visualizar, por ejemplo, tanto texto plano como texto formateado en HTML en el mismo campo como se muestra a continuación:

A screenshot of a mobile application interface. At the top, there is a navigation bar with a back arrow icon, a blue information icon, the title "Introducción" in bold black font, and the text "SWADroid" below it. The main content area contains the text: "Esta asignatura es un espacio utilizado para la coordinación del proyecto SWADroid."

A screenshot of a mobile application interface. At the top, there is a navigation bar with a back arrow icon, a blue document icon, the title "Guía Docente" in bold black font, and the text "SWADroid" below it. The main content area contains the text: "This text is bold", "This text is strong", "This text is italic", "This text is emphasized", "This is computer output", and "This is subscript and superscript".

## 4.2. Desarrollo del módulo de mensajes.

El desarrollo de este módulo ha sido especialmente complicado debido, en su mayor parte, a que ya existía. En un principio se pensó en modificar el módulo existente y reutilizar otros que nos permitían, por ejemplo, descargar los usuarios para poder mostrarlos en el listado. Se comenzó por esta opción, pero conforme se iba avanzando, únicamente surgían problemas que cada vez eran más complicados de solucionar, así que se optó por implementar un nuevo módulo.

El nuevo módulo (aparte de, obviamente, enviar mensajes):

- Tiene una interfaz completamente rediseñada.
- Permite mostrar un listado con las fotos y los nombres de los usuarios de la asignatura separados en alumnos y profesores.
- Permite filtrar el listado de usuarios.
- Permite marcar usuarios para añadirlos como destinatarios.

A continuación se muestra cómo se ha implementado el nuevo módulo y cómo se han ido solucionando los diferentes problemas e inconvenientes que han ido surgiendo.

### 1. Crear la nueva clase Messages:

Esta nueva clase debe contener un listado de destinatarios (*receivers*) que son los que se añaden manualmente (los que escribimos nosotros en el campo destinatarios) y completarlo con los destinatarios que marquemos de la lista de usuarios.

En la clase Messages se llamará a una actividad que mostrará el listado de usuarios y este será el que devuelva a la actividad principal el listado de usuarios marcados. Esta actividad se invoca mediante un botón que está incluido en la ActionBar y que forma parte del rediseño de la interfaz de este módulo.

Esto se realiza de la siguiente manera:

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_addUser:
            Intent callUsersList = new Intent (getBaseContext(),
                UsersList.class);
            startActivityForResult(callUsersList, 0);
            return true;

        ...
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Como vemos, lo que se hace es invocar una nueva actividad con la clase UsersList.

## 2. Crear la clase UsersList:

De esta clase cabe destacar 3 cosas:

1. *Contiene un método para descargar el listado de usuarios:*

```
private void downloadUsersList() {  
    Intent downloadUsersList = new Intent(getApplicationContext(),  
        DownloadUsers.class);  
    startActivity(downloadUsersList);  
    setChildGroupData();  
}
```

Con esto, se invoca una nueva actividad que descarga los usuarios. La clase DownloadUsers contiene un método que es el que se encarga de realizar la petición al Web Service para descargar los usuarios:

```
protected void requestService() throws Exception {  
  
    int userRole = Constants.TEACHER_TYPE_CODE;  
    long courseCode = Constants.getSelectedCourseCode();  
    long groupCode = getIntent().getLongExtra("groupCode", 0);  
  
    // Creates webservice request, adds required params  
    // and sends request to webservice  
    createRequest(SOAPClient.CLIENT_TYPE);  
    addParam("wsKey", Constants.getLoggedUser().getWsKey());  
    addParam("courseCode", (int) courseCode);  
    addParam("groupCode", (int) groupCode);  
    addParam("userRole", userRole);  
    sendRequest(User.class, false);  
  
    if (result != null) {  
  
        // Stores users data returned by webservice response  
        ArrayList<?> res = new ArrayList<Object>((Vector<?>) result);  
        SoapObject soap = (SoapObject) res.get(1);  
        numUsers = soap.getPropertyCount();  
        for (int i = 0; i < numUsers; i++) {  
            SoapObject pii = (SoapObject) soap.getProperty(i);  
  
            long userCode = Long.valueOf(pii.getProperty("userCode")  
                .toString());  
            String userID = pii.getProperty("userID").toString();  
            String userNickname = pii.getProperty("userNickname")  
                .toString();  
            String userSurname1 = pii.getProperty("userSurname1")  
                .toString();  
            String userSurname2 = pii.getProperty("userSurname2")  
                .toString();  
            String userFirstName = pii.getProperty("userFirstname")  
                .toString();  
            String userPhoto = pii.getProperty("userPhoto").toString();
```

```

        if (userNickname.equalsIgnoreCase(Constants.NULL_VALUE)){
            userNickname = "";
        }
        if (userSurname1.equalsIgnoreCase(Constants.NULL_VALUE)){
            userSurname1 = "";
        }
        if (userSurname2.equalsIgnoreCase(Constants.NULL_VALUE)){
            userSurname2 = "";
        }
        if (userFirstName.equalsIgnoreCase(Constants.NULL_VALUE)){
            userFirstName = "";
        }
        if (userPhoto.equalsIgnoreCase(Constants.NULL_VALUE)){
            userPhoto = "";
        }

        User u = new User(
            userCode,                                // id
            null,                                     // wsKey
            userID,
            userNickname,
            userSurname1,
            userSurname2,
            userFirstName,
            userPhoto,                                // photoPath
            userRole);

        // Inserts user in database or updates it if already exists
        dbHelper.insertUser(u);
        dbHelper.insertUserCourse(userCode, courseCode, groupCode);

    }      // end for (int i=0; i < usersCount; i++)

    if (isDebuggable) {
        Log.d(TAG, "Retrieved " + numUsers + " users");
    }
}      // end if (result != null)

// Request finalized without errors
setResult(RESULT_OK);
}

```

Este método es el encargado de hacer la petición de descarga de usuarios al Web Service como ya se ha comentado y de guardar dichos usuarios en nuestra base de datos para poder mostrarlos posteriormente en forma de listado.

## *2. Permite actualizar el listado de usuarios:*

Lo único que hace este método (que en realidad es una opción del menú de la ActionBar) es invocar una actividad con la clase DownloadUsers como se hacía anteriormente. Esto vuelve a descargar el listado de usuarios y actualiza la base de datos. El código de esta funcionalidad se implementa de la siguiente manera:

```
public boolean onOptionsItemSelected(MenuItem item) {  
  
    switch (item.getItemId()) {  
        case R.id.action_refresh_users:  
            Intent refreshUserList = new Intent (getBaseContext(),  
                                              DownloadUsers.class);  
            startActivityForResult(refreshUserList, 0);  
            return true;  
  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

## *3. Muestra el listado de usuarios en un ExpandableListView*

Esta, posiblemente, sea una de las partes más importantes del nuevo módulo de mensajes. El motivo es que ahora, la persona que vaya a enviar un mensaje, podrá ver de manera clara todos los usuarios de la asignatura y así podrá seleccionar a quién quiere enviarle el mensaje sin necesidad de conocer su DNI o su apodo como sí que era necesario con el anterior diseño del módulo, motivo por el cual estaba prácticamente en desuso.

La lista expandible tiene numerosos métodos que se pueden consultar, como ya se ha dicho en varias ocasiones, en la forja del proyecto. Los más importantes son los siguientes:

- *setGroupData():*

Con este método se crean “los padres” del listado, es decir, se crean las diferentes secciones que contendrá la lista. En nuestro caso son dos secciones, profesores y alumnos. El código es el siguiente:

```
private void setGroupData() {  
    groupItem.add(getString(R.string.Filters_Teachers));  
    groupItem.add(getString(R.string.Filters_Students));  
}
```

- ***setChildGroupData():***

Este método es el encargado de llenar “los hijos”, es decir, se encarga de completar las diferentes secciones que se crean en el método setGroupData. Este es el código que posibilita llenar las secciones:

```

private void setChildGroupData() {
    List<User> child = null;
    //Clear data
    childItem.clear();

    //Add data for teachers
    child = dbHelper.getAllRows(Constants.DB_TABLE_USERS,
        "userRole='"
        + Constants.TEACHER_TYPE_CODE+"'", orderby);

    Collections.sort(child);
    childItem.add(child);

    //Add data for students
    child = dbHelper.getAllRows(Constants.DB_TABLE_USERS, "userRole<>'"
        + Constants.TEACHER_TYPE_CODE+"'", orderby);

    Collections.sort(child);
    childItem.add(child);

    Log.d(TAG, "groups size=" + childItem.size());
    Log.d(TAG, "teachers children size=" + childItem
        .get(TEACHERS_GROUP_ID)
        .size());
    Log.d(TAG, "students children size=" + childItem
        .get(STUDENTS_GROUP_ID)
        .size());

    adapter = new ExpandableStudentsListAdapter(
        this, groupItem, childItem);

    if(dbHelper.getAllRowsCount(Constants.DB_TABLE_USERS) > 0) {
        Log.d(TAG, "[setChildGroupData] Users table is not empty");
        list.setAdapter(adapter);
    } else {
        Log.d(TAG, "[setChildGroupData] Users table is empty");
        list.setOnChildClickListener(null);
    }
}

```

Los usuarios marcados se añaden a la lista de destinatarios de la siguiente manera:

```
//Accept receivers list marked at users list
if (!childItem.isEmpty()){
    for (int i = 0; i < childItem.size(); i ++){
        int tam = childItem.get(i).size();

        for(int j = 0; j < tam; j++){

            User usu = childItem.get(i).get(j);
            if (usu.isSelected()){
                String us = usu.getUserNickname().toString();
                rcvs_Aux = rcvs_Aux + "@" + us + ",";
            }
        }
    }

    //If receivers, remove the last comma
    if(!rcvs_Aux.isEmpty()){
        rcvs = rcvs_Aux.substring(0, rcvs_Aux.length()-1);
    }
}

Intent resultData = new Intent();
resultData.putExtra("ListaRcvs", rcvs);
setResult(Activity.RESULT_OK, resultData);
```

### 3. Crear la clase ExpandableStudentsListAdapter:

Para poder realizar todo esto, se ha tenido que crear esta nueva clase que contiene un adapter en la cual se generan las vistas donde se mostraran los usuarios. Del adapter, que cuenta con numerosos métodos, hay que destacar dos, que son los que se encargan de llenar las diferentes vistas:

- *getGroupView(...):*

Este método se encarga de crear y devolver la vista de lo que anteriormente llamábamos “los padres”. Este es el código:

```
public View getGroupView(int groupPosition, boolean isExpanded,
                         View convertView, ViewGroup parent) {

    TextView groupTitle;

    if (convertView == null) {
        convertView = minflater.inflate(
                    R.layout.messages_group_item, parent, false);
    }

    groupTitle = (TextView) convertView.
                  findViewById(R.id.messages_group_item_title);
    groupTitle.setText(groupItem.get(groupPosition));

    return convertView;
}
```

- ***getChildView(...):***

Este método se encarga de devolver la vista de lo que antes se ha definido como “los hijos”. Este es su código:

```

public View getChildView(int groupPosition, int childPosition,
                        boolean isLastChild, View convertView,
                        ViewGroup parent) {

    //Bitmap bMap = null;
    final User u = (User) childItem.get(groupPosition).get(childPosition);
    final String fullName = u.getUserSurname1()
        +
        + u.getUserSurname2()
        +
        +
        + ", "
        + u.getUserFirstname();

    final String photoFileName = u.getUserPhoto();
    Log.i("fileName:", "+photoFileName");

    if (convertView == null) {
        convertView = inflater.inflate(
            R.layout.messages_child_item, parent, false);
    }

    ImageView image = (ImageView) convertView.findViewById(
        R.id.imageView1);
    TextView text = (TextView) convertView.findViewById(R.id.TextView1);
    final CheckBox checkbox = (CheckBox) convertView.findViewById(
        R.id.check);

    checkbox.setOnCheckedChangeListener(
        new CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
                isChecked) {
                checkbox.setSelected(buttonView.isChecked());
                u.setSelected(isChecked);
            }
        });
}

DisplayImageOptions options = new DisplayImageOptions.Builder()
    .showImageForEmptyUri(R.raw usr_bL)
    .showImageOnFail(R.raw usr_bL)
    .cacheInMemory(true).build();

ImageLoaderConfiguration config = new ImageLoaderConfiguration
    .Builder(activity.getApplicationContext())
    .defaultDisplayImageOptions(options).build();
ImageLoader.getInstance().init(config);
ImageLoader.getInstance().displayImage(photoFileName, image);

text.setText(fullName);
text.setTag(u.getUserNickname());
checkbox.setChecked((new StudentItemModel(u)).isSelected());

return convertView;
}

```

Respecto a este código, es de destacar el método `setOnCheckedChangeListener` que es el que se encarga de dar la funcionalidad correcta al checkbox para que podamos marcarlo y comprobar si hemos seleccionado o no un usuario como destinatario y el uso de ImageLoader para la carga de las imágenes. ImageLoader forma parte de una biblioteca externa que se comentará más adelante.

#### 4. Crear la nueva interfaz:

Como se ha mencionado en apartados anteriores, uno de los temas más importantes en el rediseño del módulo de mensajes ha sido el tema de la interfaz. Esto es debido a que la antigua interfaz, era tan simple, que influía negativamente en el uso del módulo. Como se ha comentado ya varias veces, la anterior interfaz obligaba a que un usuario tuviese que conocer el DNI o el apodo en SWAD del destinatario al que quería escribir.

La nueva interfaz, aparte de ser estilo HOLO (como es ahora todo el proyecto) tiene mucha más funcionalidad que la anterior.

Esta interfaz consta de 4 partes:

- *Messages\_screen.xml:*

Contiene los elementos que componen la ventana principal que será la que se muestre cuando seleccionemos la opción escribir mensaje en el menú principal:

```
<?xml version='1.0' encoding='UTF-8'?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    android:gravity="center_horizontal"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/LinearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:orientation="vertical" >

        <EditText
            android:id="@+id/message_receivers_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="@string/message_receivers_title"
            android:textColor="@color/foreground1" />

        <EditText
            android:id="@+id/message_subject_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/message_subject_title"
            android:textColor="@color/foreground1" />

    </LinearLayout>
</RelativeLayout>
```

```

<EditText
    android:id="@+id/message_body_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/message_body_title"
    android:textColor="@color/foreground1" />
</LinearLayout>

</RelativeLayout>

```

- *Users\_listview.xml:*

Es la vista que se muestra cuando seleccionamos la opción de añadir destinatarios en la ventana principal:

```

<?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" >

    <Spinner
        android:id="@+id/spinner_messages_users_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="0.81"
        android:gravity="center_horizontal"
        android:orientation="vertical" >

        <ExpandableListView
            android:id="@+id/users_explistview"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_above="@+id/acceptList"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:background="@color/background"
            android:descendantFocusability="blocksDescendants">
        </ExpandableListView>

        <Button
            android:id="@+id/acceptList"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:text="@string/acceptMsg" />

    </RelativeLayout>
</LinearLayout>

```

- *Messages\_group\_item.xml:*

Es la vista que rellenará el método getGroupView comentado en puntos anteriores:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/gray_goose"
    android:padding="5dp" >

    <TextView
        android:id="@+id/messages_group_item_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/padLockIcon"
        android:layout_toRightOf="@+id/padLockIcon"
        android:paddingEnd="10dp"
        android:paddingLeft="35dp"
        android:paddingRight="10dp"
        android:paddingStart="35dp"
        android:textColor="@android:color/black"
        android:textSize="20sp"
        android:textStyle="bold" />

</RelativeLayout>
```

- *Messages\_child\_item.xml:*

Es la vista que rellena el método getChildView comentado anteriormente:

```
<?xml version='1.0' encoding='UTF-8'?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/background"
    android:gravity="center_vertical"
    android:orientation="horizontal"
    android:descendantFocusability="blocksDescendants">

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"
        android:contentDescription="@string/photoContentDescription"
        android:scaleType="centerCrop" />
```

```

<TextView
    android:id="@+id/TextView1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dip"
    android:layout_weight="1"
    android:text=""
    android:textColor="@color/foreground1" />

<CheckBox
    android:id="@+id/check"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>

```

El resto del código de los métodos de las diferentes clases se puede consultar, como viene siendo habitual, en la forja del proyecto.

### ***Problemas:***

El rediseño de este módulo ha sido especialmente complicado y han ido surgiendo numerosos problemas que han dado lugar a una gran cantidad de cambios en el proyecto.

El principal problema ha sido la carga de las fotografías por los siguientes motivos:

- Si descargábamos las imágenes en la SD, los dispositivos sin SD no podrían ver las imágenes de los usuarios.
- Si descargábamos las imágenes en la memoria interna del teléfono, eran visibles en la galería, lo cual no es viable ya que las fotografías de los usuarios son privadas y sólo deberían de estar disponibles dentro de la aplicación SWADroid.
- Por modificaciones y/o actualizaciones de la API de Android, las imágenes se guardaban como corruptas tanto en la SD como en la memoria interna si reutilizábamos los métodos que ya existían para descargar los usuarios y guardarlos.

Otro gran problema ha sido que, para la implementación del ExpandableListView, no se podían usar los métodos que existían para obtener datos de la base de datos.

Por último, la implementación de las nuevas funcionalidades del módulo ha dado lugar a modificaciones en las funciones del Web Service que nos permitían obtener el listado de usuarios de las asignaturas.

### ***¿Cómo se han solucionado estos problemas?***

A parte de, como se ha explicado en los puntos anteriores, implementar las nuevas clases y la nueva interfaz, se ha tenido que hacer uso de una biblioteca externa para la carga de las imágenes (universal-image-loader-1.9.2.jar de nostra13, que se puede obtener de <https://github.com/nostra13/Android-Universal-Image-Loader>).

Con esta biblioteca externa, podemos descargar las imágenes en caché y mostrarlas, lo que nos simplifica el trabajo y nos reduce el espacio en disco de la aplicación. El uso de esta biblioteca para la carga de imágenes está expuesto en el método getChildView que se ha explicado anteriormente.

Respecto al problema de la obtención de datos de la base de datos, se han tenido que modificar numerosos métodos del dbHelper para que devuelvan un resultado correcto. A modo de ejemplo, se muestra el método getAllRows que nos devuelve todas las filas de una tabla de la base de datos:

```
public <T extends Model> List<T> getAllRows(String table, String where,
String orderby) {
    List<T> result = new ArrayList<T>();
    List<Entity> rows = db.getEntityList(table, where, orderby);
    T row;

    if (rows != null) {
        for (Entity ent : rows) {
            row = createObjectByTable(table, ent);
            result.add(row);
        }
    }

    return result;
}
```

Con estas modificaciones se solucionan el problema de la carga de las fotografías y el de completar las vistas con los usuarios descargados.

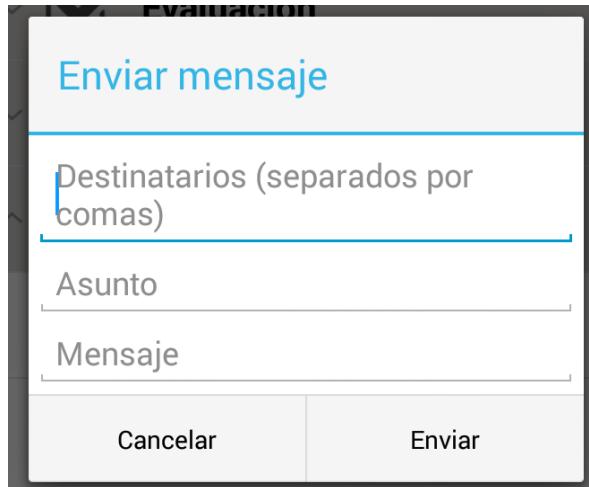
Respecto al tema de las modificaciones en el Web Service, Antonio Cañas Vargas, tutor del proyecto, es el encargado de realizarlas. En este caso concreto se han añadido modificaciones para que:

- *Cualquier usuario pueda descargar el listado de usuarios.* Anteriormente, solo se permitía descargar el listado a los profesores, por lo que solamente podrían hacer uso de las nuevas funcionalidades del módulo de mensajes ellos, reduciendo así la funcionalidad del mismo, lo que haría que todas las modificaciones realizadas fuesen inservibles.
- *Se devuelvan todos los usuarios en una única petición.*
- *Se devuelva el grupo al que pertenece el usuario (Teoría, prácticas o ninguno).*

En resumen, se ve claramente cómo implementar un nuevo módulo no es tan sencillo como parece a priori. En el caso del módulo de mensajes, para dotarlo de una buena funcionalidad se ha tenido que:

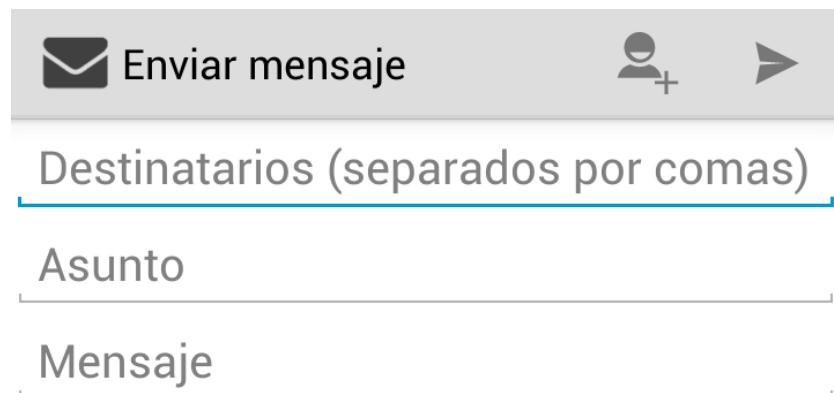
- Implementar numerosas clases nuevas.
- Modificar clases existentes.
- Modificar métodos de la base de datos.
- Modificar la interfaz.
- Modificar métodos del Web Service.

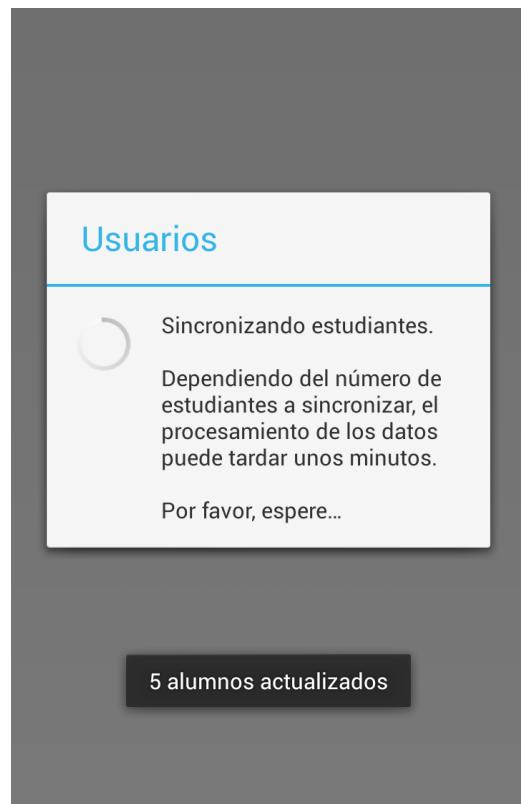
El resultado, tras implementar la nueva interfaz y las nuevas funcionalidades, ha sido pasar de un simple cuadro de diálogo que solo nos permitía escribir un mensaje:



a una vista en pantalla completa con una alta funcionalidad.

El aspecto que tiene SWADroid tras la implementación de es el siguiente:





Seleccionar Destinatarios

Profesores

	Aguilera Malagón, Antonio Manuel	<input checked="" type="checkbox"/>
	Alcalde Barros, Alejandro	<input type="checkbox"/>
	Boyero Corral, Juan Miguel	<input type="checkbox"/>
	Cañas Vargas, Antonio	<input checked="" type="checkbox"/>
	Guerrero Avilés, José Antonio	<input type="checkbox"/>

**Aceptar**



Enviar mensaje



@aguilerin,@acanas

Asunto

Mensaje

## 5. Internacionalización.

SWADroid provee soporte para varios idiomas. Antes de la realización estaba disponible en numerosos idiomas, como son el inglés, español, francés, catalán, polaco, etc. Aunque el idioma no podía ser seleccionado explícitamente por el usuario, sino que se establece automáticamente en función del idioma configurado en el sistema.

