Universidad Zaragoza

**ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL**

**Departamento de Informática e Ingeniería de Sistemas**

**Ingeniería Técnica en Informática de Gestión**

---

# "SimDetect: aplicación de seguridad para la localización de dispositivos móviles Android"

**Trabajo Fin de Carrera**

**Autor: Vicente Torres Sanz**

---

**Teruel, diciembre de 2012**

Universidad Zaragoza

**ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL**

**Departamento de Informática e Ingeniería de Sistemas**

**Ingeniería Técnica en Informática de Gestión**

---

# SimDetect: aplicación de seguridad para la localización de dispositivos móviles Android


**Autor:** Vicente Torres Sanz

**Directores:**
Francisco J. Martínez Domínguez
Manuel Fogué Cortés


**TRIBUNAL**

Presidenta: Piedad Garrido Picazo

Secretario: Manuel Fogué Cortés

Vocal: Fernando Naranjo Palomino

CALIFICACIÓN: _____

FECHA: _____

# Agradecimientos

En primer lugar me gustaría agradecer a mis directores de proyecto por la libertad que me han dado en la realización de este trabajo.

En segundo lugar agradecer a mis padres, por hacerme ver que siempre hay una nueva meta que superar. A todas las personas especiales que me han hecho más humano mi paso por la Universidad, y a esas personas del zulo que hicieron que mi trabajo fuera más ameno.

Me gustaría agradecer de manera especial, a Mª Pilar Yuste, Jesús Ibáñez, Carlos Sánchez, Jorge Navarrete y Miguel Tourón, por ser mis conejillos de indias sin ponerme trabas ni tener que recurrir a la violencia.

# Índice

# Resumen

El presente trabajo presenta SimDetect, una aplicación que permite la localización y recuperación, en caso de pérdida o robo, de los dispositivos móviles basados en Android.

La aplicación realizada en el presente proyecto funciona en un modo invisible, para no ser detectada, permitiendo la localización del terminal móvil, mediante el uso de comandos enviados mediante mensajes SMS. Además, incluye diversas funcionalidades orientadas al aumento en la seguridad en el uso de Smartphones. Por ejemplo, permite realizar un bloqueo remoto del teléfono, impidiendo su utilización mientras no se introduzca la tarjeta SIM del propietario. Además, permite obtener la foto de la persona que ha robado el terminal móvil, enviándola al propietario del mismo.

Esta aplicación se ha diseñado utilizando un paradigma de orientación a eventos, reduciendo al máximo el consumo extra de batería.

**Palabras clave:** Android, seguridad Android, localización, bloquear dispositivo, SQLite, seguridad en dispositivos móviles.

# 1.- Introducción

El ser humano, también conocido bajo la denominación científica de *Homo sapiens*, se diferencia del resto de seres vivos por las capacidades mentales que éstos poseen para inventar, aprender y utilizar estructuras lingüísticas complejas, por tener la capacidad de representarlas mediante símbolos y por tener la habilidad de construir y utilizar dispositivos tecnológicos basados en la ciencia capaces de utilizar dichas estructuras para comunicarse entre ellos. Es la única especie animal conocida del universo capaz de tener estas habilidades.

Como principio del proceso de comunicación tecnológico está la invención del telégrafo en 1833 por Johann Carl, Friedrich Gauss y Wilhelm Eduard Weber, con la posterior invención posterior del teléfono en 1871 por Antonio Meucci, llegando a nuestros días, en los que gracias a la invención de los satélites una persona se puede comunicar en cualquier parte del planeta, la invención de las sondas con las cuales se puede realizar la comunicación inter-planetaria o incluso más lejana, tal y como nos muestran las sondas Pioner y Voyager, situadas ya fuera de nuestro sistema solar.

El primer teléfono móvil fue el DynaTAC 8000X, diseñado por el ingeniero de la compañía Motorola Rudy Krolopp en 1983. Su peso era de 800 gramos, y sus dimensiones eran de 33 x 4,5 x 8,9 (alto, largo, ancho) y su precio rondaba los 4000 $.

La evolución del teléfono móvil, ha permitido una gran disminución de sus dimensiones y de su peso alcanzando sin embargo un nivel muy alto de prestaciones, similar al que puede ofrecer un computador medio, lo cual ha permitido que estos dispositivos puedan incorporar nuevas funcionalidades como por ejemplo, SMS, fotografía y vídeo digital, o acceso a Internet.

Dicha evolución ha llegado hasta los llamados smartphones, o teléfonos inteligentes, los cuales han producido una revolución en la telefonía móvil, provocando que lo que era un dispositivo móvil avanzado, se haya convertido en un computador de bolsillo, dotándolos de una capacidad de procesamiento de datos elevada y por lo tanto capaz de realizar funciones que cualquier ordenador personal podría realizar.

Las aplicaciones de estos dispositivos móviles son muy amplias, pasando desde las comunicaciones entre personas, el uso de redes sociales, la toma de imágenes, entretenimiento, o las comunicaciones en los entornos vehiculares [MTC12, MCC09, FGM11].

Estos dispositivos han dado la posibilidad de unificar en un único dispositivo funciones que antes estaban dispersas en varios, como pueden ser GPS, reproductor de música y películas, pueden acceder a Internet con conexiones de banda ancha, disponen de agenda personal que se puede sincronizar en la nube, pueden hacer fotografías y vídeo en HD, algunos incluso en 3D. Sobre todo, su capacidad de procesamiento ha provocado que se puedan instalar aplicaciones de amplia envergadura, pudiendo ya no solo visualizar documentos, sino dotarle de una suite ofimática que permite crear y editar documentos, retocar fotografías, conocer los puntos más interesantes de un lugar, o instalar juegos con los que un usuario puede disfrutar de su dispositivo móvil

Parte de todas estas funcionalidades son debidas a los Sistemas Operativos que éstos dispositivos llevan instalados (Sistemas Operativos móviles), que dan la posibilidad de que varios procesos se puedan estar ejecutando al mismo tiempo y que dan soporte para interactuar con el hardware que los teléfonos traen consigo, como pueden ser la cámara de fotos, acelerómetros, bluetooth o USB.

Un sistema operativo móvil es un sistema operativo que controla un dispositivo móvil. Los principales sistemas operativos móviles son los siguientes:

**Symbian OS:** este sistema operativo fue producto de la alianza entre las empresas Nokia, Sony Ericsson, Samsung, Siemens, Psion, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi, Panasonic y Sharp, con el objetivo de crear un sistema operativo para dispositivos móviles con la idea de competir con Palm y con Windows Mobile. En 2008 fue adquirido en su totalidad por Nokia, quien en 2010 transfirió el sistema operativo a la consultora Accenture para convertirlo en una plataforma libre. Adquirió fama entre los desarrolladores de software, ya que ofrecía un principio de API que facilitaba a los desarrolladores el realizar aplicaciones para dicha plataforma. Su facilidad de utilización por el usuario y la fiabilidad ante errores hizo que fuera la plataforma móvil más utilizada. En 2011 Nokia anunció un acuerdo con Microsoft para incorporar Windows Phone en sus terminales móviles, y anunciando que Symbian tendría soporte hasta 2016.

**iOS:** sistema operativo creado por Apple, desarrollado para el iPhone, de ahí que al principio se le conociera como Iphone OS, siendo posteriormente introducido en los dispositivos iPod Touch, iPad y Apple TV. iOS deriva de MAC OS X, que a su vez está basado en Unix Darwin BSD. Su interfaz se basa en la manipulación directa de los objetos mediante el uso de pantallas multitáctiles.

A partir de iOS 4 se incorporó un sistema multitarea, aunque solo se utilizaba para aplicaciones del sistema, haciendo una congelación en segundo plano de éstas cuando se ejecutaba una aplicación de usuario. Con esto Apple solucionaba problemas de batería y rendimiento.

En 2008 se liberó el SDK que permitía realizar aplicaciones a desarrolladores de software para esta plataforma, mediante el kit de desarrollo Xcode, que permitía realizar aplicaciones escritas en Objective-C.

Otra gran idea de Apple fue la creación del denominado App Store, en el cual cualquier desarrollador podía subir su aplicación para que el usuario pudiese descargarla/comprarla. Algunas aplicaciones se podían descargar de forma gratuita, otras eran puestas en venta por un precio mínimo de $0.99. El desarrollador recibía el 70% de las ganancias que produjera la aplicación.

**Bada:** fue desarrollado por Samsung para teléfonos móviles, también es conocido como Samsung OS, y cubre tanto teléfonos de gama baja como smartphones. Samsung lo define como una plataforma con un núcleo configurable, permitiendo en uso de cualquier núcleo de Linux. Fue anunciada el 10 de noviembre de 2009, y empresas como Twiter, EA, Capcom, Gameloft y Blockbuster lo apoyaron. Fue implantado por primera vez en el Wave S8500 y en 2010 se publicó el SDK para Bada, para facilitar el desarrollo de aplicaciones, que eran programadas en C++.

Bada ofrece distintos controles de interfaz, soporta Adobe Flash, proporcionaba soporte para interactuar con distintos sensores como sensor de movimiento, acelerómetro, magnetómetro y GPS.

Bada fue criticado por algunas limitaciones que tenía, como por ejemplo API's no abiertas, no dar acceso a la bandeja de SMS/MMS, multitarea solo para aplicaciones del SO, o la imposibilidad de instalar ninguna aplicación VoIP.

**BlackBerry OS:** fue desarrollado por Research in Motion (RIM) para los dispositivos BlackBerry en 1999. Es un sistema multitarea, y sus terminales móviles están orientados al uso profesional, ya que estos dispositivos disponen de gestor de correo electrónico, agenda, navegación Web, y sincronización en la nube con aplicaciones de Microsoft o Lotus. Identifica a cada usuario con un código único, que se denomina BlackBerry PIN, con el cual permitía a grandes empresas acceder a su servidor de correo electrónico.

Además otros fabricantes como Siemens, HTC y Sony Ericson utilizan el cliente de correo de BlackBerry.

En 2010 se presentó BlackBerry 6, que como principal novedad, incluía funcionalidades ya no solo destinado al sector profesional, en el cual perdía cuota de mercado a favor del Iphone, si no buscando acceder al usuario medio incluyendo sistemas multimedia y accesos a redes sociales. Entre estas mejoras incluía posibilidad de ejecutar juegos y aplicaciones de terceros, soporte WI-FI y reconocimiento facial.

Los desarrolladores podían realizar aplicaciones, pero ciertas funcionalidades debían ser firmadas digitalmente por RIM.

La aparición del Iphone, y del SO Android ha provocado la caída de las ventas de BlackBerry hasta niveles muy bajos, ocasionando una grave crisis interna dentro de la empresa.

**Windows Phone:** desarrollado por Microsoft para plataformas móviles, pensado para el usuario medio, con el objetivo de al igual que sucede con los PC's, lograr lo supremacía en el mercado de ventas de dispositivos móviles. Su punto fuerte es la compatibilidad con algunos productos Microsoft.

Microsoft lanzó en 2010 Windows Phone 7, con el cual buscaba revolucionar el mercado, pero haciéndolo incompatible con versiones anteriores, por lo que aplicaciones antiguas no funcionaban.

En 2011 lanzaba la versión 7.5 de su sistema operativo móvil, en la que buscaba con su asociación con Nokia, minimizar los requisitos que necesitaba el Sistema Operativo para adaptarlos a terminales básicos, de menor coste, buscando competir con otras plataformas como iOS y Android.

Incluye el denominado Marketplace, que al igual que el App Store de Apple, permite al usuario descargar/comprar aplicaciones, incluyendo también música, películas, podcast y programas de televisión, e incluyendo la posibilidad de que el usuario pueda probar una aplicación antes de comprarla.

Para el desarrollo de aplicaciones, Windows Phone habilita dos posibilidades:

- Microsoft Silverlight, que permite realizar aplicaciones visuales basadas en XAML, utilizando .NET.
- Microsoft XNA, que es un Framework que incluye una API para el desarrollo de juegos.

**HP webOS:** fue desarrollado por Palm que lo introdujo en su Palm Pre el 6 de junio de 2009, aunque posteriormente fue adquirido por la empresa HP, que en principio parecía que iba a apostar fuerte por él.

Es un sistema multitarea, basado en Linux, diseñado para dispositivos táctiles, soportando sistemas multitáctiles. Está basado en Webkit, por lo que utiliza tecnología Web HTML 5, JavaScript y CSS. Como novedad incluye un sistema llamado Synergy, que permite integrar información de diversas fuentes como Gmail, Yahoo!, Facebook o Microsoft Outlook. También integra todos los contactos en una única lista, al igual que todos los mensajes. Para sincronizar todos los datos utiliza la nube.

En 2011 HP anunció que no sacaría más productos con webOS, aunque seguiría dando soporte, liberando el código para desarrollarlo como software libre.

**Maemo:** es una plataforma de desarrollo basada en la distribución Debian de Linux. Nokia estuvo experimentando con ella como una alternativas a Symbian, sacando algunos dispositivos con él, el más conocido el N900. No fue muy bien acogido por los clientes, y fue uno de los detonantes para que Nokia firmara su alianza con Microsoft.

**Android:** este sistema operativo móvil creado por Android Inc., empresa que en 2005 fue comprada por Google. Está basado en Linux, y después de su compra fue desarrollado por la Open Handset Alliance, que está liderada por Google.

Este sistema operativo es utilizado en smartphones y tablets además de Google TV. En la actualidad es utilizado en una gran cantidad de dispositivos móviles de diferentes fabricantes, y su cuota de mercado está aumentando considerablemente hasta el punto de luchar con Symbian y iOS por ser la plataforma más utilizada.

En 2007 Google liberó la mayoría del código de Android bajo la licencia Apache, haciéndolo libre y de código libre, que junto con la publicación del SDK para el desarrollo de aplicaciones provocó que adquiriera un gran interés por los desarrolladores de aplicaciones, que podían publicar sus aplicaciones en el Market, ahora llamado Google Play, y desde el cual, cualquier usuario podía descargar/comprar gran cantidad de aplicaciones, alcanzando las 400.000 aplicaciones en enero de 2012[1] de las cuales, dos tercios son gratuitas.

El desarrollo de aplicaciones se hace mediante Java, lenguaje de programación extensamente conocido y utilizado, lo cual facilitó mucho la implementación de aplicaciones.

[1] Según Europa Press → http://www.europapress.es/portaltic/movilidad/sector/noticia-android-market-llega-400000-aplicaciones-20120105104502.html

## *1.1.- Motivación*

La gran cantidad de funcionalidades que tienen los smartphones han provocado un aumento muy considerable del precio de los terminales, llegando a alcanzar algunos modelos los 600€ o incluso 900€. Esto ha ocasionado que el teléfono sea un bien preciado debido a su precio y a la cantidad de datos personales que se encuentran en él.

Es por eso que la pérdida o el robo de estos dispositivos sea un elemento que puede resultar muy perjudicial a su dueño, ya que no solo pierde el dispositivo, sino los contactos que disponen, las fotografías y vídeos que ha realizado con él, y lo peor, la posibilidad de que otras personas puedan acceder a datos de carácter personal como puedan ser correo electrónico, o acceso a las redes sociales, con todos los perjuicios que esto puede ocasionar.

La potencia de estos nuevos dispositivos también provoca que se utilicen orientándose a la empresa, lo cual puede provocar que la perdida de los dispositivos sea mucho más valiosa. Por lo que la tarea de encontrar un terminal móvil robado/perdido puede ser vital, ya no solo por la cuantía económica del dispositivo, sino por el valor económico/sentimental de los datos contenidos en él.

Como motivación personal me gustaría añadir que la intención principal (a título personal) de este trabajo fin de carrera es aprender nuevos conocimientos que me permitan o faciliten encontrar un trabajo acabada mi vida académica. Esto además se ha visto reforzado con el aprendizaje de un nuevo paradigma de programación como lo es la orientación a eventos.

## *1.2.- Objetivos*

El objetivo del presente proyecto es profundizar en los conocimientos de la plataforma Android, explicando su organización y su arquitectura, entender su historia y cómo ha evolucionado a lo largo de corta historia.

También se comentarán los diferentes entornos de programación que existen hoy en día para desarrollar aplicaciones en Android.

Por último se presentarán las diferentes etapas del desarrollo de una aplicación realizada en Android que buscará asegurar al usuario que ninguna otra persona podrá utilizar su terminal móvil sustraído sin su permiso. Dicha aplicación almacenará los datos de la tarjeta SIM, y reconocerá cuándo se ha introducido una tarjeta distinta a la del propietario, bloqueando el dispositivo para evitar su utilización.

Los aspectos de la aplicación a realizar se encontrarán más detallados en el apartado de requisitos.

## *1.3.- Organización de la memoria*

El presente documento consta de un conjunto de apartados cuya funcionalidad se describirá a continuación.

**Marco teórico:** en este apartado se hará una introducción al sistema operativo Android, explicando su funcionamiento, su arquitectura, la evolución que ha sufrido desde su creación y en cada una de sus diferentes versiones.

**Estado del arte:** este capítulo se dará una visión general de las aplicaciones para la seguridad y localización de dispositivos móviles perdidos/robados, exponiéndose sus características generales, y presentando al finalizar una tabla comparativa de las principales aplicaciones para que se puedan visualizar de forma clara las características más importantes.

