UNIVERSIDAD
REY JUAN CARLOS

MÁSTER UNIVERSITARIO EN SOFTWARE LIBRE

Curso Académico 2011/2012

Trabajo Fin de Máster

QuiteSleep, evolución de un proyecto Android FLOSS

Autor : César Valiente Gordo

Tutor : Gregorio Robles

A mis profesores, tutor, y personas que he conocido y me han enseñado durante el transcurso de este máster.

A todos los compañeros de máster que me han acompañado.

A Samer Hassan y a Comunes Collective por toda la ayuda, apoyo y paciencia brindado a lo largo del transcurso del máster.

A mis amigos; por ser como son.

A Pablo, por estar siempre ahí.

A toda la gente maravillosa que he conocido en Amsterdam durante este último año, por su amistad y cariño.

Y como no, a mi familia; sin ellos no soy nada.

Índice

1. Resumen

QuiteSleep es una aplicación FLOSS para el sistema operativo **Android**[3] creada en 2010 como Proyecto Fin de Máster en Sistemas Telemáticos e Informáticos que hice previamente. A lo largo de todo este tiempo, y como proyecto vivo y FLOSS que es he ido mejorándolo y cambiándolo según he ido aprendiendo más, teniendo tiempo, etc.

La funcionalidad de QuiteSleep es la de bloquear de forma automática y siguiendo una configuración predefinida, llamadas en una franja de tiempo, tanto horaria como diaria. Una vez la llamada ha terminado, QuiteSleep dejará el teléfono en modo normal tanto volumen como vibración. QuiteSleep además, ofrece una funcionalidad añadida como es la de enviar mensajes de aviso indicando que se está durmiendo, ocupado, etc., tanto por ***SMS***[1] como por ***email***[2], actuando así este servicio como feedback a la persona que nos ha llamado.

En la ***Figura 1*** se puede apreciar el funcionamiento de QuiteSleep.

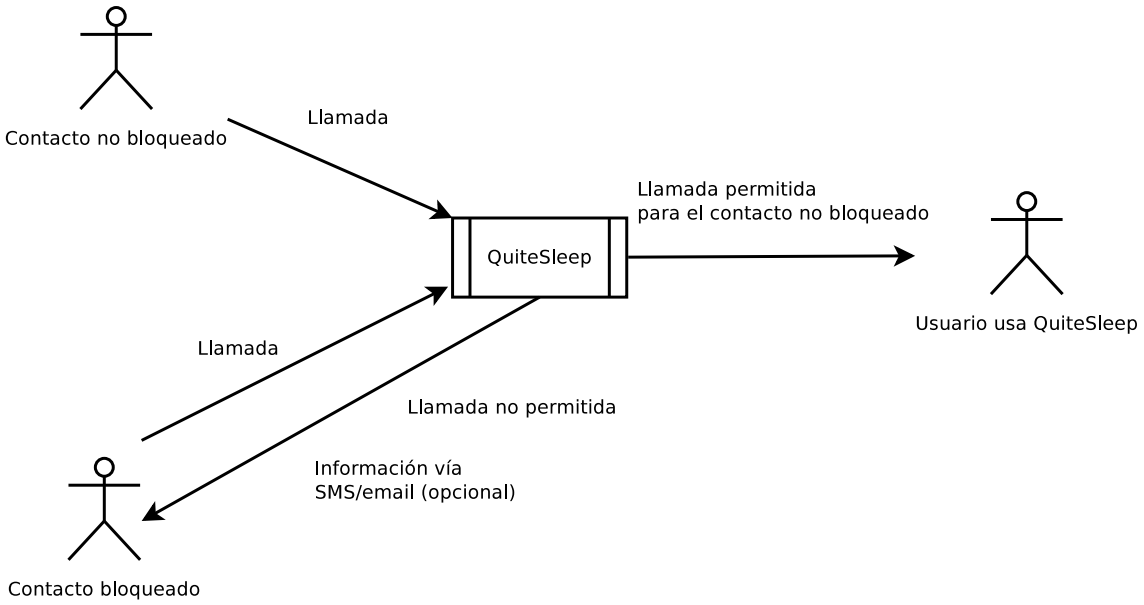

Figura 1: *Diagrama del funcionamiento de QuiteSleep.*

Indicar también que la llamada de contacto no bloqueado también puede ser bloqueada según hallamos configurado la aplicación, ya que también permite múltiples opciones de bloqueo.

[1] Short Message Service

[2] electronic mail

QuiteSleep, como se ha comentado anteriormente, es un proyecto de software libre, que respeta las cuatro máximas, postuladas por la **FSF**[3] [4], como son *libertad de uso, libertad de copia, libertad de modificación y libertad de distribución*. Así, QuiteSleep propone ser proyecto de software colaborativo, para mejorarlo, cambiarlo, adaptarlo a nuevas necesidades, etc.

Para la defensa de este proyecto vamos a hablar de dos aspectos:

- **Cambios en la implementación**, la interfaz gráfica de usuario ha sido rediseñada adaptada ahora a los nuevos patrones de interacción utilizados desde las nuevas versiones de Android. Con el cambio en la interfaz gráfica también se han cambiado multitud de elementos internos que más adelante se detallarán en profundidad.

- **Evolución del proyecto**, como proyecto FLOSS que es vamos a explicar también como ha evolucionado este proyecto desde que fue creado y publicado en Mayo de 2010 hasta como es en la actualidad con la publicación de la última versión en Mayo de 2012.

[3] Free Software Foundation

2. Introducción

QuiteSleep está incluida en un contexto muy actual como es el de las aplicaciones móviles, sector que en los últimos años ha tenido un avance muy grande tanto en creación de sistemas novedosos, como en la difusión de los mismos.

2.1. Situación histórica

Con la salida al mercado del sistema operativo *Android* en 2008, un mundo de posibilidades se abrió para para el mercado de dispositivos móviles ya que se trataba de un sistema operativo *open source*, y en el que las aplicaciones se desarrollaban en el lenguaje de programación Java, con lo que se situaba como clara competencia ante el sistema operativo *Symbian*[4] de *Nokia*[5] en el cual la programación era mucho más complicada al utilizar el lenguaje *C++*, *Windows Phone*[6] de *Microsoft*[7], para el cual se programa utilizando su plataforma de desarrollo *Visual Studio .Net*[8], que aunque permite el desarrollo en cualquiera de los lenguajes soportados por este entorno de desarrollo como C++, *C#*, o *VB*, no deja de tratarse de una plataforma privativa, para un sistema operativo completamente privativo. Finalmente hace frente al más valorado y utilizado por los desarrolladores y compradores y verdadero *hit* de ventas como es el sistema operativo de *Apple* [5] para su *iPhone*[9] llamado *iPhone OS (iOS)*[10] para el cual se desarrolla en un lenguaje parecido a C++ llamado *Objective-C*.

No solo la facilidad de desarrollo es lo que brinda Android, sino que al ser un sistema operativo libre, fabricantes de dispositivos móviles como son *HTC, Motorola, Samsung, etc.* pueden hacer uso de este sistema operativo sin realizar ningún pago de licencia, cosa que por ejemplo con el anteriormente mencionado Windows Phone no pasa, ya que como decimos este sistema operativo es privativo (el sistema operativo iOS solo está disponible para dispositivos Apple iPhone/iPad).

Tal es la aceptación que este sistema operativo ha tenido, que ante la dura competencia en

[4] http://www.symbian.org/
[5] http://www.nokia.com/
[6] http://www.microsoft.com/windowsphone/en-us/default.aspx
[7] http://www.microsoft.com/
[8] http://msdn.microsoft.com/en-us/vstudio/default.aspx
[9] http://www.apple.com/iphone/
[10] `http://es.wikipedia.org/wiki/IPhone_OS`

el sector móvil, Nokia liberó Symbian bajo la licencia *EPL*[11] y creó una fundación *(Symbian Foundation)* para mantener este proyecto software y ser mucho mejor acogido por la comunidad open source que si lo siguieran manteniendo ellos mismos.

Esto es a grandes rasgos lo que ha ocurrido en apenas cuatro años, la revolución Android está aquí y productos como *GoogleTV*[12] basado en Android, la interacción de Android con productos *Arduino* [13], y la cantidad de aplicaciones en el market y el negocio en torno a ellas, hacen de Android un sistema esencial en el mundo del software y hardware actual.

Antes incluso del lanzamiento oficial del primer teléfono con el sistema operativo Android (llamado HTC G1, HTC Dream, o Dev Phone), la comunidad de desarrolladores se volcó con Android y a su salida y sucesivamente más con el tiempo, el Android Market (ahora llamado Google Play) se ha llenado de aplicaciones desarrolladas por empresas, desarrolladores en solitario, grupos de ellos, etc. aplicaciones libres, privativas, juegos, widgets, etc. y aquí es donde entra en juego nuestra aplicación, QuiteSleep.

2.2. Motivación

La motivación en la creación de QuiteSleep en 2010 vino dada por la necesidad de tener un filtro de llamadas en teléfonos móviles con sistema operativo Android, sin hacer uso del *voicemail* [14] de serie ni de silenciar o colgar las llamadas de forma manual por el usuario.

La nueva release de QuiteSleep viene motivada por la continuación de este proyecto, por adaptarlo a los nuevos patrones de interacción, que no quede obsoleta, por seguir dando soporte a la multitud de usuarios que utilizan la aplicación y por la propia satisfacción de tener un proyecto de software libre utilizado por muchas personas a lo largo del mundo.

Esta aplicación desde su inicio fue pensada para ser publicada bajo una licencia FLOSS, esto fue tanto por afinidad ideológica a este movimiento como por querer generar comunidad y conocimiento alrededor del proyecto.

La primera versión de QuiteSleep fue publicada en el repositorio utilizado para ello una vez finalizada, con esta nueva release, los cambios se han ido haciendo y publicando simultáneamente como se hace en la mayoría de proyectos open source.

[11] Eclipse Public License

[12] http://www.google.com/tv/

[13] http://www.arduino.cc/

[14] Función que manda la llamada entrante al buzón de voz

2.3. Precedentes

Android es un sistema de reciente creación, pero en este tiempo, alrededor de cuatro años, se han desarrollado miles de aplicaciones de miles de desarrolladores, tanto empresas, como desarrolladores individuales, colectivos, agrupaciones, administraciones públicas, etc.

QuiteSleep no es la única que se ha introducido en el sector de aplicaciones que tratan de automatizar procesos cuando ocurren determinados eventos, pero si es única en cuanto a la funcionalidad concreta que ofrece.

Como veremos en la ***sección 4.1***, hay una serie de proyectos ya bien posicionados en las aplicaciones para el sistema operativo Android, que ofrecen características similares, pero siendo cada una diferente en varios aspectos, lo que hace de cada aplicación única en cuanto a funcionalidad, utilidad, interacción, etc.

Las aplicaciones son:

- **CallFillter** [6].
- **Sweet Dreams** [7].
- **Locale** [8].

Como decimos más adelante se pasará a detallar en que se basa cada una, que ofrece, y en que se parecen o son distintas a QuiteSleep.

2.4. Elementos funcionales

QuiteSleep está implementada haciendo uso del lenguaje de programación **Java** y cinco proyectos software como son:

- El ya mencionado, sistema operativo **Android**, para el que está desarrollado (para versiones iguales o mayores a la 2.2, Froyo)
- La base de datos orientada a objetos **db4o** [9]
- La librería open source **Java Mail API** [10] de *Oracle*[15]).

[15]http://www.oracle.com/index.html

- **ActionBarSherlock** [11] librería utilizada para poder utilizar el patrón de interacción *Action Bar* en la mayoría de versiones Android.

- **ViewPagerIndicator** [12] librería utilizada para poder utilizar el patrón de interacción *View Pager* en la mayoría de versiones Android.

Estos elementos son totalmente independientes entre si, tienen funcionalidades diferentes, y pueden ser utilizados conjuntamente o no (excepto el sistema operativo, que es indispensable).

Hemos hecho uso de estos cinco proyectos software basándonos en:

- La elección de un sistema operativo en el que desarrollar la aplicación.

- La elección de un modo de persistencia de datos.

- La elección de un sistema para transferencia de datos, en este caso correo electrónico, para hacer frente a nuestros objetivos y requisitos.

- La elección de una solución para tener los nuevos patrones de interacción de Android para la mayoría de versiones de este sistema operativo.

En la ***sección 3*** veremos en profundidad cada uno de estos componentes que hacen posible el desarrollo de la aplicación QuiteSleep.

3. Aspectos técnicos

En esta sección vamos a detallar tanto el lenguaje de programación en la que está implementada la aplicación, como los elementos funcionales utilizados en la misma, y que fueron mencionados anteriormente en la ***sección 2.4***.

3.1. Java

El lenguaje de programación utilizado en el desarrollo ha sido **Java**, ya que es el lenguaje de programación, impuesto por la plataforma Android para el desarrollo de aplicaciones.

Como bien es sabido, Java es un lenguaje de programación orientado a objetos creado por *Sun Microsystems* (ahora propiedad de Oracle) a principios de los años 90, Es un lenguaje muy parecido en sintaxis a C y C++, pero más sencillo que estos ya que omite el manejo de punteros y accesos a memoria que en los lenguajes anteriores suelen ser inductores de la mayoría de errores.

Java es un lenguaje puramente orientado a la multiplataforma, es decir, su propósito nació con que una aplicación creada sobre un lenguaje operativo, que no tuviera llamadas al mismo, fuera capaz de funcionar para cualquier plataforma distinta sobre la que se había desarrollado. Esto se consigue gracias a que Java es compilado en *bytecode* que es un código intermedio más abstracto que el código máquina y el cuál es interpretado en una *máquina virtual java (jvm*[16]*)*, este hecho es el que hace a Java multiplataforma, ya que existen máquinas virtuales para casi cualquier sistema operativo, con lo que el código generado en bytecode puede ser ejecutado en cualquier jvm.

Como decimos Java es un lenguaje orientado a objetos (OO), con lo que posee todas las cualidades propias de los lenguajes que utilizan este paradigma de programación, como son:

- Abstracción.
- Encapsulamiento.
- Herencia.
- Polimorfismo.

[16] Java Virtual Machine

Java además dispone de un *recolector de basura*[17] propio, con el que evitamos tener que liberar memoria explícitamente cada vez que no necesitamos ya un objeto o estructura de datos, como por ejemplo hacemos en C o C++; el recolector de basura se encarga de ello.

3.2. Android

Este proyecto ha sido concebido para uso en el sistema operativo Android. Android es un sistema operativo orientado a dispositivos móviles que usa una versión modificada del kernel[18] de Linux[19]. Esta plataforma permite el desarrollo de aplicaciones utilizando Java y a través del SDK[20] que ofrece. También es posible realizar aplicaciones más optimizadas utilizando el NDK[21] para desarrollo en lenguaje C.

3.2.1. Origen

Android fue originalmente desarrollado por *Android Inc.* compañía que fue comprada por *Google* [13] en 2005, la presentación de la plataforma Android se realizó el 5 de Noviembre de 2007 junto con la fundación de la *Open Headset Alliance*[22], que es una agrupación de compañías para desarrollar estándares abiertos para dispositivos móviles liderada por Google, algunos de los miembros son, HTC[23], Motorola[24], Qualcomm[25] y hasta 65 miembros en total. Android actualmente es desarrollado por esta agrupación de empresas.

Android está publicado como *open source* [14] bajo licencia Apache v2 [15] desde el 21 de Octubre de 2008, incluyendo la pila de red y de telefonía. Con la utilización de esta licencia, se permite la creación de aplicaciones privativas sin necesidad de difundir el código fuente ni hacer uso de la licencia Apache para difundirlas.

[17] garbage collector

[18] Es el responsable de facilitar a los distintos programas el acceso seguro al hardware del ordenador (o máquina) o en forma más básica, el encargado de gestionar recursos a través de llamadas al sistema

[19] http://www.kernel.org/

[20] Software Development Kit, es un conjunto de herramientas de desarrollo que permite a un programador desarrollar aplicaciones para un sistema concreto

[21] Native Development Kit, set de herramientas proporcionadas para desarrollo de aplicaciones que necesiten utilizar y acceder de forma mucho más eficiente a los recursos del sistema

[22] http://www.openhandsetalliance.com/

[23] http://www.htc.com/

[24] http://www.motorola.com/

[25] http://www.qualcomm.com/

Desde la primera versión publicada, se han sucedido las siguientes actualizaciones del sistema, que siguen tanto nomenclatura numérica como nomenclatura en forma de nombres de pasteles, por ejemplo Cupcake es una pequeña tarta del tamaño de un magdalena, Donut es un donut, etc. Las diferentes versiones son:

- **1.0**

  Versión del SDK lanzado en Octubre de 2008. No se define con un nombre. Lanzado junto con el primer teléfono con Android llamado HTC G1 o Dream.

- **1.1**

  Versión del SDK lanzado en Febrero de 2009. Solucionaba varios errores menores, no aportaba nuevas funcionalidades importantes.

- **1.5 Cupcake**

  Versión del SDK lanzado el 30 de Abril de 2009. Basado en el kernel de Linux 2.6.27.

- **1.6 Donut**

  Versión del SDK lanzado el 15 de Septiembre de 2009. Basado en el kernel de Linux 2.6.29.

- **2.0/2.1 Eclair**

  Versión del SDK lanzado el 26 de Octubre de 2009. Basado en el kernel de Linux 2.6.29.

- **2.2 Froyo**

  Versión del SDK lanzado el 20 de Mayo de 2010. Basado en el kernel de Linux 2.6.32.

- **2.3.x Gingerbread**

  Versión del SDK lanzado el 6 de Diciembre de 2010. Basado en el kernel de Linux 2.6.35.

- **3.0.x Honeycomb**

  Versión del SDK lanzado el 22 de Febrero de 2011, y basado en el kernel de Linux 2.6.36. El código fuente no fue liberado ya que Google argumentó que no era un código de calidad y que prefería esperarse a la siguiente versión.

- **4.0.x Ice Cream Sandwich**

  Versión del SDK lanzado el 19 de Octubre de 2011, basado en el kernel de Linux 3.0.1. El código fuente fue liberado el 14 de Noviembre del mismo año.

3.2.2. Características

El sistema operativo Android tiene una serie de características comunes a las de otros sistemas operativos y otras propias orientadas a los dispositivos móviles para los que está creado.

- Android dispone de un framework[26] para la creación de aplicaciones utilizando los recursos que provee.
- Máquina virtual **Dalvik**[27] optimizada para dispositivos móviles.
- Navegador web integrado, basado en **WebKit**[28].
- Gráficos optimizados en 2D, los gráficos en 3D se basan en la especificación **OpenGL ES 1.0**[29] (aceleración por hardware opcional). Desde la versión 2.2 de Android (Froyo), el framework también soporta la especificación **OpenGL ES 2.0**.
- Utiliza la base de datos relacional y ligera **SQLite**[16] para almacenamiento y utilización de datos estructurados.
- Soporte de los formatos de audio, vídeo e imágenes más comunes y utilizados como son **H.263, H.264 (en contenedores 3GP o MP4), MPEG4 SP, AMR, AMR-WB (en contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, OGG, WAV; JPEG, PNG, GIF, y BMP**.

[26] Estructura conceptual y tecnológica de soporte basada en módulos software concretos, con base en la cual otro proyecto software puede ser organizado y desarrollado

[27] Máquina virtual desarrollada por Dan Bornstein con contribuciones de otros ingenieros de Google

[28] Basado originalmente en el motor de renderizado KHTML del navegador web del proyecto Kde, Konqueror, es un framework open source, para aplicaciones que funciona como base para los navegadores Safari, Google Chrome, Epiphany, Midori y otros

[29] Open Graphics Library, es una especificación estándar que define una API multilenguaje y multiplataforma escribir aplicaciones que produzcan gráficos en 2D y 3D

- Telefonía GSM[30] (dependiente del hardware).
- Soporta **EDGE**[31]**, 3G y WiFi** (dependiente del hardware).
- Bluetooth con soporte para A2DP[32] y AVRCP[33].
- Soporta **cámara, GPS**[34]**, brújula, acelerómetro, pantallas táctiles con multitouch de diferentes tamaños y densidades** (dependiente del hardware).
- **Google Play Store**[35], anteriormente llamado Android Market, es una aplicación accesible desde los dispositivos móviles donde los usuarios pueden descargar tanto de forma gratuita como de pago aplicaciones realizadas por los desarrolladores. Tiene también un panel web accesible desde navegadores web, donde los desarrolladores pueden poner sus aplicaciones.
- **Multitarea** la capacidad de correr diferentes aplicaciones y procesos al mismo tiempo.

3.2.3. Arquitectura

Android tiene una serie de componentes principales que forman el esqueleto del sistema operativo, los cuales son los siguientes:

- **Aplicaciones**. Las aplicaciones base que vienen con el sistema operativo son el cliente de email, cliente de mensajería (sms/mms), calendario, mapas, navegador web, contactos, y otras. Todas las aplicaciones están desarrolladas en Java, y algunas de ellas son software privativo, con lo que no se tiene acceso a ellas cuando se obtiene el código fuente del proyecto Android, como son las aplicaciones de email (Gmail), y mapas (Google maps).

[30] Sistema Global para Comunicaciones Móviles, es un sistema estándar, completamente definido para la comunicación mediante teléfonos móviles que incorporan tecnología digital

[31] Enhaced Data rates for GSM of Evolution, es una tecnología móvil, que actúa como puente entre las redes 2G y 3G, se considera la evolución del GPRS

[32] A2DP define la calidad de la música transmitida a través de Bluetooth y que puede ser mono o estero

[33] AVRCP provee un interfaz para la manipulación de equipos de imagen y sonido

[34] Sistema de posicionamiento global, es un sistema de navegación por satélite que permite determinar en todo el mundo la posición de un objeto, persona, o vehículo con una precisión de hasta centímetros

[35] https://play.google.com/store

- **Framework de aplicaciones** Los desarrolladores tienen acceso de forma totalmente completa a las mismas APIs del framework de Android utilizado por las aplicaciones base, con lo que la diferencia entre estas y las de terceros son mínimas en cuanto a la posibilidad de utilizar los recursos del sistema. Esta arquitectura está diseñada para simplificar la reutilización de componentes, cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede acceder a esas capacidades publicadas, estando todo sujeto a diversas políticas de seguridad. Así como decimos, las aplicaciones de terceros pueden sustituir a las aplicaciones base del sistema como son las de mensajería, llamadas, etc.

- **Bibliotecas** Android provee una serie de bibliotecas en C/C++, usadas por varios componentes del sistema. Estas bibliotecas se ofrecen a los desarrolladores a través del framework de aplicaciones, como son bibliotecas de medios, bibliotecas de gráficos, bibliotecas de bases de datos (SQLite), implementación de la biblioteca C estándar, etc.

- **Runtime de Android** Android provee un set completo de bibliotecas que proporciona la mayoría de las funciones disponibles en la biblioteca estándar de Java. En Android, cada aplicación Java, tanto aplicaciones base como de terceros (ya que para el sistema son iguales) corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Esta máquina virtual que como dijimos anteriormente fue creada ad-hoc por Google para Android, está diseñada de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato *Dalvik Executable (.dex)*, el cual está optimizado para hacer uso de la menor memoria posible. La máquina virtual está basada en registros, y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida en el SDK *dx*.

- **Kernel de Linux** Como todo sistema operativo Android posee un núcleo o kernel, que como hemos dicho anteriormente es Linux, y que se encarga de los servicios base del sistema, gestión de memoria, gestión de procesos, pila de la red, y modelo de controladores, es decir el kernel actúa como una capa de abstracción entre el hardware y el software. A a lo largo de las sucesivas versiones del sistema, se han ido sustituyendo los kernels por versiones más modernos de los mismos, como en el sistema operativo GNU/Linux.

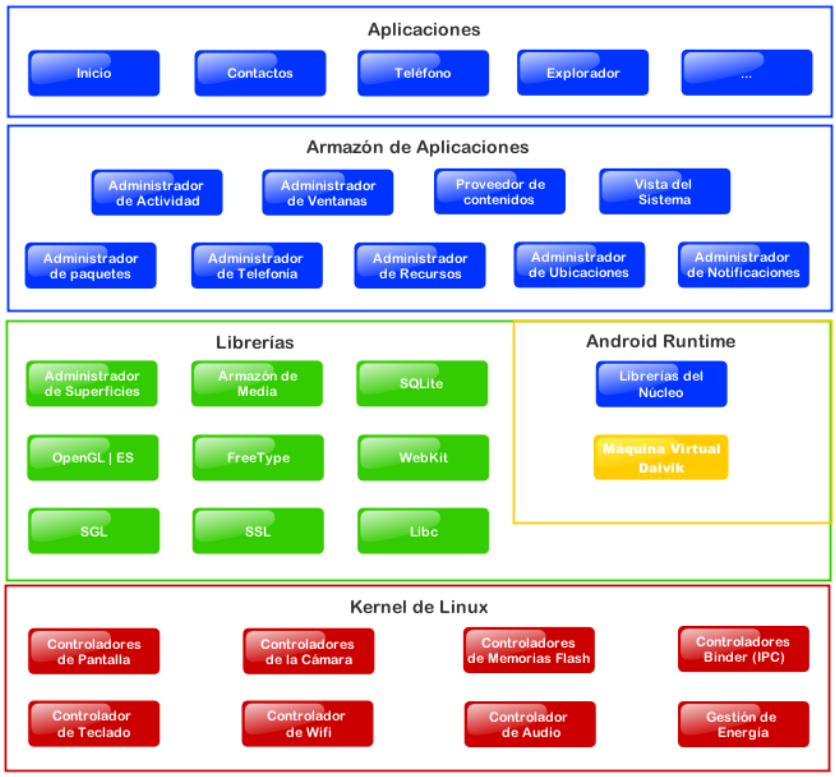

Figura 2: *Diagrama del modelo de la arquitectura Android*

Como podemos ver en la ***Figura 2***, la arquitectura Android queda dispuesta en 4 niveles, en los que cada uno abstrae a su superior del nivel inferior, es decir, la capa de *aplicaciones* utiliza el *framework de aplicaciones* pero este actúa de abstracción para el nivel *librerías* con respecto al de *aplicaciones*. Como vemos y desde nivel superior al inferior:

- **Aplicaciones.** Aquí se encuentran tanto las aplicaciones base que vienen con el sistema, como la gestión de contactos, mapas, gps, correo electrónico, etc. con las que los desarrolladores externos (terceros) desarrollan para el sistema, como vemos no hay diferencias entre ellas, todas pueden hacer uso de los mismos recursos.

- **Framework de aplicaciones.** Aquí se proveen recursos para las aplicaciones en forma de acceso a los contactos, administración de ventanas, proveedor de contenidos, etc.