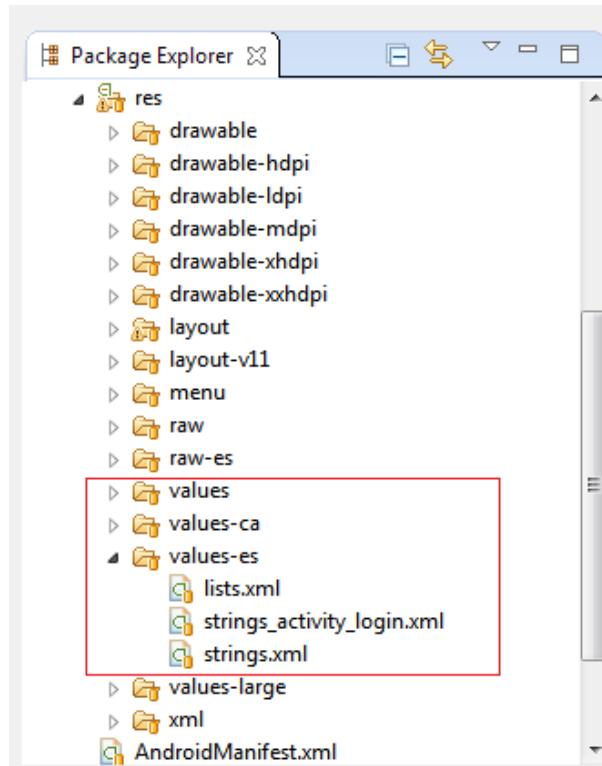
Tras varios estudios, nos dimos cuenta que la mayoría de los idiomas o bien no se usaban o bien eran usado por una minoría muy explícita, por lo que se decidió eliminarlos. Actualmente SWADroid está disponible en 3 idiomas: catalán, inglés y español.

Las cadenas de texto de cada idioma se encuentran almacenadas en archivos XML dentro de la carpeta res/values seguida del código del idioma al que corresponden. Así, la carpeta correspondiente al idioma español es res/values-es. Si no se especifica ningún código después de values, se tomará por defecto como inglés.

Es muy sencillo traducir SWADroid a otros idiomas. Sólo es necesario traducir las cadenas de texto que se encuentran entre las etiquetas <string> y </string> sin modificar en ningún caso el parámetro name de la etiqueta <string>, ya que éste es el identificador que utilizar SWADroid para acceder a la cadena de texto en el idioma apropiado.

Una vez traducidos los archivos que se encuentran en res/values-es (por ejemplo), deben incluirse en una nueva carpeta cuyo nombre debe ser res/values-[CODIGO\_IDIOMA]. Hecho esto, el nuevo idioma estará disponible en SWADroid.

Podemos ver estos archivos y los idiomas disponibles en la siguiente imagen:



## **6. Programación del proyecto.**

En este apartado se van a explicar las APIs y tecnologías utilizadas para la elaboración de este proyecto.

### **6.1 Introducción.**

Los inicios de este sistema provienen del año 2005 cuando Google compra una pequeña compañía llamada Android Inc, cuya finalidad es el desarrollo de aplicaciones para dispositivos móviles, centradas principalmente en plataformas Web.

La existencia del nuevo sistema operativo no se hace pública hasta el año 2007, momento en el que se anuncia la creación de la Open Handset Aliance. Esta organización es un consorcio de compañías relacionadas con las telecomunicaciones y la informática, cuyo principal objetivo es la promoción del sistema operativo Android y su política las licencias de código abierto.

La compañía HTC lanza a finales de septiembre de 2008 el primer teléfono móvil con este sistema operativo. Desde entonces, son ya varias las compañías que están apostando por Android. A modo de resumen estos son los tres nombres más importantes que se pueden relacionar con esta plataforma.

**Google:** esta compañía es en la actualidad una de las más poderosas del mundo, su rápido crecimiento en tan solo unos pocos años de vida hizo que en 2007 se convirtiera en la marca más valorada del mundo por delante de Microsoft o Coca-Cola, empresas que acostumbraban a dominar esta posición.

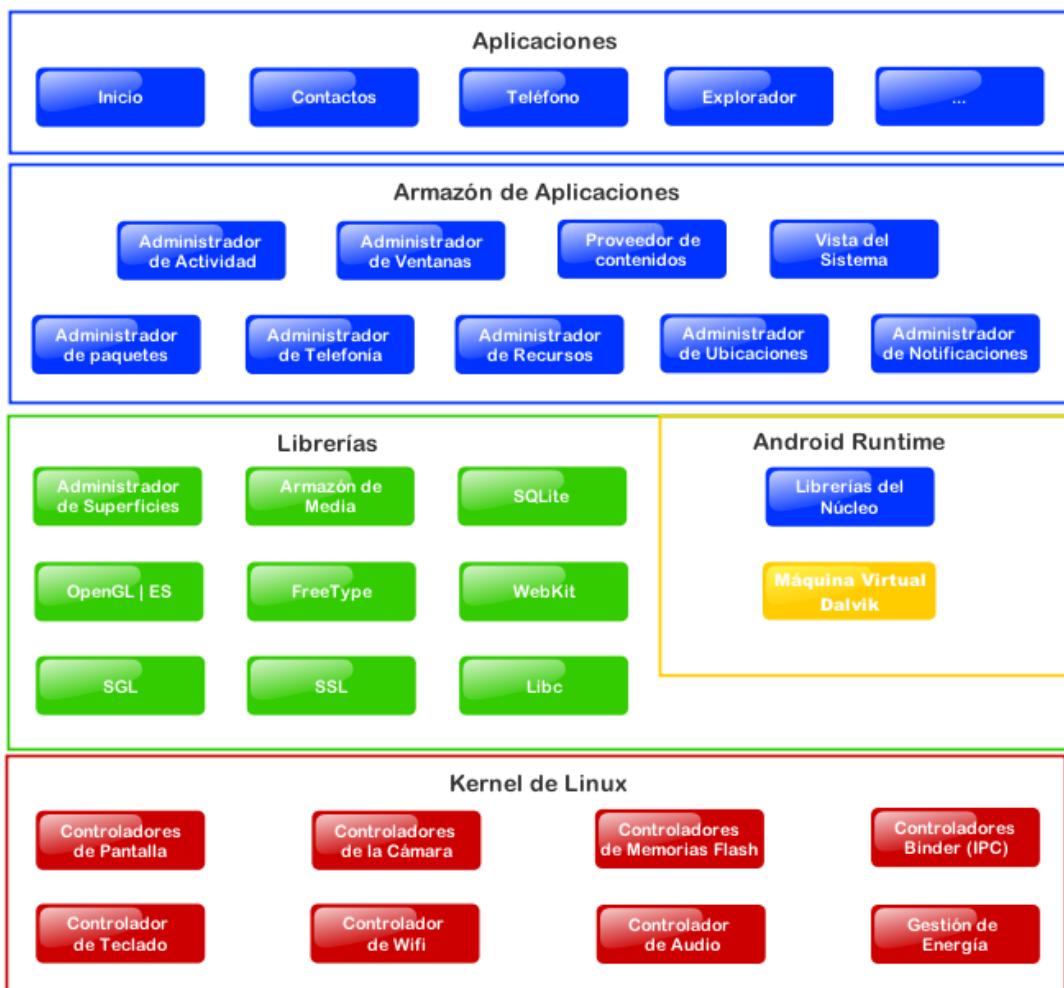
Google lidera la mencionada Open Handset Aliance. El hecho de que esta compañía de tanta importancia sea la principal promotora de Android aporta dos grandes ventajas tanto a los usuarios de sus teléfonos móviles como a los desarrolladores de aplicaciones para esta plataforma.

La primera ventaja es que esta empresa apoya actualmente diversos proyectos de software libre y licencias de código abierto. Ocurre lo mismo con el caso de Android, lo que provoca que cualquier programador comience a trabajar cuando desee sin tener que pagar ninguna licencia para desarrollar aplicaciones o darlas a conocer en el Android Market. Este último es un repositorio de aplicaciones al que se puede acceder desde cualquier Android, quedando a elección del desarrollador que su aplicación sea gratuita o de pago.

La segunda ventaja, es la cantidad de productos pertenecientes a Google tales como Gmail, Youtube, Google Maps, Google Calendar... que cuentan actualmente con un nivel de aceptación muy alto por parte de los usuarios. Todos los Android cuentan con un acceso rápido y eficiente a estos productos y, al mismo tiempo, Google proporciona APIs para que el programador pueda incluir estos servicios en sus aplicaciones de forma muy sencilla.

**Linux:** Android es un sistema operativo abierto para dispositivos móviles, y se basa en la versión 2.6 del Kernel de Linux que actúa como una capa de abstracción entre el hardware y el resto del conjunto de software, además de prestar los servicios de seguridad, gestión de memoria, gestión de procesos..., que permite que pueda ser adaptado con facilidad a otros dispositivos. Esto presenta una gran ventaja para las compañías fabricantes de teléfonos móviles.

A continuación se muestra un diagrama con los principales componentes del sistema operativo Android, que serán comentados posteriormente.



- Aplicaciones: suministra un conjunto de aplicaciones que incluye cliente de correo electrónico, proveedor de SMS, calendario, mapas, navegador web, contactos... más todas aquellas aplicaciones que el usuario desee descargar del Play Store.

- Framework de aplicaciones: proporciona acceso a los marcos utilizado por la API de las aplicaciones básicas. La arquitectura de aplicaciones se ha diseñado para simplificar la reutilización de componentes. De esta forma cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación podrá hacer uso de esas capacidades (sujeto a las limitaciones de seguridad impuestas por el marco). Este hecho permite la reutilización de componentes por parte del usuario.  
Detrás de todas las aplicaciones existen un conjunto de servicios y sistemas que incluyen un conjunto de vistas para construir las pantallas, así como proveedores de contenido que permiten que las aplicaciones de acceso a los datos de otras aplicaciones (como Contactos), o para compartir sus propios datos.
- Librerías: incluye un conjunto de librerías escritas en C y C++ utilizado diversos componentes del sistema. Estas capacidades están expuestas a los desarrolladores a través del Framework.
- Android Runtime: incluye un conjunto de librerías que proporciona la mayor parte de la funcionalidad disponible en el núcleo de las bibliotecas y está basado en el lenguaje de programación Java. Cada aplicación se ejecuta como un proceso independiente, con su propia instancia de la máquina virtual DalvÍk que se ha escrito de modo que un dispositivo puede ejecutar múltiples máquinas virtuales de manera eficiente. La máquina virtual de DalvÍk ejecuta archivos .dex cuyo formato está optimizado para un consumo de memoria mínima. La máquina virtual se basa en registros, y corre clases compiladas por un compilador de lenguaje Java que se han transformado con la herramienta incluida dx. La máquina virtual de DalvÍk se basa en el Kernel de Linux para la funcionalidad subyacente como subprocessos y de gestión de memoria a bajo nivel.
- Kernel de Linux: explicado antes de mostrar el diagrama.

**Java:** el tercer nombre importante asociado al sistema operativo Android es Java. Como se comentaba anteriormente, cualquier programador se adapta al funcionamiento de Android en muy poco tiempo, dado que el código fuente de una aplicación se escribe en Java y este es uno de los lenguajes de programación más utilizados en la actualidad. Más adelante se explicarán detalles de programación.

Además, como ventaja añadida el “*principal inconveniente*” que se suele asociar con Java es su velocidad de ejecución. En este caso, como se ha explicado antes, ese problema ni siquiera existe dado que, aunque la aplicación se escribe en Java, no es una aplicación Java ya que no se ejecuta en una máquina virtual de Java. De esta forma el programador cuenta con todas las ventajas de este lenguaje, y desaparece una de sus mayores desventajas.

## 6.2. Especificaciones técnicas.

Antes de comenzar con los detalles de programación, se explica todo lo necesario para comenzar a desarrollar aplicaciones así como los pasos necesarios para la instalación de todos sus componentes. Estos pasos pueden seguirse también de forma resumida en la página oficial de Android, en el apartado Developers → Download SDK desde el cual, además, podemos descargar todos los elementos necesarios que a continuación se explican.

Las imágenes y configuraciones que se van a mostrar son bajo las siguientes versiones de los diferentes programas:

- Eclipse 4.2.1.v20130118.
- Plugin ADT 23.0.2.1256578.
- Plugin EGit 2.2.0.201212191850-r
- JDK v1.7.0\_55

**JDK (Java Development Kit):** en primer lugar es necesario tener instalado Java, la versión mínima del JDK debe ser la 5. Puede comprobarse si se dispone de la versión necesaria introduciendo en la ventana de comandos de Windows (cmd.exe) el siguiente comando (debería mostrar al menos 1.5 para que sea la correcta):

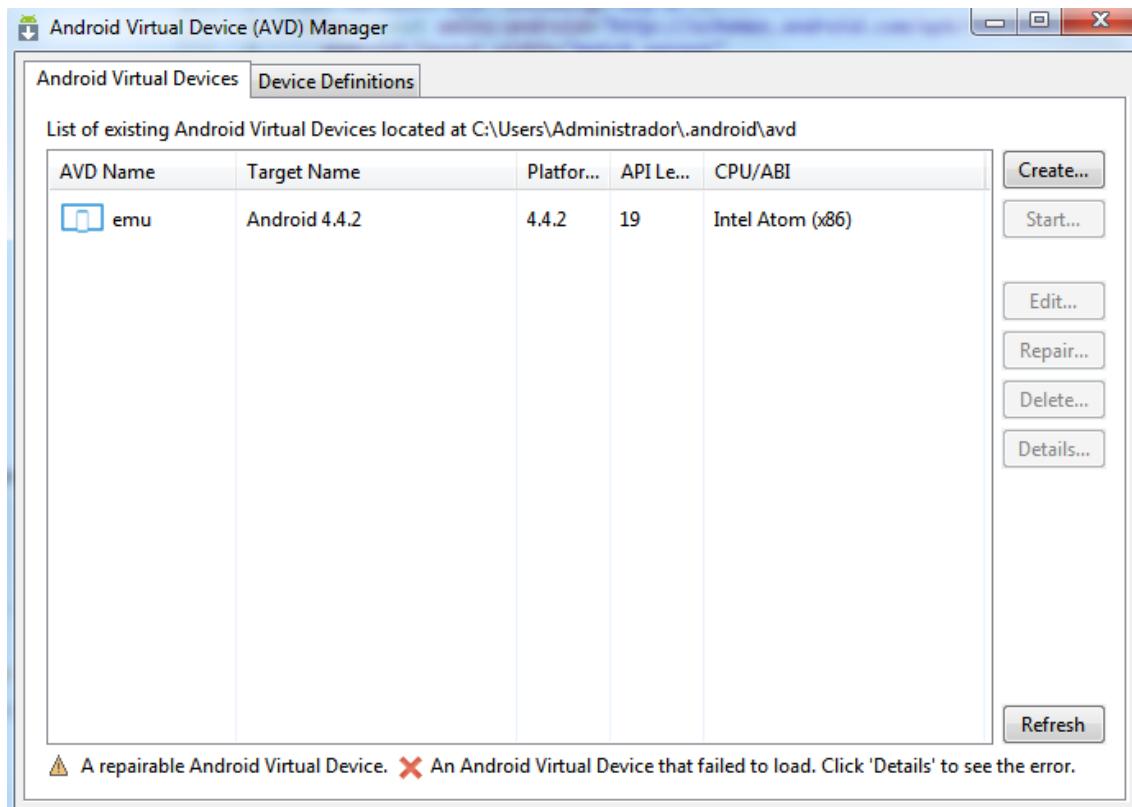
```
java -version
```

**Android SDK:** mientras que el JDK proporciona lo necesario para utilizar las clases de Java, el SDK da acceso a las clases necesarias para construir una aplicación Android. Para ello se descarga de la página oficial de Android en el apartado comentado anteriormente, seleccionando la versión correspondiente a la máquina en la que se vaya a trabajar (Mac, Windows o Linux). Esto permite guardar un archivo comprimido, que únicamente hay que descomprimir y guardar la carpeta resultante en el directorio en el que se deseé tener el SDK.

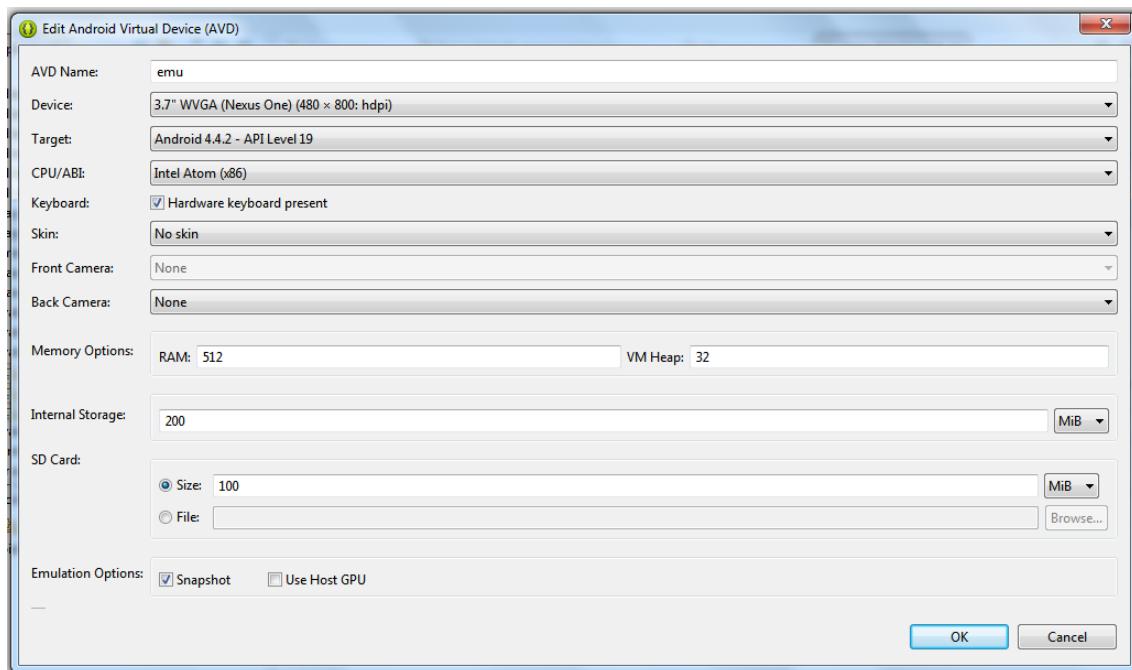
Tanto la página oficial como los libros consultados afirman que no es necesario nada más y que simplemente debe descomprimirse el archivo y posicionarlo donde se deseé.

El siguiente paso es crear un dispositivo virtual (puede realizarse en cualquier momento antes de probar la aplicación), lo que permitirá probar la aplicación en el emulador antes de instalarla en un dispositivo real.

Se pueden crear cuantos emuladores se deseé y con diversas características. Por ejemplo se puede crear uno con la versión 1.5 de Android que fue la primera en comercializarse en España y otro con la última versión para comprobar que funcione todo correctamente en diferentes versiones.



Para crearlos se accede a la opción Window → Android Virtual Device Manager y se pulsa Create en la ventana que nos aparece (es la que se ha mostrado anteriormente). En la nueva ventana se debe establecer un nombre identificador para cada dispositivo virtual, en el apartado Target la versión de sistema operativo que se desee y el tamaño que se quiera que tenga su tarjeta de memoria. Tras llenar todos los campos requeridos correctamente ya tendremos creado nuestro emulador.



**Eclipse IDE:** es un entorno de desarrollo para compilar y ejecutar aplicaciones. Si se prefiere NetBeans también cuenta con su plugin para programar en Android, pero todos los libros y la página oficial de Android se centran en la herramienta Eclipse.

La versión mínima necesaria es la 3.4, ya sea de Classic o para desarrolladores de Java. En este caso sí que es suficiente con sólo descargar el archivo y descomprimirlo, ya que no es necesaria ninguna instalación.

Además de todo esto, concretamente para este proyecto, es necesaria la instalación del plugin EGit para poder integrar el control de versiones con Eclipse.

Como ya se ha mencionado en numerosas ocasiones, SWADroid es un proyecto colaborativo en el que participan muchas personas, lo que implica que siempre debemos tener el proyecto actualizado con los últimos cambios y estos a su vez, deben estar disponibles para el resto del equipo de desarrollo. Es por eso por lo que integrar Git en Eclipse es la manera más rápida y sencilla de tener el proyecto actualizado. Esta instalación se verá en el siguiente punto.

### **6.3. Repositorio de SWADroid en GitHub.**

Git es un software de control de versiones desarrollado y mantenido por Linus Torvalds, creador del Kernel de Linux. Gracias a este software muchos desarrolladores ponen el código de sus aplicaciones a disposición de los demás con la idea de poder mejorarlo y solucionar posibles fallos así como añadir funciones. Es uno de los sistemas más utilizados por el software libre.

Una de sus principales características es el avanzado control de versiones que permite volver a una versión anterior del código sin ninguna dificultad ya que el propio sistema es quien guarda una copia de cada archivo antes de ser modificado.

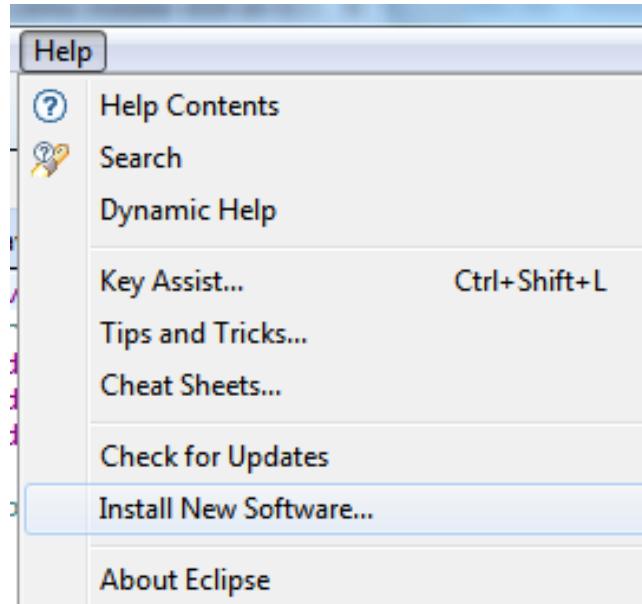
GitHub es donde se almacenan los proyectos de los usuarios utilizando el gestor de versiones. Los desarrolladores avanzados son capaces de utilizar esta herramienta desde un terminal, pero para un usuario medio, esto puede ser una verdadera pesadilla debido a la gran cantidad de comandos que existen para manejar esta plataforma. La forma más sencilla de poder programar en Git sin quebraderos de cabeza es hacerlo desde un IDE que trabaje de forma sincronizada con los repositorios seleccionados. Uno de los más utilizados es Eclipse con el plugin de egit.

SWADroid mantiene su código fuente en un repositorio público alojado en <https://github.com/AMAB/SWADroid>

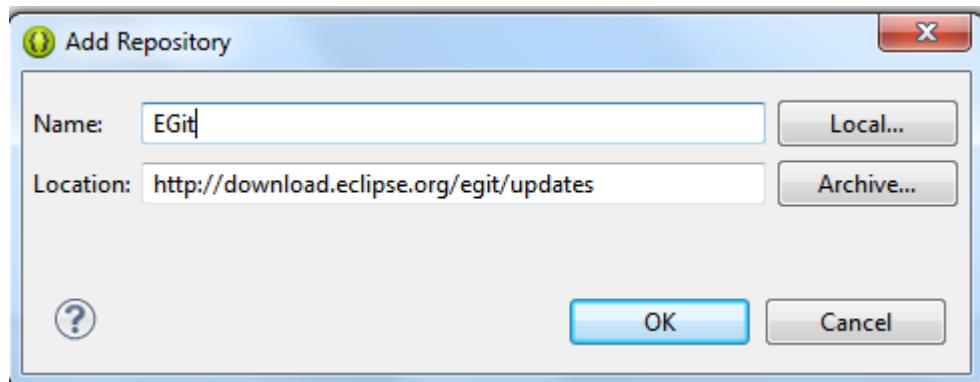
En los siguientes puntos se detallan los pasos necesarios para llevar a cabo el desarrollo de SWADroid manteniendo el código actualizado.