**Fase de análisis:** este capítulo contiene la especificación de requisitos a cumplir por la aplicación a realizar con este Trabajo Fin de Carrera, acompañado por la planificación para el desarrollo de la aplicación y una estimación de costes.

**Fase de diseño:** este capítulo contiene los distintos diagramas UML utilizados para la fase de implementación

**Fase de implementación:** este capítulo contiene distintos aspectos de la fase de implementación, como por ejemplo una selección de herramientas utilizadas para realizar la aplicación.

**Manual de usuario:** este capítulo contiene un manual para facilitar al usuario el manejo de la aplicación, explicándole sus particularidades.

**Conclusiones y trabajo futuro:** este capítulo contiene las conclusiones obtenidas con la realización de este Trabajo Fin de Carrera, así como posibles mejoras a añadir a la aplicación.

# 2.- Marco teórico

Como primer paso al desarrollo de la aplicación para el sistema operativo Android es necesario adquirir unos conocimientos previos del entorno en el que la aplicación debe instalarse. Por ello se realizará una descripción del funcionamiento de este sistema operativo, explicando su arquitectura, tal y como se expuso en los objetivos de este Trabajo Fin de Carrera.

## *2.1.- Introducción a Android*

En la primera década del siglo XXI, hablar de telefonía móvil era hablar de Nokia, y por lo tanto, lo era de hablar de Symbian que lideraba de forma clara a nivel mundial, exceptuando Estados Unidos, en el que Windows Mobile, Blackberry y Palm lideraban el sector. Sin embargo en 2007 iba a ser el año en el que todo cambiaría. Y sería a manos de un genio, Steve Jobs, quien anuncio que sacarían al mercado un dispositivo que combinaría su reproductor multimedia, teléfono y navegador Web, al que denominó iPhone. Este dispositivo de altas prestaciones generó un gran interés, pero tenía un sistema cerrado, ya que entonces solo Apple podía desarrollar aplicaciones.

La auténtica revolución llegaría con la creación de la tienda de aplicaciones App Store y con la liberación del SDK gratuito con el que cualquier desarrollador podía desarrollar aplicaciones, las cuales se podían subir al App Store.

Esto obligó a Google a adelantar sus planes, presentando un proyecto en el cual ya llevaba un tiempo trabajando. Ya había rumores de que Google sacaría a la venta su propio teléfono, pero Google iba a contraatacar presentando un proyecto llamado Android, liberando también su SDK. Google ofrecía a los fabricantes de dispositivos un sistema operativo libre de licencias, una tienda de aplicaciones con menos restricciones que el App Store, y la posibilidad de realizar aplicaciones gracias a su SDK.

A partir de entonces Google fue publicando nuevas versiones de su sistema operativo y del SDK, añadiendo servicios y funcionalidades que por la necesidad de sacar al mercado Android, no estaban en sus primeras versiones. La cuestión es que fue evolucionando hasta convertirse un serio competidor para el resto de sistemas operativos móviles. Mientras Apple se ha centrado en hardware de sus dispositivos, Android ha sido adoptado por una gran cantidad de fabricantes que han dispuesto en el mercado terminales de diferentes gamas y precios, lo cual ha permitido llegar a un espectro mayor de usuarios.

En los últimos años, gracias a la aparición de los smartphones, ha habido una batalla comercial por conseguir la supremacía en la cuota de mercado, esto se puede ver reflejado en la siguiente figura, que muestra la evolución de las ventas a nivel mundial de las plataformas móviles más importantes.

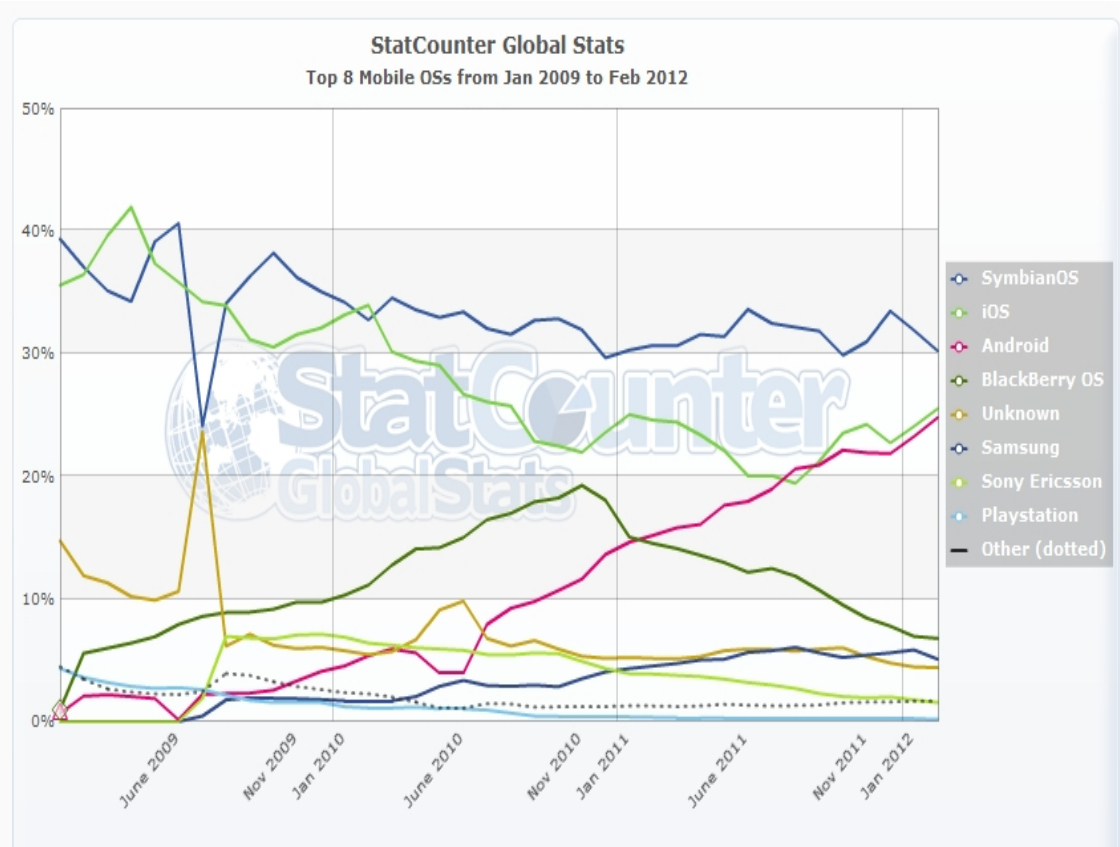



**Figura 1. S.O. móviles utilizados a nivel mundial [SOMUN]**

En la figura se observa cómo la plataforma con más cuota de mercado es SymbianOS. Esto es debido a la grandísima cantidad de ventas que tienen los terminales con dicha plataforma en la zona de Sudamérica y de Asia, que hacen que una plataforma anticuada y con fecha de caducidad siga manteniéndose como líder de utilización en el mercado mundial.

iOs, que era un dominador junto con Symbian, ha ido perdiendo cuota del mercado en una dura competencia con BlackBerry en la que ésta última le fue comiendo terreno, aunque a finales de 2010 repuntó, provocando junto con la competencia de Android, que la plataforma BlackBerry descendiera sus ventas a un nivel cercano al 7%.

Android, cuyo principal defecto es su extrema juventud, ha conseguido sin embargo un rápido aumento de cuota de mercado, pasando de tener aproximadamente un 5% de cuota de mercado a obtener cerca de un 25% y provocar una lucha de tú a tú a iOs, al que sin duda se ha salido un duro competidor.

En España la situación es sin embargo algo diferente, algo que se puede observar en la siguiente figura:

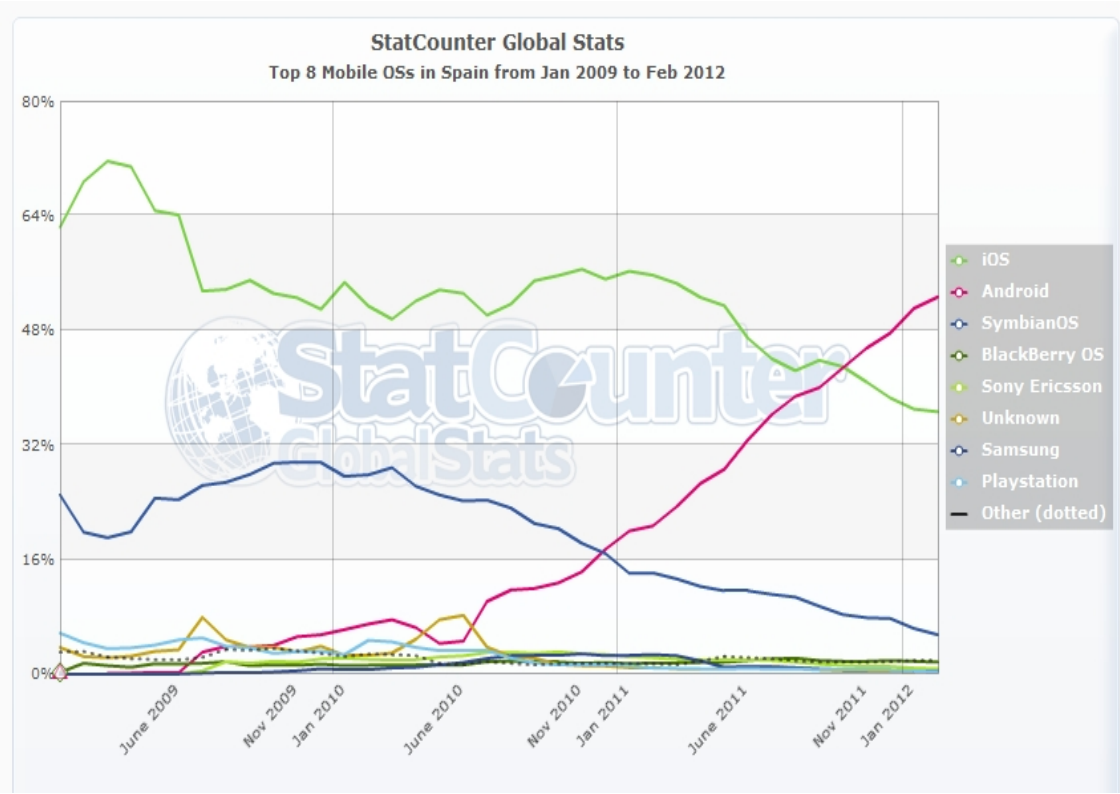



**Figura 2. S.O. móviles utilizados en España [SOESP]**

En la figura se observa la gran pérdida en el reparto del pastel de los sistemas operativos móviles de Symbian, y cómo está disminuyendo de manera considerable hasta alcanzar niveles muy marginales comparándolos con los obtenidos a nivel mundial.

En España hasta finales de 2011, el líder en cuota de mercado era iOS. En 2009 alcanzaba más de un 64% de cuota, aunque ha ido reduciéndose de manera considerable hasta ser alcanzado y superado por Android, cuyo crecimiento en apenas año y medio, se puede considerar como espectacular. El hecho de que exista tanta diferencia entre iOs y Android frente al resto de plataformas es debido a que España a pesar de la gran crisis económica sea el país en el que más smartphones se venden en el mundo, siendo el quinto en el ratio smartphones/por habitante[2], y estos dos sistemas operativos son la referencia en cuanto a smartphones se refiere.

[2] Según El Mundo → http://www.elmundo.es/blogs/elmundo/el-gadgetoblog/2011/12/19/espana-tierra-de-smartphones.html

## *2.2.- Arquitectura de Android*

Después de contextualizar Android dentro del complejo panorama de los sistemas operativos para dispositivos móviles, va siendo hora de entrar en temas algo más técnicos. A continuación se va a tratar de explicar cómo está estructurado Android como sistema operativo.

Para comenzar, destacar que Android es un Sistema monolítico, es decir, posee un núcleo grande y complejo que engloba todos los servicios del sistema operativo. Esto le otorga un rendimiento mayor, pero sin embargo, cualquier cambio a realizar en cualquier servicio requiere la recopilación del núcleo y el reinicio del sistema para aplicar los nuevos cambios.

En el siguiente diagrama se muestran los principales componentes que forman Android, agrupados en capas. Al ser un sistema monolítico cada una de estas capas necesita elementos de la capa inferior para realizar sus funciones.

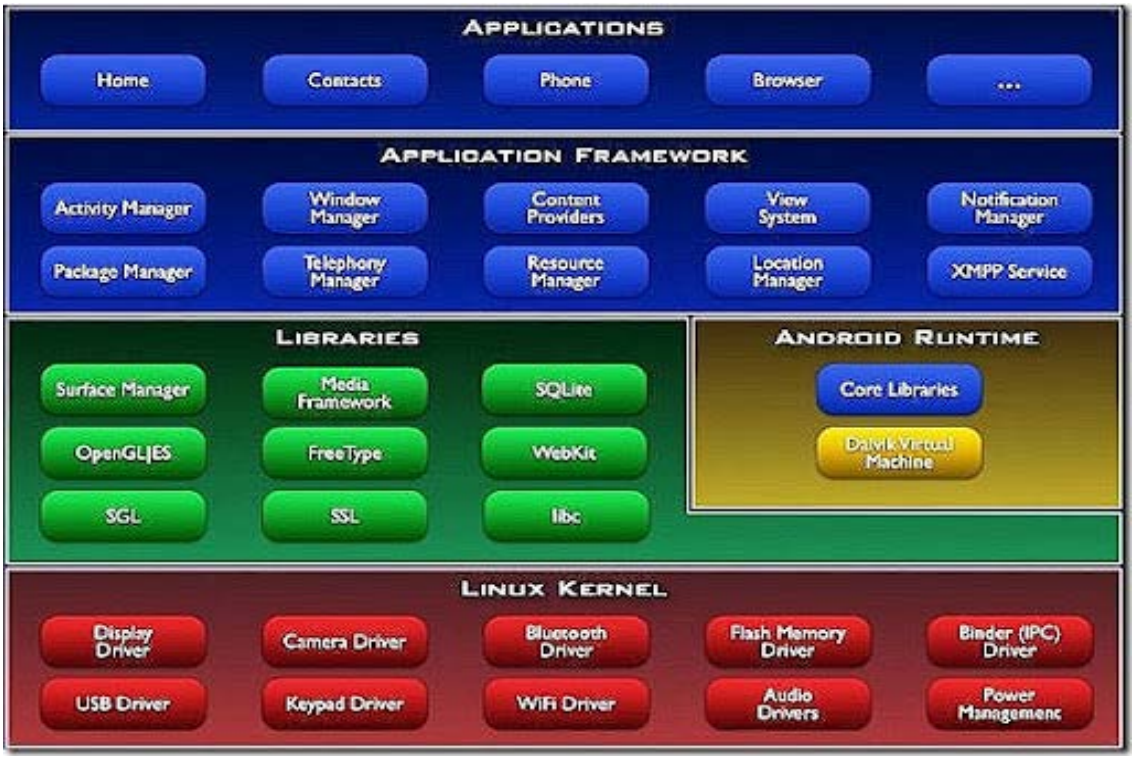



**Figura 3 Arquitectura de Android [ARQAN]**

### 2.2.1 Kernel de Linux

El sistema Android posee un kernel Linux, que varía según la versión de Android, algo que se mostrará en el apartado "Versiones de Android". Este kernel es similar al que se incluye en cualquier distribución de Linux, solo que se adapta al hardware en el que se ejecutará.

Esta capa proporciona una capa de abstracción para los elementos hardware a los que tienen que acceder a las aplicaciones, lo cual permite que se acceda a dichos componentes sin necesidad de conocer las características del hardware que se encuentra por detrás.

### 2.2.2 Librerías

Esta capa situada sobre el kernel contiene las librerías nativas de Android, que están escritas en C o C++ y son compiladas para el hardware específico del dispositivo. Su cometido es proporcionar funcionalidad a las aplicaciones, de forma que se puedan realizar de forma más eficiente.

Estas son algunas de las librerías más habituales:

***Surface Manager* (Gestor de superficies)**: se encarga de componer las imágenes que se muestran en la pantalla a partir de capas gráficas 2D y 3D. Cada vez que la aplicación pretende *dibujar* algo en la pantalla, la librería realiza los cambios en imágenes (mapas de bits) que almacena en memoria y que después combina para formar la imagen final que se envía a pantalla. Esto permite realizar con facilidad diversos efectos: superposición de elementos, transparencias, transiciones, animaciones, etc.

**OpenGL | ES (OpenGL for Embedded Systems)**: motor gráfico 3D basado en las APIs (Application Program Interface) de OpenGL ES 1.0, 1.1 (desde la versión 1.6 de Android) y 2.0 (desde la versión 2.2 de Android). Utiliza aceleración hardware (si el teléfono la proporciona) o un motor software altamente optimizado cuando no la hay.

**SGL (Scalable Graphics Library)**: desarrollada por Skia (empresa adquirida por Google en 2005) y utilizada tanto en Android como en Chrome (navegador Web de Google), se encarga de representar elementos en dos dimensiones. Es el motor gráfico 2D de Android.