- **Librerías y Android Rutime.** Aquí se ofrecen las librerías del sistema que serán accesibles a las aplicaciones por medio del uso del framework de aplicaciones (nivel superior). La máquina virtual, Dalvik, también está en este nivel.

- **Kernel de Linux.** Por último se encuentra el Kernel de Linux el cual es el encargado de ofrecer una capa de abstracción entre el software y el hardware además de todas las funciones anteriormente mencionadas, como el controlador de pantalla, controladores de la cámara, gestión de energía, etc.

3.2.4. Elementos principales en la programación con Android

Antes hemos hablado de lo que es Android como proyecto y como sistema operativo, lo que se puede hacer con el, etc. ahora vamos a ver los elementos principales que se utilizan en la programación para este sistema operativo, no se tienen por que usar todos, ni incluso ninguno, pero son los pilares en los que se fundamenta la programación en este entorno. Adelantar que en QuiteSleep hemos utilizado absolutamente todos estos elementos para las diferentes acciones que realiza la aplicación.

Indicar que estos elementos son explicados de forma completa en los libros utilizados para el desarrollo de este programa como son ***The Busy Coder's Guide to Android Development [17]*** y ***Professional Android 2 Application Development [18]***.

Así, estos son los elementos principales de la arquitectura Android para el desarrollo de aplicaciones:

- **Activities**

  La parte principal de la interfaz de usuario en Android, son las llamadas *Activities* [19], una activity es como si fuera una ventana o un *frame* en una aplicación normal de escritorio de un ordenador convencional. Por así decirlo las activities son las pantallas que aparecen en cualquier aplicación Android, cuando pasamos de una pantalla a otra, lo que estamos haciendo es pasar de una activity a otra.

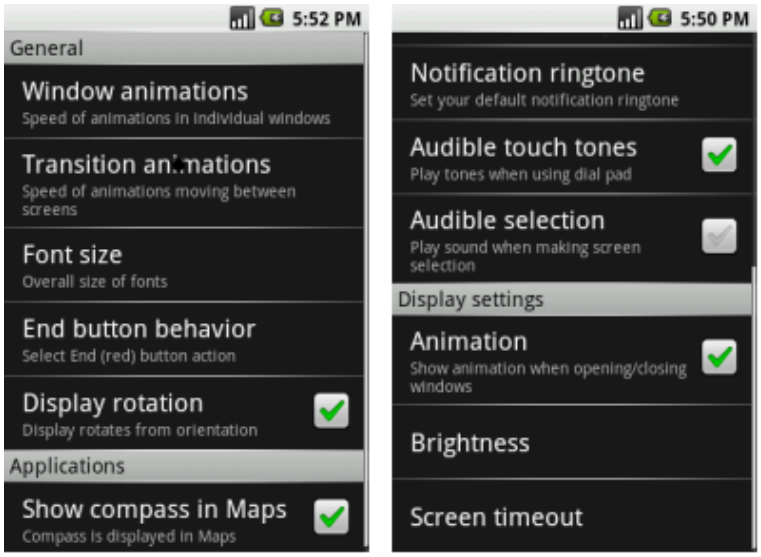

Figura 3: *Ejemplo de dos Activities diferentes*

- **Content Providers**

  Los *Content Providers* [36] [20], proveen un nivel de abstracción para cualquier tipo de dato almacenado en el dispositivo que es accesible por múltiples aplicaciones. Los datos internos del teléfono son accesibles para cualquier aplicación haciendo uso de este mecanismo. El modelo de desarrollo de Android, fomenta el poder poner los datos de cada aplicación de cada desarrollador accesible para el resto de las aplicaciones, esto se realiza construyendo tu propio content provider.

- **Intents**

  Los *intents* [21] son mensajes del sistema que son lanzados dentro del dispositivo, notificando a las aplicaciones de varios eventos, como cambios en el estado del hardware (por ejemplo cuando una SD card es insertada), datos que son recibidos (por ejemplo cuando llega un SMS), eventos de las aplicaciones (por ejemplo tu aplicación es lanzada desde el menú principal), etc. No solo se puede responder a los intents, sino que los desarrolladores pueden crear sus propios intents para lanzar otras actividades (muy utilizado) o para saber cuando se dan situaciones específicas.

- **Services**

[36] Proveedores de contenido

Las activities, content providers e intents tiene un ciclo corto de vida y pueden ser terminadas en cualquier momento, incluso sin previo aviso. Los services [22] están diseñados para mantenerse en funcionamiento en *background*, si es necesario, independientemente de la activity que lo pueda lanzar. Se pueden utilizar services para checkear actualizaciones del correo electrónico en background, o escuchar música mientras se está realizando otra acción, etc.

3.3. db4o

Aunque este proyecto utiliza la base de datos propia de Android, como es SQLite, principalmente utiliza esta base de datos en la mayoría de gestiones de acceso a los mismos.

db4o es un motor de base de datos open source orientado a objetos, sus siglas corresponden a **D**ata**B**ase **4(for) O**bjects. Está desarrollado para trabajar con desarrollos en *Java* y en *.Net*. Db4o está escrito en Java y .Net, y provee las respectivas Apis para estos lenguajes de programación. Db4o funciona en cualquier sistema operativo que soporte estos lenguajes y es ofrecido en múltiples licencias, como son **GNU GPL** [23], **the db4o Opensource Compatibility License(dOCL)**[37] y una **licencia comercial** [24] para uso en *software propietario* [25].

3.3.1. Origen

El término Base de datos orientada a objetos data sobre 1985, aunque las primeras aproximaciones en esta área son de mediados de 1970. Los primeros sistemas gestores de bases de datos orientados a objetos se lanzaron comercialmente a principios de 1990, donde fue añadido el concepto de persistencia en los lenguajes orientados a objetos.

Una segunda oleada te este tipo de bases de datos sucedió a principios de 2000, cuando las bases de datos orientadas a objetos escritas completamente en lenguajes que también eran orientados a objetos aparecieron en el mercado. Db4o es uno de estos ejemplos, escrita enteramente en Java y C# y orientada a las las mismas plataformas

El proyecto db4o fue iniciado en el 2000, por *Carl Rosenberg* y su primera versión fue lanzada en 2001.

En 2004 junto al lanzamiento comercial de db4o se crea *Db4objects Inc.* destinada a labores comerciales y de promoción de esta base de datos, así como a la gestión de su comunidad de usuarios, que desde 2001 no ha hecho más que crecer.

En 2008, Db4objects Inc, vende su producto estrella, la base de datos orientada a objetos, db4o a **Versant Corp.**[38] empresa que ya contaba con otra base de datos orientada a objetos como era Versant Object Database para Java y Versant Fast Objects para .Net. Por su parte Db4objects Inc. cambia su nombre a Servo, y pasa a dejar de lado el negocio de bases de datos, centrándose en la venta de soluciones para gestión de datos de usuarios a proveedores de comunicación

[37] http://www.db4o.com/about/company/legalpolicies/docl.aspx

[38] http://www.versant.com/

inalámbrica.

El CTO de Db4objects Inc. y responsable de la linea db4o, Carl Rosenberg pasa a trabajar también para Versant Corp. donde sigue estando a cargo de este producto.

3.3.2. Licencias

db4o hace uso de múltiple licenciamiento, orientado sobre todo a dar la máxima flexibilidad a los desarrolladores y empresas que desean utilizar db4o como base de datos para sus productos software, permitiendo tanto licenciamiento como software libre como licenciamiento para software privativo.

- **Licenciamiento como software propietario**

  Orientada al desarrollador o empresa que use db4o en su aplicación, y que quiera distribuir su trabajo sin ser software libre, es decir, que quiera distribuir su software como software privativo. También esta licencia es orientada para aquellos que no quieren hacer uso de la licencia GNU GPL (descrita a continuación) como licencia para su obra que utiliza la base de datos orientada a objetos, db4o.

- **GNU GPL v3**

  db4o hace uso de la licencia general GNU GPL v3, para uso en aplicaciones que quieran hacer uso de esta licencia y/o para uso en aplicaciones que necesiten usar esta base de datos, pero no quieran hacer frente a un desembolso económico por su licencia.

  Originalmente db4o hacía uso de la licencia GNU GPLv2, pero cambiaron a esta última versión.

  Versant Corp. en su interpretación de la GNU GPL hace mención a que el licenciamiento mediante el uso de esta licencia viene siendo como cualquier otro software publicado con esta ella, y además, comenta una serie de situaciones en las cuales el software estará basado (derivado de) en db4o (situaciones incluyentes pero no limitadas solo a estas):

  - Si compilas el software contra software de db4o.
  - Si tu software contiene referencias específicas al software de db4o.
  - Si tu software requiere del uso del software de db4o; o

- Si tu software usa una api propietaria contra el software de db4o.

En todas estas situaciones, el desarrollador o compañía deberá hacer público el código fuente en correspondencia y aceptación de la licencia GNU GPL.

- **db4o Open Source Compatibility License (dOCL)**

  db4o tiene también un tipo de licencia especial creado para db4o de forma ad-hoc. En 2007, db4o introdujo esta tercera opción de licenciamiento para poder otorgar más flexibilidad de elección a los desarrolladores y empresas para los que hasta el momento solo existían dos tipos de licenciamiento, o adquirir una licencia como software privativo o publicarlo mediante una licencia GNU GPL.

  Esta nueva licencia otorga más posibilidades de licenciamiento a los desarrolladores y empresas en proyectos free/open source que usan db4o pero que no quieren o pueden publicar su proyecto enteramente bajo licencia GNU GPL, permitiendo así licenciar el código bajo alguna de estas licencias:

  - GNU Library (or "Lesser") General Public License (LGPL), versions 2.0 and 2.1
  - Apache Software License, versions 1.0, 1.1, or 2.0
  - BSD License, as copyrighted on July 22, 1999
  - Eclipse Public License, version 1.0
  - MIT/X11 License
  - Mozilla Public License, version 1.1

  El 8/4/2010, Versant realizó un comunicado en el que se ampliaban estas licencias con la suma de:

  - Microsoft Public License (Ms-PL)
  - Microsoft Reciprocal License (Ms-RL)
  - Code Project Open 1.02 License

  Según el anuncio, Versant hizo esto para poder llegar a mas proyectos FLOSS, que por cuestiones de licenciamiento no podían hacer frente a las restricciones de la licencia GNU GPL, como:

- Novell's Mono
- Redhat's Fedora-Linux
- Eclipse (Apogee)
- Spring
- JPOX
- Apache-Lucene (Gdata)
- RSSOwl newsreader
- Arum DataEye and Funambol (SyncML)

Como dicen, con esta ampliación de licenciamiento, creen que db4o ya se podría usar en el 94 % de los proyectos FLOSS hechos en Java y .Net existentes.

Para poder hacer uso de la licencia dOCL, el desarrollador o empresa debe registrar su proyecto en la sección de Proyectos de db4o, y en su blog, y enviar un mail a su community manager, con la aceptación de los términos de la licencia dOCL.

3.3.3. ¿Por qué db4o?

Como se verá posteriormente con más detalle, db4o nos ofrece un medio para dar consistencia a nuestros datos en forma de base de datos completamente orientada a objetos con lo que la facilidad de uso en el desarrollo es muy grande, no hay necesidad de *mapear* un objeto en una fila, ni sus tipos de datos en las correspondientes columnas de una tabla, trabajar con objetos complejos es trivial y no es necesario hacer uso de herramientas que pasan del modelo orientado a objetos al relacional como ***Hibernate***[39] que requiere de librerías adicionales y pesadas así como tiempo de proceso y complejidad mucho mayores.

[39]http://www.hibernate.org/

3.4. Java Mail API

Java Mail API es una expansión de Java que facilita el envío y recepción de email desde código. Este paquete no viene con la distribución estándar de Java (Java SE) y hay que descargarlo de forma adicional. Si viene incluida, sin embargo, en la versión empresarial de Java (Java EE).

3.4.1. Características

Java Mail implementa el protocolo **SMTP** [40] los diferentes protocolos de correo **POP3** [41] e **IMAP** [42], así como los distintos tipos de conexión con servidores de correo como son **TLS**[43], **SSL** [44], autenticación con usuario y password, etc.

3.4.2. Licencias

Java Mail desde el 2 de Marzo de 2009 es software libre, alojándose el proyecto en el sitio que Sun tiene para ello como es el **Proyecto Kenai** [45] y desde el 10 de Marzo de ese año también disponible en el repositorio Maven que Sun tiene para ello.

Las licencias con las que está licenciado este proyecto son:

- **CDDL-1.0**

  Common Development and Distribution License es una licencia de software libre creada por *Sun Microsystems* y basada en *Mozilla Public License (MPL)* versión 1.1. Los archivos que usan esta licencia pueden ser combinados con otros tanto de licencias privativas como de licencias libres. Está aprobada como software libre con FSF, OSI [26] y Debian [46], pero no es compatible con la licencia GNU GPL.

  Algunos de los proyectos que utilizan esta licencia además de Java Mail son:

  - OpenSolaris
  - NetBeans

[40] Simple Mail Transport Protocol
[41] Post Office Protocol
[42] Internet Message Access Protocol
[43] Transport Layer Security
[44] Secure socket Layer
[45] http://kenai.com/
[46] http://www.debian.org/

- GlassFish
- Project DReaM

- **GNU GPL v2**

  Es seguramente con la GNU GPL v3, la licencia más utilizada en proyectos de software libre, es del tipo de *licencias libres robustas* y es **Copyleft**[47], está aprobada por la FSF (sus creadores), OSI y Debian, pero no es compatible con su evolución la GNU GPL v3.

- **BSD**

  Esta licencia fue una de las primeras licencias de software libre, su nombre viene de *Berkeley Software Distribution*, es una licencia de *tipo permisivo* y está aprobada como software libre por la FSF, OSI y Debian, es compatible con la licencia GNU GPL (la licencia BSD de tres clausulas), pero no es Copyleft.

3.4.3. ¿Por qué Java Mail?

Como hemos indicado antes con db4o, el uso de Java Mail será explicado con más detalle en secciones posteriores, pero indicar que el motivo de elegir Java Mail como librería para realizar el envío de correo electrónico es debido a que Android no permite el envío del mismo sin la autorización de forma interactiva por parte del usuario, y como veremos, eso no es lo que queremos.

3.5. ActionBarSherlock y ViewPagerIndicator

ActionBarSherlock y ViewPagerIndicator son dos proyectos FLOSS creados por **Jake Wharton** para poder utilizar los patrones de interacción *ActionBar y ViewPager* en la mayoría de las versiones de Android abstrayendo mucho la implementación que de otra forma habría que hacer si implementáramos estos patrones por nosotros mismos desde cero.

[47] Obliga a los proyectos derivados de proyectos que usen esta licencia a otorgar los mismos privilegios con los que fue recibida la obra, para así mantener la cadena siempre de igual forma

3.5.1. Características

- **ActionBarSherlock** es una librería para la implementación del patrón de interacción ActionBar, que era utilizado por multitud de aplicaciones Android, y que no fue hasta la versión de Android 3.0 cuando Google no puso de forma nativa este patrón en el SDK. Con esta librería de la misma forma que se puede crear el action bar en versiones Android 3.0 en adelante, también se puede en versiones 2.x.

- **ViewPagerIndicator** es una librería para la implementación del patrón de interacción ViewPager, este nuevo patrón fue incluido en el paquete de compatibilidad revisión 3 [48] y está disponible para versiones Android 1.6 en adelante. ViewPagerIndicator hace posible de forma mucho más sencilla e intuitiva la implementación de este patrón incluyendo recursos gráficos y diferentes modos de visualización.

3.5.2. Licencias

Ambos, ActionBarSherlock y ViewPagerIndicator están bajo la licencia **Apache License v2**[49].

Apache License v2, es seguramente, la **licencia permisiva** más importante y utilizada en el mundo FLOSS. Esta licencia, permite al usuario del software elegir como va a publicar un software hecho por el mismo que hace uso de otro bajo esta licencia, es decir un usuario que ha creado una aplicación que hace uso de una librería bajo esta licencia cuando publique su aplicación podrá elegir si publicarla bajo esta misma licencia manteniendo los derechos otorgados por el autor original de la librería, cambiar a una licencia compatible (y si se quiere y puede incluso más restrictiva) o incluso publicarla bajo una licencia privativa sin necesidad de liberar el código fuente. Esta tipo de licencias para unos es la más libre ya que da mas libertad de elección al usuario, pero para otros no lo es tanto ya que es posible romper en un cierto punto la libertad otorgada por el software originalmente a la comunidad.

[48] http://android-developers.blogspot.com/2011/08/horizontal-view-swiping-with-view pager.html

[49] www.apache.org/licenses/LICENSE-2.0.html

3.5.3. ¿Por qué ActionBarSherlock y ViewPagerIndicator?

La razón de utilizar estos dos proyectos FLOSS en vez de hacerlo desde cero, es que con estos dos proyectos nos liberamos de mucho trabajo que tendríamos que hacer para implementar estos dos patrones de interacción en la mayoría de versiones Android, también son proyectos muy utilizados, muy activos y documentados y completamente software libre con licencia compatible con la licencia GNU GPLv3 que tiene desde sus orígenes QuiteSleep, con lo que se trata de un software idóneo para nuestras necesidades.

4. Antecedentes y objetivos

4.1. Aplicaciones similares

QuiteSleep en 2010 trató de llenar un segmento en las aplicaciones Android que no estaba muy explotado, que era el de controlar las llamadas entrantes y responder de una forma u otra cuando estas llegaban.

Como pudimos ver si investigábamos un poco en los sitios web que trataban de indexar las aplicaciones del market para ofrecer una navegación mucho más intuitiva como pueden ser **AndroidZoom**[50] y **AndroLib**[51], había aplicaciones que filtraban llamadas asociadas a números concretos, y otros que filtraban según una situación previamente configurada.

Actualmente, en 2012, vemos que el panorama no ha cambiado mucho, básicamente no encontramos con las mismas aplicaciones que explicábamos en la memoria anterior, y que pasamos a explicar a continuación:

- **CallFilter**

  Esta aplicación desarrollada por la empresa japonesa *telemarks*, realiza un bloqueo de llamadas entrantes para los números de contactos individuales asociados a los contactos que tenemos en nuestra agenda personal. Es decir, en esta aplicación elegimos que números de teléfono asociados a nuestros contactos (esto se realiza de uno en uno) queremos bloquear. Una vez definimos que teléfonos vamos a bloquear, cuando nuestro móvil reciba una llamada de algún número de los considerados bloqueados y la aplicación esté activa, esta rechazará la llamada realizando la acción de **colgar** (en ingles, *hang up*), una vez haya bloqueado la llamada el teléfono pasará a estar otra vez a la espera.

[50] http://www.androidzoom.com/

[51] http://www.androlib.com/

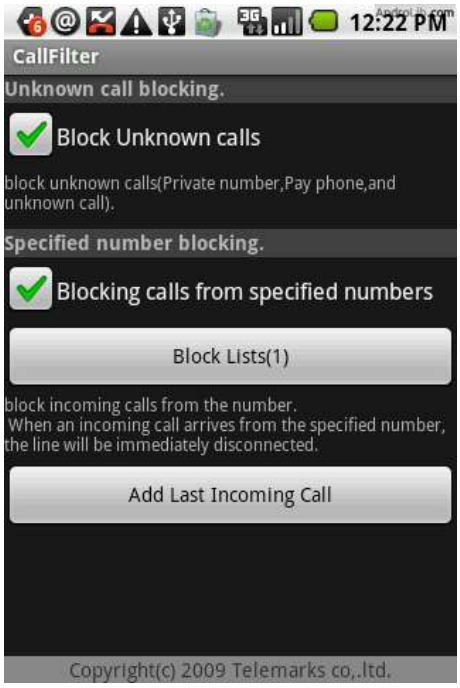

Figura 4: *Imagen de la configuración de CallFilter*

Esta aplicación tiene una serie de características, como son:

- Interfaz simple y sencilla.
- Funcionamiento muy bueno.
- Proceso para bloquear varios teléfonos de una única persona largo y tedioso.
- Únicamente en inglés.
- No tiene historial de llamadas bloqueadas.
- Gratuita.
- Software privativo.

- **Sweet Dreams**

  Sweet Dreams es una aplicación de los españoles *Inizziativa* los cuales fueron ganadores con esta aplicación del último concurso de aplicaciones de Google, el llamado *Android Developer Challenge 2 (ADC 2)*[52] realizado en verano de 2009.

[52]http://code.google.com/intl/es-ES/android/adc/

Sweet Dreams es una aplicación destinada a **silenciar** el teléfono móvil para no interrumpir tus sueños (al estilo de QuiteSleep), y que incluye varios tipos de configuraciones donde también se puede indicar que se desactiven ciertas opciones del teléfono para ahorrar batería. En cuanto a la funcionalidad parecida a QuiteSleep, podemos indicar al teléfono que se ponga en modo silencio entre dos franjas de tiempo en los días de la semana establecidos, también podemos silenciar el teléfono en según que localización nos encontremos haciendo uso del GPS del teléfono, y por último podemos silenciar el teléfono si detecta que nos encontramos en un entorno de absoluto silencio, función que realiza haciendo uso de los micrófonos del mismo.

Figura 5: *Imagen de la configuración mediante intervalo horario de Sweet Dreams*

Vamos a ver ahora una serie de características de esta aplicación:

- Es la ganadora del ADC2, con lo que atesora calidad y funcionalidad.
- Es una Aplicación 100 % española.
- Aplicación con interfaz vistosa y funcional.
- Muy configurable. Muchas opciones, puede que demasiadas, lleva tiempo configurarlas todas.

- Algunas funciones son realmente ingeniosas ¿útiles?.
- Buen funcionamiento.
- Gratuita.
- Software privativo.

- **Locale**

  La ganadora del primer ADC en 2008, Locale es una aplicación creada por alumnos de informática del **MIT**[53] en su último año de estudios, para una asignatura llamada *Building mobile applications with Android*. Una vez desarrollada la aplicación estos alumnos crearon una empresa llamada *TwoFortyFourAM* [28] en la que hacen negocio con esta aplicación. Al principio Locale se trató de una versión Beta, que era gratuita y la cual fue probada y utilizada por miles de personas, para pasar después a ser una aplicación únicamente de pago. Locale actúa de forma muy similar a Sweet Dreams, pero con mucha más funcionalidad, ya que no solo se encarga de silenciar el teléfono móvil, si no que actúa sobre todos los elementos que consumen batería del mismo, como es activar/desactivar el 3G, el WiFi, permitir notificaciones, establecer un volumen específico dependiendo de la acción, hora, día, etc. Locale cuenta con una API para poder desarrollar plugins de terceros e integrarlos con ella, para hacer así un entorno mucho más funcional.

[53] http://web.mit.edu/

Figura 6: *Imagen de la configuración mediante situaciones o sitios en Locale*

Algunas características de Locale son:

- Fue de las primera aplicaciones en aparecer y sin duda la de más calidad y funcionalidad.
- Tuvo un proceso en fase Beta largo donde pulieron todos los posibles bugs que salían para así convertirla en una aplicación muy estable.
- Múltiples acciones, no solo correspondientes con los tonos de llamada.
- Múltiples configuraciones.
- Fácil utilización, compleja, completa y sencilla.
- Ahora mismo es de pago con un precio de 10.99$ (dólares).
- Los plugins también son, en su mayoría de pago. Si queremos funcionalidad extra deberemos pagar por ella.
- Es software privativo.

Después de ver como el panorama no ha cambiado mucho, podemos seguir haciendo frente a las mismas aplicaciones que también han ido evolucionando.

4.2. Evolución de QuiteSleep (versiones)

A lo largo de estos dos años, QuiteSleep ha ido evolucionando con respecto a:

- Las peticiones de los usuarios en cuanto a nueva funcionalidad.
- La evolución de Android como sistema operativo.
- El conocimiento personal sobre la plataforma Android.

4.2.1. QuiteSleep 1.x.x

En Mayo de 2010, como hemos dicho anteriormente, se publicó la primera versión de QuiteSleep, esa versión contenía como elementos destacados, los siguientes:

- Navegación mediante pestañas (tabs).
- Gestión de contactos mediante el uso de la bbdd db4o.
- Envío de feedback al usuario mediante email y sms.
- Bloqueo de llamadas (silenciado) pertenecientes a los contactos seleccionados.
- Ayuda e información sobre la aplicación.
- Notificación para saber que la aplicación está activa.

Esta versión fue la primera y estuvo activa en los diversos markets unos 9 meses hasta que la siguiente versión de QuiteSleep fue terminada y lanzada.

4.2.2. QuiteSleep 2.x.x

Esta versión de QuiteSleep es seguramente la versión con más cambios en cuanto a la funcionalidad de la misma, esta versión fue lanzada en Febrero de 2011, y contenía las siguientes mejoras:

- Mejorado diverso funcionamiento interno.

- Nuevos tipos de configuración en el bloqueo de llamadas. Se ofrecían diferentes formas de elegir como se debía comportar la aplicación cuando recibía llamadas (no solo de los contactos en la agenda). Exactamente se ofrecía ahora estos nuevos modos de configuración:

    1. Bloquear todas las llamadas, esta opción permitía al usuario que todas las llamadas entrantes en su teléfono fueran bloqueadas sin importar si el número de la llamada pertenecía a un contacto bloqueado o no, e incluso a un contacto desconocido.
    2. Bloquear solo los contactos bloqueados, esta era la opción original que tenía las versiones 1.x.x de QuiteSleep, con esta opción solo se bloquean las llamadas pertenecientes a los contactos que tengamos definidos como bloqueados.
    3. Bloquear solo a desconocidos, con esta opción dábamos al usuario la opción de bloquear las llamadas recibidas de números desconocidos sin tener en cuenta a los contactos de nuestra agenda o los definidos como bloqueados.
    4. Bloquear desconocidos y bloqueados, seguramente esta era la opción más demandada por los usuarios, en esta opción las llamadas bloqueadas son las desconocidas (las llamadas entrantes que no pertenecen a ninguno de nuestros contactos) y las llamadas pertenecientes a los contactos que tenemos definidos como bloqueados.

- Nuevo modo de bloqueo, originalmente en versiones 1.x.x de QuiteSleep solo silenciábamos las llamadas pertinentes de forma que cuando nos llegaba la llamada, QuiteSleep la silenciaba, pero no era colgada. Elegimos silenciar la llamada en vez de colgar por dos motivos:

    1. A nuestro modo de ver era el mejor modo de realizar la acción de bloqueo, puesto que no era una acción agresiva ya que no realizábamos el efecto de colgar, que muchas veces puede ser malinterpretado.
    2. Utilizando la api dada en aquellos momentos, no era posible realizar el efecto de colgar automático, solo silenciar.