### 6.3.1. Instalar el plugin EGit.

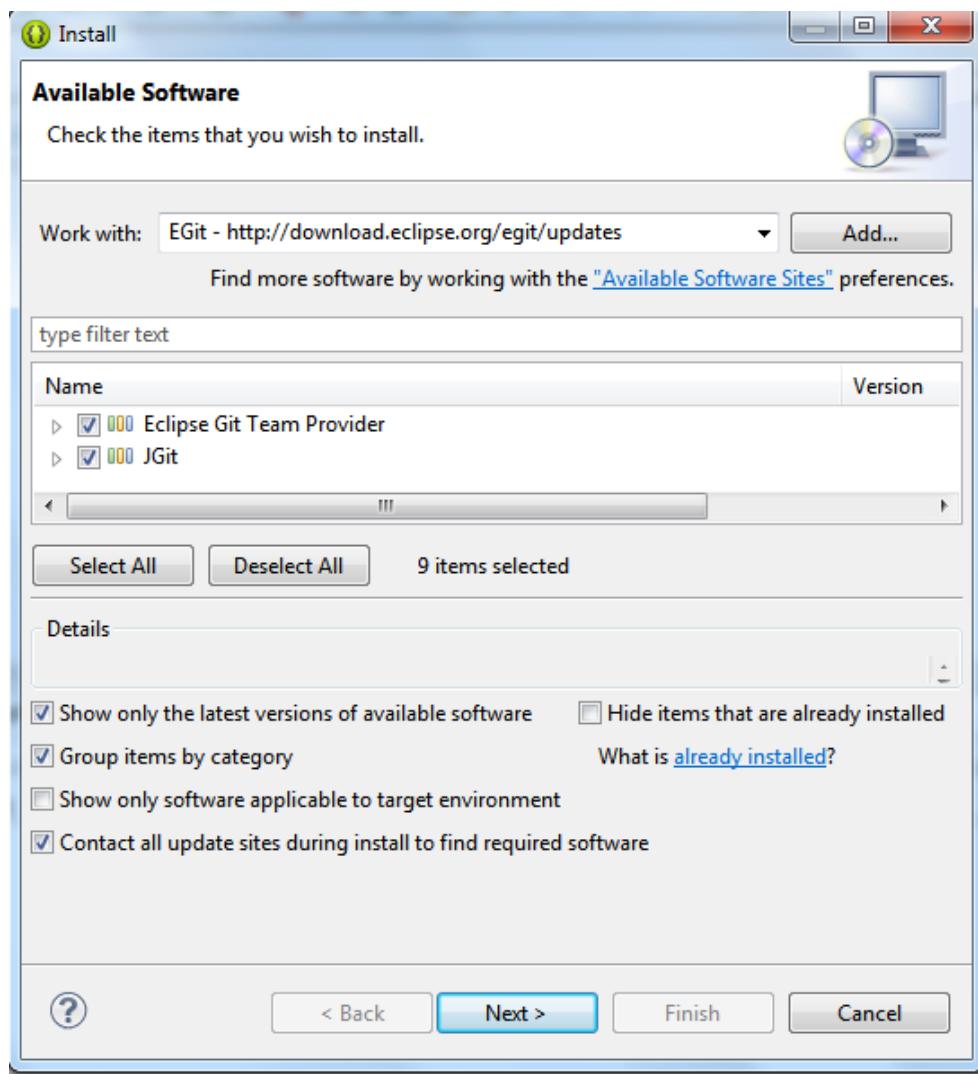
En la pestaña Help seleccionamos “Install New Software...”.



En la ventana que nos aparece pulsamos “Add...” y rellenamos los datos como en la siguiente imagen:



A continuación marcamos todos los archivos que aparecen para instalar y continuamos:



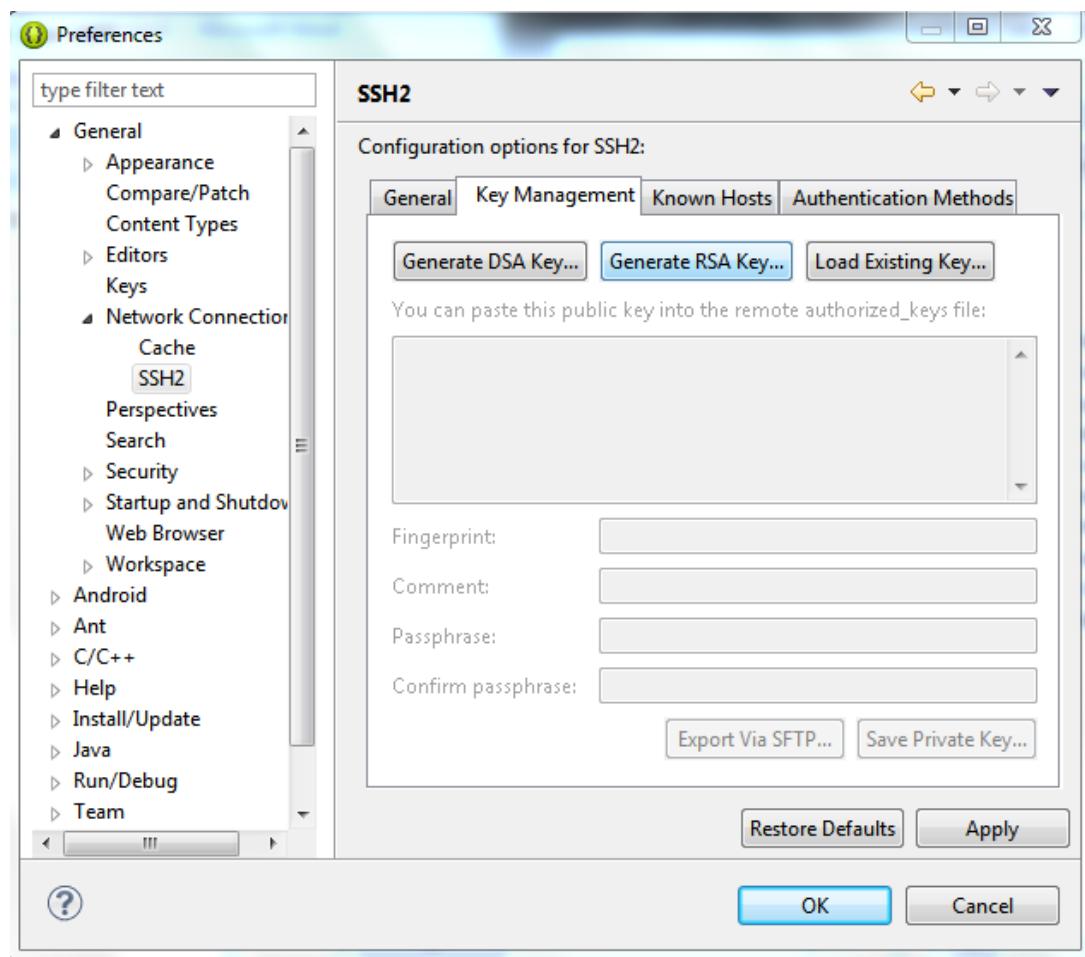
Continuamos, aceptamos la licencia y comenzará la instalación. Una vez finalizada ya tenemos incorporado el plugin EGit en Eclipse.

### 6.3.2. Generar claves SSH.

Una vez instalado debemos crear una “key ssh” que nos permitirá comunicarnos con el servidor github y subir nuestros proyectos de manera segura.

La generación de la clave SSH se puede realizar de dos maneras:

- Desde el propio Eclipse: vamos a **Window > Preferences** y dentro de este vamos a **General > Network Connections > SSH2**. En la pestaña **Key Management** creamos una nueva key rsa. Al crearse nos dará una key para utilizar en servidores con los que nos comunicaremos, en nuestro caso con github.



Como se puede ver en la anterior imagen, es posible, además de crear una nueva clave, cargar una que ya tengamos creada.

- Desde la consola de comandos:

*1.- Comprobar si ya existe una clave SSH en el ordenador*

```
$ cd ~/.ssh
```

*Si el directorio .ssh no existe, ir al paso 3., en otro caso, ir al paso 2.*

*2.- Realizar una copia de seguridad y eliminar las claves SSH existentes*

```
$ ls
    config id_rsa id_rsa.pub known_hosts

$ mkdir key_backup
$ cp id_rsa* key_backup
$ rm id_rsa*
```

*3.- Generar una nueva clave SSH*

```
ssh-keygen -t rsa -C "your_email@youremail.com"
```

Generating public/private rsa key pair.

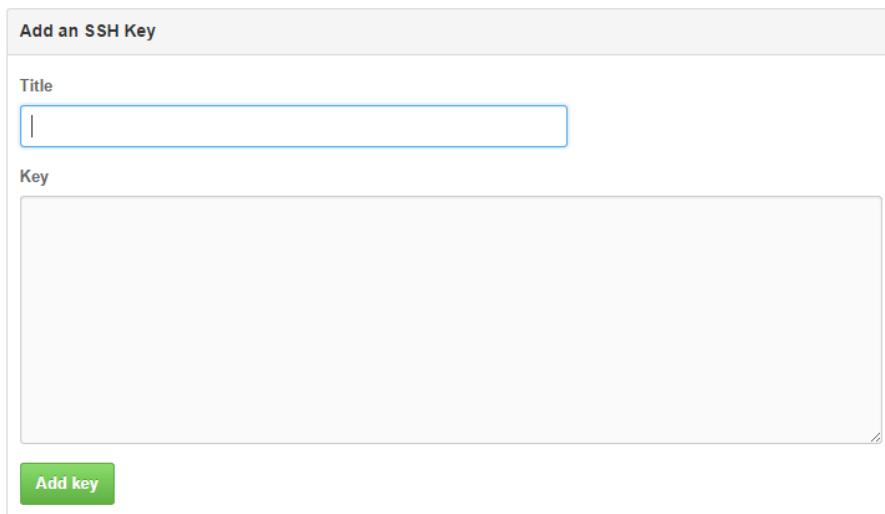
Enter file in which to save the key (/Users/your\_user\_directory/.ssh/id\_rsa):

<press enter>

Enter passphrase (empty for no passphrase):<enter a passphrase>

Enter same passphrase again:<enter passphrase again>

Una vez hecho esto, tenemos que añadir la clave creada a nuestra cuenta de GitHub. Para ello, vamos a nuestra cuenta, seleccionamos el icono de **Settings > SSH keys > Add SSH key**.



Deberemos darle un nombre y pegar el contenido del archivo de clave que hemos generado. Una vez hecho esto, pulsamos el botón “Add key” y ya tendremos nuestra clave añadida al repositorio.

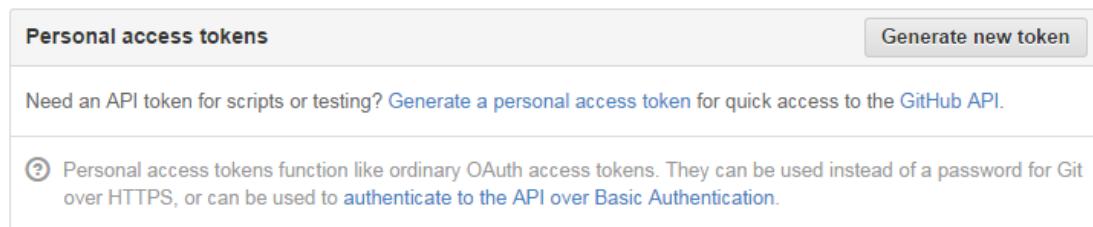
### 6.3.3. Configurar la información de desarrollador

Git lleva a cabo un seguimiento sobre quién realiza cada commit comprobando el nombre de usuario y su email. Además, GitHub utiliza esta información para asociar los commits con la cuenta de GitHub. Para configurar este aspecto es necesario ejecutar el siguiente código reemplazando el nombre de usuario y email por sus respectivos valores. El nombre debe ser el nombre real del desarrollador, no el nombre de la cuenta GitHub.

En la consola de Git escribimos:

```
$ git config --global user.name "Firstname Lastname"  
$ git config --global user.email "your_email@youremail.com"
```

Algunas herramientas se conectan a GitHub sin utilizar SSH. Para poder utilizarlas correctamente es necesario configurar un API Token. En la página web de GitHub pulsar **Settings > Account Settings** y podremos definir el nombre de usuario. Para el token, pulsar **Settings > Applications > Generate new token**



En la consola de Git se puede hacer de la siguiente manera, utilizando el nombre de usuario GitHub y el token apropiados:

```
$ git config --global github.user username  
$ git config --global github.token 0123456789yourf0123456789token
```

**NOTA:** Si se cambia la contraseña de GitHub se creará un nuevo token y será necesario repetir estos pasos.

#### 6.3.4. Importar el proyecto SWADroid y subir nuestros cambios

Durante el desarrollo del proyecto se han usado dos metodologías de trabajo con Git.

Durante los primeros meses y durante la mayor parte del desarrollo se ha usado la metodología que hace uso de “*push y pull*” para descargarnos y subir los cambios.

En la forja de SWADroid hay dos ramas diferenciadas:

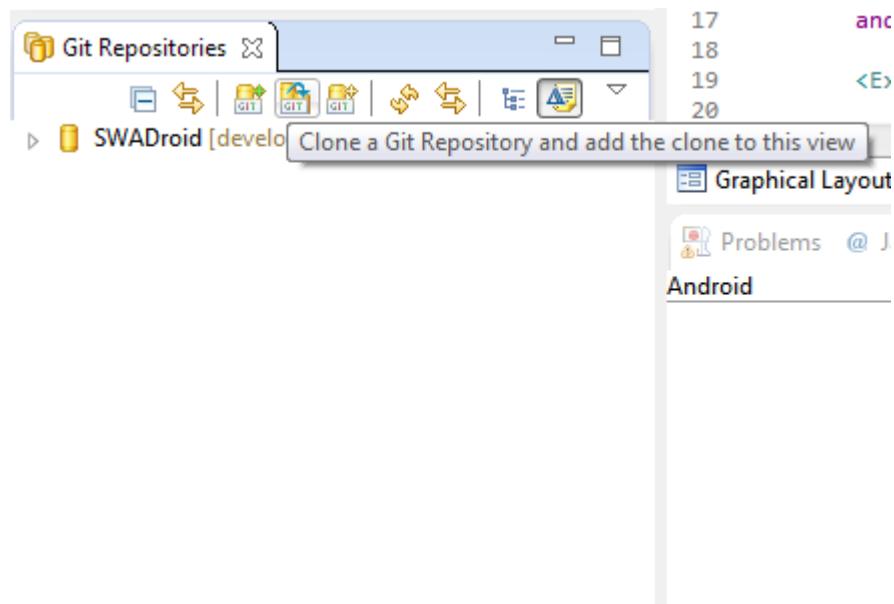
- **master**: Esta rama contiene el código de las versiones publicadas en Google Play, por lo que sólo contiene código estable. Únicamente se sube código a esta rama cuando se va a publicar una nueva versión de SWADroid en Google Play.
- **develop**: Esta rama contiene el código de las versiones en desarrollo. Los desarrolladores del proyecto trabajamos en esta rama. Aunque esta rama está para que los desarrolladores compartan su código y en ocasiones puede ocurrir que el código ni siquiera compile, es una buena práctica que los cambios que subamos a esta rama permitan compilar el código y no interfieran con el trabajo de los demás. En caso de que necesitemos compartir nuestros cambios y estos no compilen o provoquen un funcionamiento anómalo en la ejecución normal de la aplicación los subiremos comentados para desactivarlos.

El proceso para empezar a trabajar con esta metodología es el siguiente:

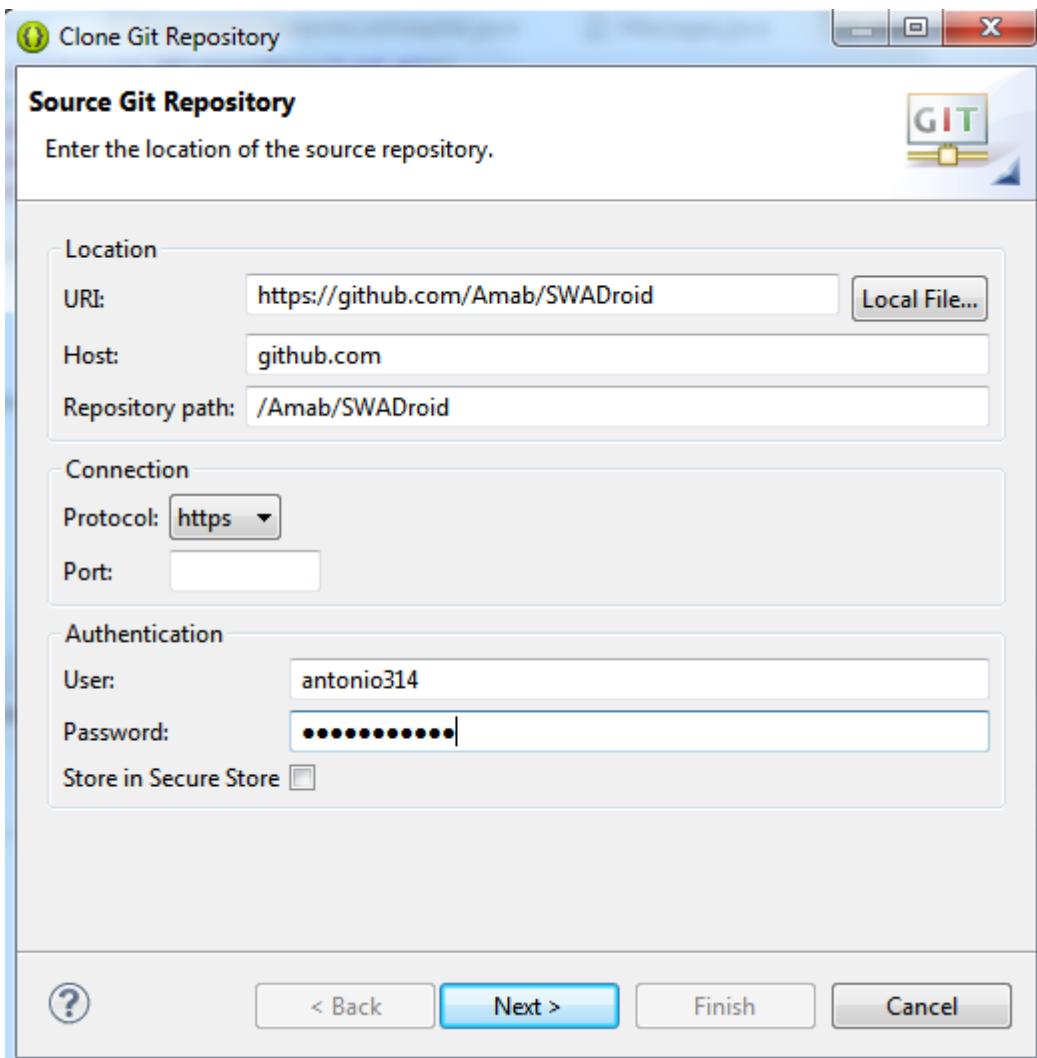
- Descargar el código de la rama develop desde la consola:

```
git clone https://github.com/Amab/SWADroid.git -b develop
```

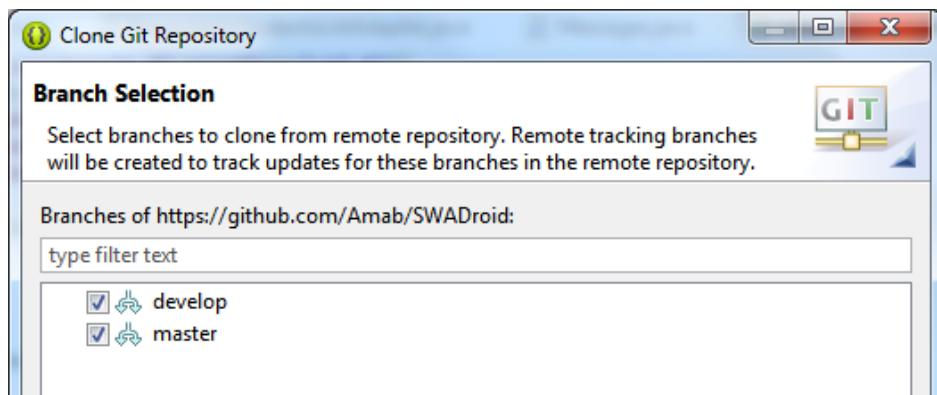
También podemos hacerlo mediante el plugin EGit de la siguiente manera:



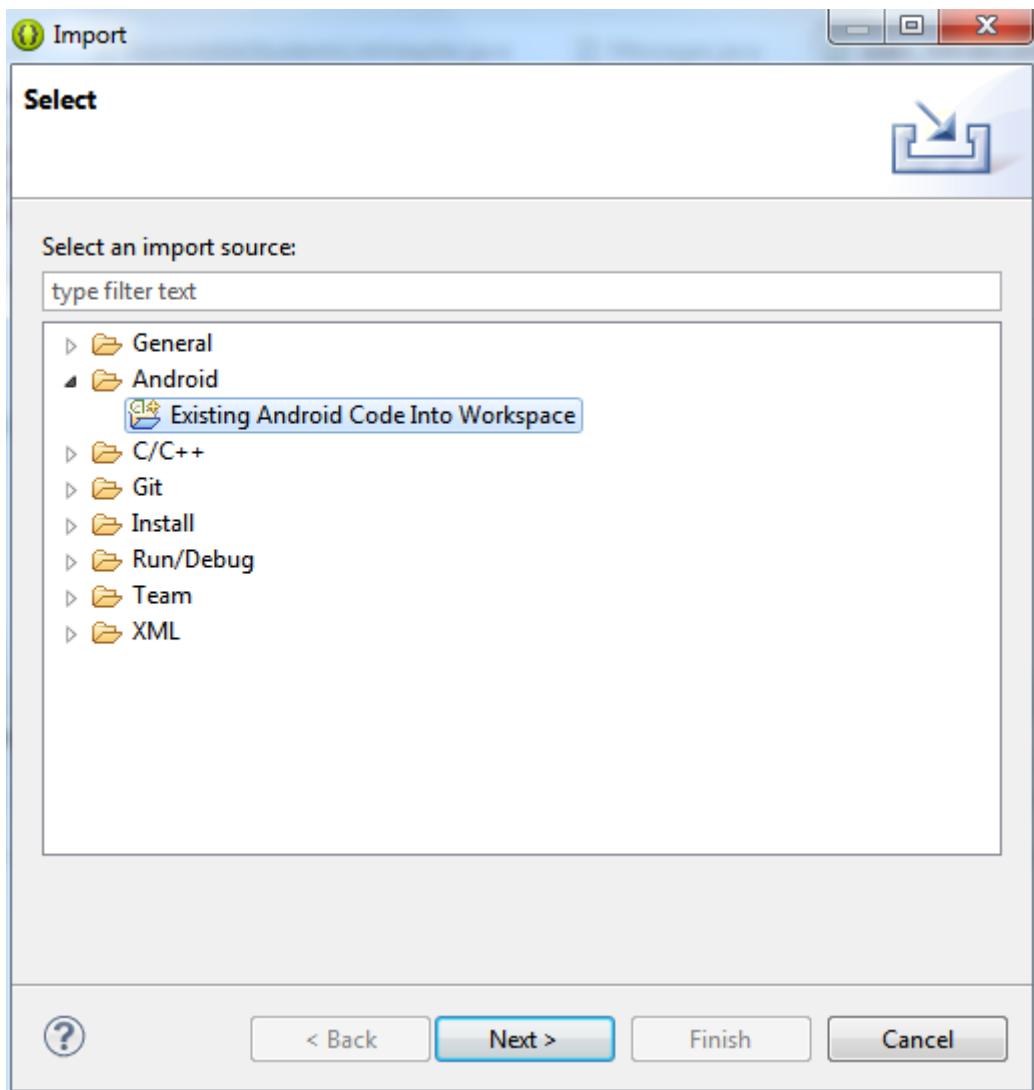
A continuación rellenamos los datos que nos aparecerán:



Seguidamente nos aparecerá una ventana en la que deberemos marcar las ramas del proyecto que queremos descargar (en el caso de que tenga más de una como es el caso de SWADroid que, como ya se ha dicho, dispone de una rama master y una rama develop).



Una vez descargado el código y configurado Git como se ha indicado en los pasos anteriores, deberemos abrirlo en Eclipse (File>Import>Existing proyecto into workspace) y ya podremos comenzar a trabajar y a interactuar con la forja.



Al clicar en Next, nos aparecerá una ventana en la que tendremos que seleccionar “Browse...” y se nos abrirá un Explorador de archivos por el que tendremos que navegar hasta donde hemos descargado el proyecto. Si se hace bien el resultado será que el proyecto aparece en el explorador de paquetes.