**Media Framework (Bibliotecas multimedia):** permiten visualizar, reproducir e incluso grabar numerosos formatos de imagen, vídeo y audio como JPG, GIF, PNG, MPEG4, AVC (H.264), MP3, AAC o AMR.

**FreeType**: permite mostrar fuentes tipográficas, tanto basadas en mapas de bits como vectoriales.

**SSL (Secure Sockets Layer)**: proporciona seguridad al acceder a Internet por medio de criptografía.

**SQLite**: motor de bases de datos relacionales, disponible para todas las aplicaciones.

**WebKit**: motor Web utilizado por el navegador Web. Es el mismo motor que utilizan Google Chrome y Safari (navegador de Apple).

**libc (Biblioteca C de sistema)**: está basada en la implementación de Berkeley Software Distribution (BSD), pero optimizada para sistemas Linux. Proporciona funcionalidad básica para la ejecución de las aplicaciones.

### 2.2.3 Entorno de Ejecución

Esta capa se encuentra en el mismo nivel que la capa anterior debido a que también incluye librerías, a pesar de que se apoya en las librerías anteriormente comentadas.

El principal componente de esta capa es la máquina virtual Dalvik, que es la encargada de ejecutar todas las instrucciones no nativas de Android.

Las aplicaciones se programan principalmente en Java, pero esto no quiere decir que el código que se ejecuta sea un bytecode de java. Esta máquina virtual es la encargada de traducir el bytecode de las aplicaciones en código nativo entendible por el dispositivo.

Cuando se compila una aplicación Android, en lugar de generar ficheros .class o .jar con bytecode Java se generan unos archivos .dex (Dalvik Executables). Estos ficheros son el resultado de combinar los ficheros java con las librerías Android, transformándolo a un juego de instrucciones propio de la maquina Dalvik. Además, durante el proceso se reduce considerablemente el tamaño del fichero, reutilizando información y haciéndolo por lo tanto más eficiente.

Esta máquina virtual está optimizada para requerir poca memoria y permite ejecutar varias instancias de sí misma simultáneamente. Otro aspecto en el que Dalvik es muy superior a otras máquinas virtuales es en la gestión de memoria de objetos obsoletos, conocido como Garbage Collector o recolector de basura, ya que se ha implementado con nuevos algoritmos que lo hacen mucho más eficiente.

A la hora de realizar el proceso de compilación se podía realizar de dos maneras, o bien compiladas, o bien interpretadas, siendo las primeras más rápidas por ser código nativo, y siendo las segundas más versátiles puesto se podían conseguir que fueran multiplataforma, pero a cambio eran más lentas puesto que se tenía que estar interpretando continuamente de lenguajes de alto nivel a código nativo. Dalvik crea una nueva forma de compilar creando el JIT (Just In Time compilation) en el cual el bytecode generado se interpreta continuamente, pero a su vez se va guardando en una caché para no tener que traducirlo a código nativo si ya se ha ejecutado con anterioridad.

Esta novedad se introdujo en Android 2.2 y se consiguió un aumento en la velocidad de ejecución de entre 2 y 5 veces. [3]

[3] Fuente: http://www.androidcentral.com/benchmarking-android-22-froyo-against-android-21-eclair

### 2.2.4 Marco de aplicación

Esta capa está formada por todos los servicios que utilizan directamente las aplicaciones para realizar su función. La mayoría de estos servicios son librerías java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik. Entre los servicios que nos podemos encontrar están los siguientes:

**Activity Manager:** administra la llamada "pila de actividades" de cada aplicación, así como su ciclo de vida.

**Package Manager:** permite obtener información sobre los paquetes instalados en el dispositivo y gestiona la instalación de nuevos paquetes.

**Window Manager:** se encarga de administrar lo que se mostrará en pantalla.

**Telephony Manager:** facilita acceder a los datos del dispositivo, pudiendo obtener identificador, marca, y versión de Android, además de ofrecer otros servicios como acceder a los servicios de llamadas o mensajes.

**Content Providers:** crea una capa que encapsula los datos que se compartirán entre las aplicaciones, para tener control sobre cómo se accede a la información.

**Resource Manager:** se utiliza para gestionar los elementos de la aplicación que están situados fuera de la aplicación, como pueden ser imágenes o sonidos.

**View System:** maneja los elementos de las interfaces, como pueden ser botones, cuadros de texto.

**Location Manager:** permite mediante GPS, obtener las coordenadas geográficas del dispositivo.

**Notification Manager:** engloba servicios que facilitan notificar al usuario cuando se requiera su información. Entre otros, permite la utilización de sonidos, utilizar el vibrador o utilizar los LEDs del telefono.

**XMPP Service:** es un protocolo de intercambio de mensajes basado en XML.

Además también posee otros servicios como pueden ser el **Sensor Manager**, que facilita el uso con el acelerómetro, giroscopio, sensor de luminosidad, brújula, cámara (que facilita el uso de la cámara fotográfica del dispositivo móvil), o el servicio multimedia, que permite reproducir audio y visualizar imágenes y vídeo.

### 2.2.5 Aplicaciones

Esta es la última capa y en la que se encuentran todas las aplicaciones instaladas en el dispositivo. Las nativas (programadas en C o C++) y las instaladas (programadas generalmente en Java).

## *2.3.- Versiones de Android.*

La primera versión de Android fue liberada el 23 de septiembre de 2008, saliendo su actualización a la versión 1.1 el 9 de febrero de 2009, aunque la primera versión que adquirió cierta relevancia fue la 1.5.

### 2.3.1.- Android 1.5 Cupcake

Fue liberada el 30 de abril de 2009 y está basada en el kernel 2.6.27 de Linux. Incluía nuevas mejoras en la interfaz, entre las cuales se pueden destacar la posibilidad de grabar y reproducir vídeos, dando la posibilidad de subir vídeos a YouTube e imágenes a Picasa, servicio de Google para compartir fotos.
Daba soporte para Bluetooth A2DP, posibilitando la conexión automática de un auricular bluetooth a distancia. Además incorporaba widgets y transiciones de pantallas animadas.

### 2.3.2.- Android 1.6 Donut

Fue liberada el 15 de septiembre de 2009 y está basada en el kernel 2.6.29 de Linux. Como principales novedades incluía un Market más actualizado, con el cual era más fácil encontrar las aplicaciones, integraba en una misma interfaz la galería de vídeo y de fotografía. También se mejora la búsqueda por voz, haciéndola más rápida y flexible, y se mejora de la experiencia Web, incluyendo marcadores e historiales. También da soporte a CDMA/EVDO, 802.1x, VPN, y da soporte a dispositivos con pantallas WVGA, con pantalla más grande, que por entonces estaban apareciendo en el mercado.

### 2.3.3.- Android 2.0/2.1 Eclair

La versión 2.0 fue liberada el 26 de octubre de 2009; el 3 de diciembre del mismo año la versión 2.0.1; y el 12 de enero de 2010 la versión 2.1. Estaban basadas en el mismo kernel que la versión 1.6 de Android, supuso una gran renovación en la interfaz de usuario, dándole mucha más potencia y soporte para distintos tamaños de pantalla. También se cambió la interfaz del navegador, dándole soporte para HTML5, se incluyó una nueva versión de Google Maps (v.3.1.2), se dio soporte para el zoom digital, para el flash de la cámara y para bluetooth 2.1. Se mejoró el teclado digital, se incluyó soporte para pantallas multitáctil y se incluyeron fondos de pantalla animados. También logró optimizar la velocidad de hardware.

### 2.3.4.- Android 2.2 Froyo

Fue liberada el 20 de mayo de 2010, y está basada en el kernel 2.6.32 de Linux. Se optimizó el sistema, buscando mejorar su rendimiento, reduciendo la fragmentación de la memoria y buscando aumentar la ejecución de aplicaciones. Se mejoró el motor de la

aplicación Browser, incluyéndole el de Google Chrome, además de sincronización de calendario remota. Se permite desactivar el tráfico de datos a través de la red, se actualiza el Market incluyendo actualizaciones automáticas, se da soporte para Adobe Flash 10.1 y para pantallas en alta definición. También se incluyen Wi-Fi hotspot y tethering por USB. También se incluye la compilación JIT, comentada con anterioridad en el apartado 2.1.2.3.

### 2.3.5.- Android 2.3 Gingerbread

Fue liberada el 6 de diciembre de 2010, y está basada en el kernel 2.6.35.7 de Linux. Vuelve a actualizar el diseño de la interfaz, ofrece soportes para pantallas extra grandes. Ofrece soporte a tecnología VoIP, decodificadores de audio tipo AAC, para diferentes tipos de sensores, para múltiples cámaras, además de ofrecer nuevos efectos de audio, mejores gráficos para diseñadores de juegos y un control de energía mejorado.

### 2.3.6.- Android 3.0/3.1/3.2 Honeycomb

Esta versión se realizó pensando exclusivamente en el mercado de las tablets. Se requiere una API distinta y las aplicaciones creadas para estas versiones de Android podían resultar incompatibles con dispositivos con versiones 2.X. Lo cual se entendió como un error de diseño, que fue subsanado con la versión 4.0, que volvía a unificar el mundo de los móviles y de los tablets, y por lo tanto, sus aplicaciones.

Esta versión de Android fue liberada el 5 de Enero del 2011, y está basada en el kernel 2.6.36 de Linux. Entre sus características más importantes, destacar la creación de escritorios 3D con widgets rediseñados y un nuevo diseño de la interfaz, mejoras en el sistema multitarea, en el navegador web, y en la conexión a redes Wi-Fi. También añadía soporte para videochat mediante Google Talk, y variedad de accesorios USB.

### 2.3.7.- Android 4.0 Ice Cream

Como ya se ha comentado con anterioridad, esta versión unifica el mundo de los móviles y de los tablets, pudiendo ser instalado en cualquiera de estos dispositivos, y por lo tanto, funcionando las aplicaciones indistintamente ya sea un móvil o un tablet, gracias a la creación de un Framework único para las aplicaciones.

Fue liberada el 20 de Octubre de 2011, y está basada en el kernel 3.0.8 de Linux. Incorpora una nueva interfaz más limpia e incorporando una nueva fuente, llamada “Roboto”, nuevos botones, widgets más organizados, corrector de texto mejorado, mejora del sistema multitarea, gestor de tráfico de datos de Internet, nuevo software gestor de la cámara, con nuevas utilidades, y la posibilidad de realizar aceleración por hardware, pudiendo ser la interfaz dibujada por la GPU.

Además, incorpora nuevas funcionalidades como reconocimiento de voz del usuario, reconocimiento facial, soporte para contenedores MKV, sistema de ficheros más fácil de manipular, y más herramientas para que el usuario pueda controlar las aplicaciones, pudiendo cerrarlas y liberar memoria.

Como se ha podido observar, Android ha sacado al mercado una gran cantidad de versiones en su corta existencia, añadiendo nuevas funcionalidades lo cual ha posibilitado que sus ventas aumenten de manera espectacular provocando una dura competencia en la que empresas como Microsoft o Blackberry se han quedado muy por detrás, y otras como Nokia o Apple vean amenazada su supremacía en el mercado.

## *2.4.- Mercado de Android*

Un aspecto importante a la hora de desarrollar aplicaciones en Android es el nivel de API que se debe utilizar. Cada versión de Android tiene una API distinta, en la cual se han desarrollado más librerías, se han actualizado otras o incluso se han abandonado por otras más desarrolladas. Esto ocasiona que cuando se desarrolla una aplicación no tiene por qué funcionar en todas las versiones de Android, y se ha de ser muy cuidadoso en ese aspecto. En la siguiente tabla se puede observar la API que se ha de utilizar según la versión de Android para la que se quiera desarrollar.

| Plataforma | Nivel de API |
|---|---|
| 4.1 Jelly Bean | 16 |
| 4.0.x *Ice Cream Sandwich* | 15 |
| 3.x.x *Honeycomb* | 11-13 |
| 2.3.x *Gingerbread* | 9-10 |
| 2.2 *Froyo* | 8 |
| 2.1 *Eclair* | 7 |
| 1.6 *Donut* | 4 |
| 1.5 *Cupcake* | 3 |

**Tabla 1 Nivel de API según la versión de Android**

La creación y evolución de Android ha ocasionado que existan dispositivos con todas las versiones disponibles. El uso de estas diferentes versiones de Android se puede observar en la siguiente tabla:

| **Plataforma** | **Nombre** | **Distribución** |
|---|---|---|
| Android 1.5 | Cupcake | 0.1% |
| Android 1.6 | Donut | 0.4% |
| Android 2.1 | Eclair | 3.4% |
| Android 2.2 | Froyo | 12.9% |
| Android 2.3 - 2.3.2 | Gingerbread | 0.3% |
| Android 2.3.3 - 2.3.7 | | 55.5% |
| Android 3.0/Android 3.1 | Honeycomb | 1.5% |
| Android 3.2 | | 1.5% |
| Android 4.0 | Ice Cream Sandwich | 23.7% |
| Android 4.1 | Jelly Bean | 1.8% |

**Tabla 2 Porcentaje uso de las versiones Android [UVAND]**

Si pasamos los datos de la tabla a un gráfico de sectores, se pueden observar la proporcionalidad de los datos de la utilización de las versiones de Android, quedando de la siguiente manera:

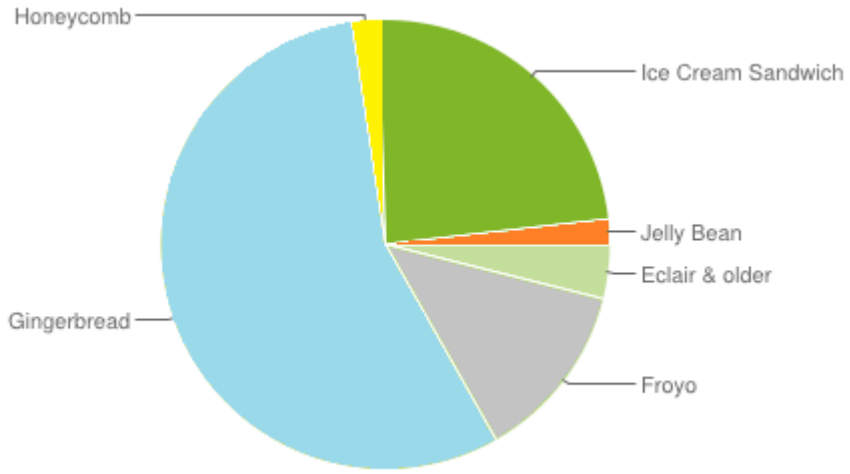



**Figura 4 Diagrama de uso de las versiones de Android [UVAND]**

En el siguiente grafico proporcionado por Google se puede observar la evolución de las diferentes versiones desde octubre del 2011:

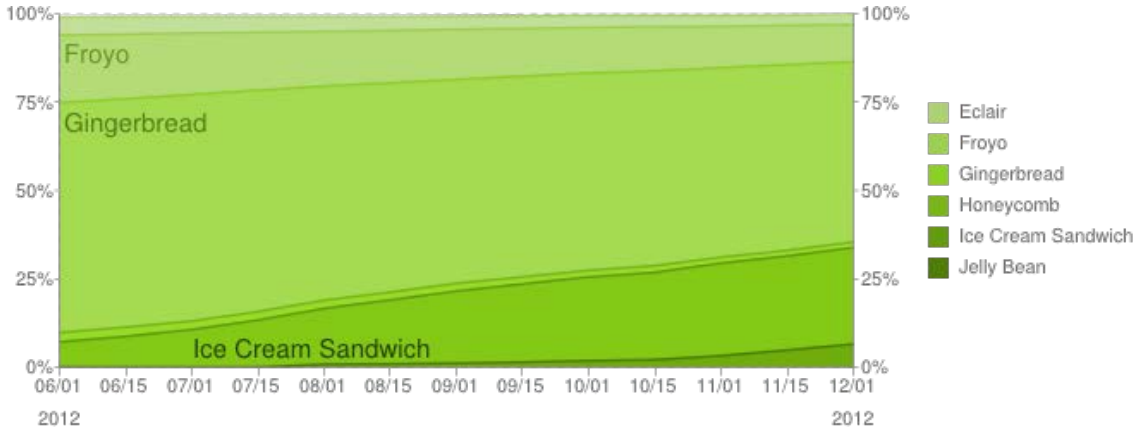



**Figura 5 Evolución de la utilización de las versiones de Android [UVAND]**

Como curiosidad acerca de los nombres de las versiones de Android, comentar que las versiones de Android reciben el nombre de postres en inglés, empezando cada versión por una letra distinta siguiendo un orden alfabético.

- C: Cupcake (v1.5), magdalena glaseada.
- D: Donut (v1.6), rosquilla.
- E: Éclair (v2.0/v2.1), pastel francés conocido en España como pepito.
- F: Froyo (v2.2), (abreviatura de «frozen yogurt») yogur helado.
- G: Gingerbread (v2.3), pan de jengibre.
- H: Honeycomb (v3.0/v3.1/v3.2), panal.
- I: Ice Cream Sandwich (v4.0), sandwich de helado.

## *2.5.- Entornos de desarrollo*

Se pueden desarrollar aplicaciones en Android de diversas maneras, incluso sin utilizar ningún entorno de desarrollo, aunque esto no sería productivo. Por ello, se utilizan dichos entornos, para facilitar al programador su tarea y aumentar su productividad.