  En este tiempo hubo muchas reviews de la aplicación en el market de Google en las que los usuarios pedían que el modo de bloqueo de la llamada fuera, o al menos se pudiera

también, colgar la llamada automáticamente. En principio para versiones anteriores lo había desestimado por las razones que he dicho antes, pero para esta y debido a la demanda de usuarios pidiéndolo y que por ejemplo la aplicación *CallFilter* lo hacia y yo no sabía por que, me parecía un reto hacerlo, así que me puse manos a la obra. Como he dicho antes la api oficial no permite realizar el colgado de llamadas con lo que hubo que seguir otra estrategia, más *"hacker"*, que básicamente se trata en *overridear* la función nativa encargada de colgar llamadas en el sdk de Android, pero esto será explicado en detalle en la **sección 5.3**.

Así finalmente para esta nueva versión de Quitesleep disponíamos de dos modos de bloquear las llamadas:

- Silenciarlas de forma automática.
- Colgarlas de forma automática.

- Eliminación del botón *atrás* de diversas pantallas de la aplicación puesto que en los dispositivos Android el botón *back* es un patrón de interacción y con este botón rompemos esa navegación propia de Android.

Esta versión ha estado en el market de Google (aún continua en el de Amazon y el de Goapk) hasta Mayo de este año, cuando la versión 3.0 de QuiteSleep ha visto la luz.

4.2.3. QuiteSleep 3.x.x

Esta versión, como hemos dicho, ha sido lanzada en Mayo de este año, bajo la motivación de realizar un cambio importante en ella para la defensa de este proyecto y para adaptarla a los nuevos tiempos, ya que la apariencia de la aplicación se estaba quedando un poco obsoleta.

Los cambios no solo han sido estéticos sino también internos e importantes. A continuación vamos a contar no muy en profundidad (esto será contado más adelante) los cambios que se incluyen en esta nueva versión.

- Cambio importante en la interfaz gráfica del programa, ahora usamos dos patrones de interacción nuevos:

  1. ActionBar, una barra superior con la que se puede navegar por la aplicación y la que contiene acciones dependiendo de donde nos encontremos.

2. ViewPager, un modo de ir de una pantalla a otra de la aplicación haciendo *swipe* (deslizando) de un lado a otro. Esto sustituye a las antiguas pestañas que contaban las versiones anteriores de QuiteSleep.

   Junto con estos nuevos patrones también se han cambiado botones y estilos para hacer la interfaz visualmente homogénea.

- Con estos cambios en la interfaz se ha cambiado el modo de navegación utilizado en la aplicación.
- Refactorización del código, se ha cambiado de forma importante la estructura interna y distribución de paquetes de la aplicación así como nombres de diferentes clases, eliminación y/o mejora de código, etc.
- Nueva forma de visualizar la ayuda y la información acerca de la aplicación y el autor utilizando un nuevo widget muy útil y vistoso (SilidingDrawer).
- Fragments, los fragments son una nueva forma de mostrar el contenido en las diferentes pantallas de la aplicación en trabajo conjunto con las Activities, en esta versión se ha migrado completamente la misma al uso de fragments y activities.
- Diversas mejoras en el funcionamiento del programa, como utilizar nuevas formas de realizar el trabajo en background, etc.

Así como vemos la evolución de QuiteSleep ha sido bastante buena, con mejoras importantes a lo largo del tiempo y con releases bastante diferenciadas e identificativas, con lo que podemos ver como ha evolucionado y mejorado la aplicación a lo largo del tiempo.

4.3. Objetivos

QuiteSleep 2.0.4 fue la última versión publicada en el market, esta nueva versión debido a los numerosos cambios que teníamos en mente es la versión 3.0. Para esta nueva release los objetivos principales, respecto a la nueva implementación son:

- Nuevo branch en la forja del proyecto para no cambiar la versión anterior de QuiteSleep.
- Refactorización del código, cambio de la estructura del proyecto.

- Rediseño de la interfaz de usuario siguiendo los últimos patrones de interacción que las Android guidelines nos ofrecen.
- Customización de elementos visuales.
- Adaptación de la aplicación al uso de Fragments.
- Cambio en diversa funcionalidad interna de la aplicación.

4.3.1. Nuevo branch en la forja

QuiteSleep se encuentra alojado en la forja de **Google Code**[54] ahí este proyecto ha estado desde su primera release, para esta nueva release se necesitaban los proyectos ActionBarSherlock y ViewPagerIndicator, así que se decidió añadirlos a la forja. También se creo un proyecto aparte para esta release a lo que código de QuiteSleep se refiere para no interferir en la anterior versión que era totalmente estable.

4.3.2. Refactorización de código

La estructura anterior del proyecto no era del todo ordenada y satisfactoria así que se decidió refactorizar el proyecto original (estructura y nombre de paquetes) para esta nueva versión, así como supresión de código no utilizado, modificación del mismo en pequeñas zonas, etc.

4.3.3. Rediseño de la interfaz de usuario

Quizá el aspecto más importante de esta release es el aspecto visual que aunque para ello hace uso de cambios en multitud de aspectos internos en la aplicación, el usuario final lo que va a ver es un rediseño de la interfaz de usuario. Este cambio es necesario para adaptar la aplicación a los nuevos patrones de interacción que ahora tiene Android, de no actualizar esto, la aplicación corría el riesgo de quedarse un poco anticuada tanto visualmente como funcionalmente con multitud de código que las diferentes versiones del SDK de Android iba poniendo en estado *deprecated*[55].

[54] http://code.google.com/p/quitesleep/

[55] Código que es perfectamente usable, pero es marcado para indicar que en las próximas versiones va a desaparecer.

4.3.4. Customización de elementos visuales

En las anteriores releases de QuiteSleep se utilizaban los botones por defecto de Android, para esta nueva versión y para *dar un poco de color* a la aplicación se decidió sustituir los botones por defecto, por otros customizados y seguir así con la estética que da el ActionBar y el ViewPager. Se crearon botones y se modificaron los mismos con la herramienta *draw9patch*[56] que se utiliza para que los recursos visuales puedan ser escalados sin perder la calidad de los mismos.

También en este objetivo, quizá incluso más importante que los botones sea la customización del ActionBar para que la misma esté acorde al nuevo estilo visual de la aplicación.

4.3.5. Fragments

Fragments seguramente fue el cambio más importante que la versión de Android 3.0 Honeycomb trabajo en el SDK, y es una vuelta de tuerca a lo que en Android se denominan Activities, las activities son por así decirlo las diferentes pantallas que el usuario va a ver e interactuar con ellas, Fragments lo que propone es que en las actvities podamos crear contenido, sustituirlo, borrarlo, etc. sin tener que utilizar nuevas activities, siendo este contenido los propios fragments. Esto es muy útil para poder visualizar la información de forma diferente en los modos portrait y landscape de móviles y tablets. También es obligatorio utilizarlo por ejemplo con el ViewPager.

4.3.6. Cambio en aspectos funcionales

En anteriores versiones de QuiteSleep era común bloquear la interfaz gráfica de usuario mientras que, por ejemplo, se cargaban datos desde la base de datos, en este nueva versión se ha cambiado muchos de estos escenarios procesando datos sin por ello bloquear la interfaz gráfica de usuario (UI Thread) y por tanto *bloquear al usuario*.

[56] http://developer.android.com/guide/developing/tools/draw9patch.html

5. Descripción informática

5.1. Metodología a seguir

En todo proyecto de desarrollo software se realizan una serie de actividades (ciclos) entre la idea inicial u original del mismo y el producto final, que obtenemos cuando está totalmente acabado, y listo para su entrega.

El plan de trabajo que se ha llevado a cabo para la realización de este proyecto ha consistido en la utilización de una determinada metodología o modelo de desarrollo, la cual establece el orden en el que se llevan a cabo las tareas pendientes de realizar durante toda la duración del proyecto.

El modelo de desarrollo escogido ha sido el **modelo de desarrollo en espiral**. Este tipo de metodología o modelo está indicado para proyectos que varían en el tiempo atendiendo a los nuevos requisitos o a la modificación de los viejos.

El modelo en espiral se basa en la necesidad de separar el comportamiento final en varias actividades más sencillas, que conjuntamente, formarán el comportamiento final del producto. Este modelo se caracteriza por la realización de fases o tareas las cuales son:

- Análisis de Requisitos
- Análisis del Riesgo
- Desarrollar, verificar y validar (probar)
- Planificar

Lo verdaderamente importante de esta metodología es que nos aporta cierta flexibilidad en cuanto a posibles cambios de los requisitos iniciales y que nos ofrece, además, la posibilidad de ir construyendo prototipos que se producen como fase final de cada ciclo y los cuales nos servirán para comprobar si hemos cumplido con los objetivos o requisitos que nos propusimos al comienzo del ciclo, para luego evaluarlo en la fase final de planificación para poder pasar o no a la siguiente actividad.

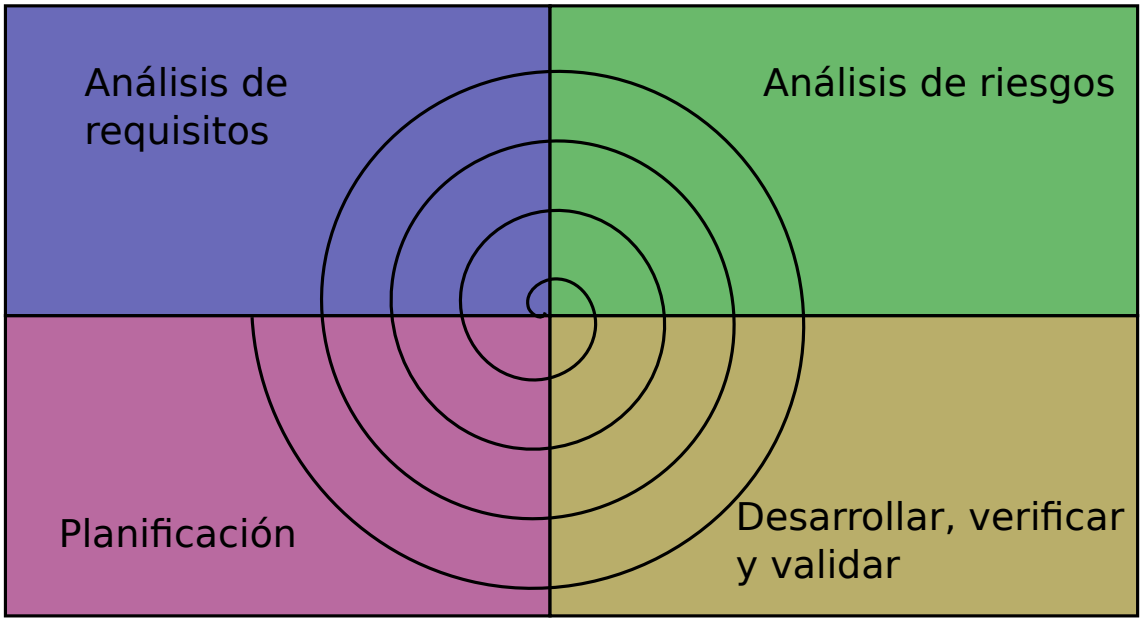

Figura 7: *Modelo en Espiral*

Este modelo en espiral, fue adaptado a nuestro propio proyecto para que fuera totalmente orientado a el:

- **Análisis de Requisitos (Comunicación con el cliente)**

  Como cualquier otro proyecto, sea del orden que sea, el nuestro también tiene captura de requisitos y su correspondiente análisis. En esta fase el cliente especifica los requisitos al inicio del ciclo de la espiral, esto se corresponde con la realización de una reunión donde el tutor del proyecto nos da una serie de pautas u objetivos a cumplir o hacia donde debemos orientarnos. Una serie de requisitos, con los que se obtiene una idea general del objetivo del proyecto en cada ciclo. Aclarar, que la planificación **inicial** o previa solo se realiza una vez, no obstante volveremos a la captura de requisitos tantas veces como sea necesario para obtener nuevos requisitos o modificar los que anteriormente ya estaban fijados.

- **Análisis de Riesgos**

  Analizamos los requisitos dados por el cliente, estudiamos los riesgos potenciales y seleccionamos una o varias alternativas para reducir o eliminar riesgos. También en esta fase se evalúa el tiempo que necesitaremos para realizar cada ciclo, si es viable la actividad a realizar, y lo más importante si merece la pena. Así, nos disponemos a diseñar la

estructura de nuestra aplicación definiendo el paradigma de programación más indicado, el lenguaje de programación que más se adapte al paradigma escogido, los pertinentes diagramas para aclararnos las ideas y tenerlo todo más claro, etc.

- **Desarrollar, verificar y validar**

  Una vez que ya tenemos el diseño inicial nos disponemos a codificar la aplicación adaptando las ideas del cliente en los requisitos iniciales a lo que el programa realizará para obtener esos objetivos.

  Una vez desarrollada parte de la aplicación (o mientras la desarrollamos), la tenemos que verificar realizándole una batería de pruebas para ver si lo realizado hasta el momento se adapta a lo que teníamos diseñado, y por tanto si se adapta a las necesidades del cliente.

- **Planificar la siguiente iteración**

  Una vez llegados a esta fase revisamos todo lo hecho hasta el momento, y con ello, decidimos si continuamos con la actividad siguiente y la planificamos, o si debemos pararnos y analizar alguna fase anterior al no haberla superado como esperábamos.

Así, una vez completada una actividad volveremos tantas veces como sea necesario a realizar todas las fases para llevar a buen fin el desarrollo del proyecto. Todas estas fases y cada actividad (iteración), con sus respectivos prototipos, serán validadas, corrigiendo errores, ampliando y modificando requisitos, etc. para la correcta conclusión del proyecto.

5.2. Fases del Proyecto

Hemos dividido el proceso de creación del proyecto en iteraciones, o ciclos, con lo que una iteración completa no es la realización del proyecto en su forma total, sino la finalización de una meta propuesta en una etapa anterior y por tanto el intento de desarrollar ese propósito. Para cada iteración se realizaron diseños, bocetos, se creó código de ejemplo que para etapas maduras se ha desestimado, etc.

5.2.1. Fase 0: Planificación

- **Análisis de requisitos**

Para empezar en esta fase, se mantuvo una reunión inicial (una serie de correos electrónicos, al encontrarme en el extranjero) con el tutor para ver que es lo que se iba a tratar de conseguir, es decir, el objetivo final del proyecto.

- **Análisis de riesgo** En esta etapa no debemos evaluar mucho ya que es una etapa de toma de contacto y no hay riesgos que asumir ni *deadlines*[57] que cumplir.

- **Desarrollo, verificación y validación**

  En esta etapa no hubo nada de desarrollo, solo fue un estudio y verificación del estado del proyecto en el momento, que cambios se podían hacer cuales no, cuales más fáciles, cuales más difíciles, etc.

- **Planificación**

  Para la siguiente fase se piensa como se va a desarrollar el proyecto de forma global, y se divide este siguiendo los objetivos que mencionamos anteriormente en la **sección 4.3**.

5.2.2. Fase 1: Creación de un nuevo branch

- **Análisis de requisitos**

  Para esta etapa lo que queremos hacer es añadir las librerías ActionBarSherlock y ViewPagerIndicator y aislar el proyecto anterior de esta nueva release creando un nuevo branch donde estarán los cambios de esta nueva versión.

- **Análisis de riesgo**

  El mayor riesgo que tenía era, debido a que no soy experto con **Mercurial**[58], perder la historia del repositorio de la anterior versión de QuiteSleep. Con lo que debía tener cuidado con ello si no quería perder toda la historia de commits anterior.

- **Desarrollo, validación y verificación**

  Se procedió a importar los proyectos ActionBarSherlock y ViewPagerIndicator y se procedió a crear el branch par la nueva versión, finalmente todo fue de forma correcta siguiendo la documentación que Mercurial tiene.

[57] tiempos límite

[58] http://mercurial.selenic.com/

- **Planificación**

  Evaluamos lo que hemos realizado hasta el momento y vemos que la progresión es correcta, la planificación que teníamos para esta fase y que deja todo listo para empezar con la implementación de la nueva release está lista, con lo que damos el visto bueno para continuar con la siguiente iteración del proyecto.

5.2.3. Fase 2: Refactorización del proyecto

- **Análisis de requisitos**

  En esta fase lo que queremos es realizar una reorganización de los paquetes que componen el proyecto, así como una reorganización de las clases, renombre de algunas de ellas, movimiento de diversas funciones o variables estáticas a diferentes clases, etc.

  Dentro de la refactorización también vamos a efectuar la supresión de diferentes clases que ya no necesitamos o cambio de alguna funcionalidad básica que creemos ya no es necesaria o que se puede mejorar de alguna forma sin necesidad de cambiar en exceso el código.

- **Análisis de riesgo**

  Esta fase es crucial para el desarrollo del proyecto, puesto que aquí vamos a realizar una refactorización importante de la estructura del proyecto y de nombres de diferentes clases, reorganización de funciones, variables estáticas, etc.

  Gracias al entorno de desarrollo **Eclipse**[59] la tarea de la refactorzación es mucho más sencilla que si no tuviéramos esta herramienta o parecidas. Con Eclipse podemos mover y renombrar paquetes, clases, etc. de forma mucho más sencilla e intuitiva.

  Así, no esperamos muchos problemas si todo lo hacemos de forma organizada.

- **Desarrollo, validación y verificación**

  En el desarrollo de esta fase como he dicho anteriormente he utilizado el entorno de desarrollo Eclipse que facilita de manera importante la refactorización y reorganización del código fuente.

[59] http://www.eclipse.org/

Aquí he seguido unos pasos:

- **Reorganización de los paquetes del proyecto.** En esta primera parte se pasó a la reorganización de los paquetes que conforman el proyecto. En releases anteriores de QuiteSleep los paquetes estaban agrupados casi de forma individual y con como máximo dos niveles de profundidad, con esta refactorización se reorganizaron los paquetes involucrados en la interfaz gráfica de usuario, tareas en background, base de datos y acceso a los mismos, notificaciones, gestión de logs, etc.
- **Gestión de clases, funciones y variables estáticas.** En esta segunda parte se pasó a la reorganización de alguna de la funcionalidad básica de alguna clase en concreto y de las variables estáticas utilizadas como por ejemplo las utilizadas en diversas interfaces que solo estaban creadas para ello.
- **Creación de la clase QuiteSleepApp**. Esto ha sido algo que he aprendido durante mi último año trabajando profesionalmente en el desarrollo Android, y es algo muy útil, se trata de crear una clase que extienda de la clase Application de Android, esta clase debe ser especificada en el manifest, y lo que hace es que será la primera clase ”lanzada“ cuando lancemos la aplicación con lo que en ella podemos realizar instanciaciones de objetos siguiendo el modelo *singleton*[60] para que tengamos acceso a diversa funcionalidad durante toda la vida de la app y desde cualquier clase de la misma sin necesidad de realizar nuevas instanciaciones de objetos y gasto de memoria. La funcionalidad principal de esta clase es crear un método que inicializa y provee un método para obtener el *application context* de nuestra aplicación, acceso a las *user preferences*, etc.

Una vez terminado el desarrollo de esta fase, comprobamos que la aplicación sigue consistente y no se ha *”roto“* con lo que damos por bueno todos estos cambios.

- **Planificación**

  En esta fase hemos realizado la reestructuración de nuestro proyecto, realizado diversos cambios en clases concretas, añadido la clase QuitesleepApp, renombrado de clases, etc.

[60] `http://en.wikipedia.org/wiki/Singleton_pattern`

con lo que ahora mismo el proyecto está listo para pasar a la siguiente fase y empezar la implementación propiamente dicha de la nueva release.

5.2.4. Fase 3: ViewPagerIndicator

- **Análisis de requisitos**

  En esta fase queremos realizar el cambio del anterior modelo de navegación entre la diferente funcionalidad que posee la aplicación, esto es, entre las acciones de:

  - Contactos.
  - Horario (planificación).
  - Configuración.
  - Histórico (logs).

  En versiones anteriores de QuiteSleep, la navegación se hacía siguiendo el modelo de *pestañas o tabs* utilizado en muchas aplicaciones no solo móviles sino también de escritorio, desde siempre este modelo de navegación se ha venido utilizando en las interfaces de usuario.

Figura 8: *Ejemplo de navegación antigua en QuiteSleep*

En la **Figura 8** podemos ver como era la navegación anterior en QuiteSleep.

Ahora con los dispositivos táctiles donde el usuario puede realizar diversos gestos en las aplicaciones para ir a determinadas partes de la misma, es más natural realizar gestos que directamente *pinchar* a donde queremos ir. Aquí entra este nuevo modelo de interacción donde el usuario puede tanto pinchar en las diferentes pestañas donde quiere ir, siguiendo el modelo tradicional, o de forma más intuitiva, arrastrar o deslizar (swipe) con un dedo la pantalla de lado a lado horizontalmente.

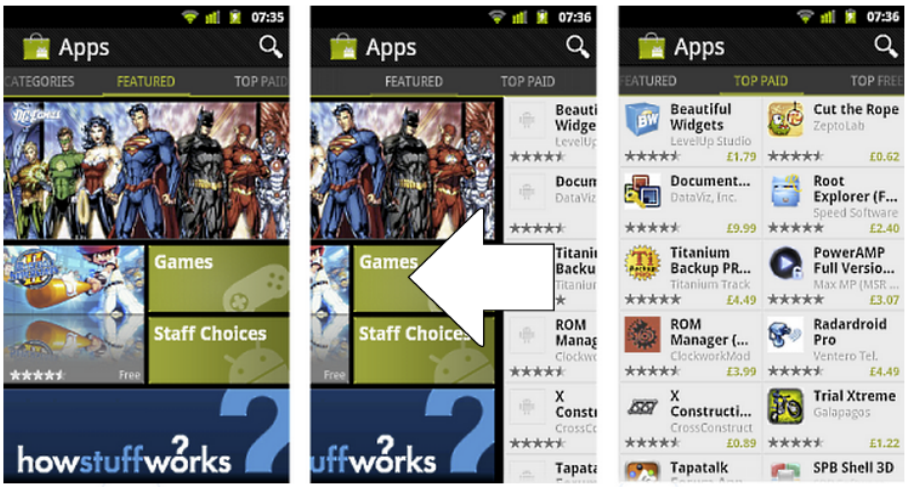

Figura 9: *Ejemplo de navegación haciendo swipe*

En la **Figura 9** podemos ver un ejemplo de lo que decimos, haciendo swipe o deslizando el dedo sobre la pantalla de forma horizontal, podemos ir de una a otra pantalla de la aplicación de forma más intuitiva.

Así que este nuevo modelo de navegación explicado es lo que queremos introducir e implementar en esta fase.

- **Análisis de riesgos**

  El riesgo que corremos aquí a la hora de implementar este nuevo modo de navegación es que vamos a romper la anterior forma de interacción con el usuario a la hora de moverse por las diferentes partes de la aplicación, con lo que la aplicación no va a ser funcional hasta que terminemos esta fase.

  En cuanto a romper también el antiguo modo de navegación, creemos que no es un problema, ya que este modo de navegación poco a poco se está introduciendo en la mayoría

de las aplicaciones de calidad que siguen los patrones de interacción de Google, como su propia aplicación *Google+*[61], la aplicación de audio de Google *Google Play Music*[62] y muchas otras, con lo que este cambio, se puede decir, que no es nuevo para el usuario Android.

Otro de los riesgos es que desde esta fase, y porque es requisito para la utilización del ViewPagerIndicator, empezaremos a cambiar Activities por FragmentActivities o directamente por Fragments, con lo que el cambio anteriormente mencionado en la **sección 4.3** a propósito del cambio de Activities por Fragments se va a hacer de forma continuada a lo largo de toda la remodelación de la interfaz gráfica y no como una tarea concreta alejada y aislada de esto (en la última iteración se explicará en detalle estos cambios realizados en cuanto a activities y fragments).

- **Desarrollo, verificación y validación**

  El desarrollo de esta fase se realiza siguiendo la documentación que la propia librería tiene en su sitio web, sin demasiados problemas:

  - Se crea la estructura del ViewPager en el xml del layout principal.
  - Se crea el objeto ViewPager que es el objeto que contiene la funcionalidad de poder desplazarnos haciendo swipe.
  - Se crea el objeto PagerIndicator que es el encargado de mostrar visualmente en que parte de la app nos encontramos (los títulos).
  - Se cambian las anteriores activities que contenían la funcionalidad de la aplicación, como eran los contactos, horario, configuración e historial por fragments.
  - Se cambia la clase Main que era la activity original que extendía de *TabActivity*, para que ahora pueda albergar los fragments anteriormente creados que ahora albergan toda la funcionalidad de las diferentes partes de la aplicación. Para ello se crea la clase *BaseFragmentActivity* que extiende de *SherlockFragmentActivity* y que será la clase de la que todas las antiguas activities extenderán, para ahora ser *FragmentActivities* y poder compartir futura funcionalidad común como es el uso del ActionBar.

[61] https://play.google.com/store/apps/details?id=com.google.android.apps.plus
[62] https://play.google.com/store/apps/details?id=com.google.android.music

Finalmente para dar acabada esta iteración, comprobamos como todas *las piezas* encajan de forma conjunta obteniendo la funcionalidad que deseábamos al principio y que la aplicación vuelve a estar totalmente funcional.

Es seguro que las diferentes clases que hemos creado serán cambiadas pronto para poder trabajar de forma conjunta con el patrón de interacción ActionBar.

- **Planificación**

  Para finalizar esta fase, pensamos en como debemos abordar la siguiente, en la que tendremos que trabajar para que los cambios que realicemos en ella tengan el mínimo impacto en la funcionalidad creada aquí y por tanto no perder demasiado tiempo en subsanar fallos.

5.2.5. Fase 4: ActionBarSherlock

- **Análisis de requisitos**

  En esta fase vamos a implementar el patrón de interacción ActionBar haciendo uso de la librería open source antes mencionada ActionBarSherlock, nuestra meta es esa, que todas las *pantallas* que forman la aplicación hagan uso del ActionBar y que sea posible su uso para la navegación que queremos ofrecer.

  Para que veamos una muestra de que es lo que queremos conseguir con el ActionBar, en la **Figura 10** podemos ver una action bar con el logo de una aplicación, una serie de tabs, y una serie de botones, uno para la cámara y otro, desplegable, que contiene más opciones.

Figura 10: *Ejemplo de una ActionBar*

En nuestro caso la ActionBar que queremos realizar solo contendría el icono de la aplicación y una serie de botones dependiendo de la pantalla en la que nos encontremos.