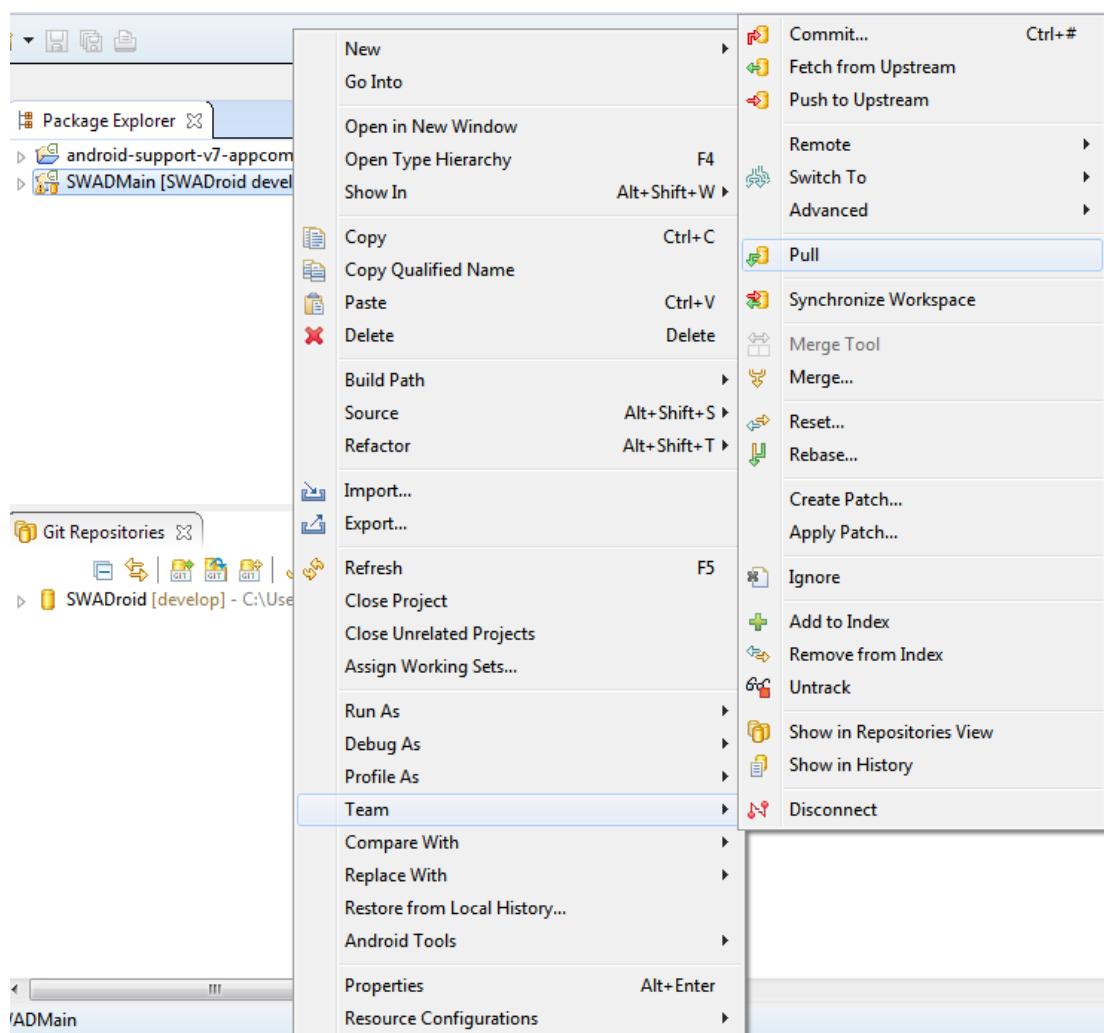
Para trabajar en conjunto con el resto del equipo de desarrollo, se usó, como ya se ha dicho, durante la mayor parte del proyecto, la metodología “*pull & push*”:

- Pull: se usa para actualizar el código del proyecto, es decir, para descargar los cambios que los demás integrantes del proyecto han subido a la forja.

Se puede usar por consola mediante el siguiente comando:

```
git pull
```

O mediante el plugin EGit haciendo clic con el botón derecho sobre el proyecto y:



Si el proyecto está actualizado, nos dirá “Already up-to-date” si no, nos descargará las modificaciones y las incluirá en nuestra copia local del proyecto.

- Push: se usa para subir nuestras modificaciones a la forja.

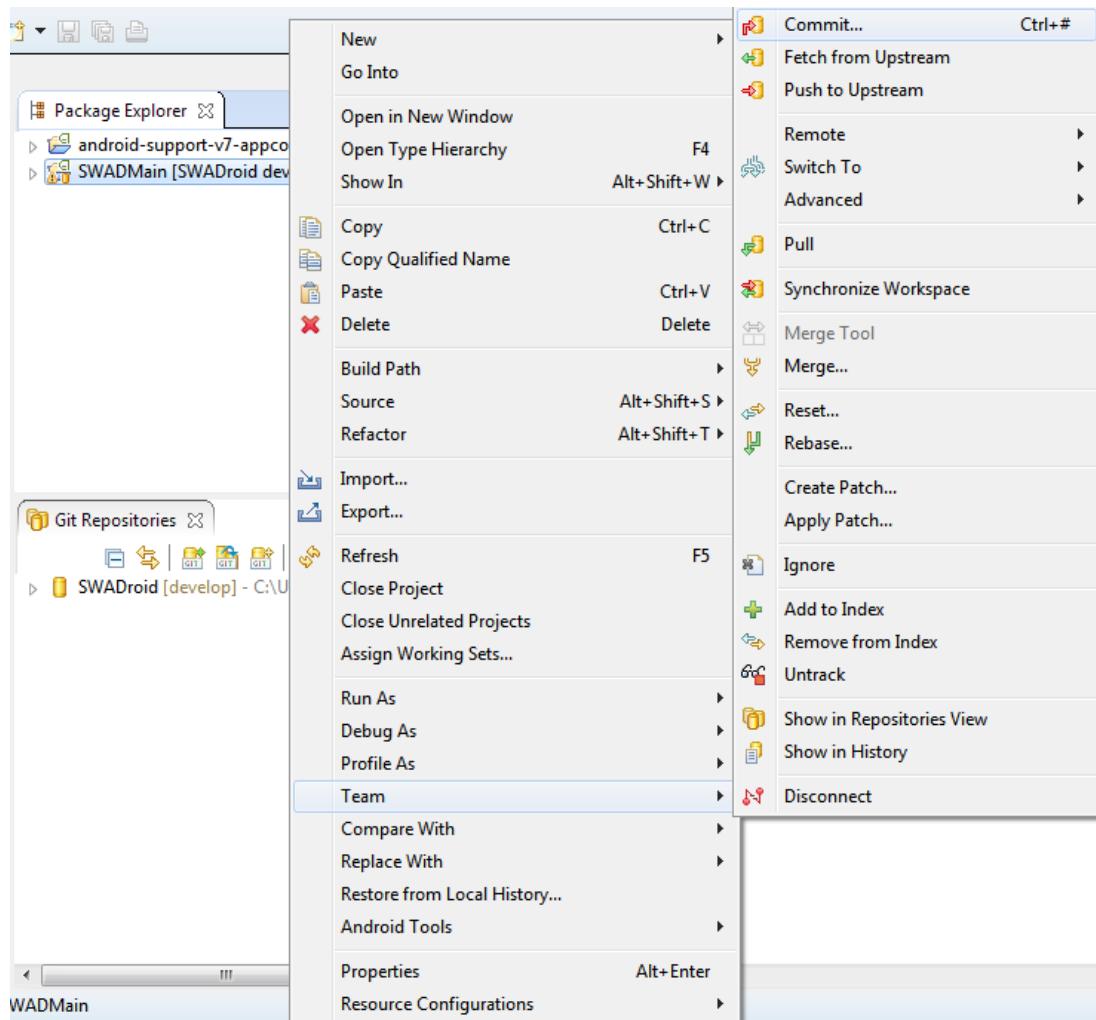
Se puede usar por consola mediante el siguiente comando:

```
git push
```

Generalmente, esto dará error y nos dirá que antes de poder hacer un “push” debemos hacer un “commit”. Hacer un commit es guardar nuestros cambios localmente antes de poder subirlos a la forja. Esto se realiza mediante los siguientes comandos:

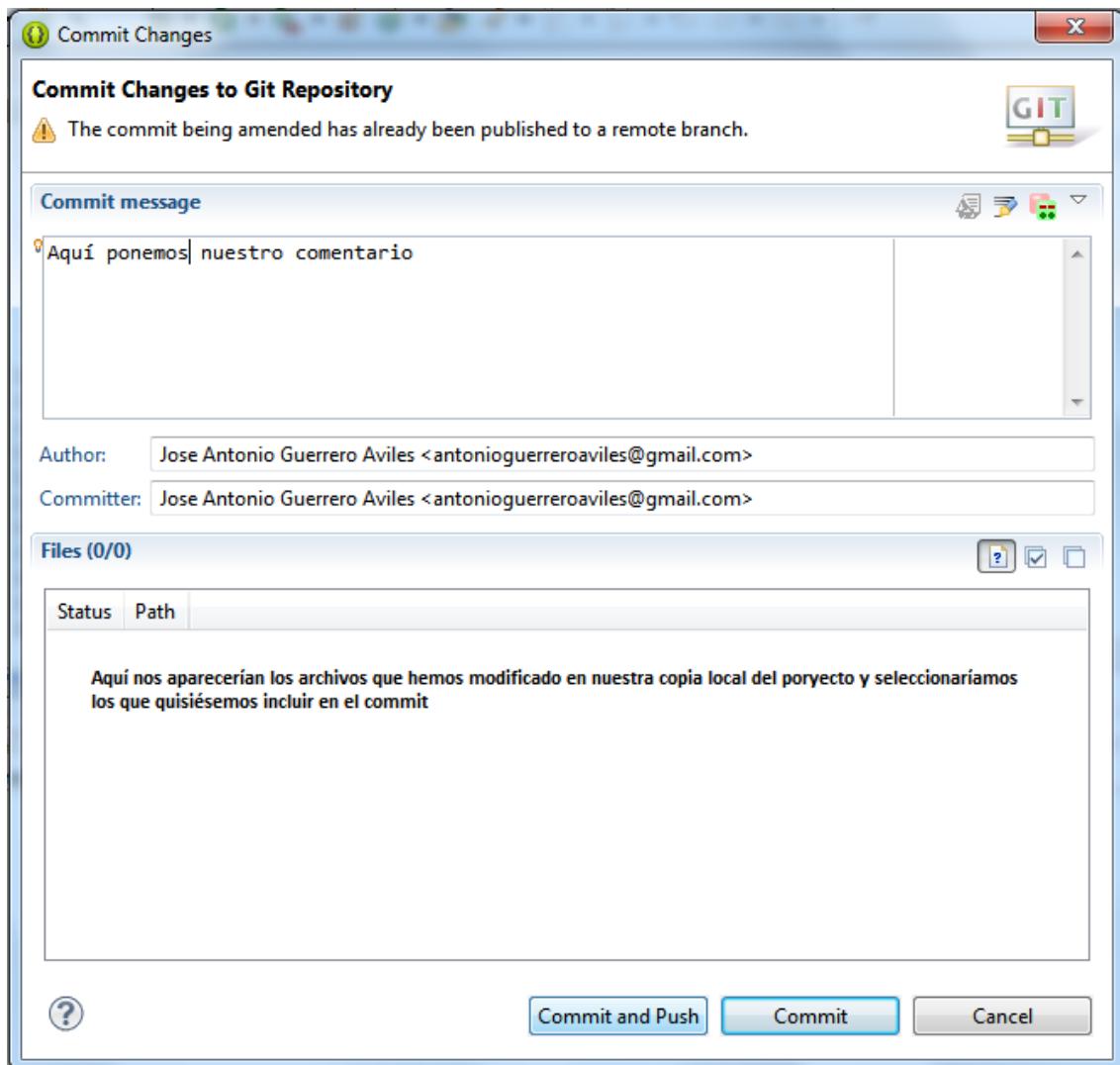
```
git add * (añade todos los archivos al futuro commit)
git commit -m "el mensaje que se incluirá en el commit" (crea el commit)
git push (sube los cambios incluidos en el commit a la forja)
```

O mediante el plugin EGit de Eclipse:



El primer paso para subir nuestros cambios a la forja es guardarlos localmente como se ha dicho anteriormente.

Una vez seleccionada esta opción nos aparecerá la ventana en la que deberemos completar los datos del commit y el push:



Si seleccionamos “Commit”, los cambios se guardarán de forma local (se trackearán) y si seleccionamos “Commit and Push” aparte de guardarse localmente, se subirán a la forja.

Durante aproximadamente el último mes de proyecto, se cambió la metodología de trabajo y se pasó de los “push & pull” a “fork & pull request”

Los objetivos de cambiar de metodología son los siguientes:

- Mantener el repositorio oficial limpio de cosas a medias y en un estado de “liberación inmediata”, es decir, que podamos liberar una nueva versión a producción en cualquier momento.
- No pisarnos los unos a los otros tocando directamente cosas que pueden afectar a los demás y utilizar un mecanismo de revisión antes de integrar los cambios en el repositorio oficial.

## Cambios

Se comienza a utilizar la metodología *fork & pull* de GitHub a base de forks. A partir de ahora ya no se tendrá acceso directo al repositorio oficial del proyecto, es decir, no se podrán subir cambios directamente (el push fallará por falta de permisos). En lugar de eso os crearéis un fork para cada uno y trabajareis en él.

La metodología de desarrollo es ahora la siguiente:

1. **Actualizar el fork personal** con los últimos cambios del repositorio oficial.
2. **Hacer el cambio** en el fork personal.

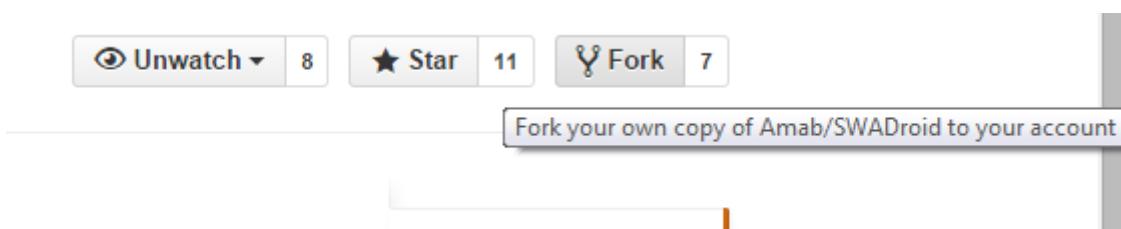
Cuando el cambio esté terminado, **enviar un pull request** contra la rama develop del repositorio oficial para su revisión y, en su caso, integración en el código del proyecto.

- o Si el pull request es aceptado, el cambio se da por completado y se integra en el código del proyecto.
- o Si el pull request es rechazado, se deben corregir los problemas que han provocado su rechazo y enviar un nuevo pull request.

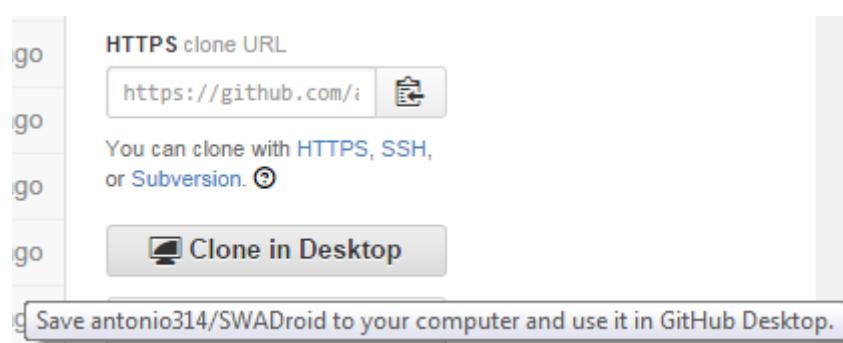
Para el envío de los pull requests seguimos algunas normas:

- Los pull requests deben ir **contra la rama develop** del repositorio oficial. Todos los pull requests que vayan contra la rama master serán **rechazados**.
- Sólo se enviarán pull requests de **cambios ya terminados y probados**. Todos los pull requests de cambios a medio hacer serán **rechazados**.

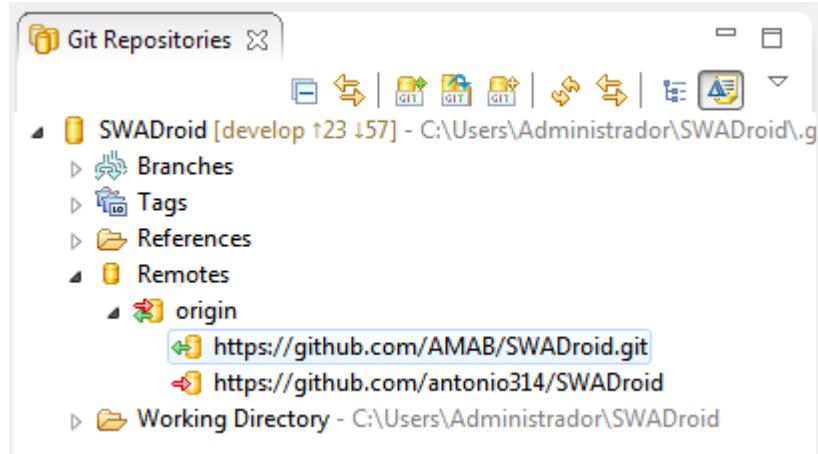
Para hacer un fork nos dirigimos a la forja del proyecto y hacemos clic en Fork como muestra la siguiente imagen:



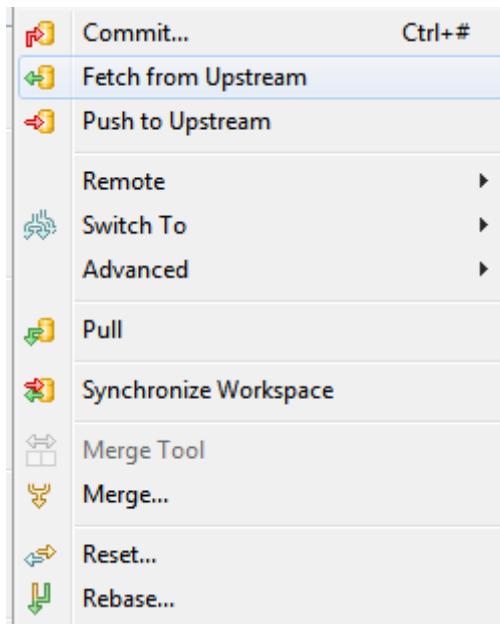
Ahora ya tenemos el proyecto “copiado” en nuestra cuenta de GitHub y solo tendremos que descargarlo usando alguna de las opciones disponibles:



Para mantener nuestro fork actualizado, debemos hacer que “el pull” apunte a la forja del proyecto, no a nuestra forja (hacer que el *upstream* apunte a <https://github.com/Amab/SWADroid>):



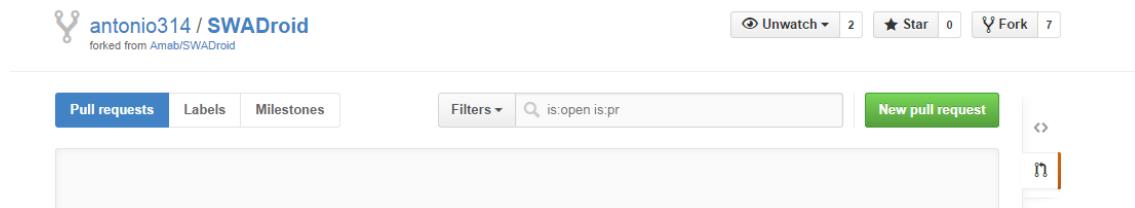
Ahora hacemos el pull que hacíamos antes, desde lo que hemos definido como upstream. Para ello, hacemos clic con el botón derecho sobre nuestro proyecto, luego seleccionamos Team y a continuación seleccionamos el pull, que en este caso se llama “Fetch from upstream”:



A continuación, se descargarán los cambios que haya en la forja (si es que los hay) y deberemos hacer “Merge...” para incluir dichos cambios en nuestro fork. Una vez hecho esto ya podemos seguir trabajando en el proyecto.

Para subir los cambios a la forja debemos realizar dos pasos:

1. Hacer un commit de los cambios y subirlos a nuestro repositorio (que es donde está el fork que hemos realizado) de alguna de las maneras que hemos visto anteriormente.
2. Enviar una solicitud “Pull request” sobre la forja del proyecto. Para ello nos dirigimos a nuestro fork del proyecto dentro de la página de Github y elegimos la opción Pull Request como se ve en la siguiente imagen:



Seleccionamos “New Pull Request” y nos aparecerá una ventana en la que deberemos elegir “Create Pull Request” y completar los datos:



Ahora para enviarlo, solamente debemos clicar en “Create Pull request”.

Si el Pull Request es aceptado, los cambios estarán subidos a la forja, si se rechaza, deberemos corregir los errores y volver a enviar la solicitud.

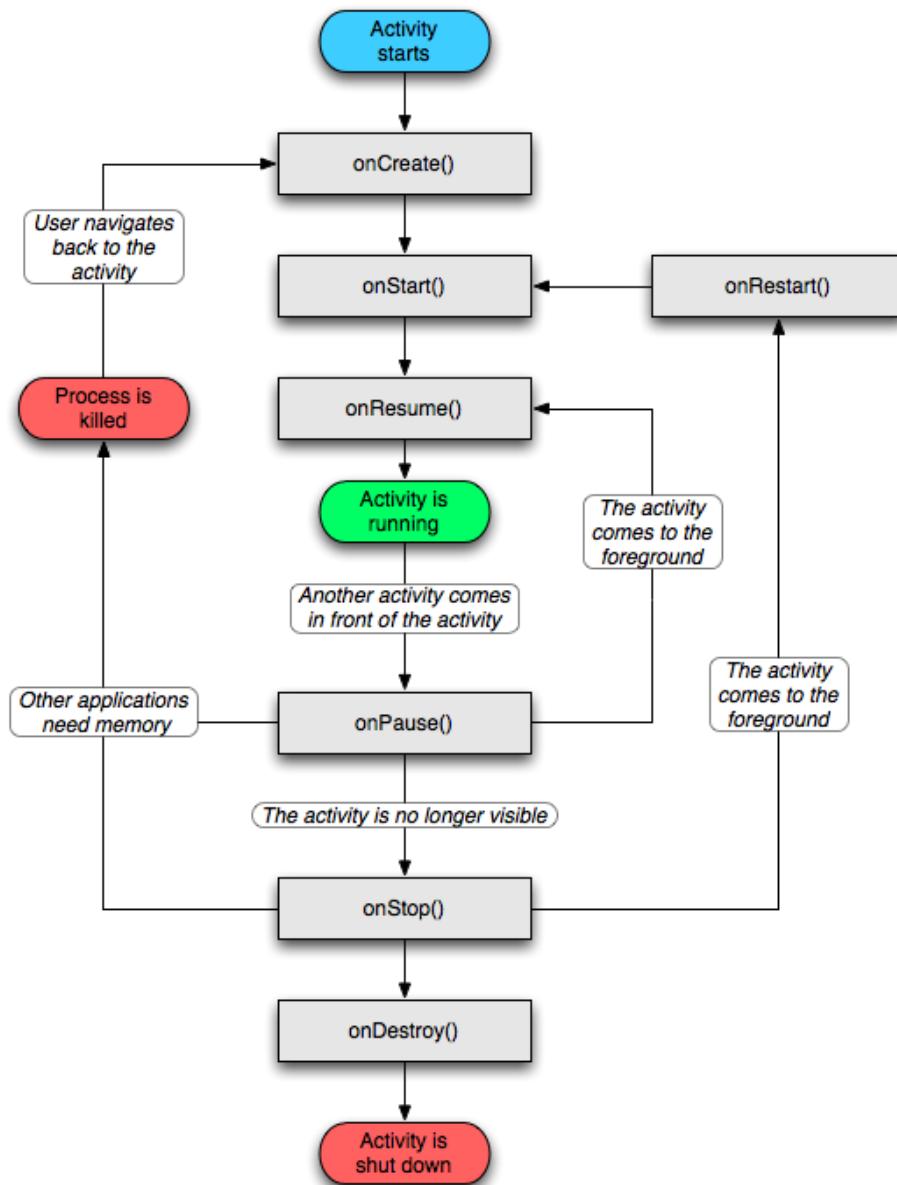
## 6.4 API de Android.