El entorno más utilizado para el desarrollo de aplicaciones en Android es Eclipse, pero como se podrá ver a continuación no es el único, ya que se van a exponer diferentes alternativas con las cuales se pueden realizar aplicaciones.

### 2.5.1.- Corona SDK

Corona SDK es una herramienta creada por la compañía Ansca, que facilita el desarrollo de aplicaciones para iOS y Android, exportando los proyectos en forma de aplicaciones nativas para dichas plataformas. Posee un motor físico muy avanzado, lo que lo convierte en una extraordinaria opción para el desarrollo de juegos.
Como principales ventajas se pueden encontrar:

- **Integración automáticamente con OpenGL-ES:** que facilita la manipulación de imágenes en pantalla.
- **Desarrollo multiplataforma:** ya que permite crear aplicaciones tanto para iOS como para Android.
- **Administración del dispositivo:** posee controles nativos que permite el acceso a dispositivos hardware como pueden ser la cámara, el GPS, el acelerómetro.
- **Fácil de aprender**.

Como principal desventaja habría que comentar que como Ansca no está integrada ni en Apple ni en Android, existen ciertas limitaciones. Sin embargo, Ansca está trabajando para corregirlas.


**Figura 6 Interfaz de Corona SDK [RUB]**

Corona SDK es gratuito, pero si se desea darle un uso comercial a la aplicación hay que comprar una licencia que vale 199 dólares, y que solo permite desarrollar una plataforma, ya sea Android o iOS. En caso de querer trabajar con ambas plataformas, será necesario pagar una licencia de 349 dólares al año.

Para realizar aplicaciones en Android se suele utilizar Java como lenguaje de programación, algo que en Corona SDK es una excepción, ya que se utiliza LUA.

LUA es un lenguaje de programación imperativo, estructurado y muy ligero, y que tiene mucho parecido con JavaScript. Es muy utilizado para la creación de juegos debido a velocidad, su sencillez y su fácil portabilidad, siendo utilizado por ejemplo, en homebrews de la consola PSP, en juegos como World of Warcraft, S.T.A.L.K.E.R.: Shadow of Chernobyl, Worms 4: Mayhem, en diversas modificaciones (mods) de juegos como Half-Life 2, Wolfenstein y Grand Theft Auto, y en portabilidades de juegos entre PSP y Wii.

### 2.5.2.- Ruboto

Ruboto es otra opción que permite desarrollar aplicaciónes en Android mediante la utilización de la plataforma Ruby. A su vez también está basado en JRuby, que es una implementación 100% Java del lenguaje Ruby liberado bajo licencia CPL/GPL/LGPL, que funciona correctamente dentro de la máquina virtual de Java, y que por lo tanto, permite la utilización de cualquier aplicación Java.

Esto facilita que una aplicación desarrollada con Ruboto pueda utilizar la API de Android.


**Figura 7 Interfaz de Ruboto [RUB]**

### 2.5.3.- Rhomobile Rodes

Rhomobile Rodes es un Framework basado en Ruby que permite realizar aplicaciones multiplataforma permitiendo desarrollar aplicaciones multiplataforma para diversos sistemas operativos como pueden ser iOS, Android, RIM, Windows Mobile y Windows 7. Permite la utilización del hardware del dispositivo, como puede ser la cámara, el Bluetooth o el GPS.

Rhomobile Rodes es gratuita y de código abierto bajo licencia del MIT, aunque al igual que Corona SDK, si se desea comercializar las aplicaciones creadas se debe de comprar una licencia.

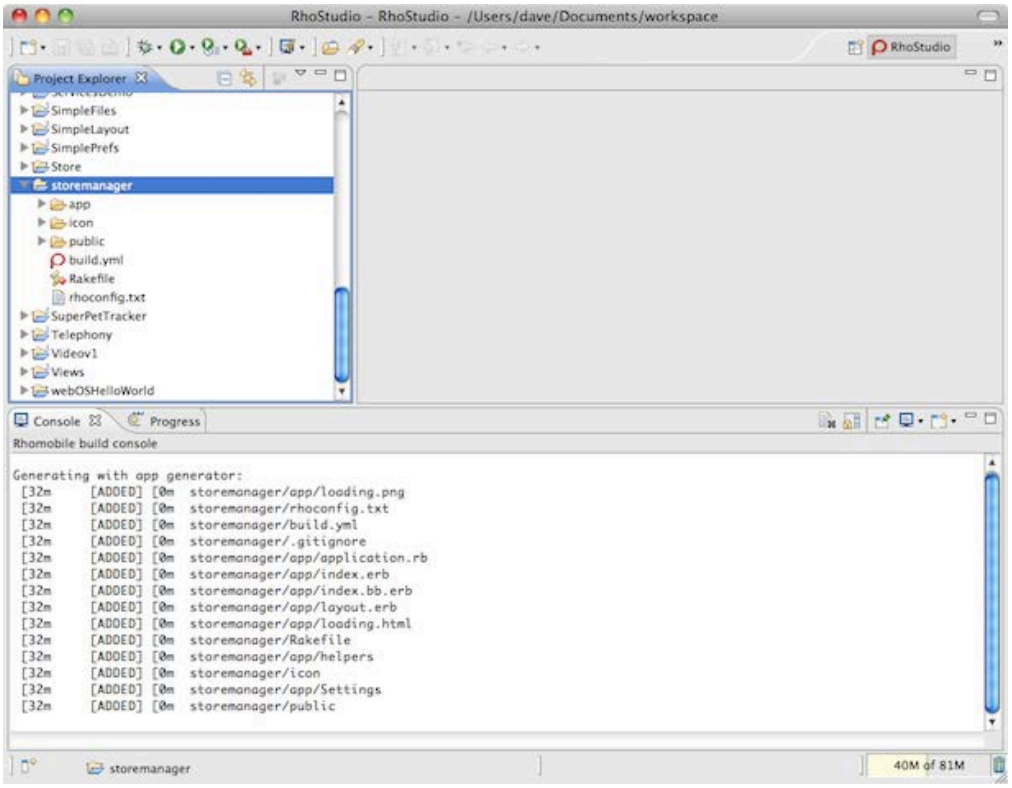



**Figura 8 Interfaz de Rhomobile Rodes [RHR]**

### 2.5.4.- Basic4Android

Basic4Android es otro entorno de programación, muy intuitivo que utiliza un lenguaje de programación muy similar a Visual Basic. Se necesita tener Windows, .Net Framework, Java y Android SDK. A pesar de no programar en Java, en realidad al compilar lo que se hace es transformar esa aplicación a código Java.

Una de las principales ventajas es la gran cantidad de librerías que posee, facilitando mucho el desarrollo de aplicaciones. Además, hay que destacar soporte para:

- Base de datos en SQLite
- Bluetooth
- GPS
- Cámara
- Servicios Web y en red
- JSON
- Reconocimiento de voz
- AdMob

Otra gran ventaja es el diseñador de interfaces incorporado, que facilita mucho a la hora de realizar las interfaces de la aplicación.

Una gran desventaja es que hay que comprar una licencia de 69 dólares para la licencia de desarrollador o de 99 dólares (licencia Enterprise). Y que a pesar de tener bastantes librerías, tiene limitaciones a la hora de poder realizar ciertas aplicaciones.

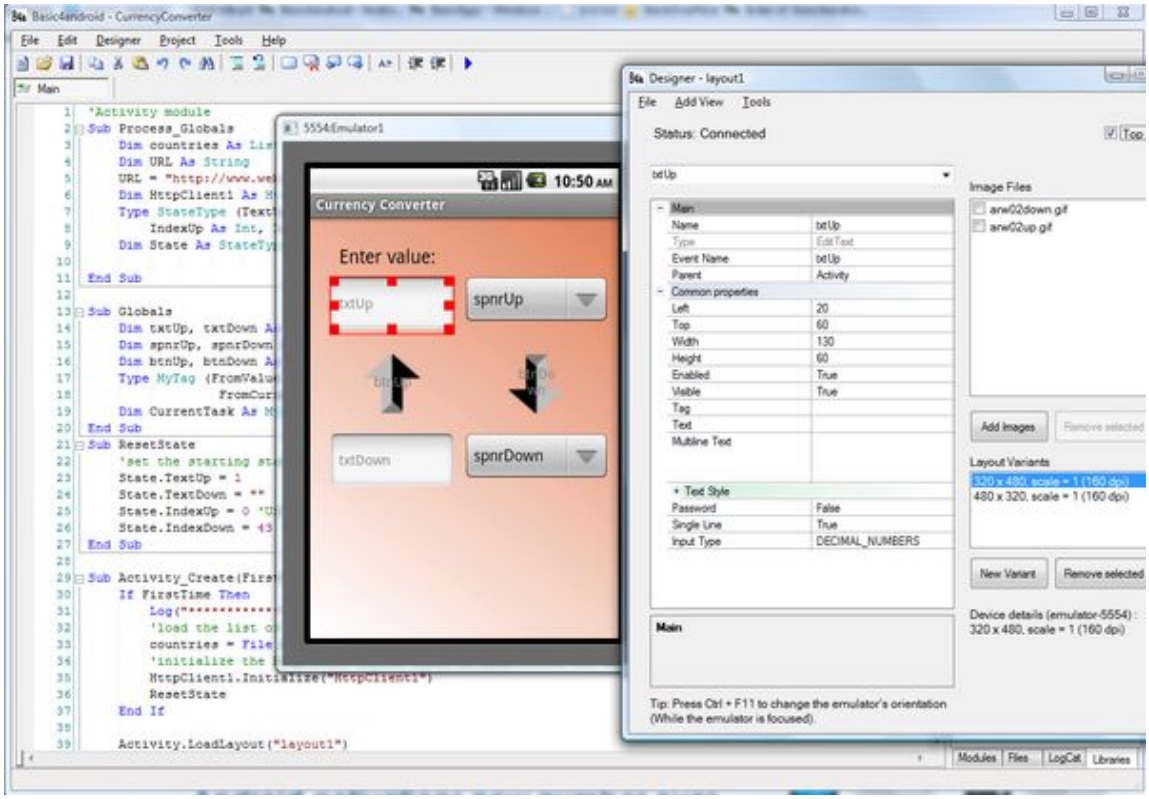



**Figura 9 Interfaz de Basic4Android [B4A]**

### 2.5.5.- Monodroid

Monodroid es un SDK que permite desarrollar aplicaciones Software para iOS, Android y Windows Phone 7, con .NET, utilizándose C# como lenguaje de programación. Para desarrollar aplicaciones en Monodroid, se requiere Visual Studio

Como gran novedad, este entorno busca la creación de las aplicaciones independientemente de su arquitectura, pudiendo realizar una aplicación, y que ésta funcione tanto en Android, como en iOS o Windows Phone, sin tener que realizar ninguna especialización en el código.

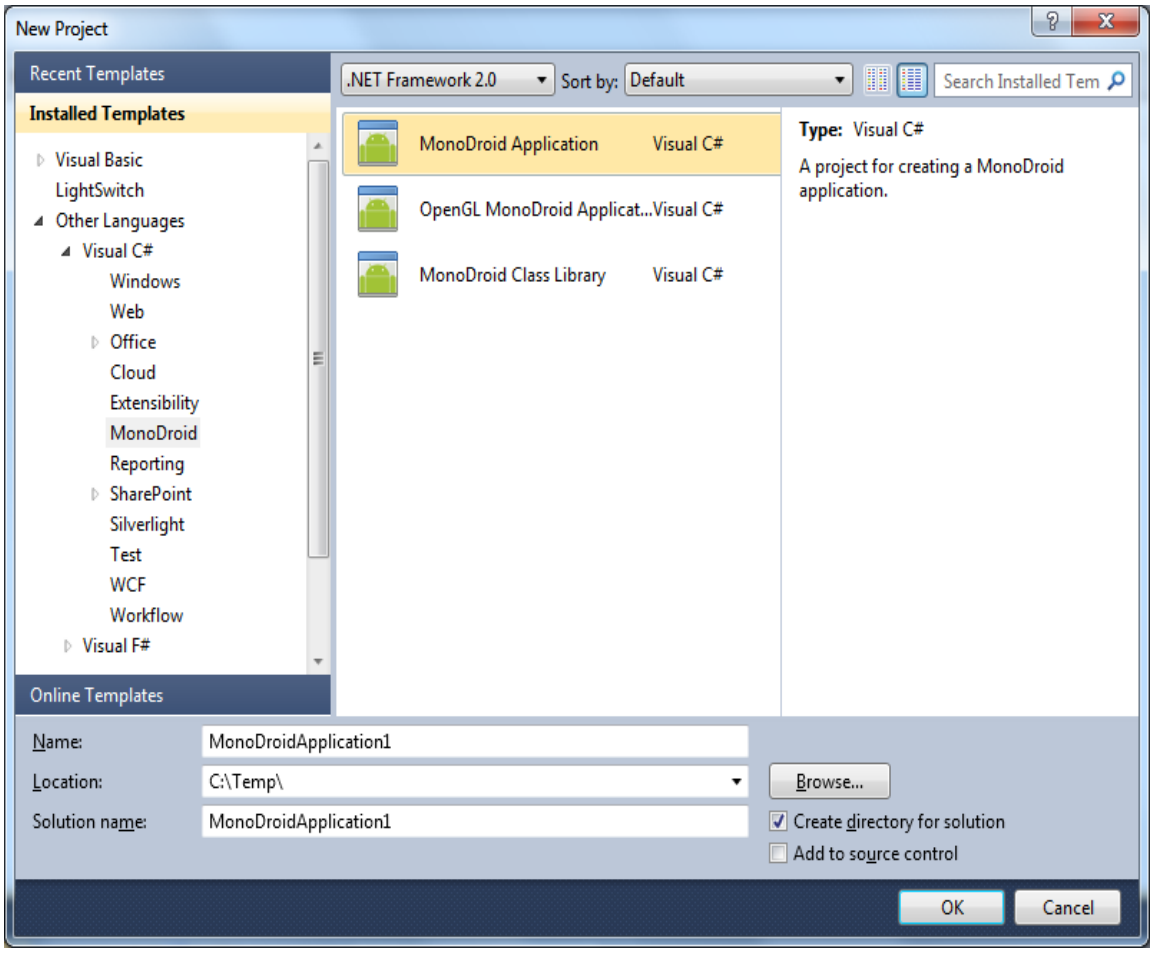



**Figura 10 Interfaz de Monodroid [MOD]**

### 2.5.6.- App Inventor

App inventor es una aplicación creada por Google para que cualquier persona con interés y sin conocimientos de programación, pueda crearse su propia aplicación móvil de interés general, ya sea para su empresa o para su casa.

Un año después de prometedores resultados de este programa, Google decidió apartarlo liberando el código para todo aquel que quisiera continuar el camino que había empezado Google, siendo el MIT el que recogió el guante. Se creó el centro MIT para el aprendizaje móvil y se trabajó duramente para que este proyecto saliera a la luz. El MIT ofrece un soporte muy completo, con manuales y bastante información, a pesar de encontrarse en fase Beta.

Permite acceder a diferentes dispositivos como Bluetooth, GPS, cámara, pudiendo utilizarla de forma muy sencilla. La forma de desarrollar aplicaciones es la siguiente:

1. Se diseña la aplicación creando la interfaz y sus componentes.
2. En el editor de bloques, se juntan los componentes y se les asigna la funcionalidad deseada.
3. Se prueba la aplicación en un dispositivo virtual o en el propio teléfono.

Como ventajas habría que destacar la increíble facilidad de utilización, y el no ser necesario ningún conocimiento de programación. Como desventajas, la nula flexibilidad que ofrece, ya que tiene alguna limitación, y la no posibilidad de ver/editar el código de la aplicación, lo cual daría la posibilidad de flexibilizar mucho las aplicaciones cubriendo las carencias que posee App Inventor. Sin duda esta es una herramienta con mucho potencial, que es interesante y que podría tener futuro.

Esta herramienta era la que se pretendía utilizar en la realización de la aplicación, pero debido a las limitaciones de la herramienta se tuvo que descartar. Estas limitaciones se comentarán en el resumen de las herramientas.