Con el cambio al patrón ActionBar queremos también respetar los estándares de Android con este patrón y que las anteriores versiones de QuiteSleep no respetaban con respecto

a las imágenes utilizadas en las acciones que se mostraban al presionar el botón menú y que ahora se corresponden con las mismas en el ActionBar, eran imágenes en color (lo normal es que sean en escala de grises). Así que como tarea final en esta fase vamos a cambiar también las imágenes (recursos/assets) utilizadas para las acciones.

- **Análisis de Riesgos**

  En esta fase el mayor reto que tenemos es la de que toda pantalla de la aplicación haga uso del ActionBar pero que cada una guarde *su personalidad* es decir que la funcionalidad que antes tenían muchas pantallas al pulsar la tecla *menú* ahora esa funcionalidad deberá estar integrada en la propia ActionBar.

  Otro posible riesgo que tenemos es el de los posibles problemas que nos pueden surgir al usar ahora la librería ActionBarSherlock con las pantallas que ya tenemos implementadas y que usan el ViewPagerIndicator, en teoría y al ser desarrolladas por el mismo desarrollador ambas librerías no deberíamos tener muchos problemas, pero debemos tener esto en mente.

  Para el cambio de recursos para las diferentes acciones en el ActionBar no esperamos muchos problemas puesto que conocemos la existencia del **Android Asset Studio**[63] que es un sitio web (front end de un proyecto de Google llamado android-ui-utils) que está indicado justamente para eso, para crear los assets necesarios para una aplicación Android respetando los estándares y *guidelines* de Google.

- **Desarrollo, verificación y validación**

  El desarrollo lo comenzamos adaptando la clase que habíamos creado en la anterior fase *BaseFragmentActivity*, para que ahora pueda estar integrado el ActionBarSherlock en ella de forma que todas las clases que extiendan de ella tengan acceso, haciendo uso de la herencia, a la funcionalidad del padre y por ello a la propia ActionBar.

  En esta clase creamos el ActionBar, hacemos que el resto de clases que anteriormente eran activities, extiendan de ella, y ya en ellas hacemos que cada una extienda el ActionBar añadiendo la funcionalidad correspondiente, por ejemplo en pantallas de primer nivel, como la que ahora engloba el ViewPager, el ActionBar solo mostrará el icono de la

[63] http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

aplicación, pero por ejemplo en la pantalla de *añadir contactos* que es de segundo nivel de profundidad, esta pantalla deberá tener el ActionBar con la posibilidad de volver hacia atrás con la pulsación sobre el icono de la aplicación (con el añadido de un indicador de atrás para que el usuario sea consciente de ello), y que además se visualice en el extremo derecho del ActionBar un icono para utilizar la funcionalidad de *añadir todos los contactos* que antes se realizaba con la pulsación del botón menú, y que ahora será con la pulsación sobre el mismo botón pero del ActionBar.

Para seguir con la descripción de la funcionalidad cambiada en las diferentes pantallas de la aplicación:

- Pantallas de **Contactos, Horario y Configuración**, estas pantallas tendrán un ActionBar simple solo mostrando el icono de la aplicación.
- **Historial**, esta pantalla tendrá ahora en la esquina superior derecha dos botones para mantener la funcionalidad que tenía anteriormente, la de *actualizar el contenido*, y *borrar el contenido*, añadiendo también un *progress loader* que no deja de ser un widget animado utilizado cuando se esta realizando una carga de datos (el historial) y que el usuario pueda ver que la aplicación está trabajando.
- **Añadir y Editar/Borrar contactos**, estas pantallas están a un nivel inferior de profundidad con lo que el ActionBar debe dar la opción al usuario de ir hacia atrás pulsando el botón de la aplicación y mostrando que se puede hacerlo con un indicador en el. Estas pantallas también tienen:
  - Añadir contactos, la opción de añadir todos los contactos.
  - Editar/Borrar contactos, la opción de borrar todos los contactos.
- **Detalles del contacto**, únicamente la opción de volver atrás.
- **Editar SMS/Email**, tienen la opción de ir hacia atrás.
- **Tipo de bloqueo**, tiene la opción de volver atrás.

Como vemos, estas son todas las diferentes pantallas que tenemos en la aplicación, como fue bastante bien diseñada en su origen para evitar tener una gran profundidad, la navegación con el ActionBar es natural y sencilla como también lo es el uso del botón *atrás* de Android.

Una vez terminada la implementación, tuvimos un problema en el que al tener que extender del tema propio del ActionBarSherlock para que la aplicación haga uso de el, por defecto el ActionBarSherlock y el ViewPagerIndicator no se veían de forma correcta, así hubo que crear un *tema* propio, extenderlo de uno del ActionBarSherlock y adaptarlo para que también funcionara de forma conjunta con el ViewPagerIndicator, y finalmente se consiguió.

Para dar por acabada esta fase vemos que todo funciona de forma correcta, tanto el ActionBar como el ViewPager que podría verse afectado por los numerosos cambios que hemos hecho, pero que como vemos, todo funciona de forma correcta.

Para finalizar esta fase, vamos a cambiar, como hemos dicho anteriormente, los recursos utilizados para las diferentes acciones localizadas en el ActionBar, así, utilizamos el sitio web mencionado anteriormente *Android Asset Studio* indicando los assets que necesitamos, estilo, etc. y este nos generará los diferentes recursos necesarios para las diferentes densidades de pantalla, que deberemos ponerlos en */res/drawable/drawable-XXX*.

Una vez creados todos los recursos y comprobado que todo funcionan de forma correcta, creemos que esta fase ya se puede dar por finalizada.

- **Planificación**

  Con la realización de esta fase hemos finalizado el objetivo más importante de los que nos propusimos, la aplicación con el uso de los modelos de interacción ActionBar y ViewPager está ahora adaptada a los nuevos tiempos y guidelines de las nuevas versiones de Android, y aunque ahora mismo, la apariencia y los componentes no están customizados, sino que usa temas y colores por defecto, la aplicación está totalmente adaptada.

  Así, podemos dar entonces esta fase por finalizada y para la siguiente fase, pensamos ahora en un componente y funcionalidad que teníamos en el aire que no es otro que que hacer con las secciones de **Ayuda** y **Acerca de**.

5.2.6. Fase 5: Ayuda y Acerca de

- **Análisis de requisitos**

  Una de las cosas que tenía dudas sobre qué hacer era respecto a la **ayuda** y **acerca de** que

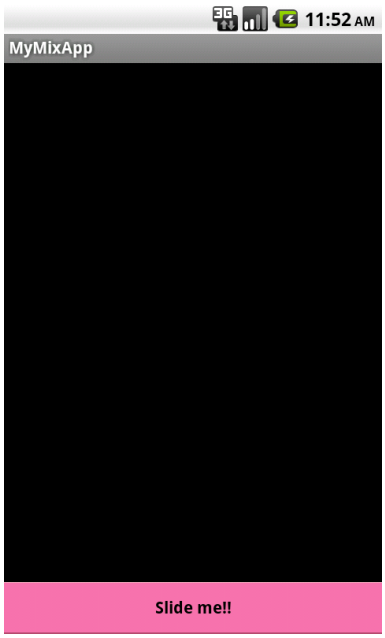

Figura 11: *SlidingDrawer por defecto.*

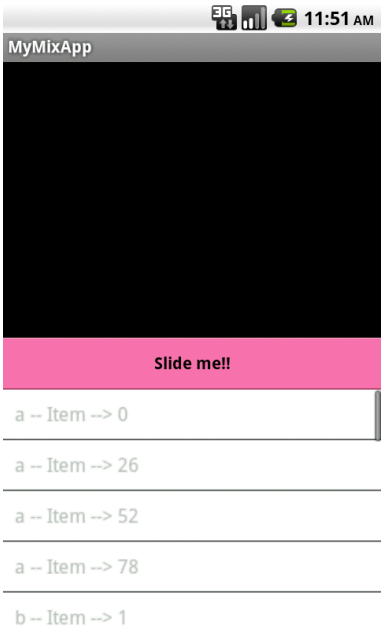

Figura 12: *SlidingDrawer mostrando parte del contenido.*

la versión anterior de QuiteSleep tenía, normalmente las aplicaciones Android no tienen ayuda y el acerca de suele estar integrado como una opción dentro de las *Preference settings* de la aplicación, pero QuiteSleep no hace uso de las preference settings, con lo que una de las opciones era eliminar tanto la ayuda como el acerca de. Pero por otra parte, en algunas reviews de la aplicación la gente me indicaba que la aplicación al principio era un poco difícil de configurar, al tener multitud de opciones, con lo que esto era un punto a favor para mantener la ayuda. Así que finalmente pensé en que ya que sabia utilizar el ViewPagerIndicator para desplazarte entre diferentes pantallas mediante un gesto (swipe), una buena idea era mostrar la ayuda y el acerca de, de forma conjunta utilizando el ViewPager. Ahora, la cuestión era ¿y donde lo mostramos? descartado el ponerlo como una opción más del action bar, puesto que me resultaba redundante que todas las ventanas lo tuvieran, pensé en que sería buena idea utilizar el componente **SlidingDrawer** que es un componente que sirve para mostrar contenido que por defecto está oculto y solo es visible un componente gráfico con el que si lo arrastramos hacia arriba (se puede configurar para que se abra de otra forma) podemos ver ese contenido.

En la **Figura 11 y 12** podemos ver como es y funciona el componente SlidingDrawer.

Teniendo entonces el ViewPager y el SlidingDrawer, la opción idónea es poner el contenido de la ayuda y acerca de en un ViewPager, y este ViewPager dentro del SlidingDrawer, con lo que el usuario siempre podría acceder a esta información desde las partes principales de la aplicación, sin ser un elemento intrusivo y de forma *diferente* a lo que viene siendo tradicional.

- **Análisis de riesgo**

  El principal riesgo de esta fase es la inclusión de estos elementos en la clase Main. Esta clase como hemos dicho anteriormente implementa el ViewPager que es utilizado para desplazarnos entre las diferentes opciones de la aplicación y hereda de una clase (BaseFragmentActivity) que contiene la inicialización del ActionBar. Con lo que ahora queremos sumar un elemento más.

- **Desarrollo, verificación y validación**

  El desarrollo de esta fase y funcionalidad es relativamente sencillo, para hacerlo seguimos estos pasos:

  - Definición del SlidingDrawer en el layout principal de la clase Main.
  - Definición dentro del componente SlidingDrawer del ViewPager y el Indicator deseado para mostrar el contenido.
  - Implementación en la clase Main del SlidingDrawer, el adapter encargado de albergar el contenido (la ayuda y el acerca de), el propio ViewPager y el Indicator concreto que para esta ocasión elegimos uno basado en círculos con los que podemos ver en que posición (página) nos encontramos del contenido.

  Una vez que tenemos todo esto implementado pasamos a las tareas de verificación y validación comprobando que lo que queríamos conseguir funciona como debe y que no hemos cambiado ninguna funcionalidad anterior que ya era correcta.

  Efectivamente todo está bien, el SlidingDrawer funciona bien mostrando el contenido de forma correcta haciendo uso del ViewPager, así que pasamos a validar esta fase.

- **Planificación**

Con esta fase damos por terminado lo que a elementos funcionales importantes se refiere. Con la esta fase, podemos dar por concluida la aplicación en cuanto a elementos de interacción, hemos cambiado la aplicación utilizando los patrones ActionBar y ViewPager y cambiando ahora la forma en la que el usuario puede acceder a la *ayuda* y la sección de *acerca de*.

Para la siguiente iteración (fase) vamos a pasar a dar los últimos retoques en cuanto al aspecto visual de la aplicación se refiere como son botones, colores, estilos, etc.

5.2.7. Fase 6: Customización de elementos visuales

- **Análisis de requisitos**

  En esta fase queremos realizar los últimos ajustes visuales de la aplicación para que podamos tener un *look and feel* acorde a nuestros gustos y necesidades.

  Para ser más concretos queremos:

  - Cambiar los botones utilizados en toda la aplicación que usaban el estilo por defecto de un botón Android. Dentro de estos botones están tanto los botones convencionales **Button**[64] e **ImageButton**[65] y el botón de estado llamado **ToggleButton**[66].
  - Cambiar el estilo por defecto del ActionBar para que se adapte al resto del estilo de la aplicación.

- **Análisis de riesgo**

  El mayor riesgo que podemos tener en esta fase es que *gastemos* mucho tiempo en ella realizando pequeños ajustes en los colores (tonalidades) que queremos usar en nuestra aplicación. Por ejemplo que elijamos un color concreto pero no nos guste y lo queramos sustituir por otro, o que no *"encaje"* con el resto de los colores que usamos, etc.

  También la creación de botones puede suponer un problema ya que deberemos hacerlos a mano utilizando algún programa de diseño o bien intentando encontrar alguna otra fuente donde la creación de los mismos sea más sencilla (no somos diseñadores).

[64] http://developer.android.com/reference/android/widget/Button.html
[65] http://developer.android.com/reference/android/widget/ImageButton.html
[66] http://developer.android.com/reference/android/widget/ToggleButton.html

Una vez tengamos los botones, tendremos que aplicarles unos cambios lo que podríamos llamar **nine-patch***earlos* que no es otra cosa que utilizando la herramienta **draw9patch** que el entorno Android nos ofrece, marcar las zonas de los botones que queremos que sean *estirables* para que podamos aplicarles el ancho y alto que queramos sino que pierdan calidad en ello al ser *estirados*.

El ActionBar que tenemos en nuestra aplicación hasta el momento utiliza el estilo **Holo Dark** que viene por defecto en la librería y que es necesario usar en la aplicación para poder usar el ActionBar, ahora queremos cambiar ese diseño por uno propio acorde al resto de los cambios, con lo que puede que esto también nos lleve mas tiempo del pensado.

- **Desarrollo, verificación y validación**

  En las versiones anteriores de QuiteSleep los botones que utilizaba la aplicación eran los botones con el estilo y diseño por defectos utilizados en el entorno Android, esto es, gris en el estado por defecto y naranja cuando el foco se centraba en ellos (utilizando por ejemplo el trackball que tienen algunos dispositivos) y naranja también cuando se presionaba.

  Para esta nueva release, mi idea inicial era basar el tema de la aplicación (los colores utilizados) en azules y negros aprovechando el tema inicial *Holo Dark* que especifiqué para utilizar en el ActionBar y el ViewPager, que está basado en negros y azules.

  Para empezar, empecé a diseñar botones utilizando la aplicación para diseño gráfico open source **Gimp**[67] y partiendo de botones open source que ya tenia, pero al no tener nada de práctica en el uso de esta herramienta los resultados eran bastante malos, no sabía como aplicar degradados, editar píxeles en zonas concretas sin afectar a otras, etc.

  Luego pasé a crear botones desde cero, pero la tarea, era, incluso más difícil, con lo que los resultados eran bastante pobres.

  Finalmente encontré un sitio web llamado **Da Button Factory**[68] que nos permite crear botones sencillos de forma visual y muy sencilla. Este sitio web no menciona que haga uso de alguna licencia, es más, en el blog de la misma web se puede encontrar multitud de comentarios fomentando el uso de la misma en cualquier escenario, con lo que supongo

[67] http://www.gimp.org/

[68] http://dabuttonfactory.com/

Figura 13: *Antiguo botón de añadir contactos.*

Figura 14: *Nuevo botón de añadir contactos.*

que estos recursos están bajo dominio publico con lo que son de uso perfectamente legal para nuestra aplicación que está bajo licencia GNU GPLv3.

Con la ayuda de este sitio web pude exportar los diferentes recursos de imágenes utilizados ahora para nuestros botones, para todos los tipos *Button, ImageButton y ToggleButton*, y para todos los estados de los mismos (normal, con el foco y pulsado).

Para el *ToggleButton*, puesto que este es un botón con estado, con Gimp lo modifique para que tuviera, en el estado pulsado, una linea de color diferente indicando que estaba activo en ese modo (a imagen y semejanza del original).

Finalmente con los botones ya creados pasé a aplicarles el *nine-patching* para que se pudieran *estirar* a placer sin perder calidad de imagen. La herramienta utilizada, como he dicho anteriormente, es la provista en las *Android tools* llamada *draw9patch*, y que al principio, para un principiante, puede resultar un poco difícil el comprender como usarla y a que aplicarlo, pero yo ya estaba bastante experimentado, con lo que fue relativamente su uso.

Las **Figuras 13** y **14** muestran los cambios realizados en el mismo botón de añadir contactos.

También indicar que se cambió el tamaño de la fuente y el color de la misma para que se ajustase mejor al nuevo botón.

Para la modificación del ActionBar en principio mi idea fue cambiar todos los recursos (assets) utilizados en la misma para poder tener el look and feel que quería ahora en consonancia con los nuevos botones, esta tarea hubiera sido un poco complicada, pero al poco de ponerme con ello apareció un *tweet* de Jake Wharton anunciando el proyecto de un miembro de la comunidad que había desarrollado un sitio web derivado del mencionado anteriormente, *Android Asset Studio* que era totalmente centrado en el look and feel del ActionBar. El sitio web es **Android Action Bar Style Generator**[69] y todo lo ge-

[69] jgilfelt.github.com/android-actionbarstylegenerator/

nerado por el mismo está licenciado bajo licencia CC-BY 3.0), con lo que es totalmente compatible con la licencia de QuiteSleep.

Una vez definido todo el estilo que queríamos para nuestra ActionBar pasamos a generar el paquete con todos los assets, con lo que estará listo para extraer en nuestra carpeta */res/drawable/drawable-XXX* (el paquete contiene los recursos necesarios para todas las densidades de pantalla soportadas).

Decir también que los recursos generados mediante esta herramienta ya están *nine-pacheados* con lo que no tenemos que aplicárselo nosotros mismos.

Una vez realizado todo esto, verificamos que nuestra nueva GUI está totalmente adaptada a las guidelines de Google para las nuevas versiones de Android, y para nuestro propio gusto personal.

- **Planificación**

  Como hemos dicho, una vez finalizada esta fase podemos dar por concluidos los cambios correspondientes al look and feel de la aplicación con lo que tenemos terminada la parte más importante de lo que nos habíamos propuesto hacer para la nueva release.

5.2.8. Fase 7: Fragments

- **Análisis de requisitos**

  Junto con los cambios relacionados a la GUI, desde la Fase 3 a las Fase 6, como hemos dicho anteriormente en ellas, hemos tenido que cambiar el uso de *Activities* e incluir también los nuevos *Fragments* para poder hacer uso por ejemplo del ViewPagerIndicator, pero el cambio al uso de Activities y Fragments no es solo relativo a esto, si no que es bueno cambiar toda la aplicación para utilizar desde esta nueva release los Fragments, y que desde ya cualquier cambio que se cree nuevo respecto a esto sea ahora mucho más fácil de implementar y cambiar puesto que nuestra aplicación ya está adaptada a Fragments.

  Resumiendo, en esta fase, que es una fase *diferente* puesto que no es una fase que se haya realizado después de crear la GUI si no que se ha hecho mientras se creaba la misma,

vemos como hemos cambiado las diferentes Activities utilizadas por FragmentActivities y diverso contenido que tenían las mismas por Fragments.

Podemos separar los cambios en tres grandes grupos:

- Cambio de Activities por FragmentActivities.
- Cambio del contenido de algunas activities por Fragments.
- Cambio de **Dialog**[70] por **DialogFragment**[71].

- **Análisis de riesgo**

  Esta fase (que como hemos dicho es una fase realizada a lo largo de las anteriores), es la fase, seguramente más importante de todas, ya que no es solo cambiar un aspecto gráfico, o utilizar y adaptar una librería, es:

  - Refactorizar paquetes.
  - Cambiar viejas clases.
  - Crear nuevas clases y sustituir las viejas.
  - Utilizar patrones de diseño.

  Como decimos esta es una fase diferente, larga, a lo largo de todo el desarrollo de esta nueva versión, con lo que disponemos de tiempo, pero también creemos que los cambios son muchos ya que por ejemplo el modo en que lanzábamos los *Dialog* ahora mismo está en modo *deprecated* que indica que no ahora, pero en el futuro esa implementación se dejara de utilizar, con lo que mejor ahora que nunca para cambiar todo lo necesario.

  Así, esta fase tenemos bastantes riesgos de *"romper"* la aplicación en varias partes, pero realizando los cambios de forma ordenada no tenemos porque dejar la aplicación en un estado *desastroso*.

- **Desarrollo, verificación y validación**

  Así, como hemos dicho en la **Fase 5.2.4**, para empezar cambiamos las activities utilizadas para mostrar las diferentes opciones de la aplicación (lo que antes era lo mostrado por las

[70] http://developer.android.com/reference/android/app/Dialog.html
[71] http://developer.android.com/reference/android/app/DialogFragment.html

pestañas y ahora lo es usando el ViewPager), por Fragments dentro del Main que ahora es en vez de una Activity una FragmentActivity.

Con esto cambiamos también la activity de los logs que era una ListActivity por una FragmentListActivity.

Como hemos dicho anteriormente, creamos unas *clases base*, tanto para las activities como para los fragments que utilizaríamos para que pudieran modificar de forma mucho más sencilla las diferentes acciones del ActionBar dependiendo de donde nos encontráramos. Las activities que hemos mencionado que cambiamos, extienden de estas con lo que hacemos un buen uso de la herencia de Java, y dejamos preparada la aplicación para la creación de nueva funcionalidad en el futuro.

Los Dialogs son seguramente el cambio más importante a realizar, ya que tenemos bastantes a lo largo de toda la aplicación, en un principio ante la cantidad de cambios internos que debíamos hacer para cambiar todos ellos, no solo el tipo de objeto, sino también funcionalidad que hay que adaptar para que funcionen con los nuevos DialogFragment, se pensó en no cambiarlos ya que se encontraban en estado deprecated y que aunque en un futuro próximo se dejarán de usar no era obligatorio, pero finalmente se siguió con el plan trazado de cambiarlos todos. Este cambio fue largo y costoso pero se realizó de forma satisfactoria.

Finalmente tenemos toda la aplicación adaptada al uso de Fragments y FragmentsActivities con lo que no solo la interfaz gráfica ha sido adaptada a las nuevas guidelines, sino que internamente la aplicación también lo ha sido.

Con esto pasamos a verificarla y validarla, con lo que damos esta parte como finalizada.

- **Planificación**

  Después de esta fase, podemos decir que la aplicación está casi lista, pero con todo lo aprendido este año que me he estado dedicando a Android, otro cambio es posible para un mejor funcionamiento de la aplicación, con lo que podemos pasar a la siguiente iteración.

5.2.9. Fase 8: Cambio en procesos internos

- **Análisis de requisitos**

En esta fase, que esperamos sea la última iteración, vamos a cambiar diversos procesos internos en los que bloqueábamos el **UI thread** que no es otra cosa que el hilo encargado de dibujar la interfaz de usuario, de atender a los eventos que se clican en ella, etc. En versiones anterior de Quitesleep y por desconocimiento se solía acceder a la base de datos desde el UI thread, con lo que esta se bloqueaba y si se quedaba demasiado tiempo así:

- El usuario podría pensar que la aplicación se había quedado bloqueada, con lo que la quitaba.
- Teníamos diferentes zonas de fallo bloqueando el UI thread, con lo que finalmente, efectivamente, la aplicación podía dar un **ANR** [72].

En ambos casos, el resultado final es el mismo, la aplicación no responde como debe, el usuario se cansa, y finalmente puede que desinstale la aplicación.

- **Análisis de riesgos**

  De inicio no creemos que el cambio en esta fase sea muy complicado de realizar ni que al realizar los cambios, estos no sean satisfactorios y la aplicación no funcione de forma correcta, con lo que no esperamos muchos problemas.

- **Desarrollo, verificación y validación**

  Como hemos dicho antes, las anteriores versiones de QuiteSleep adolecían de un problema que era el bloqueo del UI thread cuando accedíamos a la base de datos, en esta nueva versión hemos sustituido esas tareas por unas clases que hemos creado específicamente para esto que extienden de la clase de Android **AsyncTask**[73]. AsyncTask es una clase del SDK Android para realizar tareas en *under the hood* (en segundo plano) para no bloquear el UI thread, es utilizada principalmente para realizar peticiones a servidores web, acceder a bases de datos, procesar los mismos, etc.

  En nuestro caso, creamos unas clases para utilizar en los dos puntos de fallo donde bloqueábamos el UI thread:

  - Cargar contactos.

[72] Application Not Responding.

[73] http://developer.android.com/reference/android/os/AsyncTask.html

- Cargar logs.

En estos puntos de fallo, accedíamos a la bbdd sin mostrar nada mientras el usuario esperaba sin que se mostrara nada a que se cargaran las listas, con lo que ni el usuario sabía que pasaba, ni era correcto la forma de hacerlo.

Ahora, con las clases que creamos para ello *LoadContactsDataTask* y *LoadLogsTask*, realizamos estas tareas en segundo plano en otro thread sin bloquear el UI thread que antes bloqueábamos dando los mencionados problemas.

En la aplicación hay otro tipo de tareas que puede dar la impresión que bloqueamos el UI thread pero que no es así, por ejemplo cuando la aplicación se inicia por primera vez, se realiza una sincronización de los contactos de la bbdd interna del dispositivo con la utilizada por la aplicación, cuando esto ocurre, mostramos un *dialog* con un *progress loader* mostrando que se está trabajando, esto, aunque el usuario no pueda interactuar en ese momento con la UI no es bloquearla, sino que está siendo bloqueada a propósito. Lo mismo ocurre cuando añadimos todos los contactos como bloqueados y cuando los quitamos todos de ese estado.

Una vez implementadas estas clases pasamos a su verificación y validación comprobando que el funcionamiento es correcto y que la aplicación al completo funciona de forma óptima.

- **Planificación**

  Como vemos, después de este cambio, las últimas acciones que debíamos hacer para finalizar el proceso de la implementación para la nueva versión de QuiteSleep han sido realizadas, también esta última con éxito, con lo que procedemos a finalizar esta iteración, y con esto todo el proceso de la nueva implementación de la nueva versión.

5.3. Detalles de la implementación

Java y el paradigma de programación orientado a objetos nos permite diseñar nuestra aplicación siguiendo el modelo de clases. Para el modelo de datos, al estar utilizando una base de datos orientada a objetos nuestro modelo de clases utilizado en el paquete de base de datos será nuestro modelo de datos. En este tipo de bbdd el modelo de datos es el modelo de clases, en

contraposición al usual modelo entidad relación que utilizan la mayoría de las bbdd. Por ejemplo la bbdd nativa que trae consigo Android, que es **SQLite** sigue este modelo relacional que está basado en tablas.

Con la refactorización completa que realizamos y que podemos ver en la **Fase ??**, la organización de paquetes difiere bastante de lo que era originalmente y que se mantuvo sin muchos cambios hasta la versión anterior (2.0.4), así que vamos a pasar a explicar la arquitectura de la aplicación siguiendo el modelo de clases de la misma.