En este punto se comentan, de forma resumida, las clases más importantes pertenecientes al API de Android que se han usado. Estas clases se usan como cualquier otra clase Java. Se han nombrado como las más importantes, entre otras cosas, porque con ellas (y sus clases descendientes) se podrían construir aplicaciones bastante complejas sin necesidad de usar ninguna clase más.

### 6.4.1 Activity

Es la principal clase de Android, y cada pantalla que se muestra en la aplicación es una clase que hereda de Activity. Se podría decir que una actividad equivale a una pantalla.

A continuación se muestra el ciclo de vida de una actividad:



En el diagrama se observan los métodos que se pueden sobrescribir si se desea realizar una acción en concreto, y cuándo se ejecutan. No es necesario ni obligatorio implementar ninguno de ellos; el único que debemos implementar siempre (al menos si se desea que la actividad muestre algo) es el método `onCreate()` donde se define todo lo que se necesite realizar antes de que la pantalla se muestre.

Vamos a ver, de forma resumida, qué hace cada uno de esos métodos:

- **onCreate():** Se dispara cuando la *Activity* es llamada por primera vez. Aquí es donde debemos crear la inicialización normal de la aplicación, crear vistas, hacer los bind de los datos, etc. Este método te da acceso al estado de la aplicación cuando se cerró.
- **onRestart():** Se ejecuta cuando tu *Activity* ha sido parada, y quieres volver a utilizarla.
- **onStart():** Se ejecuta cuando la *Activity* se está mostrando apenas en la pantalla del dispositivo del usuario.
- **onResume():** Se ejecuta una vez que la *Activity* ha terminado de cargarse en el dispositivo y el usuario empieza a interactuar con la aplicación.
- **onPause():** Se ejecuta cuando el sistema arranca una nueva *Activity* que necesitará los recursos del sistema centrados en ella. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva *Activity*.
- **onStop():** Se ejecuta cuando la *Activity* ya no es visible para el usuario porque otra *Activity* ha pasado a primer plano.
- **onDestroy():** Esta es la llamada final de la *Activity*, después de ésta, es totalmente destruida. Esto ocurre debido a los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Si quisieramos volver a ejecutar la *Activity* se arrancaría un nuevo ciclo de vida.

Se ha hablado mucho de *Activity* pero, ¿qué es realmente?

Una actividad es una pantalla en la que pueden aparecer elementos visuales en ella. Además, cada uno de ellos debe ser un objeto que herede de la clase *View*. Estos elementos visuales pueden ser definidos mediante código o a través de ficheros XML (recomendado).

La segunda forma se lleva a cabo mediante el siguiente código:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    eventCode = getIntent().getLongExtra("eventCode", 0);  
    setContentView(R.layout.messages_screen);  
    setTitle(R.string.messagesModuleLabel);  
    getSupportActionBar().setIcon(R.drawable.msg);  
  
    if (savedInstanceState != null)  
        writeData();  
  
    setMETHOD_NAME("sendMessage");  
}
```

En este caso, este es el comienzo del *onCreate()* en el que se mostrará la vista principal de la pantalla para enviar los mensajes. Antes de llevar a cabo una acción con un elemento visual, debe llamarse al método de la clase superior. Como en este caso se desea que la apariencia visual se lea desde un fichero XML, se llama al método que toma un contenido visual y, a través de clase *R.java*, se accede al XML que se desea mostrar.

La clase que se desee mostrar también puede heredar de otra clase que herede de *Activity*, como, por ejemplo *TabActivity*, que construirá la pantalla con un panel con tantas pestañas como se definan, un *ListActivity*, que muestra elementos en forma de lista y sus correspondientes métodos para tratarla, o *MapActivity*, proporcionada por un API de Google. También se pueden reutilizar las Activities ya creadas anteriormente en SWADroid.

#### 6.4.2 Intent.

La mejor forma de definir la clase Intent es como un nexo de unión. Su uso más frecuente es llamar a una Activity desde otra, o para realizar una acción en una actividad.

Esta clase permite que el programador se despreocupe del flujo de datos de su aplicación, ya que la clase Intent se encarga de ello y, puesto que todos los Android disponen de un botón volver, no es necesario ningún control de flujo.

La clase Intent también se encarga de controlar a qué actividad se debe regresar cuando se pulse el botón del dispositivo para volver atrás. Si es necesario, se podría realizar un control de flujo de datos a través de todos los métodos que definen el ciclo de vida.

A modo de ejemplo, vamos a ver como se hace uso de la clase Intent para añadir el listado de usuarios desde la pantalla de mensajes al campo destinatarios:

```
Intent callUsersList = new Intent (getBaseContext(), UsersList.class);
startActivityForResult(callUsersList, 0);
```

En este ejemplo, se abre la ventana correspondiente a la función UsersList. Con startActivityForResult() la actividad llamante a la espera de que la nueva actividad finalice y se le asigne un identificador. En la clase UsersList se devuelven los usuarios marcados:

```
//Accept receivers list marked at users list
if (!childItem.isEmpty()){
    for (int i = 0; i < childItem.size(); i ++){
        int tam = childItem.get(i).size();
        for(int j = 0; j < tam; j++){
            User usu = childItem.get(i).get(j);
            if (usu.isSelected()){
                String us = usu.getUserNickname().toString();
                rcvs_Aux = rcvs_Aux + "@" + us + ",";
            }
        }
    }
    //If receivers, remove the last comma
    if(!rcvs_Aux.isEmpty()) {
        rcvs = rcvs_Aux.substring(0, rcvs_Aux.length()-1);
    }
}
Intent resultData = new Intent();
resultData.putExtra("ListaRcvs", rcvs);
setResult(Activity.RESULT_OK, resultData);
finish();
```

Cuando se deseé cerrar una clase Activity se puede llamar al método finish(). Antes de hacerlo se puede indicar el resultado de la acción de la actividad. Si la actividad ha realizado su tarea correctamente se puede llamar dentro de ella al método setResult(RESULT\_OK) para informar a la actividad llamante; en otro caso se puede llamar al método setResult(RESULT\_CANCELED).

RESULT\_OK/CANCELED son constantes que pertenece a la clase Activity.

Intent podría considerarse un nexo de unión y un modo de realizar acciones en el dispositivo.

No sirve exclusivamente para pasar de una actividad a otra, sino que, como se muestra en el siguiente ejemplo, también sirve para lanzar acciones que el dispositivo móvil lleva integradas, como por ejemplo realizar una llamada. En este caso, al crear el Intent es necesario indicarle el número al que se desea llamar:

```
Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:"+ numeTlf));  
startActivity(intent);
```

### 6.4.3 View.

Todo objeto que se desee mostrar por pantalla debe heredar de la clase View. Se pueden definir clases propias que hereden de View, o usar las que proporciona el API de Android. Los objetos View se pueden organizar y ordenar dentro de objetos de clase Layout. Con un Layout, por ejemplo, se puede definir que los componentes se agreguen de manera horizontal o vertical en una línea. También es posible definirlos en un formato de tabla controlando las posiciones en columnas y filas.

Entre los layouts proporcionados por Android los siguientes son los más básicos:

- LinearLayout: este Layout ordena a sus elementos en una línea en sentido horizontal o vertical, agregándolos uno por uno en el orden en que se van definiendo.
- FrameLayout: es el tipo más simple de Layout que permite agregar un solo elemento en un espacio en blanco definido.
- TableLayout: este Layout agrega sus elementos en columnas y filas. Las filas deben ser insertadas una por una con un RowLayout.
- RelativeLayout: permite agregar elementos respecto a uno previo. Esto significa que el posicionamiento se administra según la posición del elemento anterior.

Las interfaces se definen de forma jerárquica, por lo que es posible usar tantos elementos como se deseé, combinándolos o introduciendo unos dentro de otros. Otra forma de organizar los objetos de clase View es utilizar ViewGroups, que también pueden contener Layouts si se desea.

Estos son los más comunes:

- Gallery: utilizado para desplegar listados de imágenes en un componente con Scroll.
- GridView: despliega una tabla con Scroll de m columnas y n filas.
- ListView: despliega una lista con Scroll de una columna.
- ScrollView: una columna vertical de elementos con Scroll. Dentro de la jerarquía, el ScrollView sólo puede contener un elemento, si se desea que el scroll se realice sobre varios Views estos deberán estar dentro de un ViewGroup o Layout, que será incluido dentro del ScrollView.
- ExpandableListView: Permite realizar la misma función que un ListView, pero con la posibilidad de tener más de una categoría, además de poder mostrar y ocultar (expandir y contraer) el contenido de dichas categorías.

Los objetos más utilizados de clase View son:

- Button: elemento que muestra un botón en la pantalla con el objetivo de que el usuario realice alguna interacción con él. Se le puede colocar una imagen de fondo, pero para construir un botón que sea una imagen existe también una clase similar llamada ImageButton.
- CheckBox: caja de texto seleccionable que permite evaluar dos estados del elemento: seleccionado y no seleccionado.
- EditText: elemento que permite editar texto dentro del mismo.
- TextView: etiqueta de texto no editable por el usuario.
- RadioButton: elemento seleccionable parecido al CheckBox, pero con la diferencia de que una vez que se ha seleccionado no regresa a su estado anterior si este está definido dentro de un RadioGroup.
- Spinner: listado de elementos oculto que solo aparece cuando se pulsa sobre él. Cuando está oculto sólo se puede ver el elemento seleccionado de todo el listado.
- WebView: como se explicó en otros puntos, permite cargar contenido Web dentro de nuestra aplicación sin necesidad de abrir el navegador Web.

Con estas tres clases, y sus correspondientes clases descendientes, como ya se ha dicho con anterioridad, se podrían construir aplicaciones bastante complejas.

El proceso se pueden resumir de la siguiente manera: una Activity es una ventana que se muestra por pantalla, todo lo que se desee incluir en la pantalla tiene que ser descendiente de View o agrupaciones de estos objetos como Layouts y ViewGroups. Por último, los Intent permiten pasar de un Activity a otro, o realizar acciones definidas por el teléfono sobre él.

También existe la clase Service, de funciones similares a un Activity pero ejecutándose en segundo plano. Es decir, es una actividad en background que se puede ejecutar de forma indefinida y que no mostrará nada por pantalla durante su ejecución.

## 6.5. Temporización del Proyecto.

El tiempo invertido en el desarrollo del proyecto ha sido llevado a cabo mediante la plataforma clockingit (<http://www.clockingit.com/>).

Clockingit es una aplicación web libre que nos permite crear tareas, asignarlas a personas y añadir el tiempo que se ha ido invirtiendo en ellas.

En el proyecto que se ha creado en dicha aplicación están registrados como participantes José Antonio Guerrero Avilés (encargado de crear las nuevas funcionalidades para SWADroid) y Antonio Cañas Vargas (director del proyecto y encargado de implementar y modificar las funciones del Web Service).

**PFC SWADroid**

Miembros [Antonio Guerrero](#), [Antonio Cañas](#)  
Creado 15:33 09/10/2013  
Tareas 5 Abrir / 21 Total  
Trabajo -4sem. 1d 7h 27min. [0min. Estimado]  
pendiente 2sem. 3h 30min. [0min. Estimado]  
Trabajo hecho

Proyecto para llevar un control del tiempo que he invertido en las diferentes tareas que he ido realizando a lo largo de todo el Proyecto Fin de Carrera.

Como se observa en la imagen, el tiempo invertido en la realización del proyecto es de **6 semanas, 1 día, 10 horas y 57 minutos**.

La configuración del tiempo del proyecto es la siguiente:

Duración jornada	8h 0min.
Días por semana	5 días

Por lo tanto, el tiempo invertido en horas es de unas **270 horas** (a lo que habría que sumarle la realización de la presentación para exponer el proyecto y las horas que aún quedan para terminar de redactar esta memoria).

Cabe destacar que esas son las horas invertidas trabajando desde el ordenador. ¿Qué quiere decir esto? Como se ha comentado en numerosas ocasiones, esto es un proyecto colaborativo y por ende, es imposible contabilizar las horas invertidas en:

- Tutorías con Antonio Cañas.
- Llamadas telefónicas a miembros del equipo.
- Quedadas con miembros del equipo.
- Cientos de correos electrónicos enviados entre los diferentes miembros.
- Videollamadas y videoconferencias.

Y en general, toda la comunicación y el feedback constante que debe haber en el equipo de desarrollo durante todo el proyecto.

En esta temporización tampoco se incluye la totalidad del tiempo invertido en preparar la el “VIII Concurso Universitario de Software Libre” en el cuál SWADroid participó y fue premiado tal y como se puede apreciar en las imágenes que a continuación aparecen:



# Concurso Universitario De Software Libre

[Portada](#)[Noticias](#)[Proyectos](#)[Prensa](#)[Bases](#)[Colabora](#)[FAQ](#)[Premios Locales](#)[Comité](#)[Fase F](#)

## Proyectos

Hay 80 proyectos con un total de 122 estudiantes inscritos.

### Ampliación de SWADroid

Se pretende ampliar la aplicación SWADroid, un cliente móvil para acceder a algunas funcionalidades de la plataforma SWAD en dispositivos móviles Android.

**Jose Antonio Guerrero Avilés**

Universidad: Universidad de Granada Ingeniería informática

Granada

**Alejandro Alcalde Barros**

Universidad: Universidad de Granada Ingeniería informática

Granada



En puntos anteriores se mencionaba que el módulo de mensajes había sido especialmente complicado. Ya se han explicado todos los problemas que han ido surgiendo y se ve, claramente, que sí que ha sido complicado, pero para concretizar aún más vamos a ver unas gráficas en las que se muestran los tiempos invertidos en algunas de las tareas realizadas durante el proyecto y posteriormente el invertido en implementar el módulo de mensajes:

The screenshot shows a task entry titled '#5 TRABAJO DIARIO (ABRIR) (1 SEM. 3D 3H 20MIN. / 0MIN.)'. The task was created on 16/10/2013 by Antonio Guerrero and last updated on 02/09/2014. The 'Información' section includes a 'Resumen' field containing 'Trabajo diario', an empty 'Etiquetas' field, and a 'Descripción' field with the following text: 'Tiempo invertido en tutorías, investigación, recopilación de información, etc. (el trabajo del proyecto que no es una actividad específica como puede ser implementar el módulo "tal" o diseñar el módulo "cuál")'.

Esta tarea por ejemplo, se ha llamado “Trabajo diario” y, como se puede ver en su descripción, estaba pensada para añadir ahí todo el trabajo que no es “de campo” es decir, todo lo que no sea una tarea específica. Aquí se han incluido las consultas de libros, las investigaciones, etc. Estaba concebida en un principio, para contabilizar también el tiempo invertido en la comunicación, pero desde el primer momento se ha optado por no contabilizar ese tiempo ya que era constante y exponencial. Conforme avanzaba el proyecto, cada vez había que mantener una mayor comunicación entre los integrantes del equipo de desarrollo y cada vez pasábamos más tiempo discutiendo o comentando cosas.

En la siguiente imagen se pueden ver algunos de los trabajos realizados en esta tarea:

The timeline shows four entries for task #5 Trabajo diario:

- Wednesday, 23 July 2014**: 14:23 - 15:15 | #5 Trabajo diario | 52min. | AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]. Description: Descargada e instalada la nueva versión de Eclipse+ADT.
- Tuesday, 22 July 2014**: 15:16 - 16:56 | #5 Trabajo diario | 1h 40min. | AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]. Description: Actualización del SDK y el plugin ADT de Eclipse. Solución de errores en el entorno de desarrollo.
- Monday, 28 April 2014**: 16:00 - 20:30 | #5 Trabajo diario | 4h 30min. | AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]. Description: Resuelto algunos errores en el nuevo módulo de mensajes. Corregidos errores en los layout del envío de mensajes. Creación de la presentación para la final local del CUSL.
- Wednesday, 09 April 2014**: 17:15 - 18:30 | #5 Trabajo diario | 1h 15min. | AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]. Description: Añadida entrada al blog. Reinicialización del proyecto por errores al cambiar de la rama holo a la develop. Verificación de error al recibir html en las notificaciones de calificaciones.

En las siguientes imágenes vamos a ver el tiempo invertido en implementar el módulo informativo y posteriormente veremos el invertido en realizar el nuevo módulo de mensajes:

**#15 IMPLEMENTAR MODULOS INFORMATIVOS (CERRADA) (1D 5H 50MIN. / 0MIN.)**  
Creado 14:01 10/12/2013 por Antonio Guerrero - Última Actualización 13:28 09/05/2014 por Antonio Guerrero

**Información**

Resumen

Etiquetas

Descripción - Introducción
- Programa de teoría
- Programa de prácticas
- Guía docente
- FAQs
- Enlaces
- Bibliografía
- Sistema de Evaluación

"/>

Se han invertido aproximadamente unas 14 horas en terminar el módulo, a lo que habría que sumarle el tiempo dedicado a realizar pruebas y corrección de errores, el cuál se ha ido contabilizando (para la totalidad del proyecto) en:

**#17 CICLO DE PRUEBAS Y FIX DE ERRORES (ABRIR) (1D 5H 32MIN. / 0MIN.)**  
Creado 11:42 22/01/2014 por Antonio Guerrero - Última Actualización 13:58 02/09/2014 por Antonio Guerrero

**Información**

Resumen

Etiquetas

Descripción

Algunas de las tareas realizadas en la implementación del módulo informativo fueron:

03:10 - 03:45	#45 Implementar modulos informativos 35min.	AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]
Añadido sistema de evaluación dentro de la categoría evaluación		
Wednesday, 19 February 2014	13:07 - 13:52	#45 Implementar modulos informativos 45min.
AntonioSWADroid / PFC SWADroid / [Antonio Guerrero] Finalizado el módulo y añadidas todas las traducciones.		
Tuesday, 18 February 2014	19:45 - 20:30	#45 Implementar modulos informativos 45min.
AntonioSWADroid / PFC SWADroid / [Antonio Guerrero] Añadido el tratamiento de la información recibida y carga en el webview.		
Monday, 13 January 2014	13:40 - 13:52	#45 Implementar modulos informativos 12min.
AntonioSWADroid / PFC SWADroid / [Antonio Guerrero] Modificados algunos errores en el switch que trata las diferentes opciones.		
09:24 - 09:44	#45 Implementar modulos informativos 20min.	AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]
Implementado método requestService()		
09:22 - 10:07	#45 Implementar modulos informativos 45min.	AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]
Modificados los nombres de algunas variables que daban lugar a confusiones.		
Thursday, 12 December 2013	18:40 - 19:28	#45 Implementar modulos informativos 48min.
AntonioSWADroid / PFC SWADroid / [Antonio Guerrero] Añadido método onRequest de los módulos informativos		
Wednesday, 11 December 2013	12:07 - 12:42	#45 Implementar modulos informativos 35min.
AntonioSWADroid / PFC SWADroid / [Antonio Guerrero] Añadidas traducciones de los títulos y demás en los módulos informativos.		

En este punto del proyecto, pensaba que iba a ser fácil lo que me quedaba por hacer, incluso llegué a pensar que podría implementar muchas más funcionalidades puesto que en apenas 25-30 horas había terminado de implementar una nueva funcionalidad que, además de ser muy usada en SWAD no estaba disponible en SWADroid y que iba a aumentar de seguro el número de descargas de la aplicación (y así fue como veremos más adelante).

Se continuó con el trabajo y se comenzó a implementar el nuevo módulo de mensajes. Esta fue la temporización aproximada:

 #19 REDISEÑO DEL ENVIO DE MENSAJE (EN PROGRESO) (1 SEM. 4D 38MIN. / 0MIN.)  
Creado 17:19 09/04/2014 por Antonio Guerrero - Última Actualización 20:09 28/08/2014 por Antonio Guerrero

**Información**

Resumen	Rediseño del envío de mensaje
Etiquetas	
Descripción	Rediseñar el módulo de mensajes para poder enviar un nuevo mensaje seleccionando los destinatarios de una lista en lugar de tener que saberlos todos los nicknames de los destinatarios a los que queremos enviarles un mensaje y tener que escribirlos a mano.  Crear nuevas vistas para que los mensajes sean a pantalla completa y no diálogos.  Añadir opciones para filtrar la lista de usuarios.

Se ve claramente que el módulo de mensajes se ha llevado casi el 25% del tiempo total del proyecto. Se ha tardado en dotarlo de una funcionalidad completa unas 90-100 horas teniendo en cuenta el tiempo invertido en corregir bugs y en hacer pruebas y validaciones.

Ya se comentaba que había sido complicada la implementación del nuevo módulo, pero ahora, se ve claramente reflejada esa dificultad en forma de tiempo invertido.

Algunas de las tareas llevadas a cabo han sido:

Thursday, 28 August 2014  
20:04 - 21:39  #19 Rediseño del envío de mensaje  
1h 35min. AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]  
- La lista se muestra ordenada.  
- Los destinatarios se añaden correctamente al campo destinatarios del mensaje tanto si hay algo escrito como si no.  
- El mensaje se envía correctamente.

18:28 - 22:13  #19 Rediseño del envío de mensaje  
3h 45min. AntonioSWADroid / PFC SWADroid / [Antonio Guerrero]  
Nuevas modificaciones pero aun continuo sin poder ordenar la lista.

- 2h 50min.  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
Arreglados fallos en la carga de la lista pero continúa mostrándose desordenada.
- Monday, 25 August 2014  
12:48 - 18:21  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
5h 3min.  - Eliminados comentarios y permisos innecesarios.  
- Las imágenes se muestran ya correctamente, ahora el problema es que los usuarios no se muestran en el orden correcto. No se detecta el fallo. Continúo con las pruebas.  
- Falta que se termine el WS y añadir los usuarios seleccionados a los destinatarios.
- 12:47 - 13:17  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
30min.  Ya se visualizan las imágenes tras los cambios que se han añadido al proyecto general.
- Saturday, 23 August 2014  
12:23 - 16:00  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
3h 37min.  Continuo con el mismo error.
- Friday, 22 August 2014  
15:50 - 19:30  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
3h 40min.  Los archivos se guardan corruptos en la SD. No consigo solventar el error.
- Thursday, 21 August 2014  
11:26 - 14:14  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
2h 48min.  Detectado error en la carga de las imágenes del ExpandableListView. Causa: no se descargan bien las imágenes
- Monday, 18 August 2014  
16:03 - 20:02  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
3h 59min.  Arreglado el ExpandableListView.  
Ahora falla al expandir los grupos. El error está al decodificar la imagen del usuario. Parece que no se descarga bien.
- Thursday, 31 July 2014  
14:43 - 16:55  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
2h 12min. 