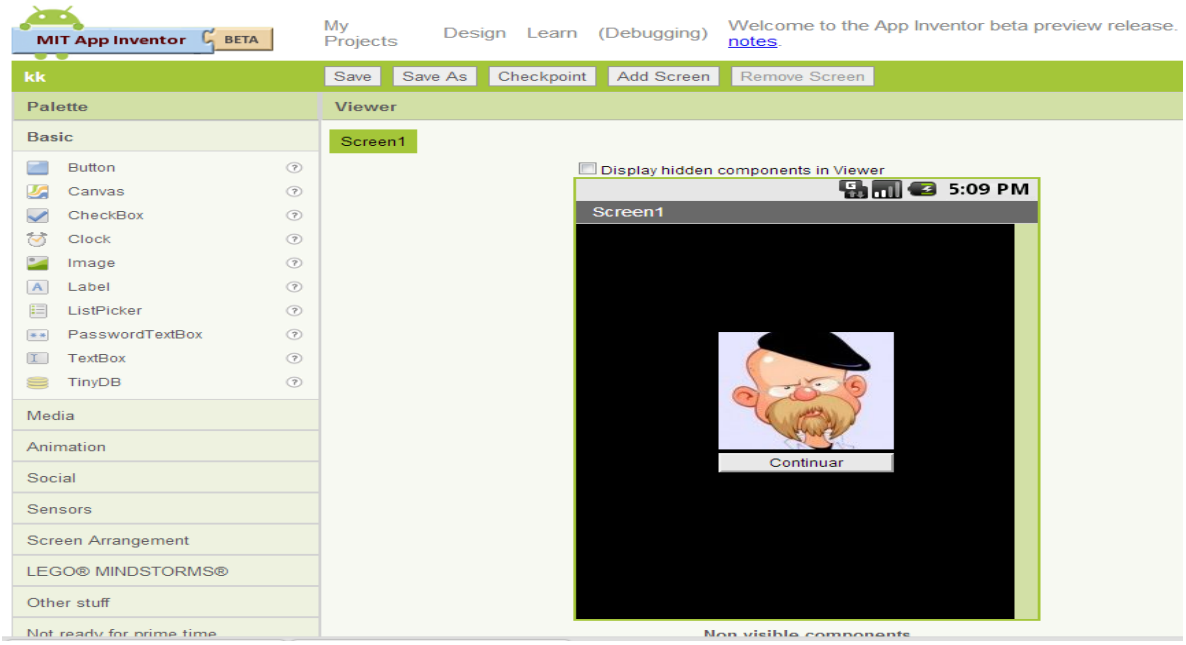



**Figura 11 Interfaz de App Inventor [APPI]**

### 2.5.7.- Netbeans

Si hay una herramienta muy conocida a la hora de desarrollar aplicaciones Java es Netbeans. Netbeans también ofrece la posibilidad de desarrollar aplicaciones para Android. Para ello se necesita:

1. Netbeans
2. SDK Android
3. Plugin de Android para Netbeans

Esto ofrece la posibilidad de desarrollar aplicaciones, en lenguaje nativo Java, con una potente herramienta, que facilita mucho a la hora de desarrollar código, permitiendo ejecutar la aplicación en un dispositivo virtual.

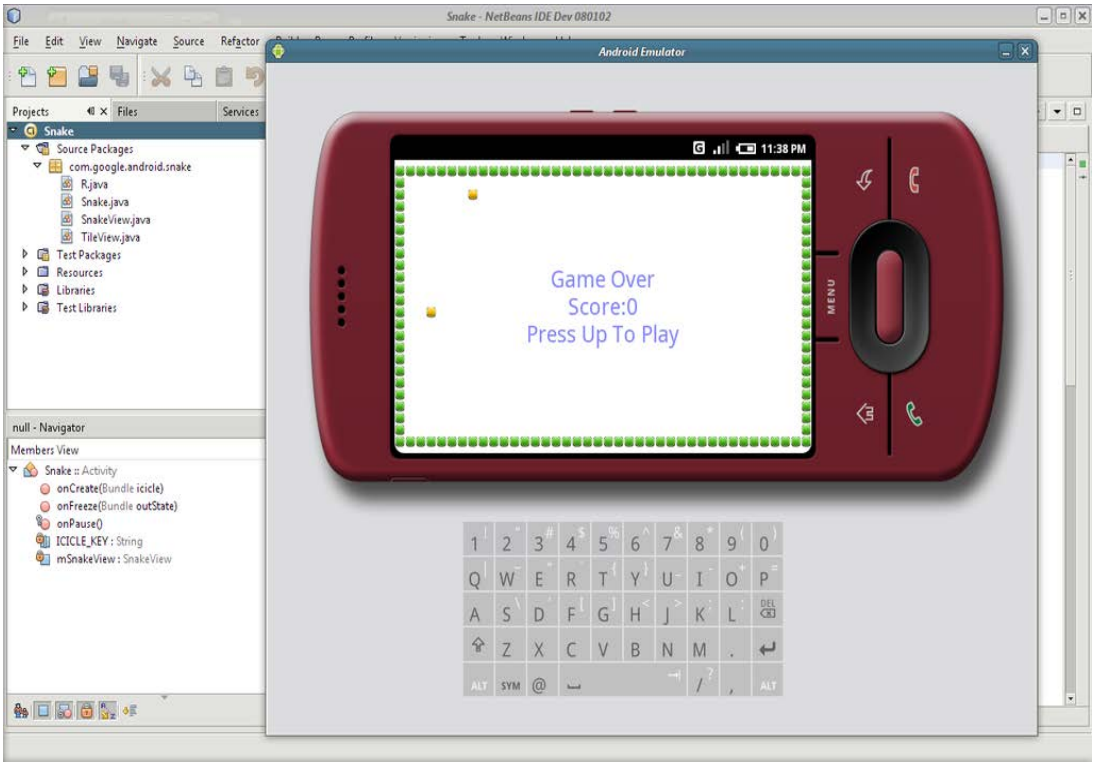



**Figura 12 Interfaz de Netbeans [NETB]**

### 2.5.8.- Eclipse

En el más utilizado a la hora de desarrollar aplicaciones para Android. El modo de funcionamiento es similar al explicado con la herramienta Netbeans. Se necesita:

1. Eclipse
2. SDK Android
3. Plugin de Android para Eclipse

Permite desarrollar aplicaciones Android en una herramienta potente, pero ligera.

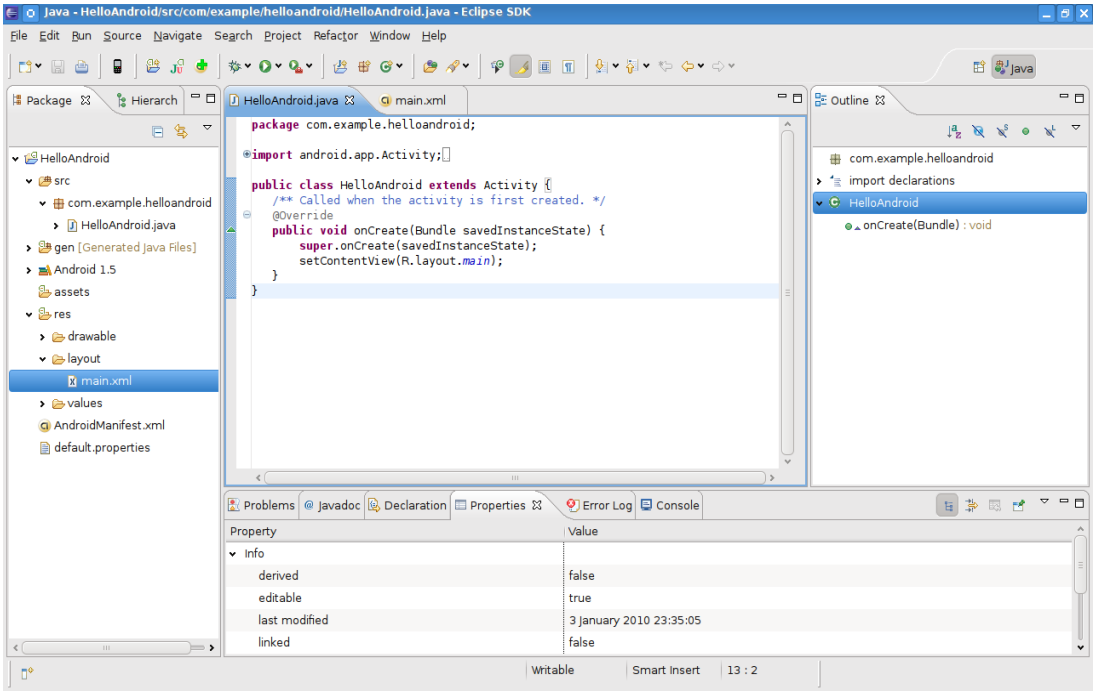



**Figura 13 Interfaz de Eclipse [ECLI]**

### 2.5.9.- Comparativa

A continuación se expondrá una tabla comparativa de los diferentes entornos de programación estudiados, justificando el porqué del entorno elegido para la realización de la aplicación de este TFC. En la tabla se ha decidido estudiar los siguientes puntos:

- **Gratuito:** A la hora de realizar una aplicación, uno de los aspectos más importantes es conocer si el IDE a utilizar va a ser gratuito o si es necesario comprar una licencia por su utilización. Se busca que su utilización sea gratuita.
- **Dificultad:** en este punto se evalúa la dificultad de utilización del IDE a la hora de utilizarlo.
- **Requiere aprendizaje:** se evalúa si requiere el aprendizaje de un nuevo lenguaje de programación distinto al nativo de Android (Java).
- **Flexibilidad:** este es uno de los puntos más importantes, dado que se ha a acceder a HW muy específico, y es imprescindible que el IDE admita flexibilidad para poder tratarlo.
- **Multiplataforma:** informa si una vez realizado el código de una aplicación se puede utilizar (migrar) a otras plataformas (iOS, Blackberry, Windows…) sin realizar modificación alguna. No era un punto importante en la realización de la aplicación, pero siempre es algo importante a tener en cuenta.
- **Modificación código:** se evalúa si ante un inconveniente o una ampliación/mejora, se permite obtener el código fuente y realizar modificaciones sobre él.

| IDE | Gratuito | Dificultad | Requiere aprendizaje | Flexibilidad | Multiplataforma | Modificación código |
|---|---|---|---|---|---|---|
| Corona SDK | NO | Media/alta | SI | NO | SI | Solo LUA |
| Ruboto | SI | Media/alta | SI | NO | NO | Solo Ruby |
| Rhomobile Rodes | SI* | Media/alta | SI | NO | SI | NO |
| Basic4Android | NO | Media/alta | SI | NO | NO | Solo Visual Basic |
| Monodroid | SI | Media | SI | NO | SI | Solo C# |
| App inventor | SI | Muy baja | NO | NO | NO | NO |
| Netbeans | SI | Media | NO | SI | NO | SI |
| Eclipse | SI | Media | NO | SI | NO | SI |

**Tabla 3 Tabla comparativa de los diferentes entornos de programación estudiados para la realización de aplicaciones Android**

* No si se desea comercializar la aplicación.

Este proyecto se ide con la intención de ser realizado con la herramienta App Inventor, algo que se tuvo que descartar debido a la imposibilidad de acceder al Hw de la tarjeta SIM y a la imposibilidad de poder obtener ni modificar el código fuente de la aplicación.

Corona SDK, Ruboto, Rhomobile Rodes, Basic4Android y Monodroid se descartaron por cuestiones similares. ´Finalmente se eligió Eclipse debido a la gran flexibilidad que tiene en su utilización, así como a la gran cantidad de información existente en la red, lo cual ha facilitado la realización de la aplicación.

# 3.- Estado del arte

La aplicación a realizar, llamada SimDetect, será una aplicación de seguridad para la localización de dispositivos móviles Android, facilitando su localización en caso de pérdida o robo. En esta sección se presentarán aplicaciones similares a la que se pretende realizar en este proyecto, haciéndose primero una introducción de las distintas aplicaciones, y presentándose para finalizar, un cuadro comparativo de las herramientas descritas.

## *3.1.- Avast! Mobile Security*

Al igual que la aplicación de escritorio, Avast es una solución software antivirus, que también se encuentra disponible para Android. Ocupa solo 2,5 MB de espacio, y ofrece los siguientes servicios:

- Escáner antivirus de aplicaciones y tarjetas SD.
- Asesor de privacidad y cortafuegos.
- Administración de aplicaciones.
- Filtro de llamadas y SMS con bloqueo.
- Alarma.
- Bloqueo remoto.
- Notificación al insertar otra tarjeta SIM.
- Oculta la aplicación.

Avast! incluye una función, llamada avast anti theft, que protege ante posibles robos o pérdidas del celular, y puede ayudar a localizar el dispositivo. Esta función está disponible para descargar una vez instalada la aplicación.

El usuario puede elegir un nombre personalizado para disfrazar la aplicación. Una vez el antirrobo está activado, el icono de la aplicación se oculta en la bandeja de aplicaciones, lo que hace difícil para los ladrones detectar o eliminar la aplicación. En caso de robo o si se inserta una SIM diferente, el teléfono puede bloquear, localizar mediante GPS o activar una sirena para detectarlo, mediante comandos SMS.

Como principal crítica, destacar que al estar la aplicación oculta, no se puede desinstalar, ni aun desinstalando Avast. Solo se puede eliminar la aplicación desde dentro del propio Avast!.

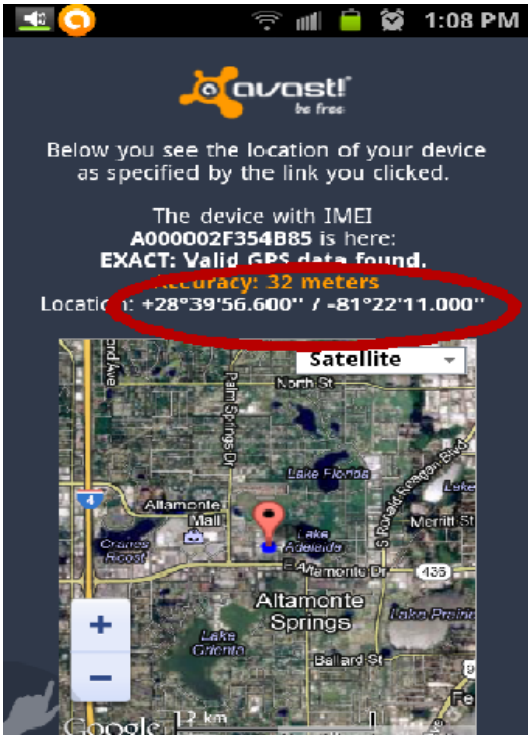



**Figura 14 Aplicación Avast!**

## *3.2.- AVG Mobilation Anti-Virus*

Aparte de ser aplicación antivirus, también ofrece la posibilidad de localizar el teléfono en caso de pérdida y/o robo mediante la utilización de Google Maps, mostrando su ubicación de forma remota. Tiene una función SHOUT, que permite subir al máximo el volumen para ayudar a localizarlo con facilidad. Permite bloquear el teléfono a través de SMS, o a través de www.avgmobilation.com, impidiendo que terceros tengan acceso a su información.

También permite crear un bloqueo de pantalla personalizado, con detalles de contacto, para facilitar la devolución del dispositivo.

Además, se puede realizar un borrado remoto de los datos del teléfono, permitiendo borrar fotografías, calendarios y mensajes, permitiendo realizar copias de seguridad en la tarjeta SD. Todo esto con solo 1,2 MB de espacio.

Esta aplicación también dispone una versión Pro de pago, que además de lo ya comentado añade:

- Protección contra malware y Spyware.
- Identificación de configuraciones del dispositivo no seguras.
- Seguridad en correos electrónicos.
- Localización del terminal en mapa.
- Alarma y bloqueo remoto.

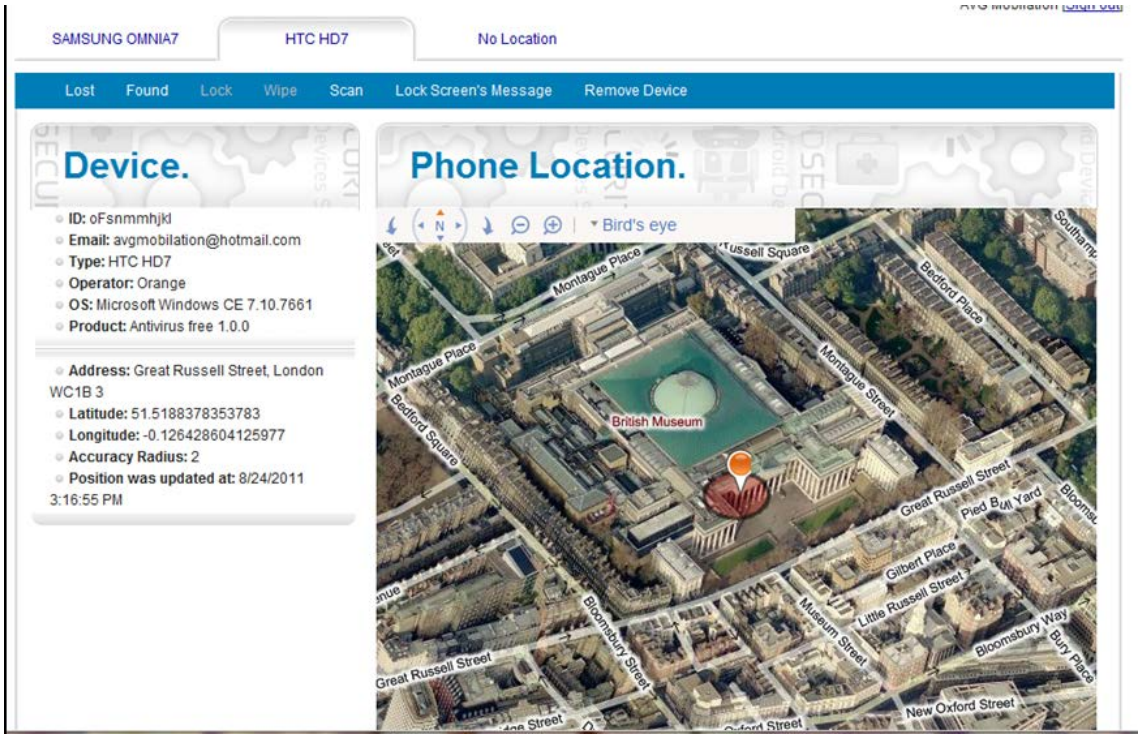



**Figura 15 Aplicación AVG Mobilation**

## *3.3.- Cerberus*

Cerberus es una de las más famosas y potentes aplicaciones en cuanto a seguridad se refiere. Una vez instalada y tras registrarse con un correo electrónico, ya se podrá acceder a la versión Web (www.ceberusapp.com). En esta Web, se puede observar la ubicación del dispositivo, pudiendo realizar estas diferentes opciones:

- Iniciar rastreo: permite localizar, con exactitud el dispositivo, mostrando sus movimientos.
- Recibir información del dispositivo: Obtiene la dirección IP, el operador y el número de serie de la tarjeta SIM.
- Bloqueo remoto: permite bloquear completamente un teléfono, pudiendo ser desbloqueado mediante un código.
- Iniciar una alarma con un mensaje: permite empezar a sonar una alarma mientras se muestra un mensaje por la pantalla.
- Obtener registro de llamadas.
- Mostrar/ocultar aplicación: puede ocultar la aplicación para evitar que sea desinstalada.
- Borrado de memoria: Borra todo el contenido de la tarjeta SD.
- Grabación: realiza grabaciones de audio, vídeo y toma fotografías de forma silenciosa.
- Llamar a teléfono: permite llamar al teléfono de la persona que posee el dispositivo perdido.