En la ***Figura 15*** podemos apreciar de forma global los paquetes utilizados en el desarrollo de la aplicación, en la que se muestran los paquetes de clases creados por nosotros y los paquetes utilizados de forma externa, como son los paquetes correspondientes a Android, db4o y Java Mail y las librerías ActionBarSherlock y ViewPagerIndicator.

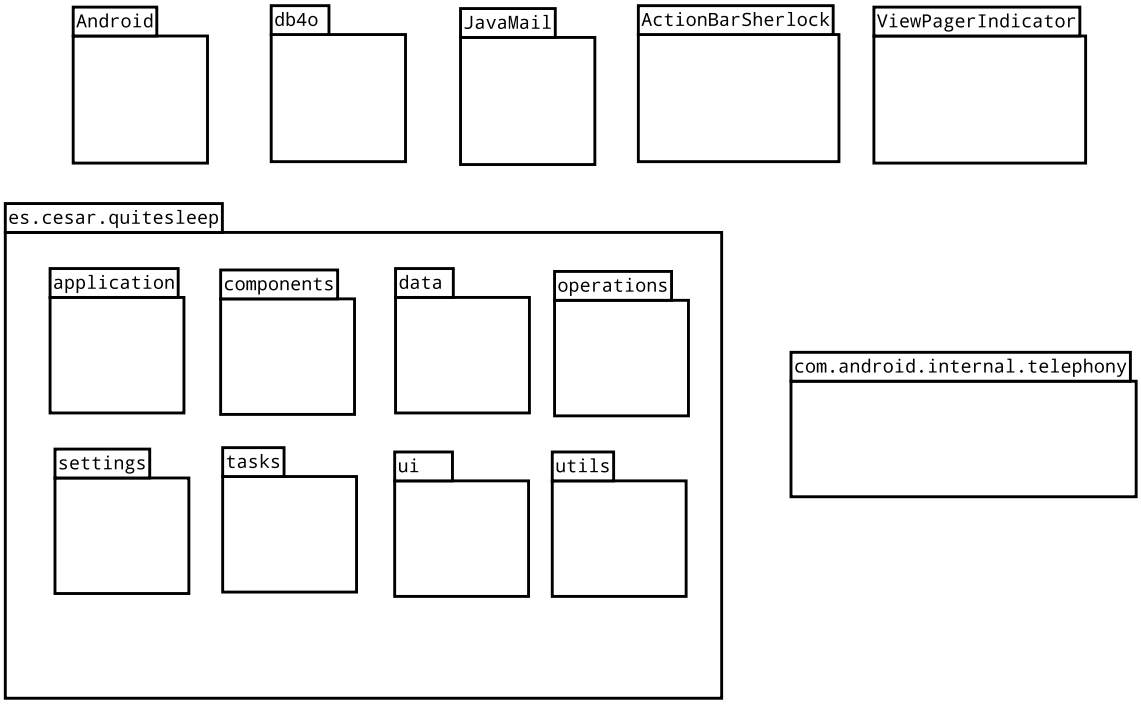

Figura 15: *Diagrama de clases global utilizado en la aplicación.*

A continuación, vamos a mostrar nuestro diseño de clases separado en los diferentes paquetes en los que hemos organizado la aplicación.

5.3.1. es.cesar.quitesleep.ui

Este es el paquete principal de la aplicación, ya que se refiere a la interfaz gráfica de usuario, en el definimos las diferentes FragmentActivities, los Fragments, Dialogs, DialogFragments, Adapters, clases base, menús y notificaciones.

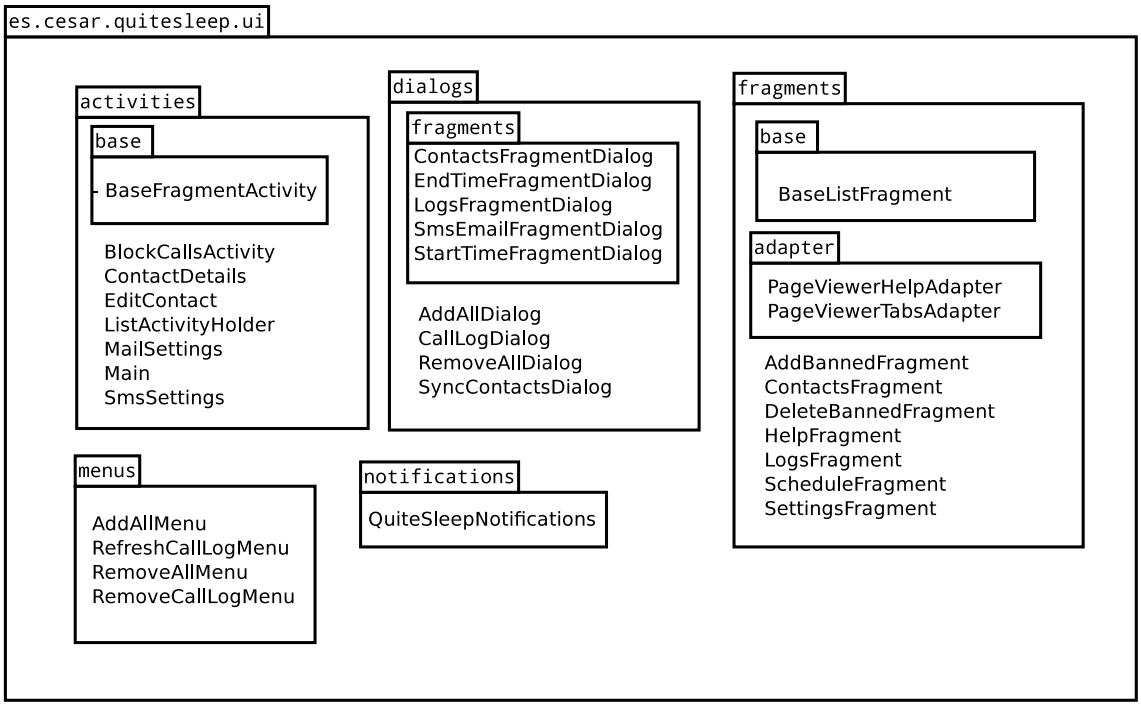

Figura 16: *Paquete de clases relativas a la interfaz de usuario*

Como vemos en la **Figura 16** el paquete correspondiente a la interfaz gráfica de usuario contiene los subpaquetes:

- **activities**, este paquete contiene un subpaquete *base* que solo tiene una clase, BaseFragmentActivity, que extiende de SherlockFragmentActivity, esta clase es utilizada por el resto de las activities, con lo que son todas en realidad FragmentActivities y permiten tanto tener contenido por si mismas como albergar otros fragments como es el caso de la clase Main utilizada para eso mismo con el ViewPager. En la **Figura 17** podemos ver de forma gráfica y más detallada la organización de las clases en este paquete y la relación entre ellas.

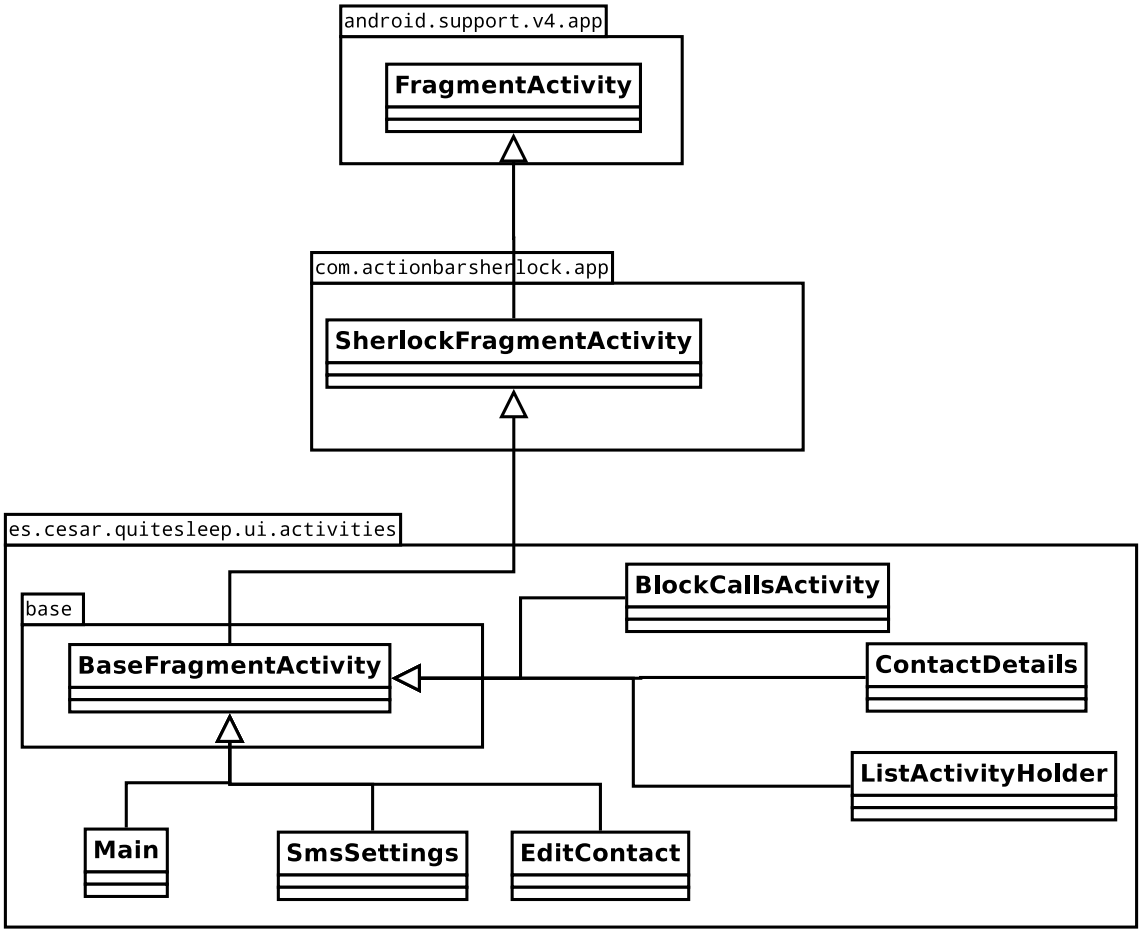

Figura 17: *Paquete de Activities (FragmentActivities)*

- **dialogs**, este paquete contiene tanto los dialogs propiamente dichos adaptados a los nuevos DialogFragments como las clases utilizadas para lanzar los *ProgressDialog* que inhabilitan la posibilidad del usuario de realizar una acción cuando queremos que no se pueda hacer nada y mientras estamos realizando una operación pesada en segundo plano, como son las operaciones de añadir todos los contactos, quitarlos de la lista de bloqueados, sincronización entre bbdd, etc.

- **fragments**, este es otro de los paquetes importantes y nuevos de esta nueva release, este paquete tiene dos subpaquetes, uno que alberga los diferentes *adapters* usados en conjunto con los correspondientes fragments para poder utilizar el ViewPagerIndicator y otro llamado *base* que sigue el modelo del anteriormente explicado activities, y que en este caso contiene una clase base que extiende del ListFragment y que utilizamos como clase abstracta para que las clases que extienden de esta realicen las mismas acciones comunes

e implementen las suyas propias.

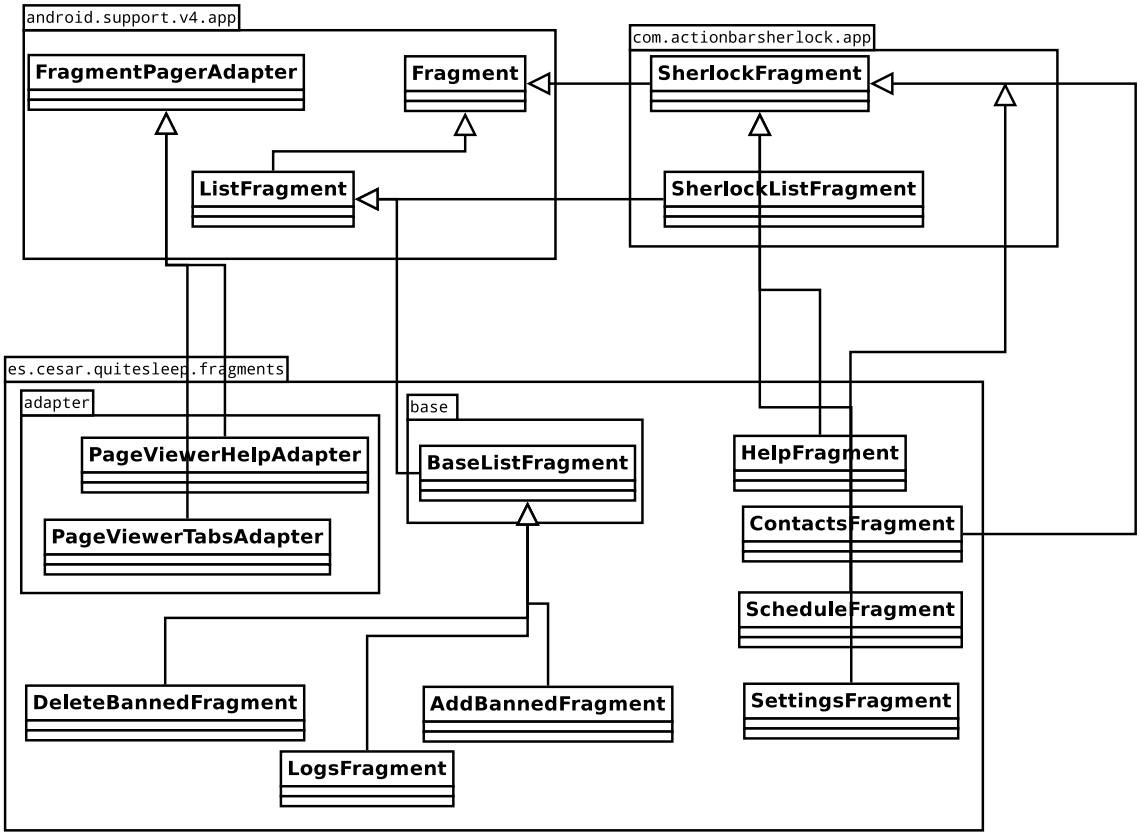

Figura 18: *Paquete de Fragments*

En la **Figura 18** podemos ver como está organizada la arquitectura de este paquete y de las clases que están en el, como tenemos la clase base para las listas que usan fragments, los adapters para usar en el ViewPager, y el resto de fragments utilizados.

- menús, en este paquete nos encontramos las clases utilizadas para la realización de las diferentes acciones localizadas ahora en el nuevo ActionBar, acciones como:

  1. Añadir todos los contactos como bloqueados (AddAllMenu).
  2. Actualizar el log del historial (RefreshCallLogMenu).
  3. Quitar todos los contactos como bloqueados (RemoveAllMenu).
  4. Eliminar el historial de logs completamente (RemoveCallLogMenu).

- notifications, este paquete incluye la clase que realiza el proceso de gestionar la notificación que es utilizada cuando se activa y desactiva el servicio de QuiteSleep.

5.3.2. es.cesar.quitesleep.application

Este paquete Contiene solo una clase llamada **QuiteSleepApp**, esta clase, como mencionamos antes en la **Fase 5.2.3**, es utilizada para poder hacer uso del *Application context* de Android a lo largo de toda la aplicación solo accediendo al método estático que tiene esta clase, con lo que gracias a esto podemos dejar de pasar el contexto a lo largo de las diferentes activities/fragments de la aplicación que hacíamos en anteriores versiones de QuiteSleep y no es muy recomendado.

Esta clase es incluida en la raíz de la aplicación del manifest, para que sea cargada nada más lanzar la aplicación.

5.3.3. es.cesar.quitesleep.components

Este paquete contiene cuatro paquetes en los que podemos encontrar clases que se encargan de diversas acciones en la aplicación.

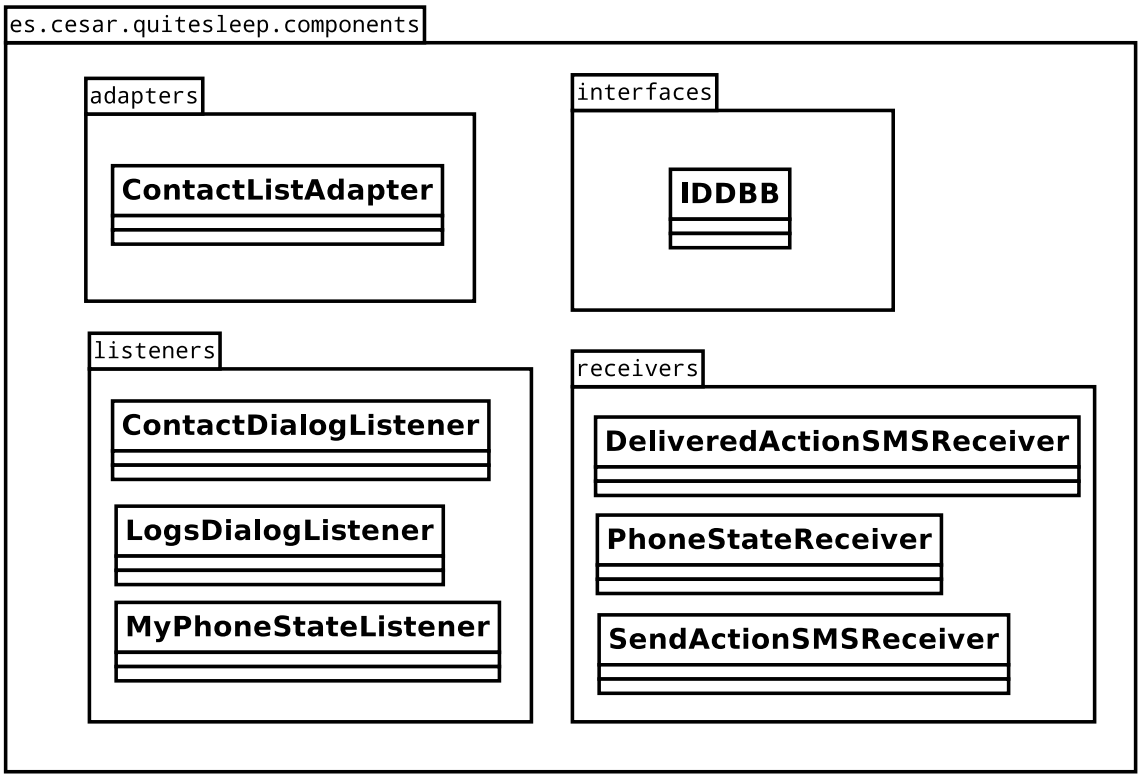

Figura 19: *Paquete components*

En la **Figura 19** podemos ver como está organizado este paquete, los paquetes que lo forman

y las clases que están contenidas en estos.

- **adapters**, este paquete contiene un *custom adapter* (ContactListAdapter) creado para la release 2.0.4 que es utilizado para poder desplazarnos por las listas de contactos de forma rápida utilizando un scroll rápido que es ordenado alfabéticamente.

- **interfaces**, este paquete contiene la clase *IDDBB* que no es más que una clase con constantes para ser utilizados en tareas correspondientes a la ddbb.

- **listeners**, este paquete contiene la antigua clase *MyPhoneStateListener*, que es utilizada para *capturar* los diferentes estados del teléfono correspondientes a las llamadas, y otras acciones relacionadas. También en este paquete tenemos las clases *ContactDialogListener* y *LogsDialogListener* que son utilizadas para capturar los diferentes eventos en los dialogos creados y mostrados al usuario cuando está apunto de realizar una acción. Estos listeners son usados como *callbacks* para interaccionar con la ui.

- **receivers**, los utilizados desde la primera versión de QuiteSleep, *DeliveredActionSMSReceiver*, *PhoneStateReceiver* y *SendActionSMSReceiver* y que se corresponden con las acciones de atender a los diferentes *BroadcastReceiver* generados por la aplicación dependiendo de las circunstancias.

5.3.4. es.cesar.quitesleep.data

Este paquete está básicamente orientado a las clases relacionadas con la base de datos, pero también contiene otros paquetes que aunque no son modelos u operaciones sobre la bbdd, si realizan operaciones con los datos.

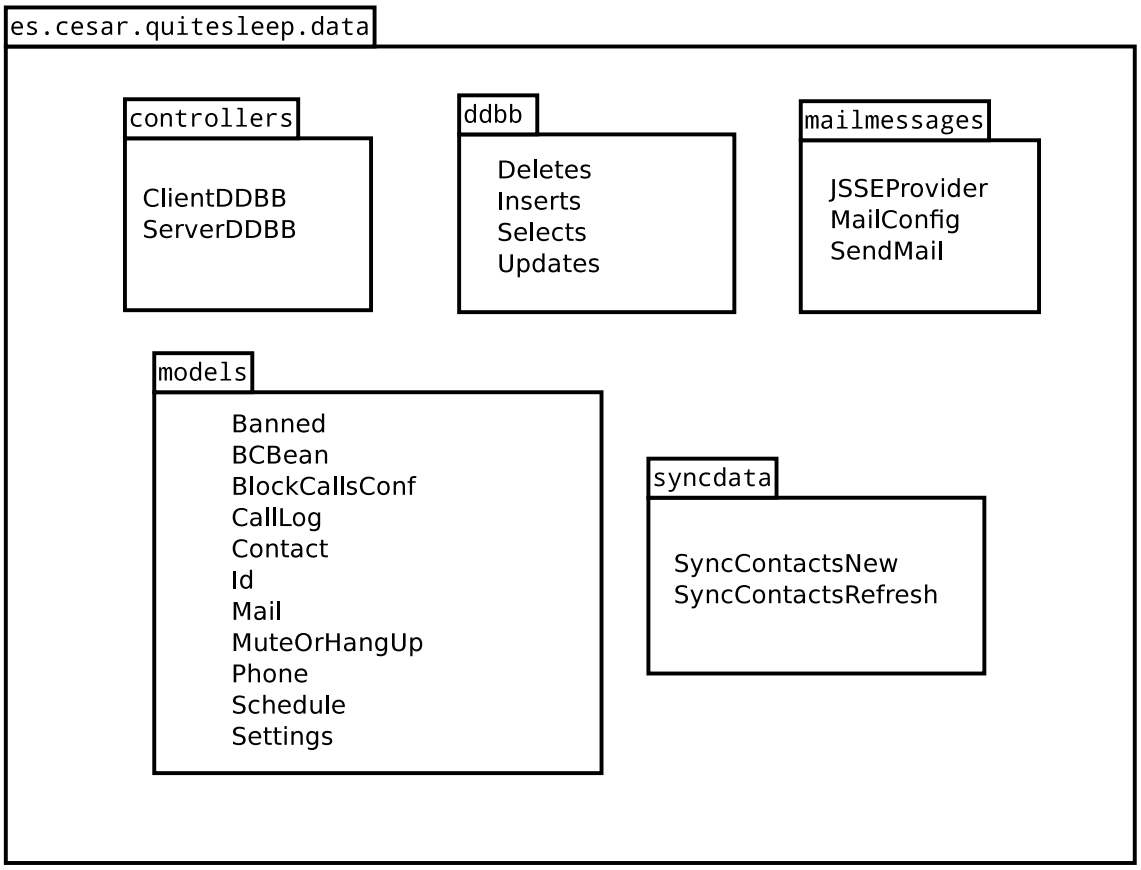

Figura 20: *Paquete data*

Como decimos, este paquete contiene los siguientes subpaquetes:

- **controllers**, este paquete contiene el *cliente y servidor* de la base de datos db4o, el servidor está a la escucha de las operaciones a realizar, desplegado en modo **singleton**, y el cliente realiza las *"queries"* pertinentes al servidor para obtener datos concretos.

- **ddbb**, este paquete contiene las clases relacionadas con las diferentes operaciones sobre la bbdd como son *selects, inserts, deletes y updates*.

- **mailmessages**, este paquete contiene el modelo de datos utilizado para configurar el envío automático de sms cuando la aplicación está configurada para hacerlo. Contiene tres clases como son *JSSEProvider, MailConfig y SendMail*, estas clases hacen uso de la librería JavaMail.

- **models**, este paquete contiene todas las clases relacionadas con el modelo de datos utilizado en la aplicación, el modelo de datos correspondiente a los contactos, a los mails,

teléfonos, etc.

- **syncdata**, este paquete contiene dos clases, *SyncContactsNew y SyncContactsRefresh* que se encargan de sincronizar la bbdd interna del sistema Android en SQLite con la bbdd interna de QuiteSleep que es db4o, una se utiliza para sincronizar la primera vez que la aplicación es lanzada y la otra bajo demanda.

5.3.5. es.cesar.quitesleep.operations

Este paquete contiene clases de diverso tipo que realizan diversas acciones en la aplicación, como:

- Comprobar el estado del servicio de la aplicación.
- Poner los contactos como bloqueados o no.
- Operaciones que son lanzadas después de mostrar dialogs.
- Operaciones relativas a las llamadas entrantes.
- Operaciones relativas a los procesos de envío de SMS y correo electrónico.
- Acciones correspondientes al inicio y parada de los servicios de QuiteSleep (el principal, el de SMS y el de email).

En la release 2.0 de QuiteSleep introdujimos una nueva clase llamada *BlockTypes* en la que diferenciamos los diferentes tipos de bloqueo que en esa versión introdujimos como eran:

- Bloquear todo.
- Bloquear solo contactos bloqueados.
- Bloquear desconocidos.
- Bloquear contactos bloqueados y desconocidos.

Además en esta clase diferenciamos el modo en que queremos actuar cuando recibimos una llamada no deseada:

- Silenciar automáticamente la llamada.

- Colgar automáticamente la llamada.