- 13:56 - 16:31  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
2h 35min. 
- Monday, 28 July 2014  
13:44 - 14:09  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
25min.  - Comprobado que se pueden descargar alumnos y profesores cambiando las opciones de la llamada al WS (hay que modificar el WS y hacer que se bajen todos a la vez)  
- Los mensajes los reciben los destinatarios correctamente
- 13:12 - 15:02  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
1h 50min.  - Cambiado el botón de enviar y añadido a la action bar.  
- La lista de usuarios se carga correctamente.  
- La lista se carga correctamente en el campo "destinatarios".  
- El mensaje, en principio (a falta de que alguno de los destinatarios responda) se envía correctamente usando este método.
- Wednesday, 06 August 2014  
12:04 - 13:57  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
1h 53min.  Solventado un error (NoSuchErrorMethod) pero aun no funciona el listado.
- Monday, 04 August 2014  
16:43 - 18:58  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
2h 15min. 
- 12:13 - 14:09  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
1h 56min.  Rediseñando el modulo de mensajes añadiendo un ExpandableListView.  
La anterior versión reutilizando código de RollCall no funciona bien ahora.
- Friday, 01 August 2014  
12:02 - 16:13  #19 Rediseño del envío de mensaje [AntonioSWADroid / PFC SWADroid](#) / [Antonio Guerrero]  
4h 11min.  Ha aparecido un error al cargar y actualizar la lista de usuarios.  
No encuentro el fallo de momento.

Si nos fijamos en las imágenes (desde abajo hacia arriba, teniendo en cuenta las fechas que aparecen) conforme avanzaba la implementación iban surgiendo más y más problemas que ha costado mucho tiempo y esfuerzo solventar. Esto ha sido debido a todos los problemas que se han ido mencionando en apartados anteriores en los que hablábamos de cómo se diseñó este módulo.

Como vistazo general a la temporización del proyecto y que incluye todas las tareas realizadas (excluyendo todo lo que ya se ha comentado con anterioridad) podemos tomar las siguientes imágenes.

Esta primera nos muestra el tiempo invertido desde que se comenzó con el proyecto hasta finales de 2013.

INFORMES				CSV
09/10/2013- 12/12/2013	Oct 13	Nov 13	Dic 13	Total
#3 Configurar, compilar y probar la aplicación desde el IDE Antonio SWADroid / PFC SWADroid /	35min.			35min.
#2 Descarga del código del proyecto y familiarización con el mismo Antonio SWADroid / PFC SWADroid /	3h 17min.			3h 17min.
#6 Implementar Introducción Antonio SWADroid / PFC SWADroid /	27min.	50min.		1h 17min.
#15 Implementar modulos informativos Antonio SWADroid / PFC SWADroid /			4h 58min.	4h 58min.
#4 Instalación del software necesario y familiarización con el mismo Antonio SWADroid / PFC SWADroid /	3h 26min.			3h 26min.
#4 Lectura memoria 1er PFC SWADroid Antonio SWADroid / PFC SWADroid /	1h 46min.			1h 46min.
#5 Trabajo diario Antonio SWADroid / PFC SWADroid /	1d 7h 45min.	1d 3h 45min.	2h	3d 5h 30min.
	3d 1h 17min.	1d 4h 35min.	6h 58min.	1sem. 4h 50min.

Y esta segunda la parte correspondiente a 2014:

INFORMES				CSV					
13/01/2014- 15/09/2014	Ene 14	Feb 14	Mar 14	Abr 14	May 14	Jul 14	Ago 14	Set 14	Total
#17 Ciclo de pruebas y Fix de errores Antonio SWADroid / PFC SWADroid /	35min.		6h 35min.				1h 45min.	4h 37min.	1d 5h 32min.
#18 Corrección de errores en los modulos informativos Antonio SWADroid / PFC SWADroid /			4h 50min.						4h 50min.
#14 Creación de iconos Antonio SWADroid / PFC SWADroid /			6h						6h
#16 Crear función getCourseInfo Antonio SWADroid / PFC SWADroid /				2d 4h					2d 4h
#2 Descarga del código del proyecto y familiarización con el mismo Antonio SWADroid / PFC SWADroid /					2d 6h 43min.				2d 6h 43min.
#15 Implementar modulos informativos Antonio SWADroid / PFC SWADroid /	1h 17min.	1h 30min.	35min.		5h 30min.				1d 52min.
#4 Instalación del software necesario y familiarización con el mismo Antonio SWADroid / PFC SWADroid /					5h 45min.				5h 45min.
#20 Redactar y maquetar memoria Antonio SWADroid / PFC SWADroid /						2h 22min.	2d 3h 33min.		2d 5h 56min.
#19 Rediseño del envío de mensajes Antonio SWADroid / PFC SWADroid /				1d 2h 50min.		2d 49min.	1sem. 4h 59min.		1sem. 4d 38min.
#5 Trabajo diario Antonio SWADroid / PFC SWADroid /	1h 18min.	4h 10min.	7h 40min.	7h	1h 25min.	3h 17min.	1d 25min.	4h 35min.	4d 5h 50min.
	3h 10min.	5h 40min.	3d 1h 40min.	4d 5h 50min.	4d 3h 23min.	2d 4h 6min.	1sem. 2d 1h 32min.	3d 4h 45min.	5sem. 1d 6h 7min.

Como se puede apreciar el inicio fue lento y constó prácticamente de tareas referentes a la familiarización con el proyecto.

Como es normal, aquí tampoco está incluido un tiempo que, obviamente, es incontable. Se trata del tiempo invertido en aprender Android. ¿Por qué es incontable? Pues simplemente porque nunca termina dicho aprendizaje. Constantemente hay que aprender funciones nuevas y métodos diferentes de hacer las cosas porque Android se va actualizando y nosotros debemos actualizarnos con él, así que es un tiempo que verdaderamente no se ha podido calcular.

A modo de resumen, sin incluir tareas se muestran los siguientes informes de tiempo:

09/10/2013- 12/12/2013		Oct 13	Nov 13	Dic 13	Total					
AntonioSWADroid / PFC SWADroid		3d 1h 17min.	1d 4h 35min.	6h 58min.	1sem. 4h 50min.					
		3d 1h 17min.	1d 4h 35min.	6h 58min.	1sem. 4h 50min.					
<hr/>										
13/01/2014- 15/09/2014		Ene 14	Feb 14	Mar 14	Abr 14	May 14	Jul 14	Ago 14	Set 14	Total
AntonioSWADroid / PFC SWADroid		3h 10min.	5h 40min.	3d 1h 40min.	4d 5h 50min.	4d 3h 23min.	2d 4h 6min.	1sem. 2d 1h 32min.	3d 4h 45min.	5sem. 1d 6h 7min.
		3h 10min.	5h 40min.	3d 1h 40min.	4d 5h 50min.	4d 3h 23min.	2d 4h 6min.	1sem. 2d 1h 32min.	3d 4h 45min.	5sem. 1d 6h 7min.

Los lapsus de tiempo que se aprecian y el desnivel de carga de trabajo en las tareas del proyecto son debidos, en su totalidad, a la alta carga lectiva que tenía en este año y la cuál no me ha permitido trabajar de una manera continuada en el proyecto.

Es por eso que, por ejemplo, tras los exámenes de febrero y junio la carga de trabajo supera altamente a la del resto de meses al estar más liberado tanto de exámenes como de prácticas.

Esta desigualdad en la carga se puede apreciar en la siguiente gráfica:



## 7. Estadísticas.

SWADroid ha cambiado mucho durante el desarrollo de este proyecto y esto ha sido en parte, gracias a la participación de SWADroid en el “VIII Concurso Universitario de Software Libre”.

Las funcionalidades disponibles en SWADroid a fecha 14/03/2014 (antes del concurso) eran:

- Asignatura > Documentos
- Asignatura > Archivos comunes
- Evaluación > Tests off line
- Mensajes > Notificaciones (incluyendo mensajes y consulta de calificaciones)
- Mensajes > Enviar mensaje
- Mensajes > Publicar aviso
- Usuarios > Grupos
- Usuarios > Generar QR

Y el número de instalaciones era de 11.111. Los accesos a SWAD desde SWADroid eran aproximadamente de unos 1.400 usuarios/día.

Durante la participación de SWADroid en el Concurso Universitario de Software Libre se consiguieron desarrollar las siguientes funcionalidades:

- Perfil > Envío de nueva contraseña.
- Perfil > Recordatorio de «introducir contraseña correcta».
- Perfil > ¿Por qué no funciona mi contraseña?
- Asignatura > Introducción
- Asignatura > Guía docente.
- Asignatura > Programa de teoría.
- Asignatura > Programas prácticas.
- Asignatura > Bibliografía.
- Asignatura > FAQ.
- Asignatura > Enlaces.
- Evaluación > Sistema de evaluación.

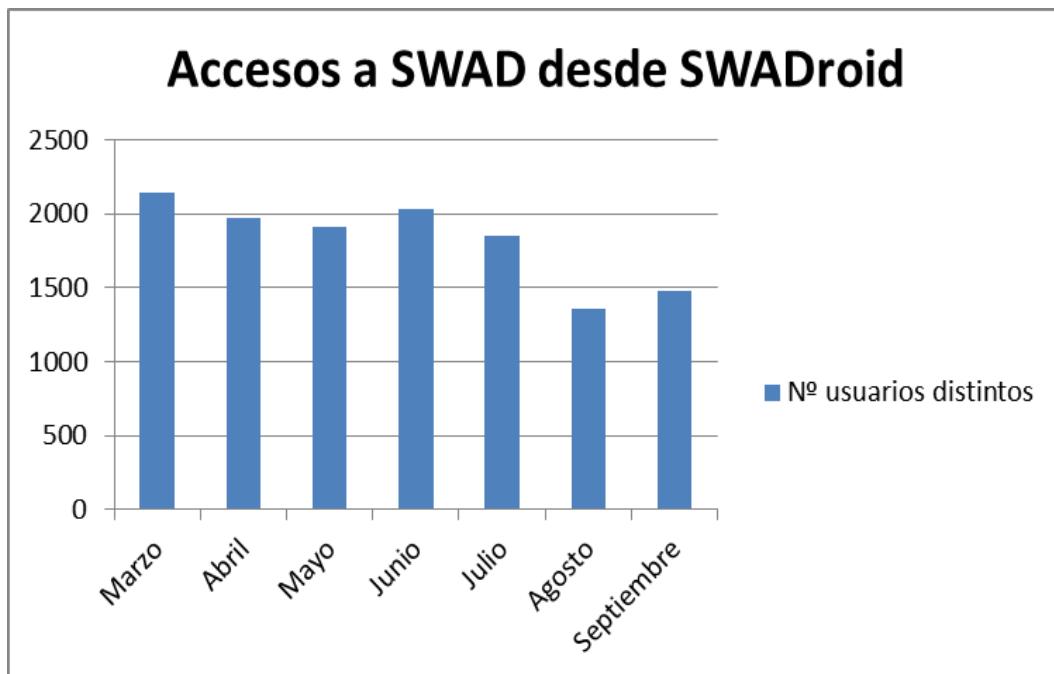
Lo que se tradujo en poder lanzar la primera versión estable de SWADroid desde que fuese creado en 2011 por Juan Miguel Boyero Corral y en un aumento del uso de la aplicación y del número de instalaciones (240 más).

Estos son los datos de acceso a SWAD desde SWADroid desde la primera actualización lanzada a Play Store durante el proyecto hasta septiembre de 2014:

Mes	Nº Total Clics
Marzo	2140
Abril	1970
Mayo	1915
Junio	2033
Julio	1847
Agosto	1363
Septiembre	1474
<b>MEDIA</b>	<b>1820</b>

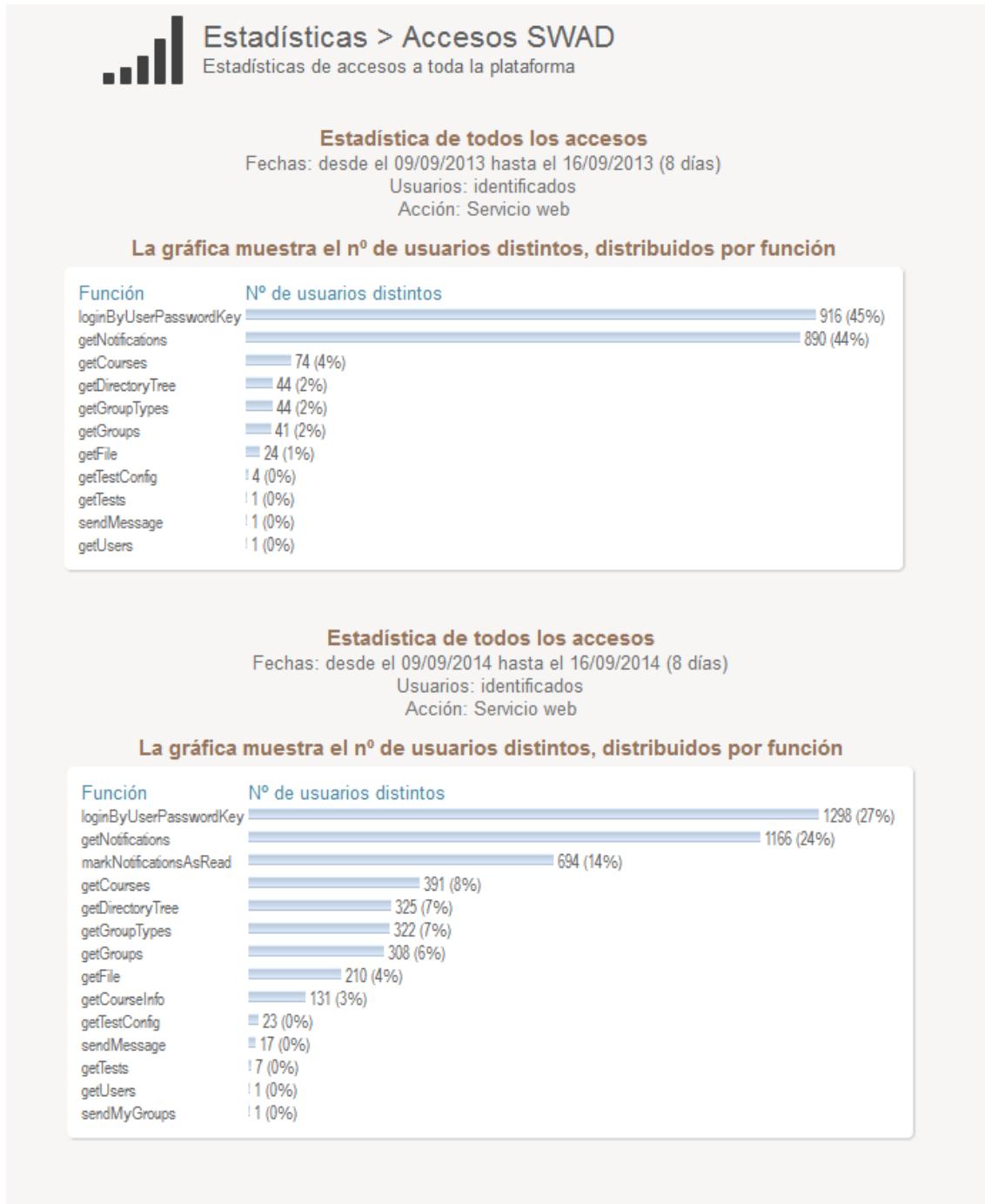
Como se puede apreciar en la tabla, la media de accesos con las modificaciones publicadas en Play Store hasta la fecha 16/09/2014 han aumentado en 420 usuarios más por día, teniendo en cuenta que estas se ven un poco perjudicadas porque están incluidos los meses de vacaciones en los que apenas hay actividad lectiva y que el mes de septiembre está hasta la fecha actual de la toma de datos (16/09/2014).

La distribución de accesos se puede ver mejor en la siguiente gráfica:



Obviamente, un aumento de 420 accesos diarios es una grandísima estadística teniendo en cuenta que son usuarios diferentes, es decir, que si un usuario accede a SWAD desde SWADroid, por ejemplo, el día 21/06/2014 a las 12:00 y lo vuelve a hacer el mismo día a las 15:00, esto solo cuenta como un único acceso.

En las siguientes gráficas se podrá ver cómo, con las nuevas modificaciones que han sido lanzadas en forma de actualización en Play Store, las estadísticas de uso de SWADroid han cambiado radicalmente, siendo mucho más usado comparando la misma fecha en 2014 (con las actualizaciones) con la del 2013 (sin las actualizaciones).



Se puede apreciar cómo, aparte de existir nuevos servicios Webs, los existentes son más usados y funcionalidades añadidas durante este proyecto, como por ejemplo obtener la información de las asignaturas (getCourseInfo) comienzan a ser más utilizadas.

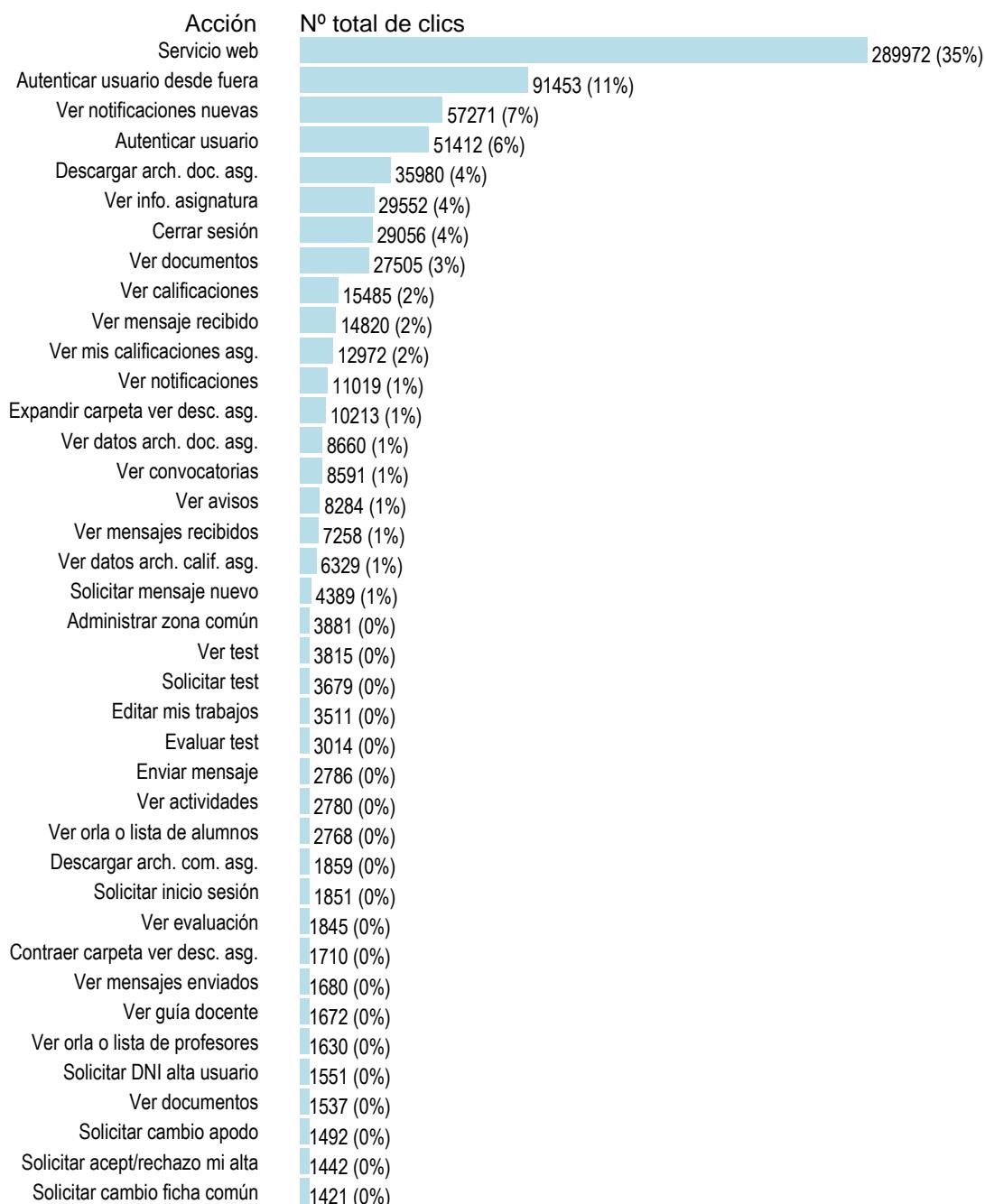
Estas estadísticas no incluyen el nuevo módulo de mensajes, pero aun así se ve como incluso este módulo (que en la instalación que tienen los usuarios continúa teniendo la funcionalidad limitada) es más usado y esto es debido, en parte, a que el resto de

modificaciones están haciendo que el uso de SWADroid s vaya extendiendo cada vez más.

Como estadísticas generales de SWAD, podemos ver en la siguiente tabla todas las funciones usadas distribuidas por número de clics, es decir, de las más usadas a las menos usadas.

Se puede comprobar cómo las funciones más usadas son las que SWADroid tiene implementadas casi en su totalidad.

Como se aprecia, el número de funciones disponibles desde la Web es enorme (aquí no están todas estas) y es totalmente imposible que SWADroid cuente con las mismas, pero sí que es posible implementar las más usadas y es precisamente eso lo que se ha hecho.



Solicitar cambio ficha asig.	1348 (0%)
Ver mensaje enviado	1304 (0%)
Crear ZIP doc. asg.	1148 (0%)
Ver últimos clics	1139 (0%)
Ver teoría	1092 (0%)
Descargar arch. doc. asg.	1025 (0%)
Seleccionar fechas para mis result. test	1019 (0%)
Solicitar unión a grupos	994 (0%)
Aceptar mi alta	989 (0%)
Ver prácticas	917 (0%)
Ver calificaciones	916 (0%)
Eliminar mensaje recibido	899 (0%)
Seleccionar una de mis asignaturas	858 (0%)
Solic. nuevo arch/carp mis act.	858 (0%)
Buscar desde asignatura	823 (0%)
Solicitar envío foto	806 (0%)
Descargar arch. doc. grp.	729 (0%)
Solicitar editar trabajos asig.	721 (0%)
Marcar notificaciones como vistas	706 (0%)
Solic. nuevo arch/carp mis trab.	705 (0%)

Si nos fijamos en la tabla que hay justo antes de esta última, podemos ver mejor lo que ya se ha comentado, que SWADroid implementa casi la totalidad de las funcionalidades más usadas en SWAD.

Todo esto ha llevado a un aumento paulatino de las instalaciones de SWADroid como muestran la siguiente tabla y gráfico:

Mes	Instalaciones
01/03/2014	11116
01/04/2014	11275
01/05/2014	11425
01/06/2014	11578
01/07/2014	11792
01/08/2014	11905
01/09/2014	11945
15/09/2014	12034



## 8. Futuras mejoras.

La elección de las mejoras en SWADroid siempre ha pasado por el análisis de las funciones más utilizadas por los usuarios de SWAD. Como podemos comprobar en la siguiente gráfica, sus funciones más utilizadas ya están incluidas en la aplicación.

En proyectos de SWADroid anteriores se pronosticaba la siguiente futura mejora (por parte de Helena Rodríguez Gijón): “*Un aspecto de gran importancia y que será necesario cambiar a medio plazo es la propia interfaz de la aplicación. Esto es debido a la gran velocidad a la que está evolucionando el mundo de los dispositivos móviles y a la continua actualización del sistema Android.*

*En estos momentos, tenemos que contemplar la irrupción de las tabletas y su más que posible extensión de uso para el m-learning. Aunque Android es capaz de redimensionar las aplicaciones para adaptarlas a los distintos dispositivos en los que se utilizan, es recomendable que la misma aplicación contemple un diseño adaptado a los distintos tamaños de dispositivos.*

*Respecto a la interfaz, también se ha de tener en cuenta la mejora gráfica y de diseño que está haciéndose en Android desde su salida. Esta mejora ha sido especialmente acusada en la nueva versión 4.0, por lo que en un espacio relativamente corto de tiempo, si SWADroid no se adapta a estas mejoras corre el riesgo de tener un aspecto gráfico obsoleto. Para esta función sería muy recomendable que el proyecto contara con un diseñador gráfico especializado que mantuviera el diseño de la aplicación actualizado.”* Y que como se ha visto, es uno de los cambios que se han realizado durante este proyecto. Pues bien, esto es una mejora que siempre estará presente por, como bien se ha dicho, los constantes cambios en las diferentes versiones de Android.

Se podrían incluir mejoras como:

- Poder leer y escribir en los foros.
- Chat interno.
- Pasar lista.
- Subir archivos a la zona “Mis trabajos”.

Y seguramente, teniendo en cuenta el equipo de desarrollo con el que cuenta SWADroid serán implementadas en un corto-medio plazo.

Finalmente, la mejora continua tiene que estar muy centrada en la resolución de bugs. Esta aplicación cuenta con la ventaja de ser utilizada por mucha gente, debido primordialmente a la gran difusión de SWAD en el ámbito de la Universidad de Granada, por lo que existe un enorme feedback por parte de los usuarios que utilizan distintas versiones de Android. Este feedback debe ser aprovechado tanto para solucionar fallos como para implementar los cambios y mejoras que sean más demandados por los usuarios.

## 9. Conclusiones.

En un principio, topé con el proyecto de casualidad pero desde un primer momento me llamó la atención. Durante un tiempo estuve barajando los pros y los contras, hablando con Antonio Cañas Vargas sobre la dificultad y perspectiva del mismo. Ampliar SWADroid me interesaba principalmente por dos motivos: era una gran idea participar en un proyecto para ampliar una aplicación que yo mismo usaba desde hacía ya mucho tiempo y además iba a “aprender” (entrecomillado porque, como ya se ha dicho en muchas ocasiones, Android va cambiando constantemente y debemos estar siempre al día) a programar en Android, cosa que llevaba tiempo queriendo hacer.

Por otro lado, su aspecto colaborativo era también una gran motivación. En este sentido, valoré muy bien que la informática real normalmente se lleva a cabo en grupos de trabajo y que el éxito de los proyectos depende, muchas veces, de la capacidad de integración y coordinación de estos equipos.

Ahora, tras todo el trabajo realizado, puedo decir que participar en el desarrollo de SWADroid ha superado mis expectativas. Esto ha sido en gran medida debido a todo lo que he podido aprender.

Durante la realización del proyecto he podido comprobar que la dificultad del mismo no ha estado en si he tenido que implementar más o menos cosas, o en si esas cosas eran más o menos complejas, sino que ha estado en que la finalización completa o no de una funcionalidad venía de la mano de una comunicación constante y activa con el resto del equipo de desarrollo.

Existen casos en los que si un miembro del equipo se atasca en algo que otro miembro necesita para una funcionalidad que está implementando, no podrá terminarla por completo. Esta dificultad existe debido a que los integrantes del equipo de desarrollo no están centrados simplemente en programar SWADroid sino que tienen sus estudios y/o trabajos y aportan al proyecto cuando tienen tiempo libre, lo que podría considerarse una dificultad añadida.

Esto me hace darme cuenta de lo que realmente supone trabajar en equipo y la dificultad que esto tiene. En este caso, somos 5-6 programadores como mucho, pero en casos en los que se tenga que coordinar un amplio grupo de personas debe de ser extremadamente complicado.