Tiene un coste de 2.99€ permitiendo rastrear hasta 5 móviles diferentes.

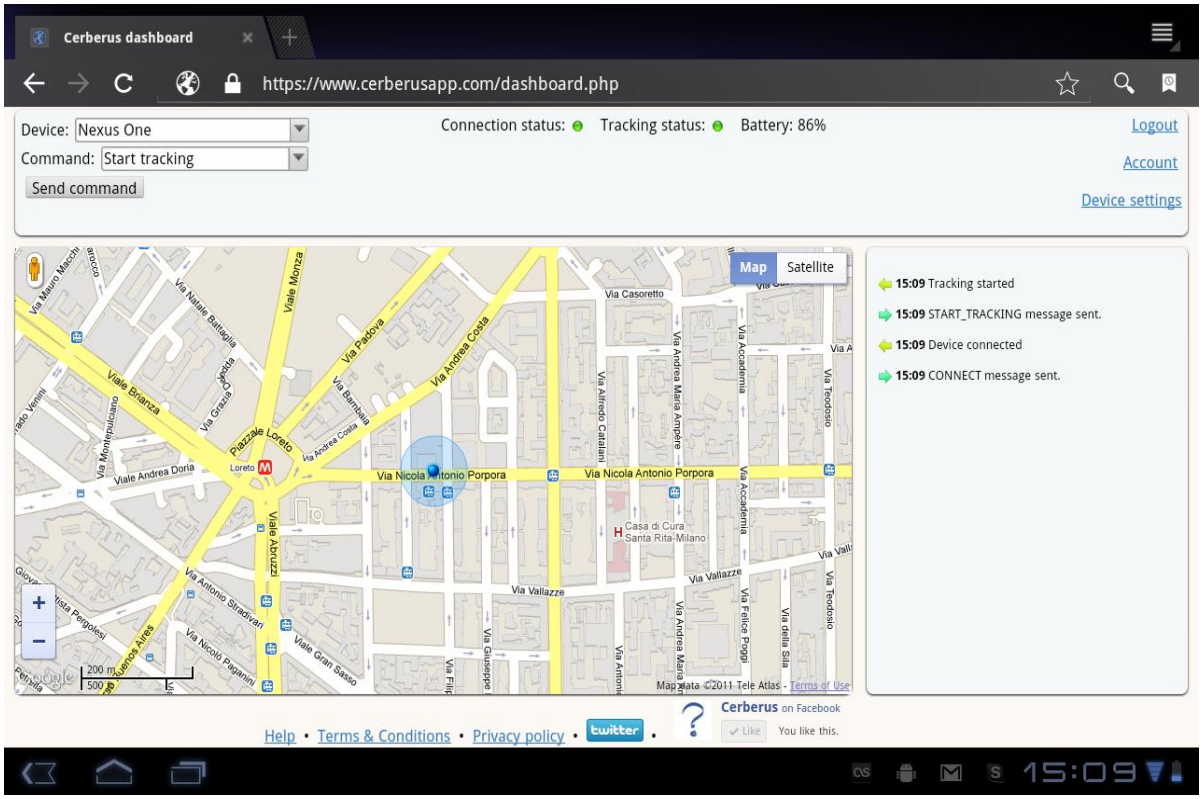



**Figura 16 Aplicación Cerberus localizando un terminal.**

## *3.4.- LookOut*

LookOut es una aplicación de seguridad integral para móviles, que está presente en plataformas como Android, Blackberry o Windows Mobile que permite realizar copias de seguridad, buscar malware en el terminal, o localizarlo en caso de pérdida.

Existen dos versiones, una gratuita y otra de pago, que mediante el pago de 2.99$ al mes o 29,99$ al año añade funcionalidades no disponibles en la versión gratuita, como por ejemplo la de bloquearlo y borrar los datos del dispositivo. Tiene las siguientes características:

- Localización: permite la localización del dispositivo usando el GPS, mostrando su ubicación en un mapa vía Web (Google Maps).
- Alarma: permite activar una alarma para poder oírla si el dispositivo se encuentra cerca.
- Copia de seguridad: realiza copia de seguridad de los contactos, subiéndola a la aplicación Web.

La aplicación de pago Premium añade las siguientes funcionalidades:

- Bloquear el dispositivo.
- Borrar todos los datos de tarjeta SD.
- Copia de seguridad.

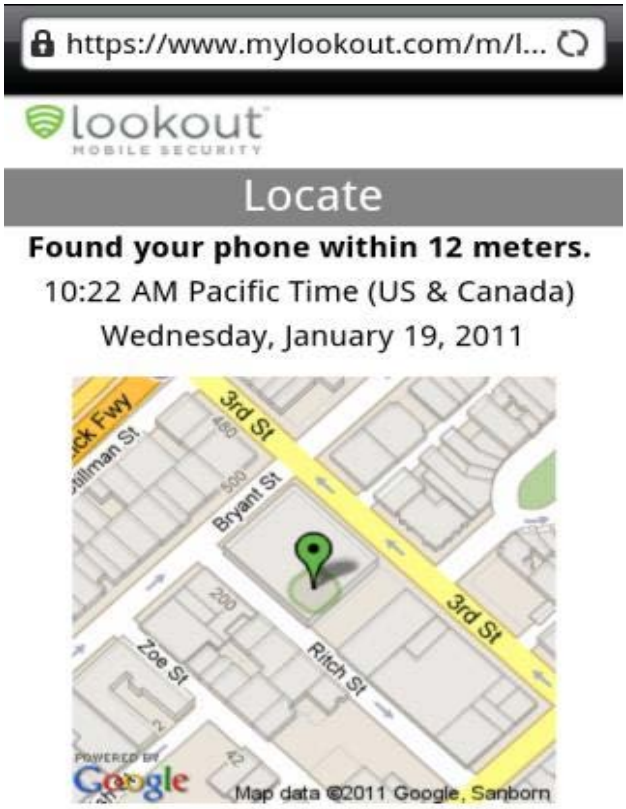



**Figura 17 Aplicación Lookout localizando un terminal.**

## *3.5.- McAfee WaveSecure*

WaveSecure es una herramienta de seguridad, muy premiada, ganadora por ejemplo de la Android Developer Challenge 2, y que cuenta con el beneplácito de Google.

El funcionamiento de esta aplicación es sencillo, cuando detecta que la tarjeta SIM introducida no es la correcta, manda un SMS al número anterior informando del cambio de tarjeta. Entre sus funciones destacar las siguientes:

- Copias de seguridad: permite realizar copias de seguridad de mensajes, contactos, llamadas, archivos multimedia en la nube, pudiendo restaurarlos cuando sea necesario.
- Bloqueo de terminal: WaveSecure ofrece la posibilidad de bloquear el terminar remotamente, enviando un SMS o mediante la aplicación Web. También se puede bloquear automáticamente cuando se detecte un cambio en la SIM.
- Alarma.
- Borrado de datos: permite borrar los datos del terminal.
- Localización: permite localizar el dispositivo, mostrando su ubicación en un mapa.
- Rastreo de la tarjeta SIM introducida y las llamadas realizadas.

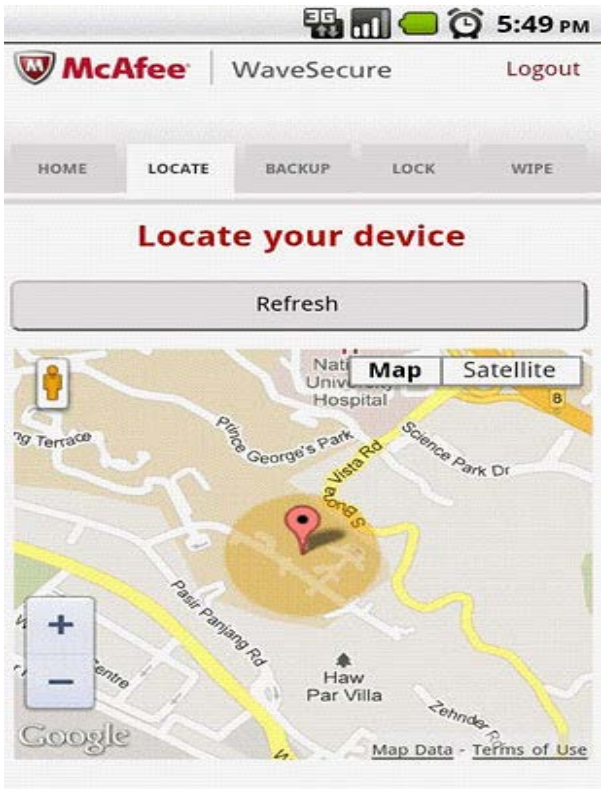



**Figura 18 Aplicación McAfee WaveSecure localizando un terminal.**

## *3.6.- Norton Mobile Security*

Norton es una conocida suite antivirus, que por supuesto, no podía faltar a su cita con Android. Protege el terminal y la privacidad de los datos importantes en caso de pérdida. Para proceder a la localización del terminal, se puede utilizar la aplicación Web o bien mediante comandos SMS.

Dispone de una versión específica para tablets, en la cual solo se puede buscar el dispositivo mediante la aplicación Web.

Para las tablets, dispone de los siguientes servicios:

- Bloqueo remoto: permite bloquear el dispositivo remotamente mediante interfaz Web, permitiendo mostrar un mensaje en la pantalla bloqueada.
- Fotografía: permite realizar fotografías de la persona poseedora de la tablet.
- Localización: permite localizar el terminal en un mapa.

Para dispositivos móviles permite:

- Localizar, bloquear y borrar datos del terminal mediante comandos SMS.
- Localizar el móvil en un mapa mediante la Web.
- Permite enviar mensajes al poseedor del terminal.
- Realizar fotografía.

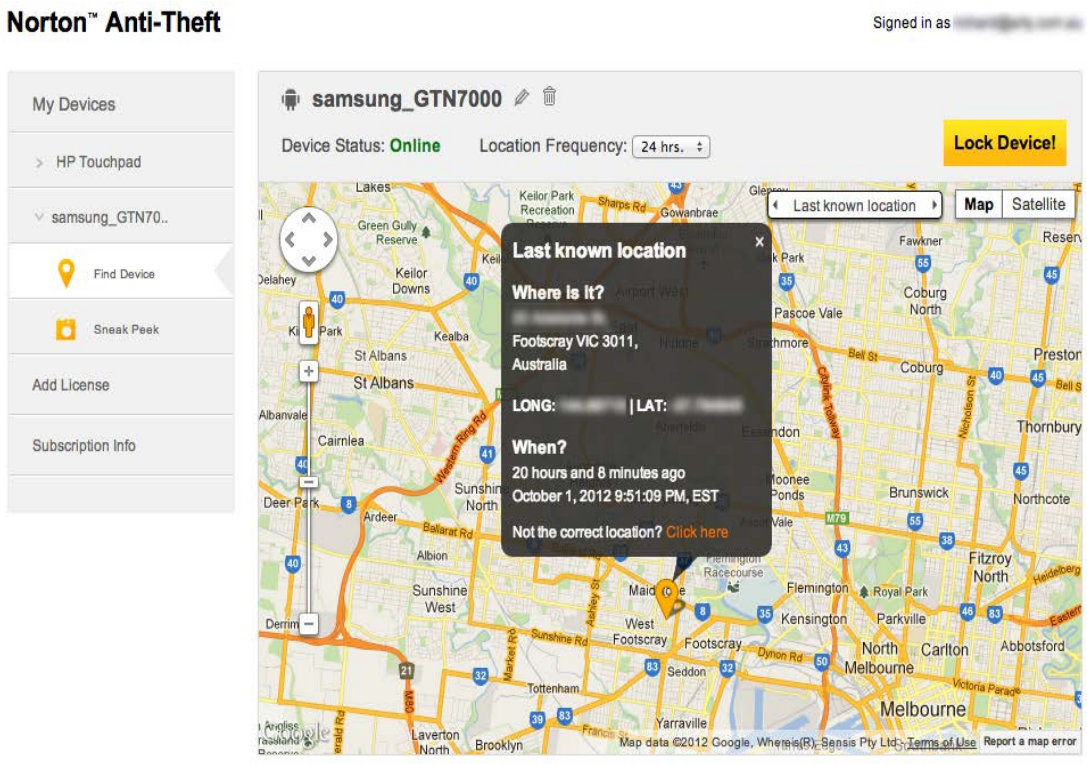



**Figura 19 Aplicación Norton Mobile Security localizando un terminal.**

### *3.7.- PhoneLocator Pro*

Phonelocator es un aplicación de monitoreo que permite a los usuarios localizar y recuperar los dispositivos perdidos o robados. Sus características más importantes son las siguientes:

- Protección con contraseña para evitar desactivar o desinstalar.
- Permite esconder la aplicación.
- Permite enviar la ubicación del terminal de forma temporal.
- Realiza fotografía de la persona poseedora del terminal.
- Notifica el cambio de la tarjeta SIM.
- Esconde los mensajes SMS enviados y recibidos de forma remota.
- Alarma.

Ha recibido algunas críticas por parte de usuarios aduciendo que es poco configurable, y no tiene un funcionamiento correcto, teniendo en cuenta que es una aplicación de pago (su coste es de 2.88 dólares) y hay aplicaciones gratuitas más completas.

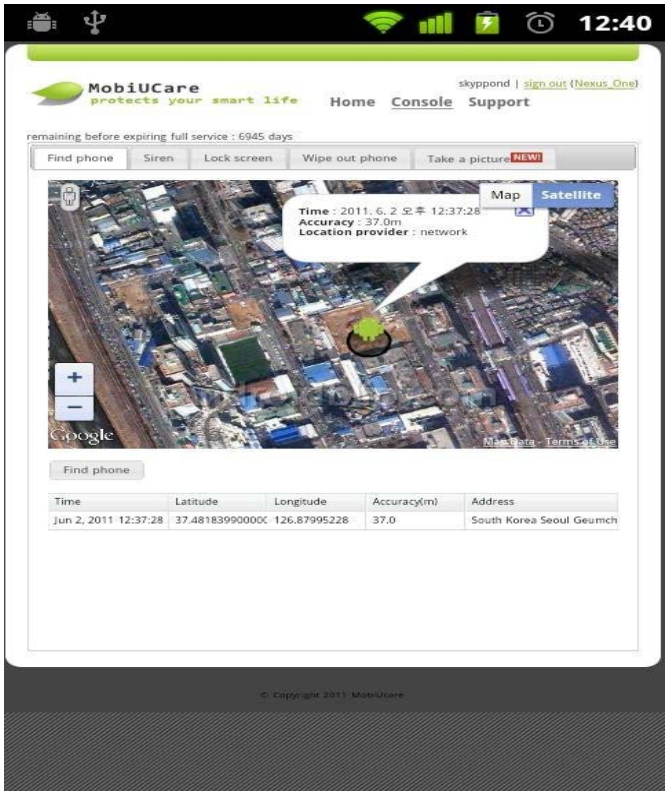



**Figura 6 Aplicación PhoneLocator localizando un terminal.**

## *3.8.- Seek Droid*

Es otra aplicación que permite localizar un dispositivo perdido o robado. La aplicación se ejecuta en segundo plano y lo que hace es conectarse al servidor del fabricante de la aplicación para ayudar a localizar el dispositivo. Para ello. se puede operar de dos maneras, ya sea mediante Web o mediante SMS. Las características más importantes que posee son las siguientes:

- Control de múltiples dispositivos con una única cuenta.
- Localización de dispositivos, mostrando su ubicación en un mapa.
- Realizar seguimientos del dispositivo.
- Alarma.
- Posibilidad de enviar mensajes que se muestren en pantalla.
- Bloquear dispositivo a través de Internet o de SMS.
- Recuperar las llamadas recientes.
- Borrar de forma remota los datos del teléfono y de la tarjeta SD.
- Ocultar la aplicación.
- Obtener datos del terminal (IMEI, SIM serial…).