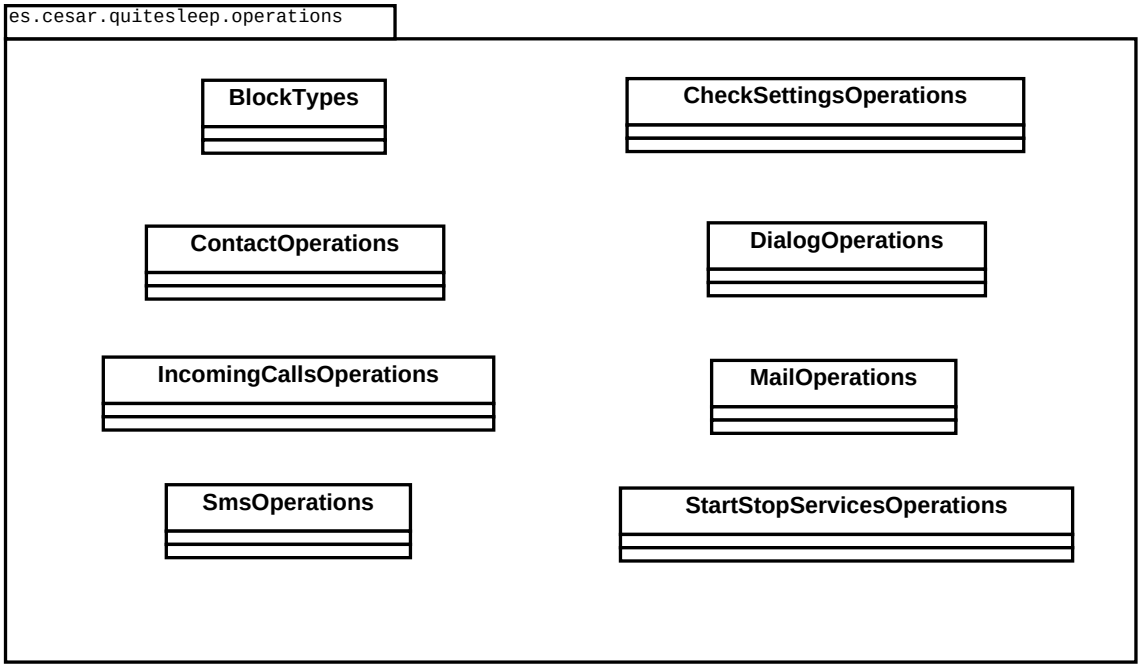

Figura 21: *Paquete operations.*

En la **Figura 21** podemos ver la organización de este paquete junto con las clases que contiene.

5.3.6. es.cesar.quitesleep.settings

Este paquete incluye las clases que son utilizadas para contener objetos estáticos y utilizarlos muchas veces de forma directa como si fuera una ***caché***.

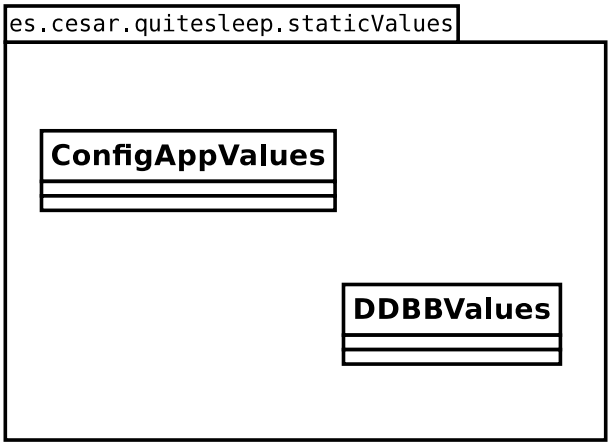

Figura 22: *Paquete de clases que contienen objetos estáticos.*

5.3.7. es.cesar.quitesleep.tasks

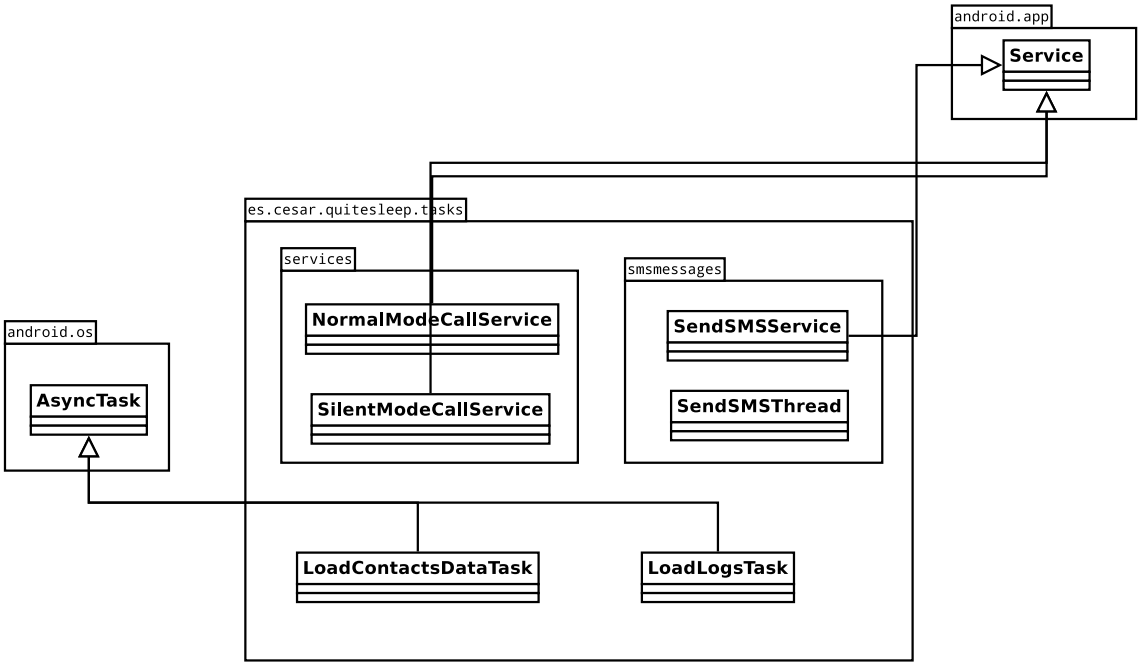

Figura 23: *Paquete tasks.*

Este paquete incluye otros paquetes con funcionalidad diferenciada:

- **services**, este paquete contiene las clases que extienden de la clase Android *Service* y que realizan tareas en background correspondientes con fijar el estado del teléfono en un modo concreto: silenciado o normal.

- **smsmessages**, este paquete contiene las clases relacionadas con la configuración, gestión y envío de SMS, contiene dos clases que realizan la misma función pero implementadas de forma diferente ya que una de ellas utiliza *services* de Android y otra *threads* de Java, se han creado para probar cual de las dos es más eficiente, pero actuando las dos de forma exactamente igual. Actualmente se utiliza la versión que usa threads de java.

- **LoadContactsDataTask** y **LoadLogsTask**, estas clases son las que creamos durante la **Fase ??** que mencionamos anteriormente, ambas extienden de la clase Android **AsyncTask** y son utilizadas para realizar tareas en otro thread evitando así bloquear el UI thread de la aplicación y por tanto evitar problemas tanto de percepción al usuario por mantener la aplicación bloqueada como de funcionamiento respetando la arquitectura

Android. Estas dos clases son utilizadas para el acceso a la bbdd para cargar contenido correspondiente a los contactos como contenido correspondiente a los logs.

5.3.8. es.cesar.quitesleep.utils

Paquete correspondiente a clases utilizadas para realizar acciones de diversa índole como:

- Tipo de dato utilizado para el envío de correo electrónico.
- Control y gestión de excepciones.
- Control y gestión de logs.
- Control y gestión de notificaciones de tipo *toast*.
- Codificación de datos utilizando *SHA1* [74].
- Parseo de datos utilizando *tokens*.

[74] http://en.wikipedia.org/wiki/SHA-1

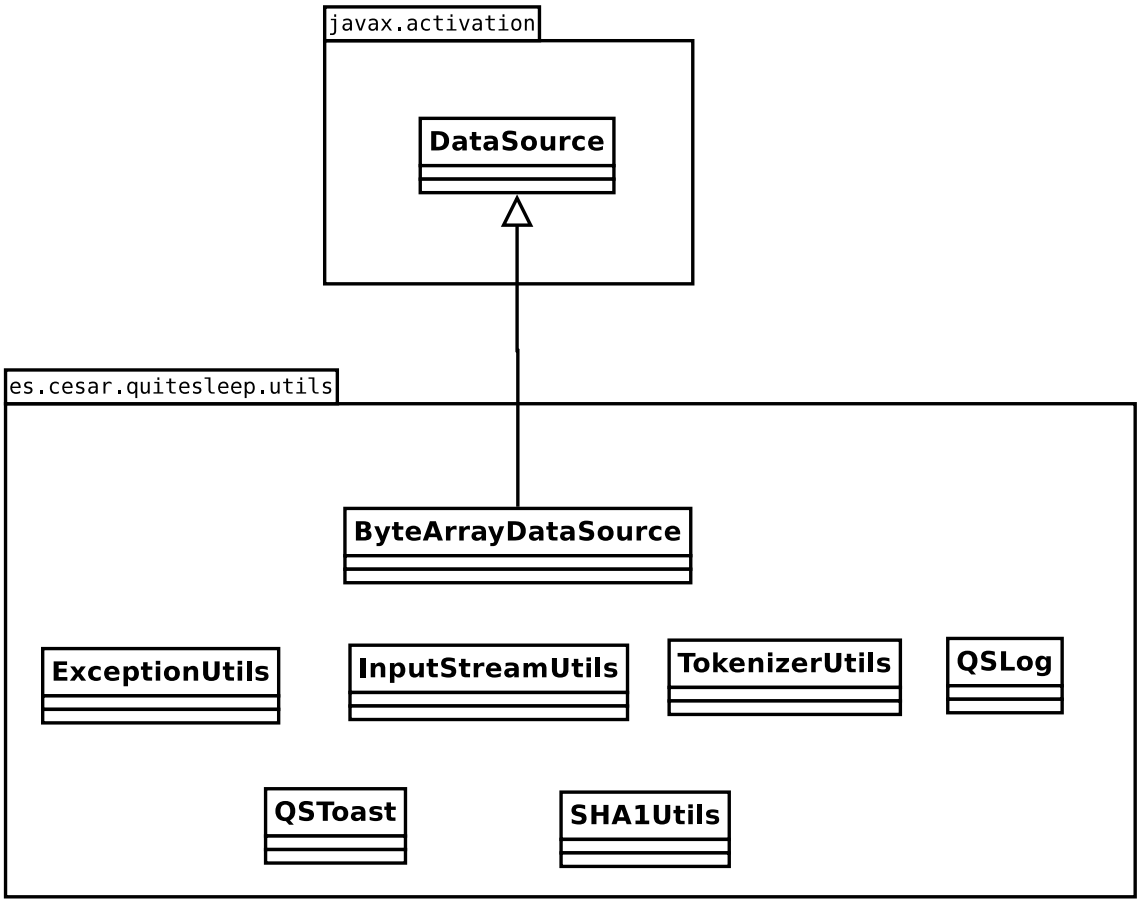

Figura 24: *Paquete de clases correspondiente a utilidades varias.*

5.3.9. com.android.internal.telephony

Este paquete fue incluido en la versión 2.0 de Android, pero creemos que su funcionalidad merece la pena un comentario aquí.

Debido a la nueva funcionalidad que nos propusimos para la versión 2.0 de permitir que el usuario pudiera escoger entre silenciar una llamada o colgarla de forma automática, y que fue muy demandada por los usuarios, nos propusimos realizar esta tarea.

El método de colgar la llamada como podemos ver si accedemos al código fuente de Android está allí pero no en la api, con lo que no podemos utilizar ese método y hay que hacerlo abordando el problema de otra forma.

1. Para ello lo que se hizo fue crear un paquete siguiendo el mismo nombre que la clase en la que está contenida.

2. En vez de un archivo .java, lo que tenemos que crear es un archivo *.aidl* que contiene el stub de la interfaz *Telephony* que es la que tiene el método de colgar la llamada.

3. En nuestro código debemos crear una implementación de la interfaz Telephony, pero ahora la interfaz a utilizar es la que tenemos definida en el .aidl.

4. Con esa implementación ya podemos llamar al método *endCall()* con el que se nos permite colgar la llamada desde código.

Así, con esta implementación, que no es muy normal usar en aplicaciones convencionales, pudimos acceder al método que deseábamos pero que no teníamos disponible.

Esto puede traer consigo problemas y es que ahora estamos sujetos a la implementación interna del código de Android con lo que si cambia tendremos problemas y no podremos colgar las llamadas. Tendremos que estar atentos a los cambios del sistema operativo con respecto a esa interfaz para poder actualizarla si es debido.

6. Evolución de QuiteSleep como proyecto FLOSS

QuiteSleep se trata de un proyecto completamente **FLOSS**.

Desde la concepción de la aplicación, Quitesleep siempre fue pensada para ser un proyecto FLOSS, no solo el estar bajo una licencia de software libre, sino estar alojado en una forja, tener diferentes recursos disponibles para los usuarios, no solo código fuente, si no también documentación, binarios diferentes *tarballs* de las diferentes versiones, vías para contactar con el autor y crear comunidad alrededor, etc. pero no solo de recursos o código vive el software libre, en esta sección hablaremos de la evolución de QuiteSleep a lo largo de su vida, desde que fue concebido allá por Febrero de 2010 hasta hoy, que cambios ha sufrido, como ha mejorado, estadísticas útiles, datos de interés, datos curiosos, datos personales, etc.

Vamos a ir contando a lo largo de esta sección, los diferentes puntos de interés mencionados intentando seguir un orden cronológico desde lo que fue en principio, como han ido cambiando (si es que lo han hecho), y como han quedado.

6.1. Licencia

Nuestro propósito desde siempre era publicar nuestra aplicación bajo una licencia robusta que fuese Copyleft, la más usada de este tipo es la GNU GPL, con lo que nuestra intención era publicarla bajo esta licencia.

El por qué de utilizar una licencia robusta es algo, como muchas cosas, personal, yo me declaro mucho más afín a este tipo de licencias en las que se preserva la libertad otorgada por el autor del software a toda la comunidad alrededor del mismo y durante toda su vida.

Que la aplicación pudiera ser utilizada completamente o parte de ella como software privativo era algo que no quería con lo que la decisión era clara, y ¿por qué GNU GPLv3?, debido a la gran proliferación de licencias libres, que en la práctica es un problema, la decisión de utilizar la licencia GNU GPLv3 era clara por ser la más utilizada en el mundo del software libre, por estar muy documentada, y por otorgar los derechos que queríamos otorgar a nuestros usuarios.

Originalmente, el software adicional que utilizamos para la creación de la primera release era completamente libre, como se detalla a continuación:

- Android, publicado en casi su totalidad bajo la licencia Apache License 2.0

- db4o, publicado mediante licencia dual, como hemos dicho en la *sección 3.3.2* db4o si queremos utilizarlo de forma libre nos permite utilizarlo bajo varias licencias, siendo la que a nosotros nos interesa la licencia GNU GPL v3.
- Java Mail API, publicada bajo varias licencias entre ellas GNU GPL v2 y BSD, la que a nosotros nos interesa es esta última ya que GNU GPL v2 no es compatible con GNU GPL v3, que es como nosotros queremos liberar nuestra aplicación.
- Iconos, fuentes y demás recursos, utilizamos licencias libres compatibles con GNU GPL v3, como son licencias *CC-BY-SA v3* o *dominio público*, que no es exactamente una licencia pero nos da total libertad de uso.

Con la inclusión de dos nuevas librerías como son las anteriormente mencionadas *ActionBarSherlock y ViewPagerIndicator*, debíamos comprobar también, antes de poder usarlas, si las licencias utilizadas en ellas eran compatibles con la de nuestra aplicación, y así fue, dado que ambas librerías están bajo una licencia **Apache License v2** que es totalmente compatible con la licencia GNU GPLv3.

Toda clase y archivo xml en nuestra aplicación contiene una cabecera que indica la licencia que estamos utilizando, el copyright, etc.

```
/*
 Copyright 2010 Cesar Valiente Gordo

 This file is part of QuiteSleep.

   QuiteSleep is free software: you can redistribute it and/or
   modify it under the terms of the GNU General Public License
   as published by the Free Software Foundation, either
   version 3 of the License, or (at your option) any later
   version.

   QuiteSleep is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty
   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
    See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public
    License along with QuiteSleep.  If not, see
    <http://www.gnu.org/licenses/>.
*/
```

Esta cabecera fue la utilizada en la primera release de la aplicación, a lo largo de las diferentes releases se ha ido cambiando el año del copyright, por el de 2010-2011 y actualmente por 2010-2012.

Para indicar el tipo de licencias que utilizan otro tipo de recursos que utilizamos como son las imágenes, fuentes, etc. tenemos un fichero de texto en la carpeta correspondiente indicando el tipo de licencia que utiliza.

Finalmente como toda aplicación de software libre, tenemos los archivos *COPYING* con la licencia en formato texto plano y *README* con datos correspondientes al proyecto, autor, contacto, etc.

En verano de 2011, tuve un problema que afectaba seriamente a la aplicación en términos de la aplicación como proyecto de software libre, y es que se publicó una aplicación en el market de Google que era exactamente igual a QuiteSleep pero con publicidad.

Esto no sería un problema, ni mucho menos, es más es todo un orgullo que otros desarrolladores hagan un fork de tu proyecto, creen otra aplicación o la modifiquen y la publiquen e intenten ganar dinero, el software libre está para eso, pero este usuario no publicó los cambios que había realizado en ningún sitio, su código no estaba disponible y mi autoría del proyecto, en la sección de Acerca de había sido modificada y cambiada con el nombre del *"nuevo"* autor, con lo que estaba infringiendo la licencia GNU GPLv3 con la que publiqué la aplicación:

1. Por publicar la aplicación pero no publicar el código con los cambios, con lo que la aplicación pasaba a ser privativa, cosa que no se puede con la licencia GNU GPLv3.

2. Por eliminar mi autoría de la aplicación.

Con esto me puse en contacto con los profesores del máster que me aconsejaron sabiamente como debía actuar y con Google para que eliminara esa aplicación del market ya que estaba infringiendo la licencia que yo había puesto a mi proyecto.

Aquí es donde podemos ver que la libertad que da Google a los desarrolladores a la hora de subir aplicaciones a su market está muy bien, pero cuando hay problemas esto no funciona tan bien; estuve enviando correos bastante a menudo a Google para que eliminara la aplicación a lo largo del tiempo y no fue hasta más o menos finales/primeros de este año cuando la aplicación ilegal y otras del mismo autor (hacia lo mismo con otras aplicaciones FLOSS) fueron eliminadas del market.

6.2. Proyecto en una forja de software libre

Para la publicación y mantenimiento de una comunidad sobre un proyecto de software, hay que publicar este haciendo uso de una de las múltiples forjas de software que hay disponibles como *SourceForge*[75]*, BerliOS,*[76]*, Google Code [33], Launchpad*[77]*, etc.*, que nos proveen no solo de espacio para poder alojar nuestro proyecto, sino que también nos ofrecen todo tipo de herramientas con las que poder llevar un proyecto colaborativo a buen puerto, como son, el repositorio para el código fuente, uso de foros, blogs, *bugtracker*[78], descargas de diferentes versiones de tarballs, documentación, historial, etc.

Últimamente está en alza **GitHub**[79] debido a la gran infraestructura creada en esta forja, el sistema social que tiene con respecto a los desarrolladores, y por el uso de Git, que seguramente sea el SCM de más uso en la actualidad, en, al menos, el mundo open source.

Finalmente nosotros nos decidimos por Google Code, ya que los proyectos open source de Android en general, se suelen alojar allí, debido a la delgada linea que separa Android y todos los servicios de Google.

Google Code se trata, como decimos, de la forja para proyectos software de Google, en apariencia más sencilla que otros como SourceForge o BerliOS, Google Code ofrece todo lo que se le puede pedir a una forja, como son las características antes mencionadas.

El sitio del proyecto se *reserva* al poco de comenzar el proyecto, y en el se anuncia en que estamos trabajando y cuando, más o menos, pensamos tenerlo terminado, y se planea su subida.

El día ***18/05/2010*** se decide publicar el proyecto, en el sitio del Proyecto en Google Code

[75] http://sourceforge.net/

[76] http://www.berlios.de/

[77] https://launchpad.net/

[78] Sitio dedicado exclusivamente al seguimiento de errores en un proyecto software

[79] https://github.com/

[34], donde se utilizan los siguientes recursos:

- Proyecto en un repositorio de tipo distribuido como es *Mercurial* [35].
- Bugtracker.
- Descripción e historial.
- Aplicación lista para instalar y utilizar, en formato .apk de Android.
- Tarball con el código fuente, archivos de configuración, recursos, etc.
- Documentación en formato de API utilizando JavaDoc.

La ***Figura 25*** muestra la página principal del proyecto en Google Code.

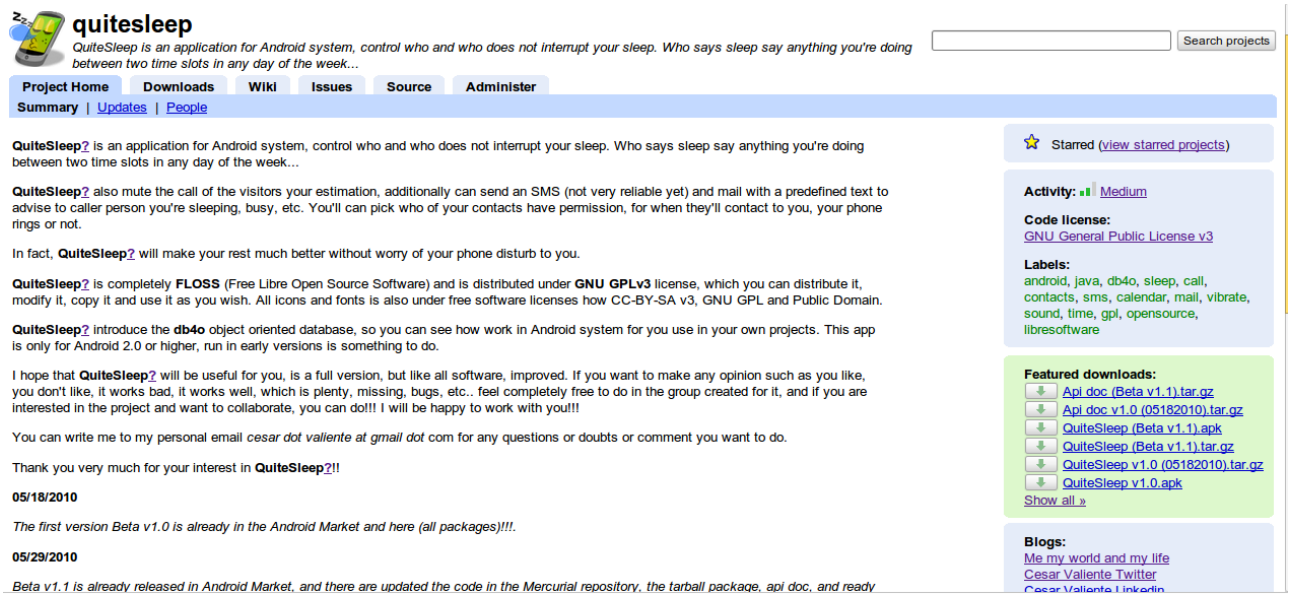

Figura 25: *Captura de pantalla del proyecto QuiteSleep en Google Code.*

Junto con el proyecto en el sitio de la forja utilizada se crea también un grupo en *Google Groups*[80] para que la posible comunidad creada en torno al proyecto, tenga un sitio web donde poner todo lo relativo al mismo, con noticias, comentarios, bugs, etc.

Como vemos en la ***Figura 26*** el grupo del proyecto en Google Groups[81] es un sitio web sencillo que admite miembros, y posts donde se cuenta todo lo relativo a el.

[80] http://groups.google.com/

[81] http://groups.google.com/group/QuiteSleep

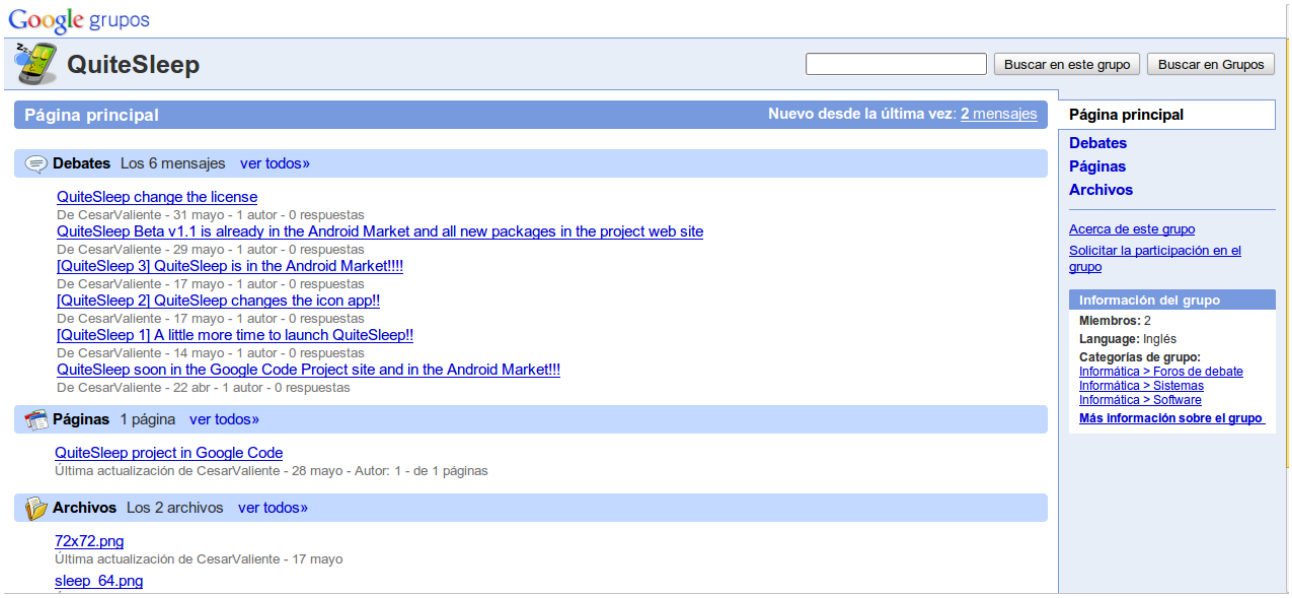

Figura 26: *Captura de pantalla del proyecto QuiteSleep en Google Groups.*

A lo largo de todo este tiempo desde la primera publicación de la aplicación, se han ido añadiendo diferentes y nuevos recursos al sitio web del proyecto como son:

- Diferentes binarios (.apk) con las diferentes versiones (releases) de la aplicación.
- Diferentes paquetes de documentación, con las actualizaciones de las nuevas clases.
- Diferentes paquetes de código fuente (tarballs) listo para poder ser importado en Eclipse y utilizar y desarrollar o modificar la aplicación.
- Memoria y transparencias utilizadas en la defensa del Proyecto Fin de Máster de la primera versión de QuiteSleep.
- Otros documentos, como un artículo que salió publicado en **dZone**[82] pero en formato *.pdf* o las transparencias utilizadas en la presentación que hice en la **Tenerife Lan Party 2010 (TLP2K10)**[83].

En la forja también se pueden ver otros datos de interés como son bugs, los commits realizados, el número de descargas, número de clones del proyecto, pero todo esto se pasará a contar en detalle más adelante.