En estos momentos, SWADroid es mucho más funcional y su difusión entre la comunidad universitaria de Granada ha crecido considerablemente, no sólo por la inclusión de nuevas características, sino por su mayor difusión, por su presentación a la comunidad de Software Libre mediante la participación y premio en el “VIII Concurso Universitario de Software Libre” y, además, por el aumento de descargas y de comentarios positivos de los usuarios.

A nivel personal, el proyecto me ha aportado una auténtica inmersión en el desarrollo de aplicaciones para dispositivos móviles. Teniendo en cuenta el crecimiento y evolución continuos de este campo, el conocimiento de las particularidades del diseño y programación de aplicaciones Android supone una buena oportunidad de acceso al mundo laboral una vez finalizada la Ingeniería.

Por otro lado, el uso del control de versiones distribuido Git en un marco de desarrollo colaborativo, ha supuesto la adquisición de una serie de habilidades y conocimientos de gran utilidad para el desarrollo de proyectos en el futuro.

La coordinación del grupo de trabajo en el Hackathon realizado durante el “VIII Concurso Universitario de Software Libre” y la presentación y defensa que realicé del proyecto para el mismo han supuesto también una experiencia que valoro muy positivamente.

Finalmente, la participación en este proyecto me ha ligado de tal manera a él, que, personalmente, voy a continuar colaborando con SWADroid aún finalizado este proyecto fin de carrera. Esto es debido a que es un proyecto que cuenta que una actividad constante por parte de Antonio Cañas Vargas y Juan Miguel Boyero Corral (creador de la aplicación) que siguen trabajando para mejorar la aplicación todavía más, lo que convierte a SWADroid en un proyecto que no va a “morir” de momento y lo que invita a continuar participando en él; porque viendo la aceptación que SWADroid tiene (y que cada día aumenta más) me gustaría que futuros usuarios pudiesen disponer de una aplicación que contase casi con la totalidad de funciones de las que dispone la versión Web, ya que es lo que a mí me hubiese gustado tener; y porque pienso que esto puede ayudar mucho a los usuarios en sus estudios, teniendo acceso a todo SWAD desde cualquier lugar y de una manera mucho más rápida.

## 10. Bibliografía.

### [1] El Gran libro de Android avanzado.

1<sup>a</sup> edición.

Autor: Tomás Gironés, Jesús.

Editorial: MARCOMBO, S.A

### [2] El Gran libro de Android [actualizado a la versión 4.2].

3<sup>a</sup> edición.

Autor: Tomás Gironés, Jesús.

Editorial: MARCOMBO, S.A

### [3] Android. Desarrollo de aplicaciones ganadoras.

Edición 2013.

Autor: Wei-Meng Lee.

Editorial: Anaya Multimedia.

### [4]Android: programación de dispositivos móviles a través de ejemplos.

Edición 2012.

Autor: José Enrique Amaro Soriano

Editorial: MARCOMBO, S.A

### [5] Pinceladas sobre Android.

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

### [6] Principales componentes de Android

- <http://www.insdout.com/tutoriales/primeros-pasos-para-programacion-en-android.htm>
- <http://androideity.com/2011/07/05/componentes-de-una-aplicacion-android/>

### [7] Ciclo de vida de las actividades en Android

- <http://telekita.wordpress.com/2012/02/03/ciclo-de-vida-de-una-activity/>
- <http://androideity.com/2011/07/06/ciclo-de-vida-de-una-actividad/>

### [8] Recursos para programar en Android

<http://www.whatstnew.com/2011/03/07/recursos-en-espanol-para-aprender-a-programar-en-android/>

### [9] Curso de Programación Android

<http://elbauldelprogramador.com/curso-programacion-android/>

### [10] Estadísticas sobre el uso de Android y del Software Libre

- <http://www.marketingcharts.com/online/in-the-us-time-spent-with-mobile-apps-now-exceeds-the-desktop-web-41153/>
- <http://www.flurry.com/bid/109749/Apps-Solidify-Leadership-Six-Years-into-the-Mobile-Revolution#.VBg175R5N5I>
- <http://www.cenatic.es/hemeroteca-de-cenatic/1-actualidad-cenatic/39555-el-uso-del-software-libre-en-las-empresas-espanolas-sigue-creciendo-segun-el-ine>
- <http://www.cenatic.es/dossier/encuesta-ine>
- <http://www.portalprogramas.com/descargas/estudio-valoracion-software-libre.pdf>

**[11] SWAD. Para conseguir estadísticas de uso de SWAD y SWADroid.**  
<https://swad.ugr.es/es>

**[12] Guía de usuario de Git**  
- [http://wiki.eclipse.org/EGit/User\\_Guide](http://wiki.eclipse.org/EGit/User_Guide)  
- <http://rogerdudler.github.io/git-guide/index.es.html>

**[13] Mini tutorial Git y Comandos**  
<http://elbauldelprogramador.com/mini-tutorial-y-chuleta-de-comandos-git/>

**[14] Funciones del Servicio Web**  
<https://openswad.org/ws/>

**[15] Hacer un fork de un repositorio en GitHub**  
<https://help.github.com/articles/fork-a-repo>

**[16] Mantener actualizado un fork en GitHub**  
<https://help.github.com/articles/syncing-a-fork>

**[17] Como crear un Pull Request en GitHub**  
<https://help.github.com/articles/creating-a-pull-request>

**[18] Ayuda online de la forja GitHub**  
<http://help.github.com/>

**[19] Formatear texto en HTML**  
[http://www.w3schools.com/html/html\\_formatting.asp](http://www.w3schools.com/html/html_formatting.asp)

**[20] Página web de preguntas y respuestas orientada a desarrolladores**  
<http://stackoverflow.com/>

**[21] Memoria proyecto SWADroid. Juan Miguel Boyero Corral.**  
<http://www.slideshare.net/Marown/memoria-proyecto-fin-de-carrera-swadroid>

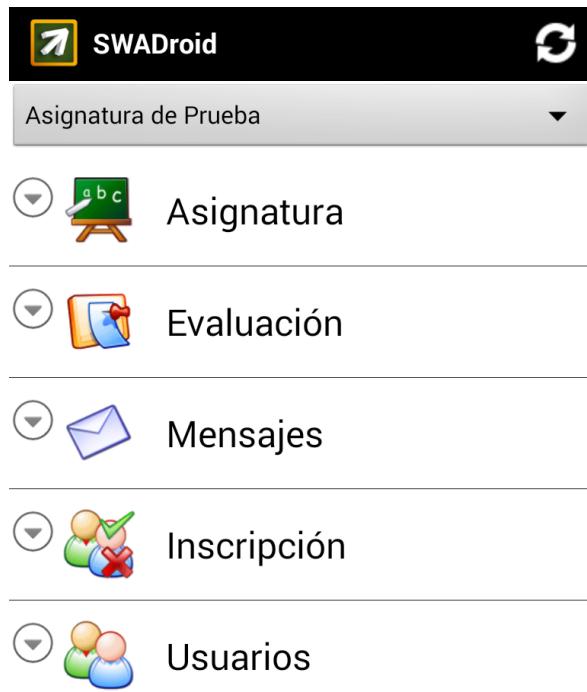
**[22] Memoria proyecto SWADroid. Antonio Aguilera Malagón. (Inhabilitado. Consultado el 15/05/2014)**  
[https://www.dropbox.com/s/7qn6cvb1xlbx3n3/PFC\\_Antonio\\_Aguilera.pdf](https://www.dropbox.com/s/7qn6cvb1xlbx3n3/PFC_Antonio_Aguilera.pdf)

**[23] Presentación proyecto SWADroid. Helena Rodríguez Gijón.**  
[http://prezi.com/c10wy1\\_jgqyx/swadroid-ampliacion/](http://prezi.com/c10wy1_jgqyx/swadroid-ampliacion/)

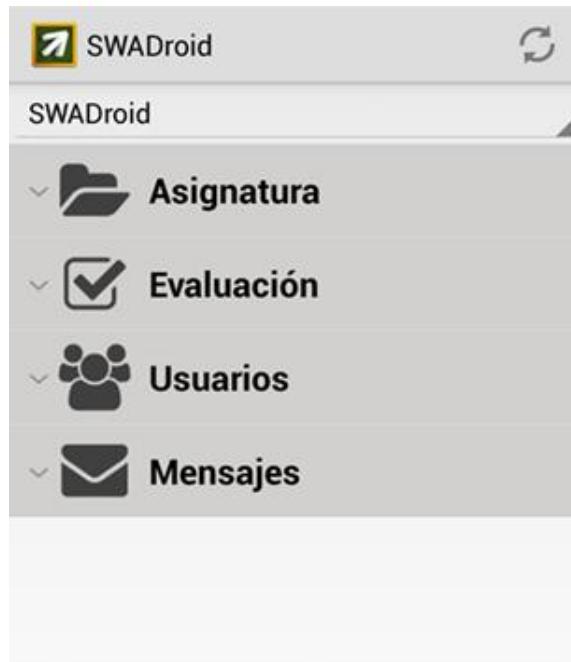
## 11. Apéndices.

### 11.1. La nueva interfaz

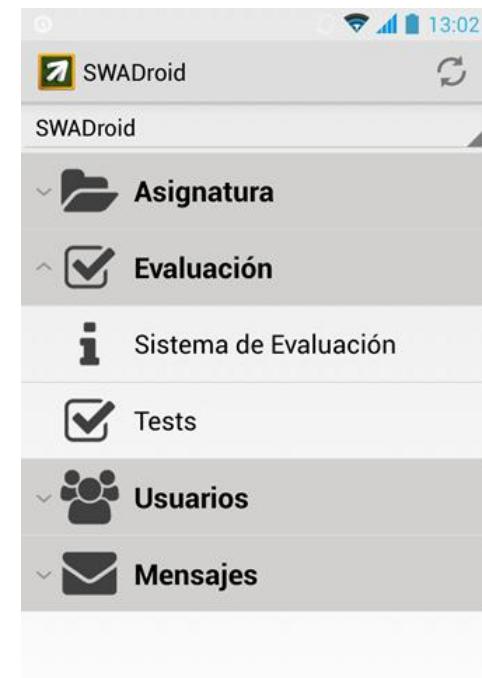
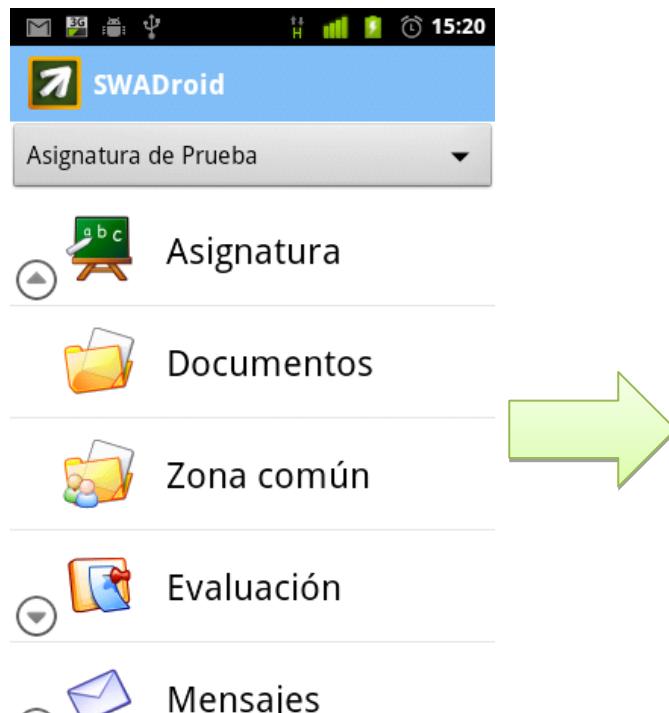
Pantalla principal con la antigua interfaz:



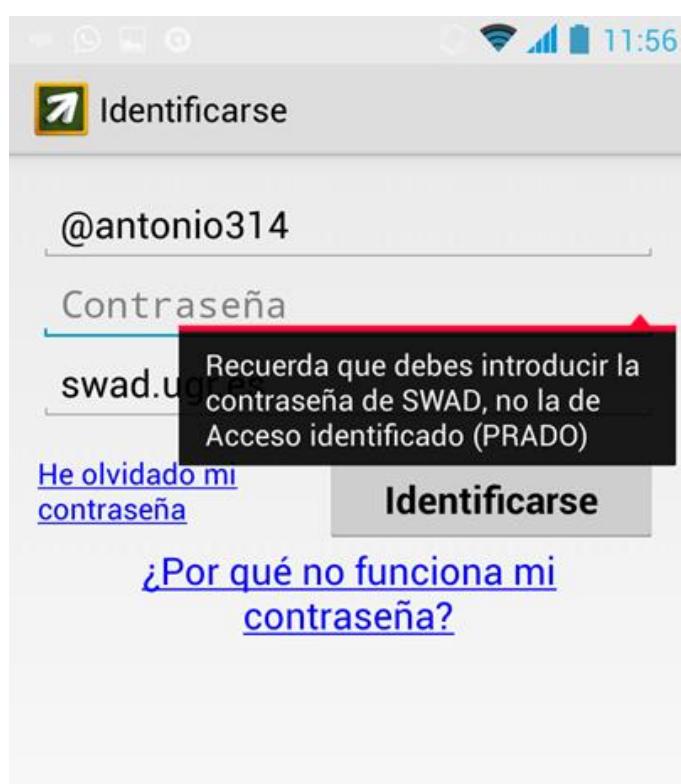
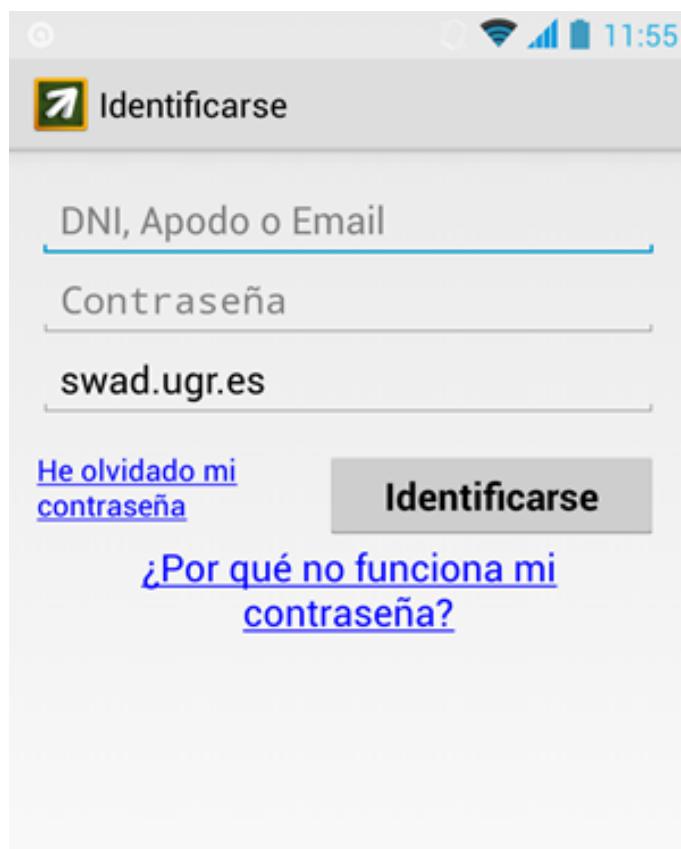
Pantalla principal con la nueva interfaz:



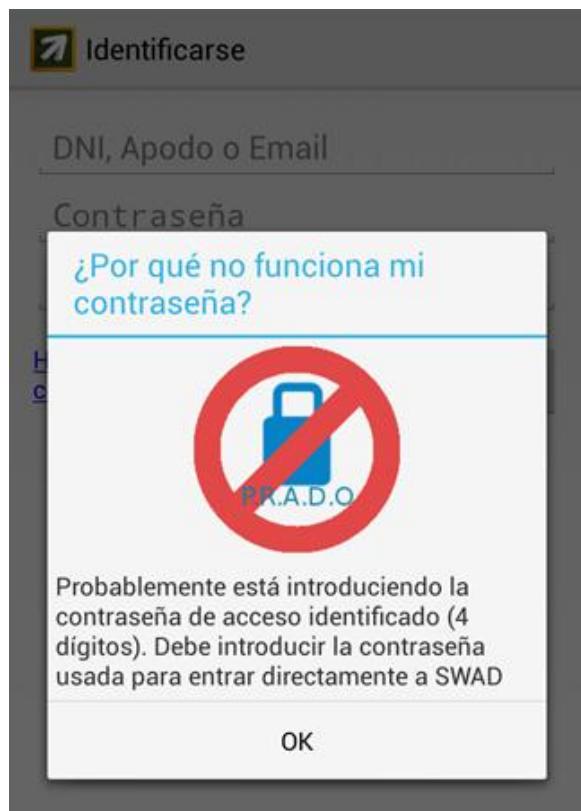
Consulta de la información de las asignaturas (antes no disponible):



Nueva página de “Primer inicio”:

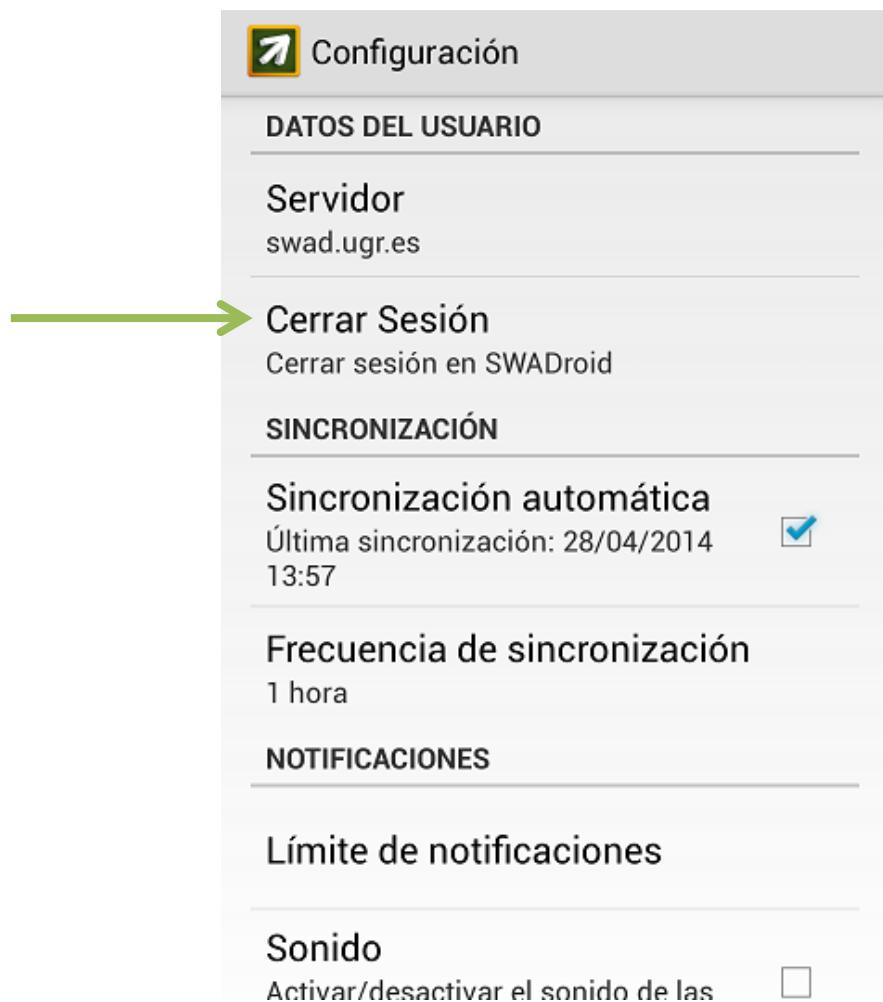


Aviso de contraseña errónea y recuperación de contraseña:



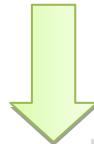
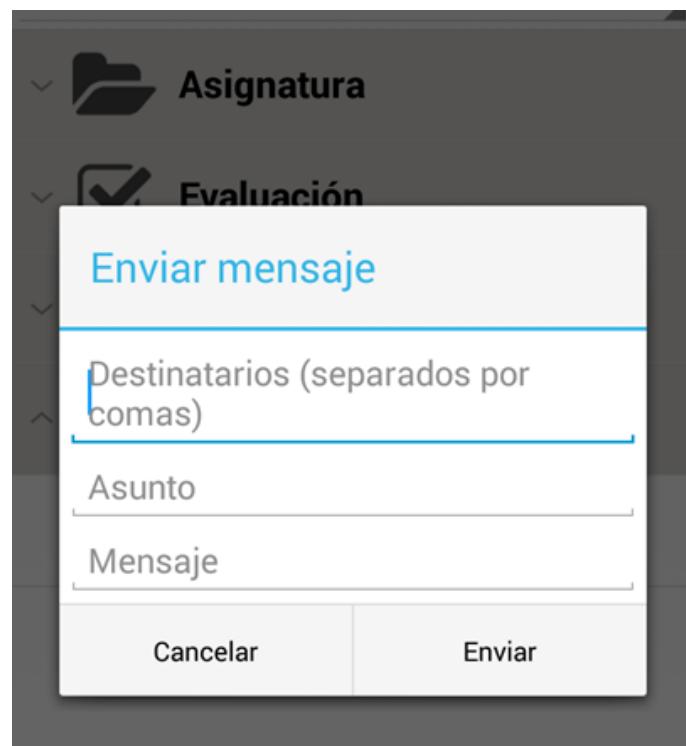
This screenshot illustrates the password recovery process. On the left, a mobile device displays the URL "swad.ugr.es". A modal window is open with the heading "He olvidado mi contraseña" and the option "Solicitar una nueva contraseña". Below this is a field labeled "DNI, Apodo o Email" with a green arrow pointing to it from the right side of the image. At the bottom of the modal are "Cancelar" and "OK" buttons. On the right, the "Identificarse" page is shown with the user identifier "@antonio314" and the password field filled with "swad.ugr.es". Below these fields is another "He olvidado mi contraseña" link. A large green arrow points from the "OK" button on the mobile device's modal to this link on the desktop page. To the right of the desktop page is a callout box containing the text "¿Por qué no funciona mi contraseña?". At the bottom right of the desktop page is a message box stating "Contraseña cambiada, comprueba tu email" (Password changed, check your email).

Cierre de sesión:



Algunos módulos con el cambio de interfaz:





Seleccionar Destinatarios

Profesores

	Aguilera Malagón, Antonio Manuel	<input type="checkbox"/>
	Alcalde Barros, Alejandro	<input type="checkbox"/>
	Boyero Corral, Juan Miguel	<input type="checkbox"/>
	Cañas Vargas, Antonio	<input type="checkbox"/>
	Guerrero Avilés, José Antonio	<input type="checkbox"/>

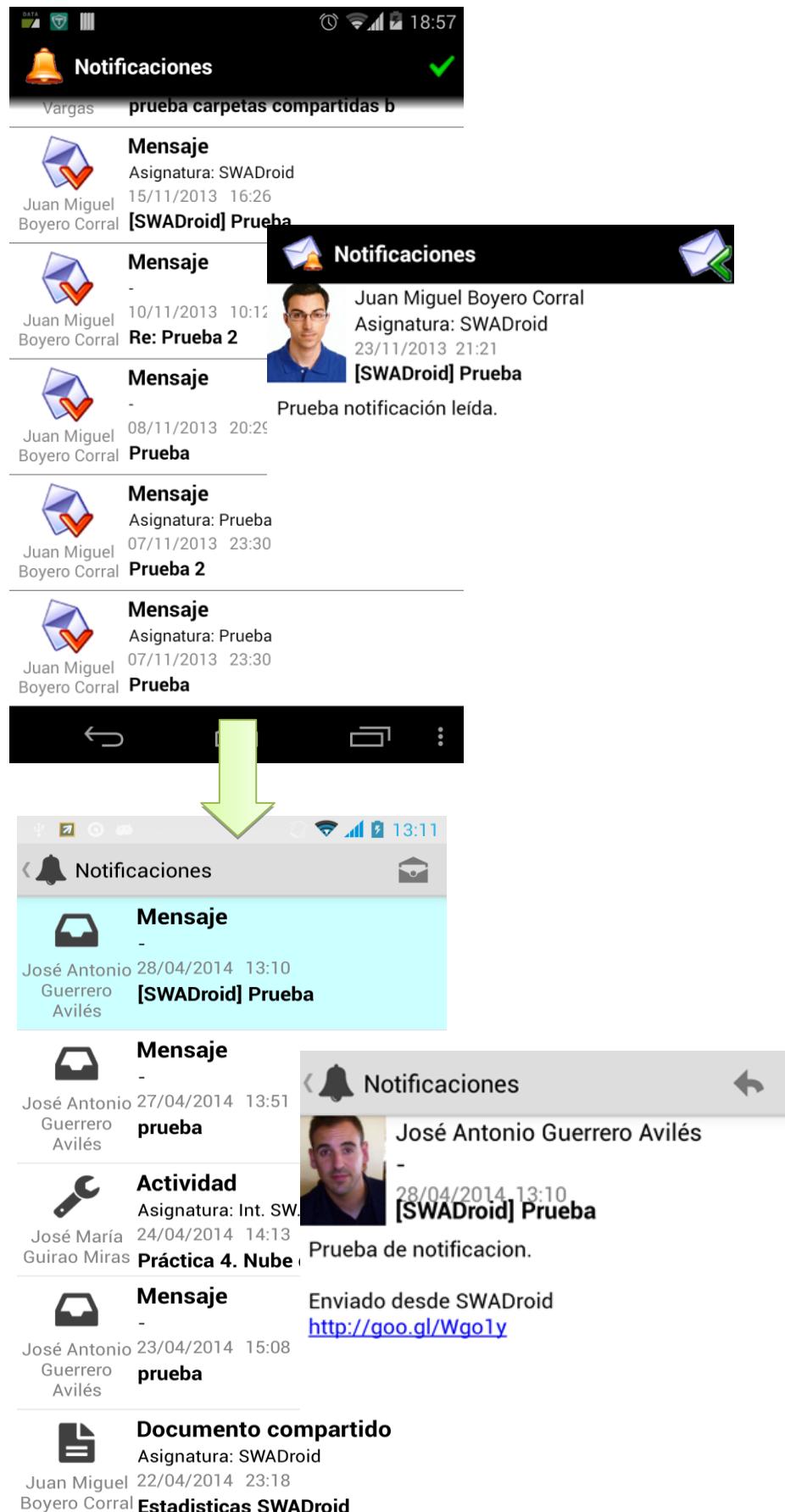
Enviar mensaje

Destinatarios (separados por comas)

Asunto

Mensaje

Aceptar

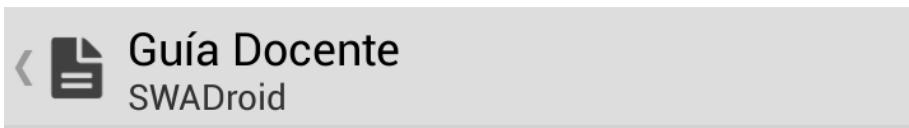
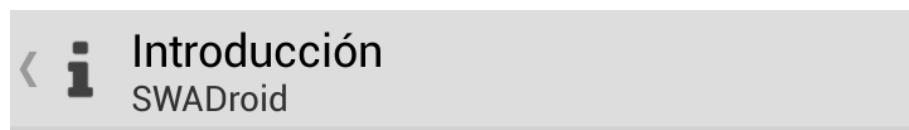


## 11.2. Módulo informativo. La potencia del WebView.

En este apartado vamos a ver una serie de capturas de pantalla que nos ilustrarán la verdadera potencia del WebView.

Se ha comentado durante la descripción del módulo informativo que este implementaba un WebView que nos permitía cargar contenido Web sin necesidad de abrir ningún navegador Web.

En las siguientes capturas de pantalla vamos a ver como este WebView es capaz de cargar en el mismo campo tanto texto plano, texto formateado en HTML o incluso una página Web.





## Programa de Teoría

Sis. Información

- 1 Información y el sistema de información en la empresa
- 2 El modelo de datos multidimensional
  - 2.1 Fundamentos del modelo de datos multidimensional: un modelo para consultas
  - 2.2 Diseño multidimensional
  - 2.3 Procesamiento de consultas y optimización
- 3 La fábrica de información corporativa
  - 3.1 Arquitectura del sistema de información: la fábrica de información corporativa
  - 3.2 Construcción de la fábrica de información corporativa



## Programa de Prácticas

SWADroid

- 1 Acentos ñáéíóú
  - 1.1 Item 1.1
  - 1.2 Item 1.2
- 2 Item 2



## Bibliografía

Sis. Información

R. Bouman, J. van Dongen: Pentaho Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL. Wiley (2009).

D. Darmawikarta: Dimensional Data Warehousing with MySQL: A Tutorial. BrainySoftware (2007).

M. Golfarelli, S. Rizzi: Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill (2009).

I. Gorbach, A. Berger, E. Melomed: Microsoft SQL Server 2008 Analysis Services Unleashed. Sams (2008).



## FAQs

Sis. Información

Teoría

---

Prácticas

---

StarTracker

---

P. A la hora de intentar ejecutar StarTracker me da el error "Printer error".

R. Hay que tener instalada una impresora en el sistema. Yo uso pdfCreator.

P. Me aparece un error avisando que no es válido el path de acceso al fichero MDB.

R. El nombre de este fichero está limitado a 8 caracteres de longitud.

P. Al iniciar la herramienta me indica que la base de datos está corrupta.

R. La base de datos ha de estar en formato 2.0 (ver el manual del desarrollador).



## Enlaces

Sis. Información

Conceptos:

Almacén de datos: [http://es.wikipedia.org/wiki/Almac%C3%A9n\\_de\\_datos](http://es.wikipedia.org/wiki/Almac%C3%A9n_de_datos)

OLAP: <http://es.wikipedia.org/wiki/OLAP> y <http://es.wikipedia.org/wiki/Cubo OLAP>

ETL: [http://es.wikipedia.org/wiki/Extract,\\_transform\\_and\\_load](http://es.wikipedia.org/wiki/Extract,_transform_and_load)

Herramientas:

Mondrian: <http://mondrian.pentaho.com/>



## Sistema de Evaluación

Sis. Información

De acuerdo a NCG71/2: Normativa de evaluación y de calificación de los estudiantes de la Universidad de Granada (aprobado en la sesión extraordinaria del Consejo de Gobierno de 20 de mayo de 2013, [http://secretariageneral.ugr.es/pages/normativa/ugr/ncg71...\)](http://secretariageneral.ugr.es/pages/normativa/ugr/ncg71...)

Teoría (50% de la calificación): Evaluación continua mediante ejercicios en clase. Examen final opcional.

Prácticas (50% de la calificación): Evaluación continua del desarrollo de las prácticas en clase. Se propondrá un itinerario no-presencial alternativo de prácticas para aquellos alumnos que acrediten debidamente, ANTES DEL 15 DE OCTUBRE DE 2013, que no pueden asistir a clase de prácticas durante el curso.

Para aprobar la asignatura será preciso obtener una calificación mayor o igual a cinco puntos en cada una de las partes.

Adicionalmente se tendrá en cuenta la participación activa en clase.

Hasta el momento se ha visto texto plano y texto formateado en HTML, pero ¿qué hay sobre la carga de una Web sin necesidad de abrir el Navegador?

En la siguiente imagen se puede ver eso mismo. Para ello se ha hecho que uno de los apartados informativos de una asignatura (en este caso el apartado Enlaces de la asignatura SWADroid) nos dirija a la web de la Escuela:

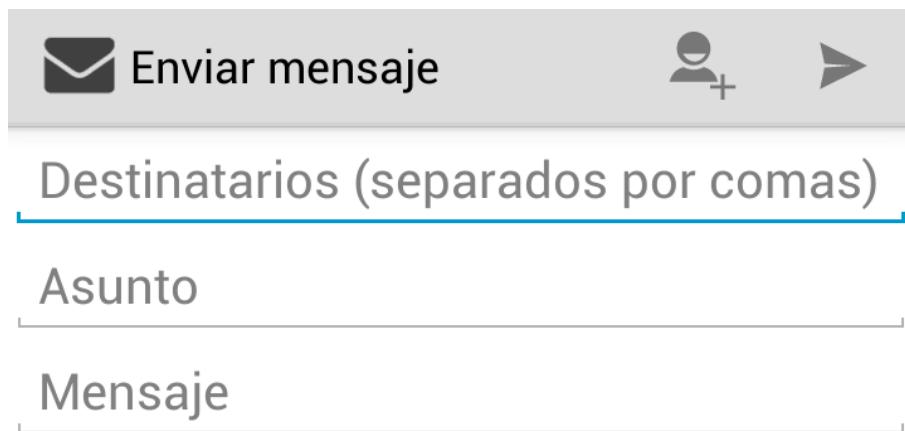


Con todas estas imágenes se puede apreciar con más claridad la gran versatilidad del WebView y lo útil que nos es en nuestro caso.

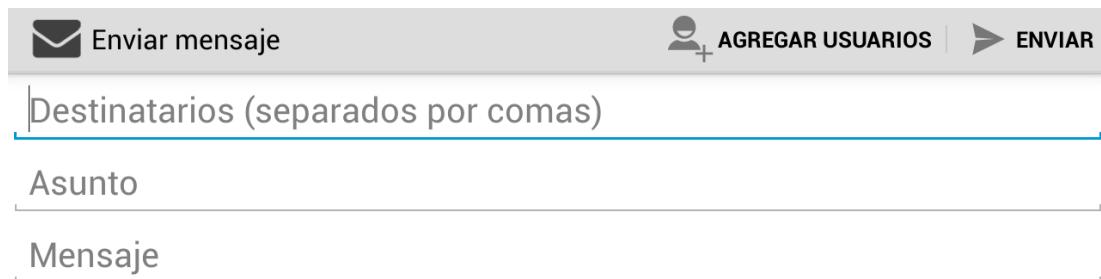
### 11.3. Módulo de mensajes. Manual de usuario.

Con la nueva implementación es necesario dar unas pinceladas sobre cómo usar el nuevo módulo. No es que sea para nada complicado su uso, pero el rediseño de la interfaz ha hecho que se eliminen botones y se creen nuevos y por tanto, es necesario explicar su uso.

La primera vista que nos aparece es la siguiente:



Esto será si la orientación del dispositivo es vertical, en cambio, si es horizontal, podemos ver qué es cada botón:



Como novedad, podemos añadir destinatarios desde un listado que nos aparecerá al pulsar el botón de agregar usuarios:



(en la pantalla horizontal se ve claramente cuál es ese botón).

Al seleccionar “Aregar usuarios” nos aparecerá la siguiente ventana tras descargar el listado de usuarios:



Esta ventana carga en varias categorías los usuarios. Si seleccionamos una de las categorías el listado se desplegará y podremos ver los usuarios con su foto y nombre.

En el caso de tener la pantalla del dispositivo en orientación vertical veremos:

Seleccionar Destinatarios

Profesores

	Aguilera Malagón, Antonio Manuel	<input type="checkbox"/>
	Alcalde Barros, Alejandro	<input type="checkbox"/>
	Boyero Corral, Juan Miguel	<input type="checkbox"/>
	Cañas Vargas, Antonio	<input type="checkbox"/>
	Guerrero Avilés, José Antonio	<input type="checkbox"/>

Aceptar

Y en el caso de que esté en horizontal:

Seleccionar Destinatarios

ACTUALIZAR

	Cañas Vargas, Antonio	<input type="checkbox"/>
	Guerrero Avilés, José Antonio	<input type="checkbox"/>

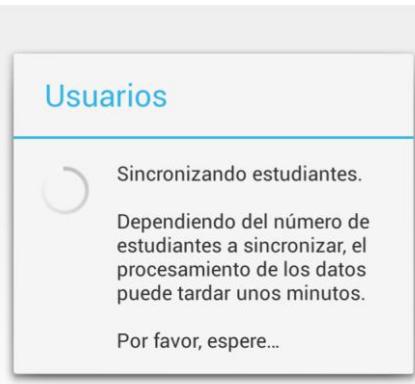
Alumnos

Aceptar

En esta pantalla tendremos la posibilidad de actualizar el listado pulsando el botón de actualizar:



(arriba a la derecha; en la vista horizontal se ve de manera clara):



O seleccionar destinatarios de la lista de usuarios y añadirlos pulsando el botón aceptar:

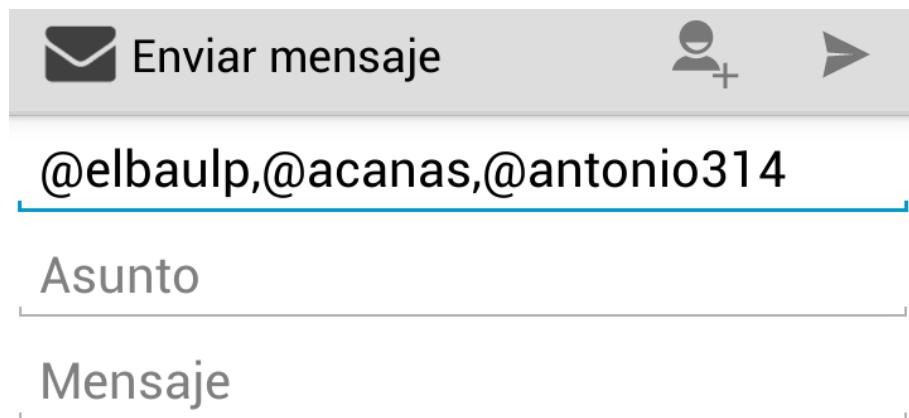
**Seleccionar Destinatarios**

**Profesores**

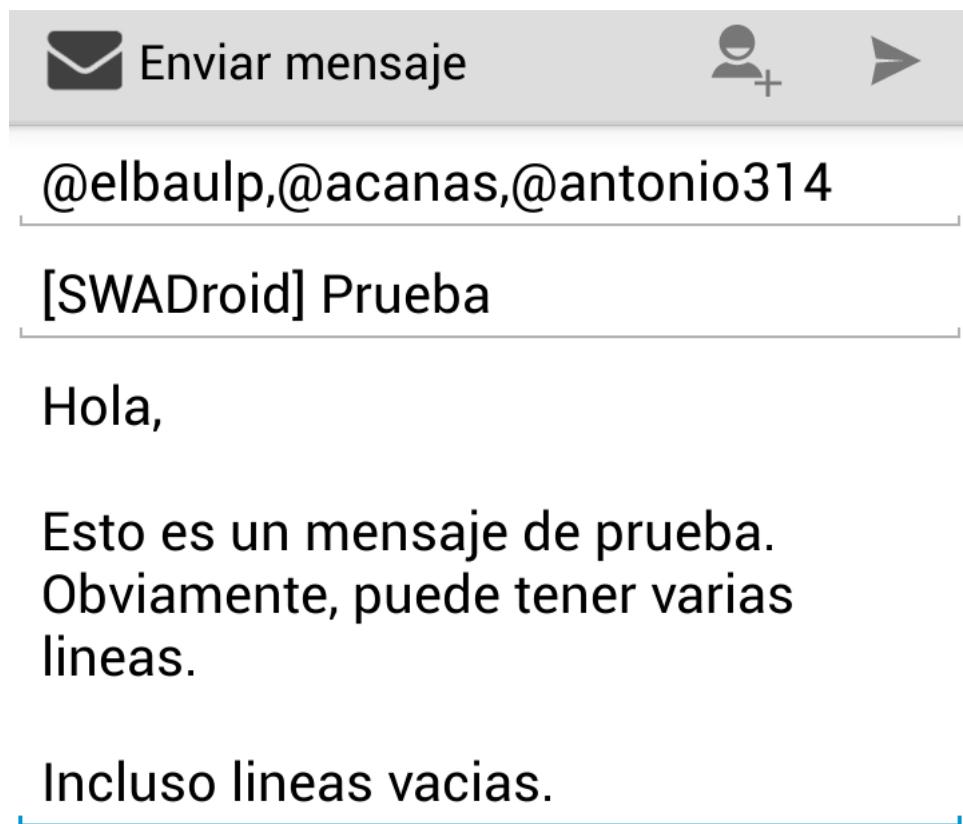
	Aguilera Malagón, Antonio Manuel	<input type="checkbox"/>
	Alcalde Barros, Alejandro	<input checked="" type="checkbox"/>
	Boyero Corral, Juan Miguel	<input type="checkbox"/>
	Cañas Vargas, Antonio	<input checked="" type="checkbox"/>
	Guerrero Avilés, José Antonio	<input checked="" type="checkbox"/>

**Aceptar**

Los usuarios se añaden al campo de destinatarios:



Podemos completar el asunto y el cuerpo del mensaje:

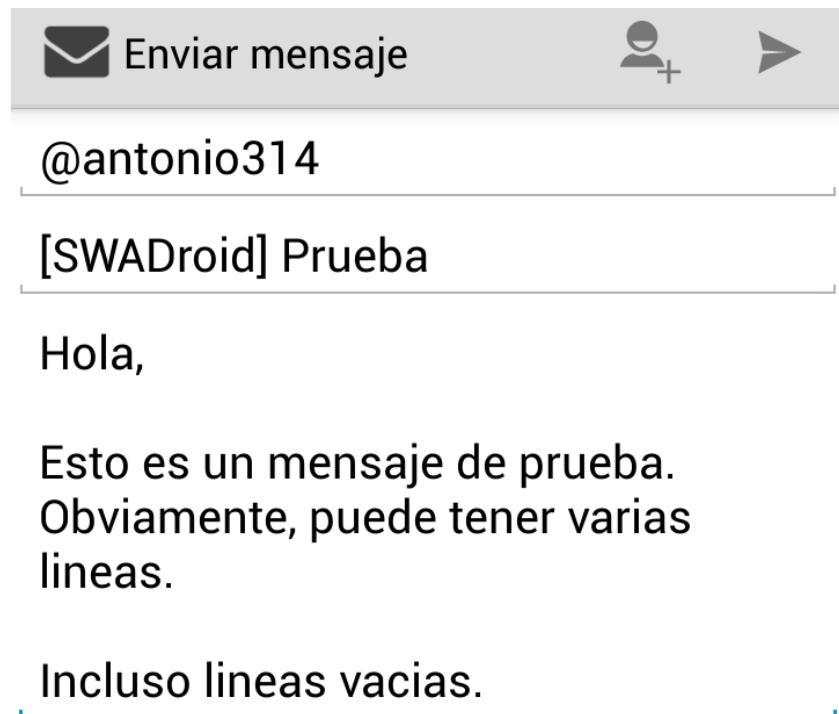
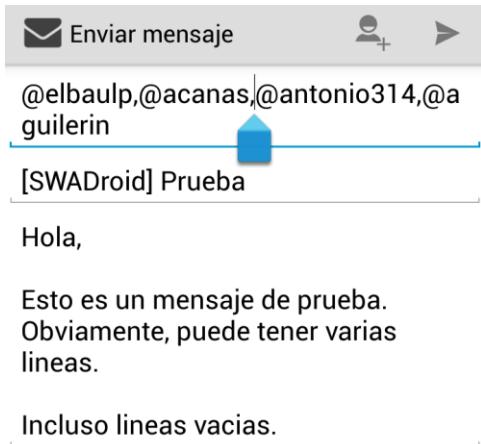


Una vez completado el mensaje podemos editar los destinatarios. Tenemos la posibilidad de añadir más usuarios de dos maneras: volviendo a seleccionar de la lista o manualmente.

En el caso de añadirlos del listado, seguimos el mismo procedimiento que se ha descrito anteriormente:

The screenshot shows a messaging application interface. At the top, there's a header with a group icon and the text "Seleccionar Destinatarios". Below it, a section titled "Profesores" lists five contacts with their names and profile pictures. The first contact, "Aguilera Malagón, Antonio Manuel", has a checked checkbox next to their name. The other four contacts have unchecked checkboxes. A large button labeled "Aceptar" is positioned below the contact list. To the right of the contacts, there's a message composition area with a "Enviar mensaje" button, a recipient icon, and a send arrow icon. The recipient field contains the email addresses: "@elbaulp,@acanas,@antonio314,@aguilerin". Below the message field, the text "[SWADroid] Prueba" is followed by a multi-line message body containing "Hola," and "Esto es un mensaje de prueba. Obviamente, puede tener varias líneas." A callout box points from the "Aceptar" button to the message body, stating: "Al pulsar aceptar se añade el nuevo usuario a la lista de destinatarios".

En el caso de hacerlo manualmente, podemos tanto añadir un usuario (eso sí, siempre que conozcamos su DNI o su apodo) o, como es lógico, eliminar alguno que ya se haya añadido:

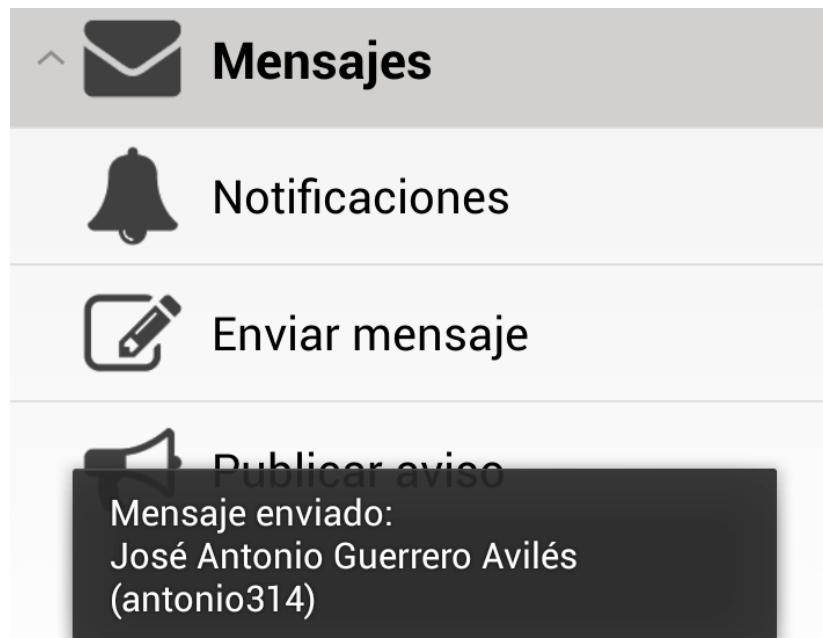


Ya estaríamos listos para enviar el mensaje pulsando el botón de enviar:

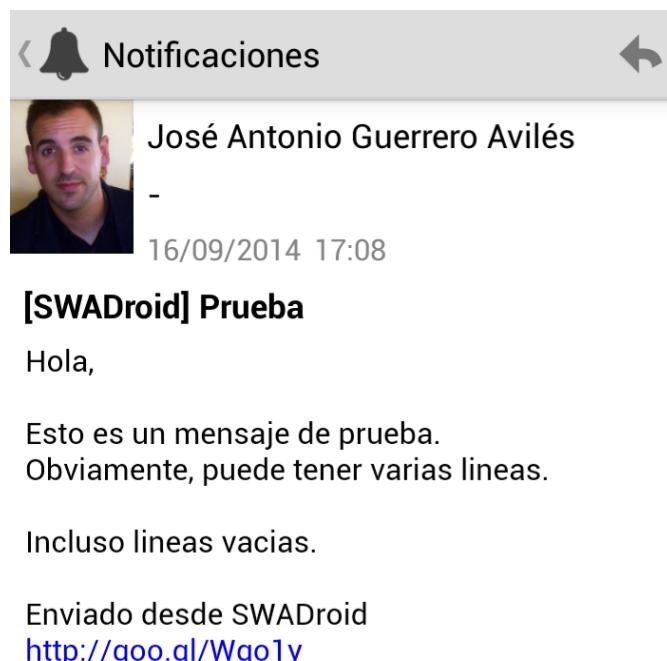


(en la vista horizontal se ve claramente este botón junto con su nombre)

Una vez pulsado, el mensaje se enviará y nos mostrará unos mensajes para que conozcamos cómo va el proceso:



El destinatario recibirá el mensaje y podrá leerlo:





Escuela Técnica Superior  
de Ingenierías Informática  
y de Telecomunicación



*ugr*

Universidad  
de Granada