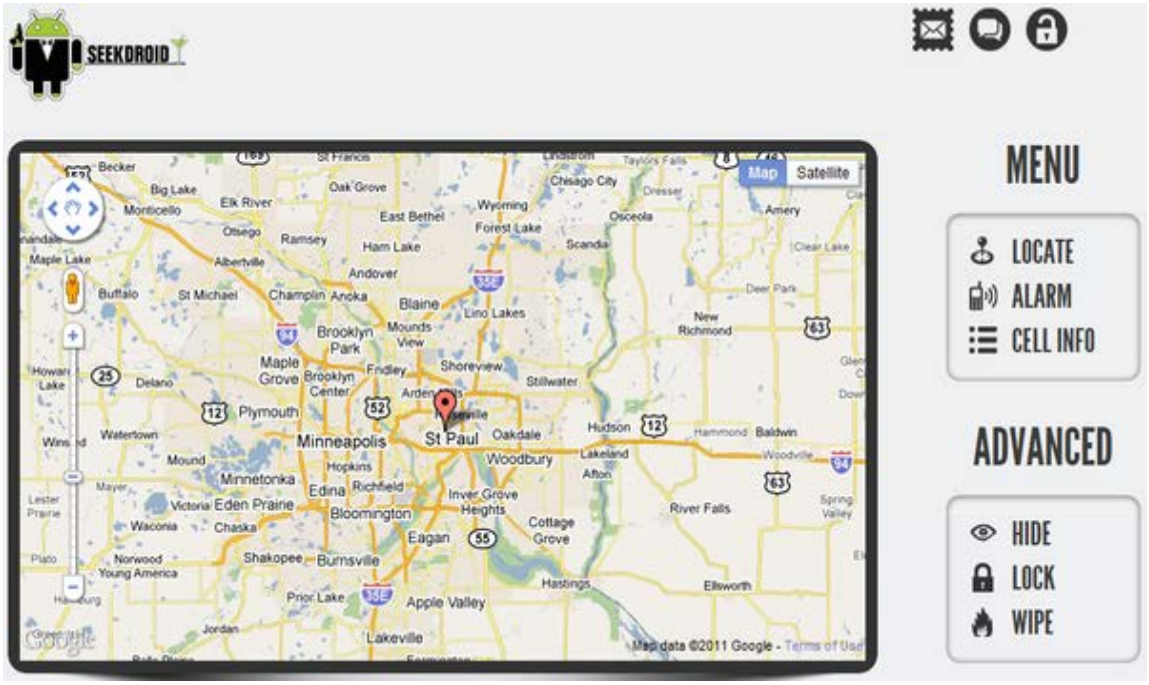



**Figura 22 Aplicacion Seek Droid localizando un terminal.**

## *3.9.- Comparativa*

Después de haber visto las principales aplicaciones que se dedican a la localización de dispositivos, en este apartado se presenta una tabla comparativa de las aplicaciones ya expuestas, resumiendo y comparando las principales características de cada una. Comentar que por motivos de claridad solo se van a exponer en la tabla aquellas aplicaciones que disponen de una mayor valoración de los usuarios en Google Play.

| Características | Avast! Mobile | AVG Mobilation | Cerberus | Lookout | Seek Droid |
|---|---|---|---|---|---|
| Notificación SIM | SI | NO | SI | NO | NO |
| Modo invisible | SI | NO | SI | NO | SI |
| Localización en mapa | SI | SI | SI | SI | SI |
| Bloqueo remoto | SI | SI | SI | SI* | SI |
| Alarma remota | SI | SI | SI | SI | SI |
| Comandos SMS | SI | SI | SI | NO | NO |
| Copia de contactos | NO | NO | NO | SI | NO |
| Borrado remoto | SI | SI | SI | SI* | SI* |
| Aplicación Web | NO | SI | SI | SI | SI |
| Información datos ladrón | NO | NO | SI | NO | SI |
| Funciona sin SIM | NO | NO | NO | NO | SI |
| Valoración Google Play (sobre 5) | 4,7 | 4,5 | 4,7 | 4,5 | 4,5 |
| Versión mínima Android | 2,2 | 1,5 | 2,2 | 1,5 | 2,2 |
| Precio | Gratuita | Gratuita | 2,99 € | Gratis / 2€ mensuales /20€ anuales | Gratuita / 2,24 € |

**Tabla 4 Comparativa de las diferentes aplicaciones de seguridad estudiadas**

* Solo versión Premium de pago.

En cuanto a las aplicaciones presentadas en la tabla comparativa destacan sobretodo dos, Avast! y Cerberus. Son las dos más completas, coincidiendo además con la valoración de los usuarios que la han utilizado. Son las únicas que notifican al usuario cuando se introduce una tarjeta SIM no válida, y ambas ofrecen la posibilidad de ocultar la aplicación para evitar su desinstalación. Cerberus además ofrece la posibilidad de buscar el dispositivo a través de su página Web, lo que evita gastos innecesarios y proporciona más anonimato al no tener que dar a conocer ningún número de teléfono. Tiene la posibilidad de obtener datos del poseedor del terminal, pudiendo captar su fotografía, los datos de la tarjeta SIM para poder realizar la denuncia, y ofrece además, la posibilidad de realizar grabaciones de audio y vídeo, capturas del escritorio del terminal.

La aplicación realizada en el presente proyecto permite la notificación al introducir otra tarjeta SIM, ya que avisará a los contactos introducidos en la aplicación de que se ha producido una intrusión. Además la aplicación dispone de una opción para deshabilitar esta protección, en caso de que voluntariamente se desee introducir otra tarjeta SIM.

Dispone de un modo invisible, es decir, la aplicación será ocultada automáticamente siempre que otra tarjeta SIM distinta se introduzca en el dispositivo. Esta medida de protección se utiliza para que en caso de robo del terminal, evitar que alguien pueda introducir su tarjeta SIM y entonces pueda desinstalar la aplicación. Para volver a hacer visible la aplicación, bastará con iniciar el dispositivo introduciendo la tarjeta SIM correcta previa validación de código PIN.

La aplicación facilita la localización del terminal móvil en un mapa previo comando SMS, utilizando el sistema GPS del dispositivo. Para ello utiliza Google Maps. Si mediante el mapa se sitúa el móvil en un lugar cercano podrá encontrarlo gracias a la alarma. La alarma sonará aunque el dispositivo se encuentre en "modo silencioso".

Dispone también de un bloqueo remoto mediante comando SMS que bloqueará su utilización hasta que se introduzca la tarjeta SIM válida.

Para intentar obtener la foto de la persona que ha robado el terminal móvil, la aplicación simula un recordatorio de la agenda, mostrando por pantalla que tiene cita con el médico. Cuando se intente posponer la cita la aplicación realizará una foto con la cámara frontal enviándola por Internet a la dirección de correo introducida en la aplicación.

También posee un comando SMS que fuerza al teléfono a que haga una llamada al número de teléfono emisor del SMS. Mediante otro comando se podrá mostrar un mensaje en la pantalla del terminal a modo de comunicación. Por ejemplo si se ha perdido el móvil se puede mostrar un mensaje que indique que si alguien encontrase el teléfono, que se pusiera en contacto con un número de teléfono. El mensaje que se mostrará será el que se indique en el comando SMS.

Además posee otras funcionalidades, como la de obtener los datos de los contactos a modo de copia de seguridad, facilita el obtener los datos del teléfono y los datos de la tarjeta SIM del posible ladrón, y además en caso de que no se inserte una tarjeta SIM en el dispositivo móvil la aplicación se ocultará para evitar que sea desinstalada.

A modo de innovación, esta aplicación no se ha diseñado utilizando un paradigma de orientación a objetos, sino que se ha utilizado una orientación a eventos, lo cual provoca una gran ventaja al usuario. No hay un consumo extra de batería por la ejecución de aplicaciones. El resto de aplicaciones siempre están en ejecución, si se utiliza un administrador de tareas se observa como siempre hay un proceso en ejecución. Con SimDetect esto no ocurrirá, ya SimDetect reacciona a eventos, como pueden ser la utilización de comandos SMS, hará la función a realizar y finalizará a la espera de obtener otro evento. Es decir, SimDetect no provoca un aumento del consumo de la batería. Algo que sin duda, el usuario agradecerá.

# 4.- Fase de análisis

Este capítulo trata sobre la fase de análisis de la aplicación, y en él se pueden encontrar tres apartados. En el primero se expondrá el documento de requisitos, en el cual se detallan cada uno de los requisitos y características que debe cumplir la aplicación. En el segundo se presentará la planificación seguida durante la realización de dicha aplicación, y en el tercer apartado se presentará una estimación de costes de la aplicación.

## *4.1.- Documento de Especificación de Requisitos*

### 4.1.1.- Introducción

En este apartado se detallará el análisis y el diseño de la aplicación, siguiendo el estándar IEEE Recommended Practice for Software Requirementes Specifications IEEE 830-1998 [IEEE 1830].

### 4.1.2.- Propósito

El objetivo de este módulo es el de plasmar de una forma precisa el análisis de la aplicación SimDetect, para lo que se definirá de una forma clara, una especificación de requisitos de las funcionalidades que tendrá la aplicación y sus posibles restricciones.

Se detallará la planificación del proyecto mediante un diagrama Gannt, comentando y explicando los aspectos más importantes de cada una de las fases. Se presentará un cálculo de costes, mediante el cálculo de puntos función, con el cual se realizará una estimación económica aproximada.

### 4.1.3.- Alcance

El objetivo es obtener un sistema software de seguridad en dispositivos móviles Android, que facilite al usuario el poder localizar el terminal móvil en caso de pérdida o robo.

### 4.1.4.- Definiciones, acrónimos

**Tarjeta SIM (*Subscriber Identity Module):*** de identificación del suscriptor. Es una tarjeta usada en teléfonos móviles y módems HSDPA o HSUPA, que permite identificarse ante la red.

**Numero de serie de la tarjeta SIM:** código que identifica de forma univoca a cada tarjeta SIM a nivel mundial.

**API (*Application Programming Interface):*** es un conjunto de procedimientos que permiten que una aplicación pueda interactuar con otra completamente distinta como una capa de abstracción.

**XML (*Extensible Markup Languaje):*** lenguaje de marcas extensible en castellano, es un metalenguaje que sirve para almacenar datos de forma estructurada.

**SMS (*Short Message Service):*** servicio de mensajes cortos es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos de texto entre teléfonos móviles

**GPS (*Global Positioning System)*** sistema de posicionamiento global, es un sistema que permite determinar, mediante la triangulación con satélites, la posición de un objeto o persona en el mundo.

**E-mail:** o correo electrónico, es un servicio de red que permite a los usuarios enviar y recibir mensajes y archivos mediante sistemas de comunicación electrónicos.

**Código de acción:** código que permitirá a la aplicación SimDetect realizar alguna acción predeterminada en caso de pérdida o sustracción del dispositivo móvil.

**Usuario propietario:** se entenderá como usuario propietario aquella persona propietaria legítima del dispositivo y que ha instalado la aplicación SimDetect.

**Usuario sustractor:** se entenderá como usuario sustractor aquella persona que ha sustraído de forma ilegítima el dispositivo móvil del usuario, privándole de su uso.

**IMEI (*International Mobile Equipment Identity):*** *Identidad Internacional de Equipo Móvil* es un código pre-grabado en los teléfonos móviles GSM que identifica al aparato unívocamente a nivel mundial, y es transmitido por el aparato a la red al conectarse a ésta.

**Operador:** aquella compañía telefónica que provee servicios de telefonía para sus clientes.

**IMSI (*International Mobile Subscriber Identity)*** Identidad Internacional del Abonado a un Móvil, es un código de identificación único para cada dispositivo de telefonía móvil, integrado en la tarjeta SIM, que permite su identificación a través de las redes GSM y UMTS.

### 4.1.5.- Descripción global

Para explicar el funcionamiento que tendrá la aplicación se dividirá en tres fases, que se diseccionarán explicando más detalladamente su funcionamiento.

La primera fase consistirá en almacenar los datos de contacto que se utilizarán en caso de robo. Se podrán insertar hasta dos números de teléfono y hasta dos e-mails. En la segunda fase, cada vez que se inicie el dispositivo, se comprobará si la tarjeta SIM es la correcta.

La tercera fase se ejecutará cuando un mensaje SMS llegue al dispositivo móvil. La aplicación lo recogerá buscando algún código predefinido en la aplicación, En caso de encontrarlo se realizarán las tareas propias de dicho código, definidas en los requisitos. En caso de no encontrar ningún código en el SMS, la aplicación finalizará.

En la primera fase, una vez instalada la aplicación, se deberá de rellenar los datos de contacto (Ver RQ 1, RQ 2) y se almacenarán de forma persistente tanto los datos de contactos como los datos de la tarjeta SIM (RQ 5).


**Figura 23 Funcionamiento de la aplicación en la primera fase.**

En la segunda fase, SimDetect se ejecutará cuando el sistema operativo Android arranque, obteniendo los datos de la tarjeta SIM introducida y comparándolos con los ya almacenados. Si los datos son los mismos la aplicación finalizará.

En caso de no ser los mismos notificará al usuario propietario que otra tarjeta SIM ha sido introducida en su dispositivo de dos maneras:

- Mediante SMS a los números introducidos en la primera fase.
- Mediante correo electrónico a la dirección introducida en la primera fase.

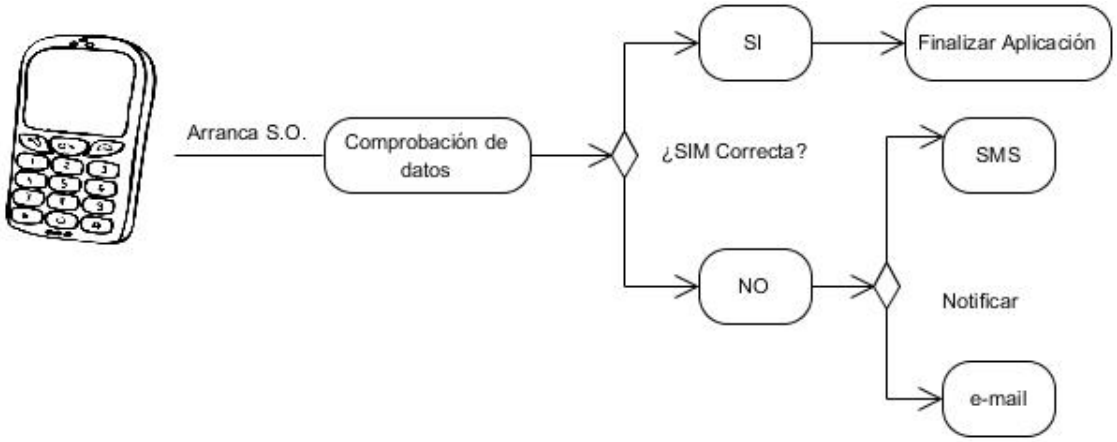



**Figura 24 Funcionamiento de la aplicación en la segunda fase.**

En la tercera fase, la aplicación arrancará automáticamente cuando el dispositivo reciba un SMS, leyéndolo y buscando ciertos códigos. Si no encuentra dichos códigos interpretará que el SMS es de tipo personal, no actuando; en caso contrario, leerá el código, y actuará según el código insertado.