[82] http://java.dzone.com/articles/real-android-apps-leveraging-1

[83] http://www.tenerife-lanparty.com/2k10/actividades/ponentes/410-cesar-valiente.ht ml

6.3. Publicación en el Market

La aplicación fue publicada en el Android Market (ahora llamado Google Play), en Mayo de 2010, junto con la publicación del proyecto en su forja.

La primera versión fue destinada a teléfonos Android 2.0 en adelante. En aquel momento la versión 2.1 era la más nueva, con lo que nuestra aplicación estaba orientada a esos nuevos terminales con las últimas versiones de Android.

Para poder subir aplicaciones al Android Market se necesitó tener una cuenta de desarrollador subscrito al Market, esto se realiza haciendo un pago de 25$ (dólares), y una vez hecho, como desarrolladores podemos subir nuestras aplicaciones, hacer dinero con ellas, comprar dispositivos de desarrollador, etc.

Mediante la *consola del desarrollador* se hace todas las gestiones pertinentes a la publicación de la aplicación, muy pocos pasos que realizar y sencillez es lo que resume el proceso. En la ***Figura 27*** podemos ver parte de la consola de desarrollador donde especificamos los datos necesarios y subimos el programa listo para instalar en formato .apk.

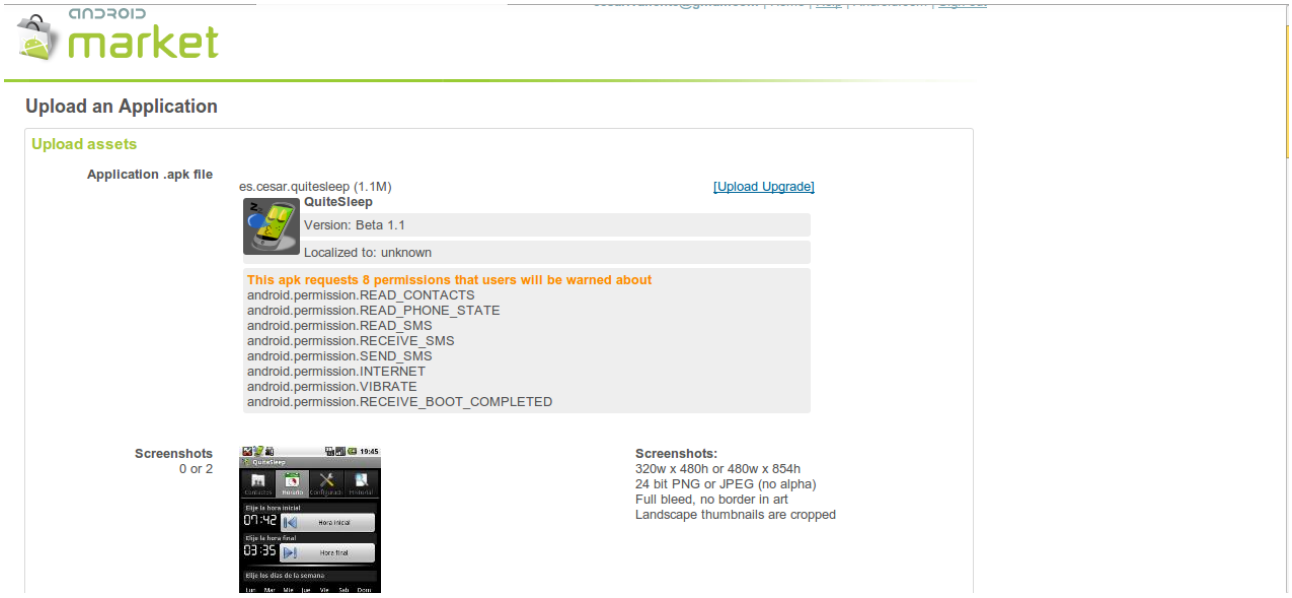

Figura 27: *Captura de pantalla de la consola de desarrollador en Android Market.*

Una vez subida la aplicación al instante estará disponible en el Android Market de todos los países en los que hayamos especificado que está disponible (todos los posibles), y lista para que los usuarios hagan uso de ella.

Después de la publicación en el Market oficial de Google, contactó conmigo una empresa

China que cuenta con un un market para su mercado nacional, esta empresa estaba muy interesada en contar con mi aplicación en su market, con lo que se ofrecieron incluso a traducirla a su idioma. Con esto acepté y la aplicación pasó a estar traducida a Chino y Chino tradicional y publicada en su market[84]. En este market, que actualizan ellos mismos, la última versión de QuiteSleep disponible es la 2.0.4.

Con estos dos markets, también se publicó la aplicación en el **Amazon Appstore**[85] sitio web del portal Amazon con el que entró en el mercado de las aplicaciones móviles. En este market está publicada la versión 2.0.4 de QuiteSleep, teniendo que publicar la nueva versión tan pronto como sea posible.

A lo largo de la historia de esta aplicación, se han ido subiendo diferentes versiones menores, y dos mayores, las versiones menores fueron lanzadas para subsanar diferentes bugs detectados por mi mismo o que veía en la consola de desarrollador como problemas que habían sido reportados por usuarios utilizando el sistema que tiene Android para ello, o mediante mensajes recibidos en mi correo electrónico.

Un ejemplo de mails reportando el mal funcionamiento de la aplicación por usuarios son:

```
 No logro que funcione en un huawei8110 com android 2.1 rooteado.
gracias


---------------------------------------------------------------


Hola Cesar:
Lo primero felicitarte por la excelente aplicacion. Me parece una
idea muy buena y espero que sigas adelante con ella y con las que
se te puedan ocurrir :) .
Queria comentarte una serie de bugs que encuentro. Tengo un
terminal htc desire con froyo. Cuando intento añadir un contacto
a la lista de bloqueados, no me lo permite. No me da ningun tipo
de error, unicamente parece que se selecciona pero no lo hace.
Para poder meter un contacto,
```

[84] http://market.goapk.com/search.php?search_key=quitesleep

[85] http://www.amazon.com/AndroidStartup-QuiteSleep/dp/B004VD436E/

debo añadir todos e ir quitando uno a uno.
Por otro lado, me pareceria una buena opcion para la siguiente version, que se pudieran bloquear tambien las llamadas anonimas o las llamadas que no esten en la lista de contactos, puesto que la cantidad de llamadas que se reciben a horarios muy mañaneros son o de teleoperadores o gente que llama por confusion.
Gracias por tu atencion, y espero una respuesta para ver si el fallo es solo mio. Gracias!

Hola! Tengo un smartphone lg optimus black, he instalado la aplicación quitesleep pero no funciona colgar la llamada, si pongo la opción silenciar si que funciona. Pero necesito que cuelgue la llamada, por qué
no funciona? Hay algo que pueda hacer? Algo que falta configurar? muchas gracias.

Las versiones mayores fueron lanzadas con importantes mejoras y cambios en la aplicación, las ideas y motivaciones para realizar estos cambios fueron mis propias ideas e intereses y por peticiones sobre la funcionalidad, realizadas por diversos usuarios a través del email, mi blog, etc.

Cabe destacar que algunas peticiones se tuvieron en cuenta y otras no por escaparse de lo que yo entiendo que la aplicación debe realizar.

Un ejemplo de emails recibidos, omitiendo los datos personales de los autores para preservar su intimidad, son:

Disculpa laas molestias.Bloquear desconocidos,se refiere a números que no los reconoce el identificador de llamadas, por ej. cabinas telefónicas?.Sería muy interesante bloquear sms.

Espero tu respuesta y desde ya.muhas gracias.mi equipo,
sony ericcson xperia 10,última versión,Argentina.Mucha suerte!

Hello! I'm from XXXX, so I hope you understand me! It's a quite good proram, but I'm missing something! Can you add buttons to block also private and unknown numbers? Or maybe a button would be good: "Only allow following numbers to ring!" The last button would be perfect for me, I'm working in the night and so I could just allow work to phone me and wake me up! All others should be blocked than! Thank you in advance!

Hola Cesar enhorabuena por tu trabajo. Existe la opción que en vez de indicar los contactos bloqueados, pudieras seleccionar los que sí quieres aceptar. Pienso que en mi caso, con lo grande que es la lista sería más fácil.
Saludos cordiales y que tengas mucha suerte con el proyecto.

Hola César.

He leído en un blog acerca de tu aplicación quitesleep y me ha parecido muy interesante. Me la he descargado y tiene muy buena pinta.

Te escribo para preguntarte acerca del comportamiento de la aplicación cuando se recibe una llamada de un número que no esta en la agenda. He leido el acerca y la ayuda pero no lo

he encontrado, quizás se me haya pasado. Puedes comentarte cómo funcionaria la aplicación en ese caso?

Mi idea es utilizar la aplicacion para bloquear todo (incluso aquellas que no tenga en mi lista) excepto las que yo diga.
Se puede hacer eso?

Un saludo y enhorabuena por la aplicación.

Como vemos los usuarios han reportado tanto errores que se han encontrado como peticiones sobre que les gustaría que la aplicación hiciera, con lo que es muy bueno este feedback creado.

6.4. Publicidad de QuiteSleep

Con la puesta en marcha del lanzamiento del proyecto en todos los sitios mencionados anteriormente, se intentó dar un poco de publicidad, dentro de lo posible, a la aplicación.

Como nuestra meta es la creación de un proyecto de software libre y que pueda ser atractivo para que otros desarrolladores se unan a el, la publicidad está centrada en la comunidad, no está orientada a descargas o utilización.

Así, se dio paso a publicitar el proyecto en los sitios open source ***Freshmeat***[86] y ***Ohloh*** [87] que son básicamente sitios web donde tienen referencias de multitud de proyectos de software libre.

En la ***Figura 28*** y ***Figura 29*** podemos ver los sitios web mencionados con sus correspondientes referencias al proyecto QuiteSleep.

Compaginando con estos sitios, también se publicitó el proyecto desde el comienzo, y aunque no tiene la capacidad de alcance que los anteriores, ni mucho menos, en el sitio web (blog) del autor. En la ***Figura 30*** podemos ver una imagen del mismo haciendo referencia a alguna de las noticias relacionadas con el proyecto.

[86] http://freshmeat.net/
[87] https://www.ohloh.net/

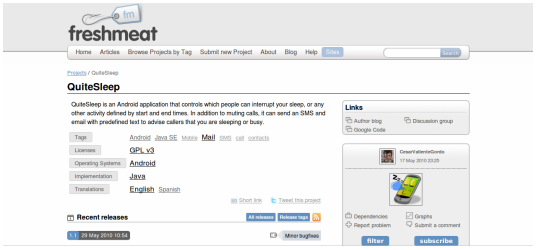

Figura 28: *QuiteSleep en Freshmeat [36]*

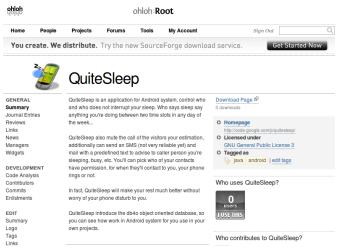

Figura 29: *QuiteSleep en Ohloh [37]*

Sitios web donde dedicados a aplicaciones Android (pero en la parte de market) y que han referenciado también el proyecto han sido *AndroidZoom*, *AndroLib*, *Bubiloop* [88], *Appbrain*[89] y *Cyrket*[90].

Figura 30: *Captura de pantalla del blog del autor con referencias al proyecto[38].*

Cada anuncio nuevo del proyecto, se ha anunciado también en el *twitter* del autor[91].

A lo largo de todo este tiempo y después de la primera publicación en el Market de Google, se pasó a publicitar también la aplicación en importantes sitios, como el sitio web más importante del mundo hispano dedicado a Android en español **android.es**[92], y quizás el sitio web

[88] http://bubiloop.com/

[89] http://www.appbrain.com/app/es.cesar.quitesleep

[90] http://www.cyrket.com/p/android/es.cesar.quitesleep/

[91] https://twitter.com/CesarValiente

[92] http://www.android.es/quitesleep-un-proyecto-floss-en-android.html

dedicado a Android más importante de las Islas Canarias, **androizados**[93] en el que hicieron una reseña a la aplicación.

Aparte de estos sitios web, también se pasó a publicitar la aplicación de manera indirecta gracias a un artículo en el prestigioso sitio web **DZone** (anteriormente mencionado) donde se publicó un breve artículo sobre la utilización de la base de datos db4o en la aplicación para que sirviera como ejemplo a desarrolladores interesados en utilizar db4o en sus aplicaciones.

6.5. Algunos números

Siempre es interesante conocer estadísticas de uso del software que uno crea, y poder ver si interesa a los usuarios, etc.

La aplicación, como hemos dicho anteriormente, fue publicada en Mayo de 2010, y tenemos los datos iniciales que teníamos cuando redactamos la pasada memoria del Proyecto Fin de Máster, con lo que podemos comparar como ha ido evolucionando la aplicación en cuanto a números se refiere desde casi el inicio hasta hoy en día.

6.5.1. Números en los markets

Cuando finalizamos de redactar la pasada memoria sobre QuiteSleep, teníamos **320 descargas**, esto era a principios/mediados de Junio 2010, este número no era para nada malo, por aquel entonces había en el market oficial de Google unas 50.000 aplicaciones, y la notoriedad en el era realmente difícil de conseguir.

En cuanto a puntuación, se tenía una valoración de cinco votos con *una media de 3 estrellas*, no estaba nada mal si tenemos en cuenta que los usuarios, al utilizar la aplicación, si esta da algún problema o no funciona como ellos creen que debe funcionar, juzgan de forma rigurosa a la baja.

También cabía destacar que QuiteSleep estaba instalada en ese momento en **114 dispositivos**, un 35 % del total de descargadas/actualmente-instaladas, estadísticas en las cuales se veía claramente, que los fallos iniciales hicieron daño. Desde que se actualizó con la siguiente revisión, ambas cantidades, descargas y aplicaciones actualmente instaladas, se vieron incrementadas.

[93] http://www.androizados.com/aplicacion-de-android-quite-sleep/

Actualmente casi dos años después de aquella primera instalación tenemos los siguientes datos a día de 24/05/2012:

- Descargas (en Google Play): **10.588**, un número muy bueno y que hace que nuestra aplicación esté en el rango de aplicaciones de 10.000 a 50.000 descargas, difícil de conseguir normalmente y menos en aplicaciones personales y FLOSS.
- Usuarios con la aplicación activa: **699**, como vemos no todo el mundo deja la aplicación instalada una vez descargada, como hemos visto a lo largo de la vida de esta aplicación, es muy difícil que la misma funcione de forma correcta en todos los dispositivos, no solo que se vea igual, o que responda igual, nuestra aplicación realiza algo muy concreto que muchas veces las diferentes operadoras o fabricantes hacen difícil realizar ese tipo de acciones (silenciar y bloquear llamadas).
- **73 ratings**, una media 3.7 con lo que es una media muy buena a nuestro modo de ver.
- Una media de 10 instalaciones diarias en el último mes.
- **10.236** descargas en el otro market donde QuiteSleep también está publicada, Goapk (mercado Chino), con lo que también es un número genial para nuestras pretensiones.
- **64** descargas en el Amazon Appstore, como vemos en este market el numero de descargas es infinitamente más pequeño que el del resto de los markets, con lo que podemos pensar, que este market no ha despegado totalmente, o la forma de búsqueda de apps no es muy eficaz, o que nuestra aplicación no interesa mucho a los usuarios de ese market (el Kindle Fire de Amazon no tiene la posibilidad de realizar/recibir llamadas), etc.

6.5.2. Números en la forja

Cuando defendimos la memoria del anterior proyecto, con respecto a los números que teníamos en la forja no eran demasiados, pero si muy buenos para el poco tiempo que llevaba la aplicación publicada, en estos números estaban incluidos el binario de la aplicación, el paquete del tarball y el de la documentación:

- **tarball**, 20 descargas.
- **.apk**, 19 descargas.

- **api doc**, 8 descargas.

Dos años después tenemos muchos más datos y bastante interesantes e importantes:

- **tarball (todas las versiones)**: 1441 descargas.
  - **1.0**: 138 descargas.
  - **1.1 (Beta)**: 176 descargas.
  - **1.1**: 285 descargas.
  - **2.0**: 842 descargas.
- **.apk (todas las versiones)**: 1618 descargas.
  - **1.0**: 106 descargas.
  - **1.1 (Beta)**: 165 descargas.
  - **1.1**: 265 descargas.
  - **2.0**: 826 descargas.
  - **2.0.4**: 210 descargas.
  - **3.0**: 46 descargas.
- **api doc (todas las versiones)**: 899 descargas.
  - **1.0**: 151 descargas.
  - **1.1 (Beta)**: 177 descargas.
  - **1.1**: 164 descargas.
  - **2.0**: 407 descargas.
- **Otros documentos**: 1050 descargas.
  - **db4o en QuiteSleep**: 373 descargas.
  - **Transparencias** utilizadas en la TLP2K10: 177 descargas.
  - **Transparencias** utilizadas en la defensa del anterior proyecto: 216 descargas.
  - **Memoria** del anterior Proyecto Fin de Máster: 284 descargas.

Como vemos tenemos unos números más que interesantes, con multitud de descargas de los diferentes recursos que ofrecemos a quien quiera utilizarlos.

Personalmente, cabe destacar que el documento de la memoria del anterior Proyecto Fin de Máster ha sido descargada 284 veces, con lo que es un orgullo que haya 284 copias de mi memoria circulando alrededor del mundo.

Cuando publicamos la anterior versión no teníamos ni gente que había guardado nuestro proyecto como favorito ni clones de nuestro proyecto, pero ahora si tenemos datos correspondientes a esto:

- Está marcado como favorito para **64 usuarios**.
- Ha sido clonado **50 veces** (en la forja de Google Code).

Vemos y nos llena de satisfacción ambos datos, es favorito de bastantes desarrolladores y ha sido clonado bastantes veces en la misma forja de Google Code, seguramente muchas más directamente en los equipos personales de los desarrolladores.

Correspondiente al número de *commits* que hemos realizado, tenemos **45 commits** en total en nuestra forja (todos hechos por mi), efectivamente no suenan a muchos commits pero podemos explicarlo basándonos en lo siguiente:

- La primera versión de QuiteSleep fue *commiteada* cuando estuvo completamente terminada y lista para su publicación en los markets.
- El resto de commits desde esa versión a la 2.0 fueron pequeñas soluciones a bugs o la traducción al chino.
- La versión 2.0 de QuiteSleep fue commiteada siguiendo el mismo procedimiento que la 1.0 es decir cuando estuvo completamente terminada.
- Desde esa versión se han ido realizando diferentes commits más asiduos llegando el punto más alto durante el trabajo de esta release, en la que teniendo en cuenta todo lo aprendido en el máster de commitear de forma asidua hemos commiteado cuando había cambios hechos y funcionales sin esperar al finalizar la release, con lo que hemos seguido mucho más el modo de desarrollo de cualquier proyecto de software libre más activo.

En general vemos que tenemos unos números muy interesantes, si nos hubieran dicho que después de dos años íbamos a tener estos números tanto en los market como en la forja no nos lo hubiéramos creído, con lo que nos podemos sentir orgullosos.

6.6. Logros y actividades

Desde el inicio del proyecto se consiguieron diferentes *logros* correspondientes a diferentes actividades que realizamos desde el principio.

6.6.1. Android Developer Lab 2010

En Febrero de 2010, acudí a un evento organizado por Google y GSyC LibreSoft, llamado **Android Developer Lab**, que se trataba una charla en la que acudía un ponente y empleado de Google para contarnos aspectos relacionados con el sistema operativo y el desarrollo. En esta sesión acudió **Reto Meier** *evangelizador* de Android para Europa, África y medio Asia. En la charla además a los presentes se nos obsequió por parte de Google de un *Nexus One* que es ni más ni menos que el teléfono que Google sacó a comienzos de este año.

6.6.2. AndroidStartup

Durante todo el proceso se acudí a diversos sitios web de referencia en el desarrollo de Android, y en el transcurso y visita de estos sitios web, un día encontré una nueva comunidad de desarrolladores Android en Madrid llamada ***AndroidStartup*** *[39]*, en ella comentaban en un post que les interesaba crear reuniones donde hablar del desarrollo en Android, lo que engloba a este sistema operativo, hablar de posibles modelos de negocio que se pueden abrir, *networking*, etc. yo me apunté y desde ese momento he sido, con los dos miembros fundadores, un valor muy importante para la comunidad ya que me puesto en contacto en numerosas ocasiones con la mayoría de grupos Android españoles como son ***android.es***, el grupo ***LibreSoft*** *[40]* del grupo GSyC de la Universidad Rey Juan Carlos, ***Android-Spa***[94] e incluso ***Google*** con los que mantengo con todos buenas relaciones en cuanto a intereses comunes, que no es más, que la difusión del conocimiento.

[94]http://www.android-spa.com/

Figura 31: *Google I/O 2010 Madrid.*

Figura 32: *Ponencia en la Tenerife Lan Party 2K10.*

En Mayo de 2010 fui invitado a presenciar la *keynote* de **Google I/O**[95] en las oficinas de Google en Madrid, donde se me ofreció si quería realizar alguna presentación una vez acabada la keynote, así que acepté, y el *20/5/2010* realicé dos presentaciones, una de mi grupo de Android junto con mis compañeros, y otra de mi aplicación para darle así la mayor publicidad posible, ya que era un evento orientado para desarrolladores principalmente.

En estos dos años de AndroidStartup, hemos pasado por diferentes etapas, etapas en las que teníamos mucha actividad y etapas en las que menos, en la anterior memoria del proyecto comenté que había sido invitado a dar una charla en la Tenerife Lan Party 2K10 en Julio, y así fue, acudí a dar una charla que estuvo basada en el desarrollo de la aplicación y por qué licenciarla como software libre, la charla fue muy bien acogida y me sirvió para vivir una experiencia inolvidable.

En la ***Figura 31 y 32*** se pueden ver un par de imágenes de mis presentaciones en la Google I/O 2010 y en la Tenerife Lan Party 2K10.

A raíz del Google I/O 2010, también conocimos a los chicos de **GTUG Madrid**[96] (Google Technology User Group), con los que desde entonces hemos hecho eventos juntos, o asistido a charlas que han ofrecido y viceversa, etc.

Esa invitación al Google I/O de 2010 fue repetida en 2011 a la que también acudí como invitado y en la que seguí conociendo gente como los chicos de **GTUG Granada**[97] y más gente

[95] http://code.google.com/intl/es-ES/events/io/2010/

[96] http://sites.google.com/site/gtugmadrid/

[97] http://www.gtugs.org/chapter.jsp?id=475001

del *mundillo Android*.

Debido a la colaboración con Google España, los miembros de AndroidStartup fuimos invitados al *Google DevFest* en la Universidad Complutense de Madrid en Septiembre de 2010 en la que seguimos conociendo a gente y estrechando lazos.

AndroidStartup también fue miembro participante del **Codemotion2012**[98] evento internacional muy importante sobre diferentes tecnologías que este año hacia su parada en Madrid.

Últimamente la actividad de AndroidStartup está en su grado más alto, con reuniones mensuales en los que quien quiera puede dar una charla sobre algo interesante o experiencias propias relacionadas con Android. Normalmente se suelen realizar en la sala *Ciball* de Madrid, pero desde hace poco tenemos también relación con **Tuenti**[99] con algunos de sus ingenieros y hemos dado alguna charla en su sede de Madrid. También tenemos excelente relación con diversas empresas como por ejemplo **Samsung España**[100] que se ponen en contacto con nosotros para ofrecer puestos de trabajo, que esto ahora mismo es algo muy importante y necesario en nuestro país y para nuestros miembros.

Actualmente AndroidStartup cuenta con **190 miembros** con lo que podemos decir, que seguramente es la comunidad de desarrolladores Android localizada en Madrid más importante y activa.

6.6.3. dVP2010

Algo que me ocurrió muy importante fue debido al uso y difusión de la bbdd utilizada en QuiteSleep, db4o, fui nombrado en Verano de 2010 **dVP2010**[101] (db4o Valuable Professional) que es un premio a nivel mundial (65 premiados) que otorga **Versant** a los profesionales que demuestran su *expertise* y fomento de su producto a lo largo del año. Seguramente yo fui uno de los primeros en utilizar db4o en una aplicación Android, y el premio fue una alegría y sorpresa para mi.

[98] http://www.codemotion.es/

[99] http://www.tuenti.com/

[100] http://www.samsung.com/

[101] http://community.versant.com/Blogs/Db4o/tabid/197/entryid/919/Default.aspx

6.6.4. Trabajo

Pero seguramente el logro más importante que me ha dado QuiteSleep es que en Mayo de 2011 conseguí un trabajo en Amsterdam (Países Bajos) como desarrollador Android, el código de QuiteSleep fue lo que enseñé como mi trabajo cuando me pidieron referencias sobre aplicaciones publicadas y código con lo que es genial el poder utilizar algo que realizaste como Proyecto Fin de Máster en tu vida profesional.

Este Junio, después de la defensa de este proyecto el día 1 de Junio, empezaré una nueva aventura profesional en Berlín (Alemania) y QuiteSleep sigue teniendo mucho la culpa de la consecución del nuevo trabajo ya que siempre me ayuda a poder dar referencias.

7. Conclusiones

Para finalizar, las conclusiones obtenidas después de realizar este proyecto en los que hemos creado una nueva release de una aplicación ya madura, y un estudio relacionado con *su vida* son:

- QuiteSleep goza de un buen lugar tanto en los diversos markets donde está publicada como en lo que a proyecto de software libre se refiere.
- No han salido muchas más aplicaciones similares a la nuestra, o al menos que puedan competir en funcionalidad, descargas, y libertad de uso.
- Aun siguiendo desarrollándose por un solo desarrollador QuiteSleep goza de muy buena salud, no siendo abandonada nunca, cuando lo normal en los proyectos fin de máster/carrera es lo que suele suceder.
- El desarrollo de esta nueva release me ha permitido aprender diversas cosas respecto a los nuevos sdk de Android, el uso de nuevos patrones de interacción y librerías que lo implementan, el uso de Fragments, que poco a poco tienden a cambiar el antiguo modo de mostrar la información y construir los layouts que tiene Android.
- Los diversos cambios que ha sufrido la aplicación han sido realmente buenos tanto por ofrecer una nueva apariencia e interacción al usuario, como internamente en la distribución de paquetes, mejora en funcionalidad, etc.
- A lo largo de todo este tiempo hemos podido ver como la fragmentación en Android es algo bueno y malo, bueno porque como software libre que es los diferentes fabricantes pueden adaptar este sistema operativo a sus necesidades, crear capas intermedias para cambiar la interfaz gráfica, etc. pero esto que es una virtud también es un problema para los desarrolladores ya que nos debemos enfrentar a diferentes resoluciones de pantalla, diferentes densidades de píxel, diferentes comportamientos en las diferentes adaptaciones del sistema operativo, con lo que supone mucho más trabajo del que ya es el de por si desarrollar una aplicación para dispositivos móviles.
- Después de dos años, la competencia en Google Play es muy dura, con un número de aplicaciones ahora de alrededor de 500.000 (hace dos años eran *"tan solo"* 50.000) destacar

en el market es algo muy difícil, que encima para aplicaciones sin diseños extraordinarios se hace todavía más difícil destacar en el, ya aunque solo sea por el aspecto gráfico.

- Los usuarios muchas veces no hacen caso a los detalles puestos por la aplicación ni a la funcionalidad real de esta, en ocasiones quieren que realice algo que en realidad no hace.
- La información que se ofrece al desarrollador en el market de Google ha mejorado de forma importante en estos dos años, quejándonos en la anterior memoria, los datos que ahora nos provee Google son bastantes, importantes, e interesantes.
- Se ha seguido desarrollando un proyecto creado en 2010, con releases menores que solucionaban pequeños problemas o añadían pequeñas características nuevas y releases mayores con importantes cambios.
- La compatibilidad de licencias es un asunto muy importante que muchas veces se descuida, pero es vital por los aspectos éticos y legales que pueden derivar. En muchas ocasiones querremos utilizar dos licencias, pero a la hora de usarlas tenemos que ver si son compatibles, con lo que puede que tengamos que hacer ajustes, utilizar otras librerías (con otras licencias), ver si se puede publicar bajo otra licencia que también satisfaga nuestros deseos, etc. Con la licencia que escogimos GNU GPLv3, vemos que acertamos plenamente puesto que a la hora de expandir el proyecto con nueva funcionalidad y utilizando nuevas librerías, el estar usando una de las principales licencias ayuda a que sea más probable que el nuevo software sea compatible con el nuestro.
- Al publicar un proyecto como software libre, y publicitarlo al menos un poco, se obtienen referencias de otros sitios web relacionados, mensajes de usuarios, desarrolladores, etc.
- La evolución de un proyecto de software libre ofrece tomar varias vías para crear nuevas implementaciones, nuevos caminos por los que orientar la aplicación, etc.
- La creación de un proyecto de software libre, y las vías de acceso que ofrecen los sitios web de referencia al mismo, hace que el número de visitas a el crezca de forma importante.
- La creación (involucración) de un proyecto de software libre, me ha permitido de una forma u otra, adentrarme en la comunidad de desarrolladores de Android, conociendo

gente, compartiendo conocimientos, teniendo nuevas experiencias, etc.

- Nunca me podría haber imaginado la repercusión que este proyecto tendría en mi vida, desde la creación de la comunidad de Android, participación en diversos eventos, premios, trabajo, sin duda gran parte de culpa de todo esto lo tiene el haber publicado la aplicación como software libre.

Como conclusión general de este trabajo y por tanto de la evolución de este proyecto de software libre, podemos decir que el software, como sabemos, nunca está terminado ni libre de fallos ni a gusto de todos, el software como cualquier otro producto puede fallar, gusta más o menos, etc. En la evolución de este proyecto hemos visto como un buen feedback ayuda a que el desarrollador pueda arreglar de forma más eficaz los errores que se dan, mejorar el software basándose en las sugerencias de los usuarios, etc.

Hemos visto como un proyecto de software libre puede generar multitud de acciones a su alrededor, desde que la gente se descargue tu código para verlo, creación de comunidad en torno o a partir de el, estar dentro del *mundillo del software libre*, etc. y no solo relativas al software, si no que nos puede dar notoriedad, prestigio, etc. pero esto último no debe ser buscado, sino que puede llegar o no según como se den las cosas.

Para finalizar, este proyecto me ha servido para crecer tanto profesionalmente como personalmente, y a lo largo de estos dos años en los que este proyecto lleva vivo, podemos ver como he evolucionado junto a el.

7.1. Futuras ampliaciones

En la anterior memoria, se incluyo esta misma sección en la que decíamos que íbamos a mejorar a lo largo de ese año 2010, como vemos no todas se hicieron realidad, ni incluso si lo hicieron fue durante ese año, como vemos esto también es un factor del software libre, y es que no puedes, o al menos es muy difícil, planificarte para realizar algo de forma altruista cuando tienes una vida. Como hemos visto a lo largo del máster en diversas asignaturas, los desarrolladores de software libre, trabajan cuando pueden y cuando quieren, con lo que finalmente y sin quererlo, hemos seguido exactamente el mismo patrón.

Algunas de las acciones (las más importantes) que planificamos en la anterior memoria, y ver las que tenemos en mente realizar para siguientes versiones son.:

- Filtrado de llamadas de números **desconocidos** –>*Realizado*.
- Filtrado de llamadas de **todos** los números –>*Realizado*.
- Exportación de la configuración actual de los contactos (bloqueados, sms y email para enviar respuesta) a fichero de algún tipo (¿xml? ¿json?) para cuando se realice una nueva actualización/importación de contactos, mantener la vieja configuración de contactos que ya estuvieran configurados. Esta exportación también es válida para mantener una copia de seguridad de la configuración. –>*No, pendiente para la próxima release porque es algo que **debemos** hacer*.
- Creación y gestión de grupos de usuarios –>*Pendiente.*
- Ayuda en formato html, accesible vía web para permitir acceso no solo desde el dispositivo móvil –>*La ayuda ahora es HTML pero no está accesible a través del web (si a través de la forja de los ficheros de ayuda, pero no de forma directa)*.
- Publicidad en sitios de referencia Android (blogs, webs, etc.) –>*Realizado*.

7.2. Implementaciones derivadas

Como en la pasada memoria, derivando de este proyecto y gracias a que es un proyecto de software libre, un abanico enorme de posibilidades se puede dar, por ejemplo:

- Software derivado de este con misma o diferente funcionalidad.
- Módulos separados que doten de más funcionalidad a la aplicación.
- Separar más profundamente la aplicación en módulos para que estos sean utilizados en otras aplicaciones o software.
- Utilización de otro hardware de ayuda, como GPS, acelerómetros, giroscopio, etc. para dotar de nueva funcionalidad a la aplicación, como por ejemplo geolocalizarnos y evitar llamadas de otro país en el que estemos, o comprobar en que posición está el teléfono y responder a las llamadas de una forma y otra dependiendo de la misma, etc.
- Se podría utilizar la cámara para enriquecer el feedback al usuario que nos llama, como por ejemplo enviar una foto de lo que está pasando en ese preciso instante, etc.

Así la riqueza que nos da que un proyecto esté basado en una idea útil y que sea completamente software libre es muy interesante de cara a realizar nuevas implementaciones al mismo.

8. Apéndice: Manual de usuario

A continuación vamos a pasar a explicar de forma sencilla y visual como se utiliza la aplicación QuiteSleep en su última versión, es decir la que hemos implementado para este proyecto y que mantiene además toda la funcionalidad de anteriores versiones.

El manual se divide en las diferentes secciones (antiguamente pestañas, hoy pantallas accesibles haciendo swipe de un lado a otro) que tenemos en la aplicación, seguiremos un orden de izquierda a derecha de las mismas, por supuesto el usuario puede configurar la aplicación siguiendo el orden que quiera pero es más intuitivo seguir el orden mencionado.

La primera vez que se instala la aplicación, y si nunca ha estado previamente instalada, se realizará una primera sincronización de los datos de los contactos del usuario. Solo se utilizarán contactos que tengan al menos un teléfono, ya que los que no tienen, no nos interesan.

8.1. Contactos

En la primera pestaña, *Contactos*, configuramos los contactos que establecemos como bloqueados.

Figura 33: *Pestaña Contactos.*

Hay tres botones:

- **Añadir**

Este botón lo utilizaremos para añadir contactos al estado de bloqueados para que la aplicación filtre sus llamadas. Como vemos en la ***Figura 34***, una vez pulsado el botón de Añadir, se muestra una lista con todos los contactos que tenemos, en el ActionBar podemos ver también como tenemos un botón que sirve para añadir todos los usuarios como bloqueados en un solo paso.

Figura 34: *Lista de contactos listos para bloquear.*

Si pulsamos el botón de *añadir todos* se notificará al usuario que que esta acción puede llevar un tiempo dependiendo de la cantidad de contactos que tengamos en nuestra base de datos. ***Figura 35***.

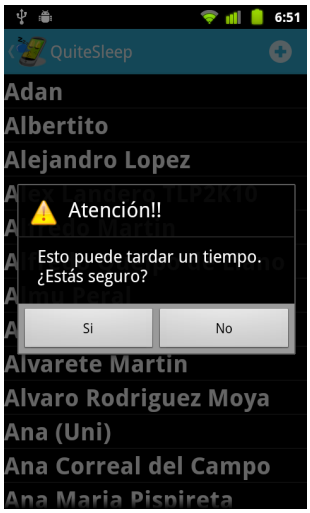

Figura 35: *Aviso antes de añadir todos los contactos a bloqueados.*

Si en vez de añadir todos, seleccionamos uno de los contactos para añadir de forma individual se nos mostrará una pantalla con los datos de ese contacto, en donde podremos configurar que números de teléfono serán usados para enviar SMS si nos llama desde alguno de ellos, o que direcciones de correo utilizar para enviar feedback, además de añadirlo como contacto bloqueado. ***Figura 36***.

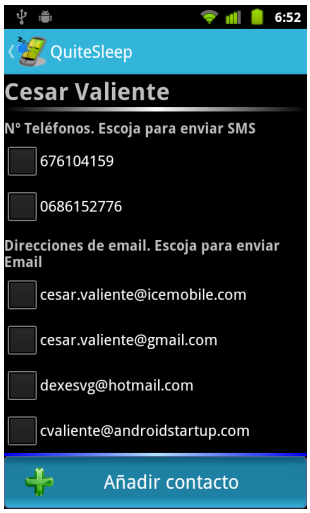

Figura 36: *Detalle de un contacto al añadirlo como bloqueado.*

- **Quitar y editar**

  Este botón se usa cuando queremos quitar algún contacto que tenemos como bloqueado de la lista de bloqueados o si queremos editar la configuración de este, modificando los números de teléfono o direcciones de correo electrónico que se utilizarán para el envío de mensajes. Sigue el mismo patrón que el botón *Añadir*, cuando lo pulsamos nos aparecerá una lista y pulsando el botón que tenemos en el ActionBar podremos quitar todos los contactos que están como bloqueados en un solo paso.

  En la figura ***Figura 37***, se puede apreciar la pantalla creada para este proceso.

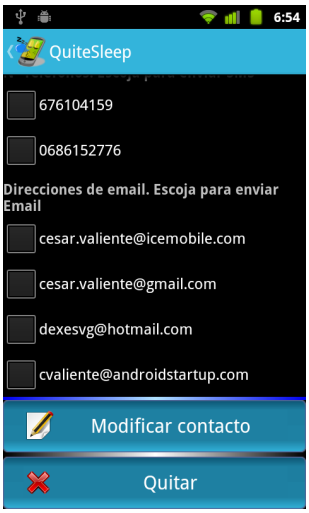

Figura 37: *Detalles de un contacto que ya está bloqueado.*

- **Sincronizar**

  Pulsando este botón se producirá una nueva sincronización/importación de contactos. Esto es útil para actualizar los contactos que maneja de forma interna el programa si hemos añadido nuevos contactos al teléfono, o si hemos modificado alguno que ya estaba.

  Cuando el usuario aprieta el botón se le comunicará que este proceso eliminará la configuración actual y que puede llevar un tiempo dependiendo de los contactos. Esta confirmación también eliminará el historial de llamadas de la aplicación. ***Figura 38***.

Figura 38: *Aviso al realizar una nueva sincronización/importación de contactos.*

8.2. Horario

En esta segunda pestaña vamos a realizar la gestión correspondiente con la franja horaria y diaria en la que la aplicación podrá realizar su cometido.

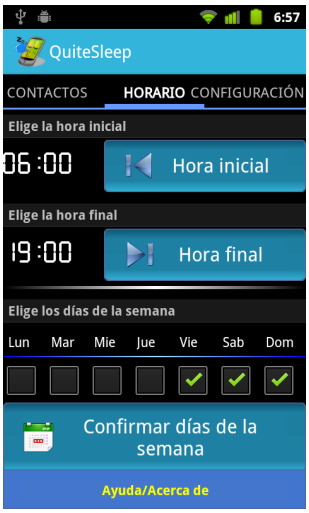

Figura 39: *Pestaña de gestión de horario (franja horaria/diaria).*

Como vemos hay tres botones que realizan las siguientes funciones:

- **Hora inicial**

  Con este botón especificaremos la hora inicial en la que la aplicación bloqueará las llamadas de los contactos bloqueados. Al pulsarlo nos aparecerá un *dialog* donde especificaremos hora y minuto, para proceder a aceptar o cancelar el cambio realizado.

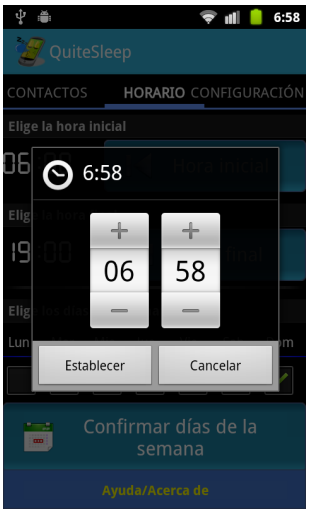

Figura 40: *Especificación de la hora en la franja horaria.*

- **Hora final**

  Es exactamente igual que el caso anterior, una vez pulsado el botón nos aparecerá un cuadro de dialogo donde podremos introducir la hora de fin de la franja horaria.

- **Días de la semana**

  Como se muestra, hay siete *checkboxes* correspondientes con los que podremos definir que días de la semana funcionará la aplicación si esta está activa.

  Una vez establecido los días que queremos, los guardamos pulsando el botón de *Confirmar días de la semana*, nos aparecerá una notificación y ya estará todo listo.

8.3. Configuración

En esta pestaña estableceremos los parámetros de configuración de los mensajes SMS y de correo electrónico, así como activar o desactivar su envío, configurar como queremos que QuiteSleep trate las llamadas entrantes, especificar el tipo de bloqueo y activar/desactivar el proceso general de QuiteSleep.

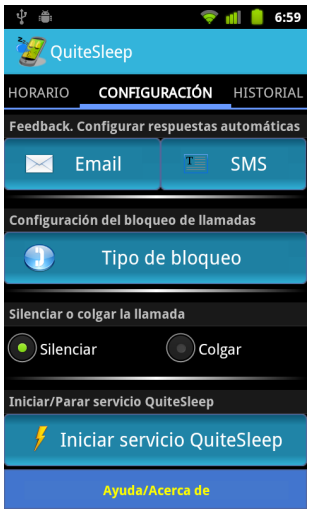

Figura 41: *Pestaña de configuración de servicios.*

- **Email**

  Si pulsamos este botón nos aparecerá una ventana donde podremos establecer los parámetros de :

- Usuario (de la cuenta de Gmail).
- Contraseña (de la cuenta de Gmail).
- Asunto (subject) del correo electrónico a enviar.
- Contenido (body) del correo electrónico a enviar.

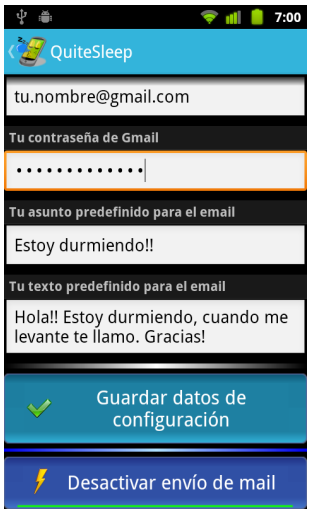

Figura 42: *Configuración del correo electrónico.*

Podremos guardar la configuración establecida, y activar o desactivar el servicio de envío de correo electrónico.

Al activar el servicio se notificará al usuario que usar esta característica puede llevar un coste dependiendo de nuestro operador. El usuario puede volver a desactivar, o modificar los datos cuando quiera.

Figura 43: *Aviso de la posible tarificación extra según el operador al activar el envío.*

- **SMS**

  Al igual que el botón de *Email*, con este botón configuraremos y activaremos el servicio de envío de SMS a los contactos que tengamos especificados que se le envíe un mensaje de texto.

  La opción a configurar es solo el texto del mensaje, ya que los mensajes de texto (SMS) no tienen título ni ningún otro dato.

  Podremos guardar la configuración (texto del mensaje) y activar o desactivar el servicio de envío de SMS, donde también nos será notificado al activarlo, que puede estar ligado a una tarificación extra según nuestro operador.

- **Tipo de Bloqueo**

  Esta opción fue introducida en QuiteSleep 2.0 para dar más control al usuario sobre que hacer cuando llega una llamada.

Figura 44: *Tipos de bloqueo disponibles.*

En la **Figura 44** podemos ver los cuatro tipos diferentes de bloqueo que ofrecemos:

- **Bloquear todas las llamadas**, si el usuario especifica esta acción no importará que usuarios tenga bloqueados, QuiteSleep bloqueará todas las llamadas entrantes.
- **Bloquear solo contactos bloqueados**, esta es la opción por defecto utilizada en QuiteSleep y única opción ofrecida en versiones 1.xx. Esta opción hace que la aplicación solo bloquee llamadas de los contactos que hemos definido como bloqueados.
- **Bloquear solo a desconocidos**, esta opción bloqueará solo llamadas desconocidas que no estén entre nuestros contactos, indistintamente de los contactos que tengamos seleccionados como bloqueados.
- **Bloquear desconocidos y bloqueados**, en esta opción, el sistema bloqueará las llamadas de teléfonos que no tengamos en nuestra agenda y también de aquellos contactos que hayamos seleccionado como bloqueados.

- **Silenciar o colgar la llamada**

  Esta es otra de las acciones introducidas desde QuiteSleep 2.0. Aquí ofrecemos al usuario dos opciones:

  - **Silenciar** la llamada entrante.

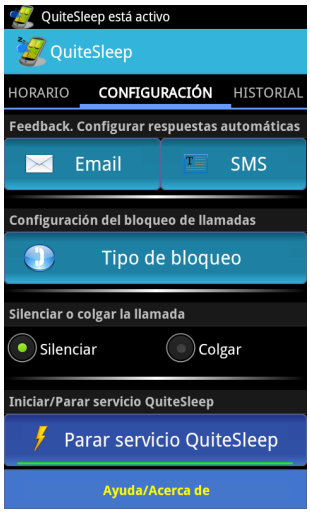

Figura 45: *Servicio QuiteSleep arrancado.*

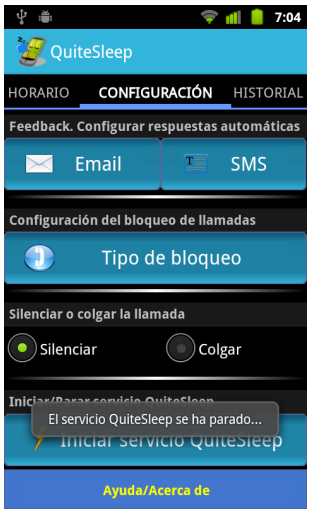

Figura 46: *Servicio QuiteSleep parado.*

- **Colgar** la llamada entrante.

Por supuesto, estas acciones son inútiles si no está activado el servicio QuiteSleep y configurado el resto de acciones de la aplicación.

- **Activar/Desactivar servicio QuiteSleep**

  Pulsando este botón activaremos o desactivaremos (según su estado anterior) el servicio QuiteSleep. Al activarlo se nos notificará mediante una notificación en la barra de estado, y al desactivarlo lo hará mediante una notificación de tipo *toast*. ***Figuras 45 y 46***.

8.4. Historial

En esta pestaña podremos ver el historial de llamadas filtradas que ha procesado el servicio de QuiteSleep.

En la ***Figura 47*** podemos ver como se muestra la lista de las llamadas que ha procesado la aplicación como son:

- Contacto.
- Teléfono.
- Fecha y hora.
- Envío de SMS.

- Envío de correo electrónico.

Figura 47: *Pestaña de historial con los botones accesibles tras pulsar la tecla de menú.*

Estos elementos no son *clickables* con lo que por mucho que los pulsemos no conseguiremos nada.

En el ActionBar ahora tenemos dos nuevos botones:

- **Actualizar**

  Si pulsamos este botón, se actualizará la lista de llamadas del historial, eliminando la lista y actualizándola con nuevos datos. Al pulsar, y para que el usuario esté seguro de lo que realiza se pedirá confirmación al usuario indicando que esta tarea puede llevar un tiempo dependiendo de los datos a actualizar.

- **Borrar**

  Si pulsamos este botón, la lista se quedará vacía y se eliminarán los datos correspondientes al histórico de llamadas filtradas. Antes de realizar esta acción, se pedirá confirmación al usuario ya que esta acción es irreversible.

8.5. Ayuda y Acerca de

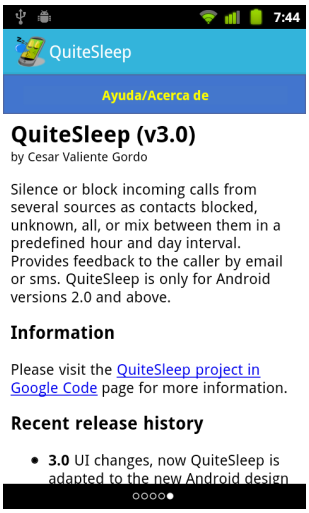

Figura 48: *Widget mostrando la Ayuda e información sobre la aplicación.*

Las secciones de *Ayuda y Acerca de* que en anteriores versiones se mostraban cuando se pulsaba el botón *menú* en las pantallas de configuración principales, en esta versión están disponibles en un widget (SlidingDrawer) en la parte inferior de la pantalla. Este widget, si se pulsa o se arrastra hacia arriba mostrará la información deseada y para pasar por las diferentes secciones de la ayuda o el acerca de, solo habrá que hacer *swipe* hacia izquierda o derecha, teniendo un indicador en la parte inferior que nos mostrará siempre donde nos encontramos.

Aclarar que esta información (ambas, ayuda y acerca de) está disponible solo en ingles.

- **Ayuda** Aquí se mostrará información del uso de la aplicación (parecido al manual de usuario aquí explicado). Se divide en cuatro secciones que se corresponden con las cuatro partes en las que está dividida la aplicación:
  - Gestión de contactos.
  - Gestión de la franja horaria y diaria.
  - Gestion general del bloqueo y feedback utilizado. Activación del servicio QuiteSleep.
  - Información sobre las llamadas controladas por la aplicación en el historial.
-

- **Acerca de** Aquí se mostrará información de la aplicación, autor, licencia, contacto, etc. En esta información aparece el enlace al sitio web del proyecto, y la dirección de correo electrónico personal del autor para hacerle toda clase de comentarios, si se pulsa alguno de ellos el sistema abrirá la aplicación correspondiente, ya sea navegador web o cliente de correo electrónico para realizar las acciones pertinentes.

Como vemos el funcionamiento de la aplicación es muy sencilla y configurarla nos llevará un tiempo determinado por el número de contactos que queramos configurar, una vez configurada, no habrá que realizar nada más salvo cuando se quiera realizar nuevos ajustes.

Ni siquiera habrá que acceder a la aplicación cuando se encienda el teléfono de nuevo, porque QutieSleep se activará (si se dejó activada) por ella misma sin intervención alguna por parte del usuario y notificándoselo en la *barra de estado*.

Bibliografía

[1] **Máster Software Libre (Libresoft, URJC)**

http://master.libresoft.es/

[2] **Memoria QuiteSleep 2010**

http://quitesleep.googlecode.com/files/MemoriaQuiteSleep.pdf

[3] **Android**

http://www.android.com

[4] **FSF**

http://www.fsf.org/

[5] **Apple**

http://www.apple.com/

[6] **CallFilter**

http://www.telemarks.co.jp/en/products/callfilter.htm

[7] **Sweet Dreams**

http://www.inizziativa.com/productos/

[8] **Locale**

http://www.twofortyfouram.com/product.html

[9] **db4o**

http://www.db4o.com/

[10] **Java Mail API**

http://java.sun.com/products/javamail/

[11] **ActionBarSherlock**

http://actionbarsherlock.com/

[12] **ViewPagerIndicator**

http://viewpagerindicator.com/

[13] **Google**

http://www.google.com/

[14] **Open Source**

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

[15] **Apache License**

http://www.apache.org/licenses/

[16] **SQLite**

http://www.sqlite.org/

[17] **The Busy Coder's Guide to Android Development**

CommonsWare LLC. ISBN 978-0-9816780-0-9 Mark L. Murphy

[18] **Professional Android 2 Application Development**

Wiley Publishing, Inc.Wrox Programmer to Programmer ISBN 978-0-470-56552-0 Reto Meier

[19] **Activity**

http://developer.android.com/intl/de/reference/android/app/Activity.html

[20] **ContentProvider**

http://developer.android.com/intl/de/guide/topics/providers/content-providers.html

[21] **Intent**

http://developer.android.com/intl/de/reference/android/content/Intent.html

[22] **Services**

http://developer.android.com/intl/de/reference/android/app/Service.html

[23] **GNU GPL**

http://www.gnu.org/licenses/gpl.html

[24] **Licencia comercial**

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

[25] **Software propietario**

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

[26] **OSI**

http://www.opensource.org/

[27] **Inizziativa**

http://www.inizziativa.com/

[28] **TwoFortyFourAM**

http://www.twofortyfouram.com/

[29] **Broadcast Receiver**

http://developer.android.com/intl/de/reference/android/content/

BroadcastReceiver.html

[30] **Iconspedia**

http://www.iconspedia.com/

[31] **Open icon library**

http://openiconlibrary.sourceforge.net/

[32] **FontStruct**

http://fontstruct.fontshop.com/

[33] **Google Code**

http://code.google.com/

[34] **QuiteSleep en Google Code**

http://code.google.com/p/quitesleep/

[35] **Mercurial**

http://mercurial.selenic.com/

[36] **QuiteSleep en Freshmeat**

http://freshmeat.net/projects/quitesleep

[37] **QuiteSleep en Ohloh**

https://www.ohloh.net/p/QuiteSleep

[38] **QuiteSleep en el blog del autor**

http://memyworldandmylife.wordpress.com/

[39] **AndroidStartup**

http://www.androidstartup.com/

[40] **LibreSoft**

http://libresoft.es/

[41] **Sukiweb (web del propietario de las fotos del evento de Google I/O)**

http://sukiweb.net/