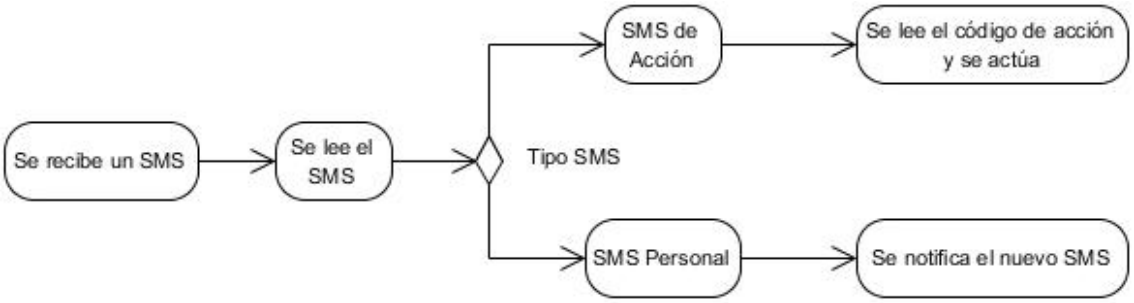



**Figura 25 Funcionamiento de la aplicación en la tercera fase.**

### 4.1.6.- Interfaces hardware

El sistema deberá tener en cuenta las diferentes interfaces hardware que se utilizarán, tal y como se refleja en el siguiente diagrama de componentes y despliegue:

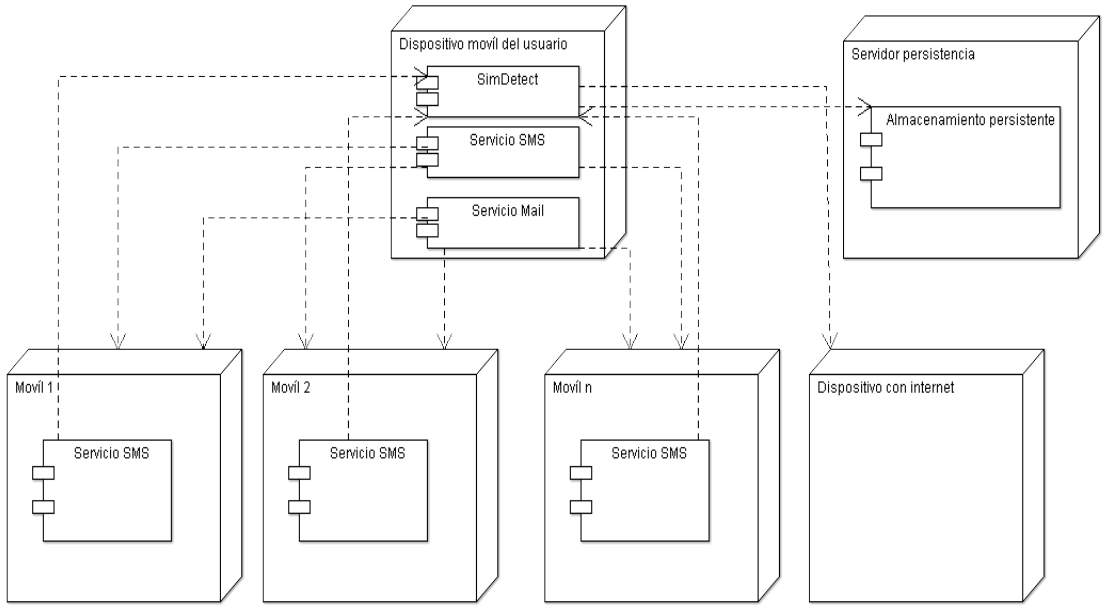



**Figura 26 Diagrama de componentes y despliegue de la aplicación.**

Con la aplicación instalada en el dispositivo del usuario propietario, la aplicación se comunicará a través de mensajes SMS. Los SMS que envíe la aplicación podrán ser leídos desde cualquier terminal móvil. A su vez los correos electrónicos enviados, se podrán leer desde cualquier dispositivo con conexión a Internet.

La aplicación almacenará los datos de la tarjeta SIM mediante un sistema que asegure la persistencia de éstos. Este sistema podrá estar contenido dentro del propio terminal móvil.

Además el dispositivo deberá tener las siguientes características hardware:

- **RQH 1.** El terminal móvil deberá tener tarjeta SIM y un receptor GPS integrado.
- **RQH 2.** El terminal móvil deberá tener cámara fotográfica frontal integrada.

### 4.1.7.- Interfaces software

En el dispositivo móvil del usuario propietario:

- **RQS 1.** Capacidad de enviar/recibir SMS.
- **RQS 2.** Android 2.3 o superior.

En el dispositivo móvil suplementario (para la localización):

- **RQS 3.** Capacidad de enviar/recibir SMS.
- **RQS 4.** Navegador Web

### 4.1.8.- Requisitos de la interfaz

- **RQI 1.** La interfaz de la aplicación se deberá mostrar en español o en inglés, según el idioma del dispositivo móvil.
- **RQI 2.** En caso de que el dispositivo móvil no sea ni inglés ni español se mostrará en inglés.
- **RQI 3.** Se buscará que las interfaces se observen correctamente en cualquier pantalla.

### 4.1.9.- Requisitos específicos

La aplicación SimDetect deberá cumplir los siguientes requisitos:

- **RQ 1.** La aplicación deberá pedir dos números de teléfono de confianza.
- **RQ 2.** La aplicación deberá pedir una dirección de correo electrónico.
- **RQ 3.** La aplicación deberá iniciarse cuando el sistema operativo Android arranque.
- **RQ 4.** Al arrancar por primera vez, la aplicación deberá mostrar la interfaz para rellenar los contactos de confianza.
- **RQ 5.** El almacenamiento de los datos será de forma persistente.

- **RQ 6.** La aplicación realizará una comprobación al arrancar el sistema operativo de los datos de la tarjeta SIM insertada, comparándolos con los almacenados con anterioridad.

- **RQ 7.** En caso de no coincidir los datos, se deberá avisar a los números de teléfono almacenados mediante SMS y a la dirección de correo mediante un correo electrónico.

- **RQ 8.** En caso de coincidir los datos, en la comprobación inicial, se deberá finalizar la aplicación.

- **RQ 9.** La aplicación deberá ser capaz de detectar y leer un SMS enviado desde cualquier terminal móvil, independientemente de su antigüedad y de su sistema operativo.

- **RQ 10.** Cuando reciba un SMS, la aplicación deberá discriminar entre SMS personales y SMS de acción.

- **RQ 11.** Los SMS de acción serán aquellos que comiencen por la palabra "signal" y contengan un código de acción.

- **RQ 12.** El resto de SMS se considerarán SMS personales.

- **RQ 13.** Los SMS de acción no deberán ser notificados al usuario sustractor ni mostrar ningún aviso.

- **RQ 14.** Los SMS de acción no deberán dejar rastro en el dispositivo móvil.

- **RQ 15.** El mensaje de acción que incorpore el código "signal alarm" deberá provocar que suene un sonido con volumen alto, de tal manera que se pueda encontrar el dispositivo móvil en un radio de alcance cercano.

- **RQ 16.** El mensaje de acción que incorpore el código "signal locate" deberá obtener las coordenadas GPS en las que se encuentra el dispositivo móvil físicamente y enviarlas al móvil remitente mediante mensaje SMS, así como a la dirección de correo electrónico mediante e-mail.

- **RQ 17.** El mensaje de acción que incorpore el código "signal blood" deberá obtener la lista de contactos del usuario sustractor enviando un SMS a cada contacto, informando que el usuario (sustractor) está utilizando un dispositivo móvil robado.

- **RQ 18.** El mensaje de acción que incorpore el código "signal block" deberá bloquear el dispositivo, impidiendo que el usuario sustractor pueda utilizarlo.

- **RQ 19.** El mensaje de acción que incorpore el código "signal text" deberá mostrar un mensaje introducido en el propio SMS y que se mostrará en la pantalla del dispositivo.

- **RQ 20.** El mensaje de acción que incorpore el código “signal recovery” deberá obtener una copia de seguridad de la lista de contactos del teléfono enviándola a la dirección de correo electrónico mediante e-mail, y al móvil remitente del mensaje SMS.

- **RQ 21.** El mensaje de acción que incorpore el código “signal data” deberá obtener IMEI, el operador, el serial de la tarjeta SIM y el IMSI para realizar la denuncia ante la autoridad competente. Estos datos serán los que se enviarán al número de teléfono remitente del SMS.

- **RQ 22.** El mensaje de acción que incorpore el código “signal camera” activará la cámara del dispositivo, buscando realizar una fotografía del usuario sustractor. Esta fotografía será enviada a la dirección de correo electrónico mediante e-mail.

- **RQ 23.** El mensaje de acción que incorpore el código “signal call” realizará una llamada desde el dispositivo al remitente del SMS, de tal manera que el usuario propietario pueda escuchar las conversaciones del usuario sustractor.

- **RQ 24.** En todo momento se buscará que el consumo de recursos sea el menor posible, para no afectar a la duración de la batería.

- **RQ 25.** La aplicación se podrá deshabilitar para poder insertar voluntariamente otra tarjeta SIM.

- **RQ 26.** Se deberá facilitar la localización del dispositivo mediante un mapa.

- **RQ 27.** Cuando un dispositivo se bloquee remotamente mediante el código “signal block”, se deberá impedir que se pueda desbloquear mediante la utilización de alguna tecla predefinida de Android.

- **RQ 28.** Cuando un dispositivo esté bloqueado remotamente mediante el código “signal block” se deberá mantener el bloqueo aunque se reinicie el dispositivo.

- **RQ 29.** Cuando un dispositivo esté bloqueado remotamente mediante el código “signal block” solo se desbloqueará introduciendo la SIM válida.

- **RQ 30.** Cuando en el dispositivo se introduzca una tarjeta SIM no válida, la aplicación deberá desaparecer de la lista de aplicaciones.

- **RQ 31.** Los datos de los contactos de confianza se podrán eliminar.

## *4.2.- Planificación estimada*

En este apartado se comentará la planificación seguida, detallándola en cada fase. Esta es la planificación seguida para la realización de dicho proyecto, ya que se ha adecuado casi a la perfección, lo planificado con lo realizado.

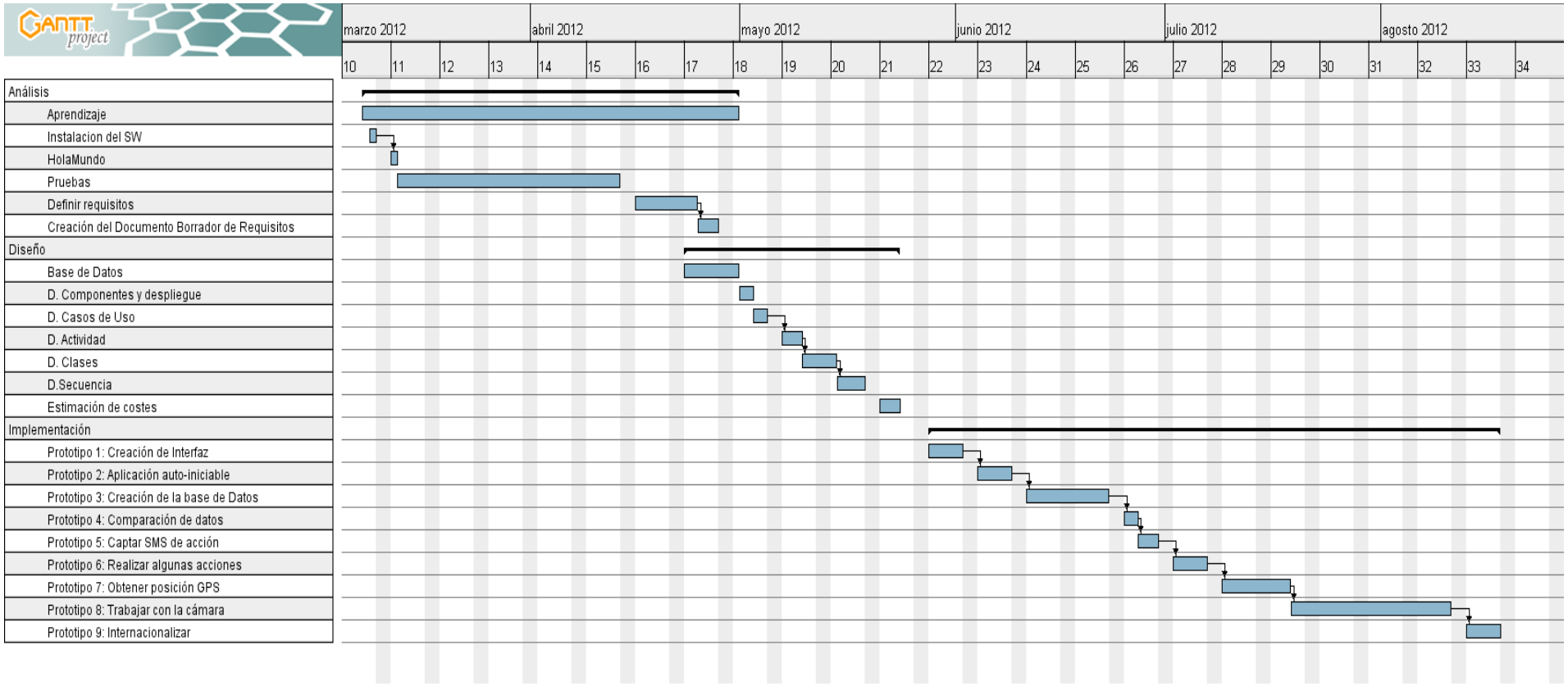



**Figura 27 Diagrama Gantt con la planificación seguida durante el desarrollo de la aplicación.**

A continuación se expondrá más detalladamente cada sección del diagrama Gantt presentado.

### 4.2.1.- Fase de Análisis

En primer lugar, se comentará la fase de análisis, en la cual se busca analizar el proyecto a realizar, realizando un análisis de riesgos, evaluando diferentes alternativas para la realización del proyecto. Para ello se han seguido los siguientes apartados:

- Comprensión del problema.
- Establecer marco de trabajo.
- Fijar las bases del diseño.
- Facilitar la verificación del cumplimiento de los objetivos.

La planificación seguida durante esta fase es la siguiente:

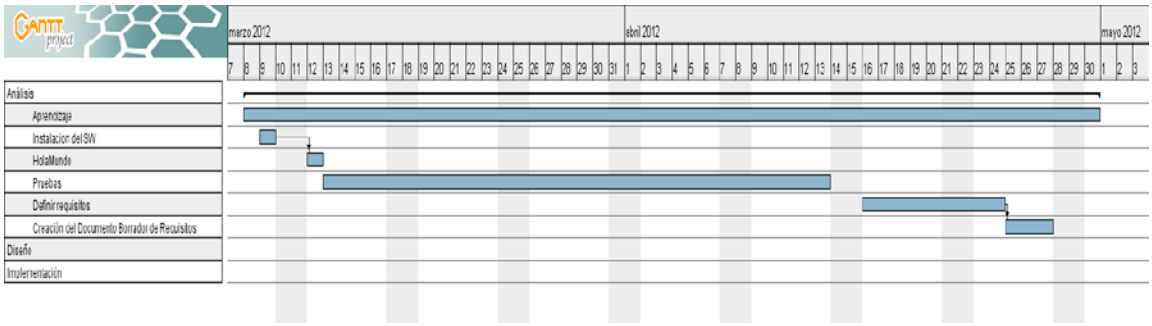



**Figura 28 Fase de análisis.**

La fase de análisis consta de las siguientes tareas:

- **Aprendizaje:** Al tratarse de un proyecto para una nueva plataforma, se deben aprender las peculiaridades, tanto de la propia plataforma, como del lenguaje de programación utilizados. Para ello, se han utilizado libros y manuales, incluidos en la sección de bibliografía.
- **Instalación del SW:** evaluación del software para desarrollar aplicaciones en Android, e instalación del sistema elegido.
- **HolaMundo:** como todo buen programador, cuando se inicia una labor en una nueva plataforma se realiza un HolaMundo.
- **Pruebas:** como no solo de HolaMundos vive un programador, se profundizan los conocimientos en la nueva plataforma mediante programas de prueba.
- **Documento Borrador de Requisitos:** se establecen los requisitos que deberá cumplir la aplicación a realizar.

Esta fase del diseño está planificada para realizarse en un total de 54 días.

### 4.2.2.- Fase de Diseño

En la fase de diseño, se busca descomponer y organizar el sistema en elementos que puedan ser desarrollados por separado, plasmándolos en diagramas estandarizados mediante el lenguaje UML.

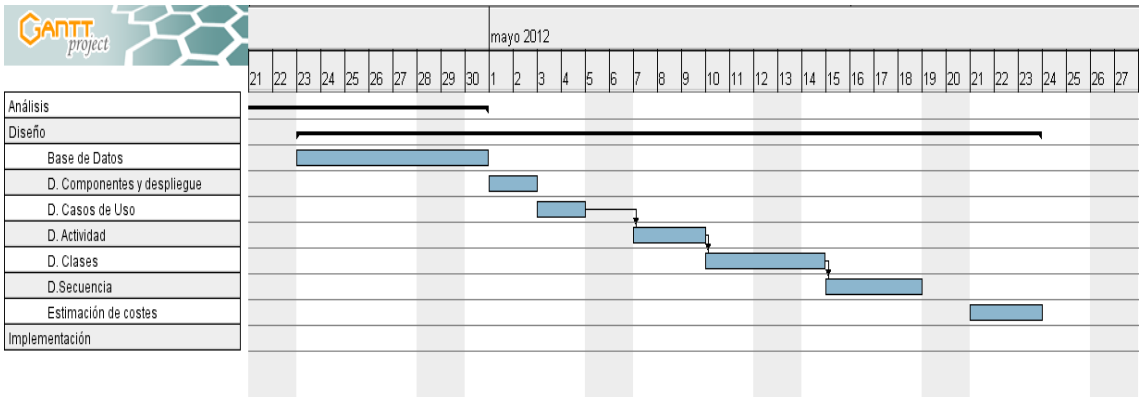



**Figura 29 Fase de diseño.**

Las tareas a realizar en esta fase son los siguientes:

- **Base de Datos:** se realizará el diseño de la base de datos utilizada en la aplicación para almacenar ciertos datos de manera persistente.
- **Diagrama de Componentes y despliegue:** se referencia de manera esquemática los componentes que formarán parte del sistema.
- **Diagrama de Casos de Uso:** se expondrán las diferentes acciones o casos que el usuario podrá realizar utilizando la aplicación.
- **Diagrama de Actividad:** se expondrán de forma esquemática las acciones a realizar en cada subapartado de la aplicación.
- **Diagramas de Clases:** se presentarán las clases a implementar con sus correspondientes métodos y atributos, para realizar las acciones a realizar expuestas en el diagrama de actividad.
- **Diagramas de Secuencia:** se presentarán qué clases y métodos se utilizarán para resolver las acciones expuestas en los diagramas de actividad.

Esta fase del diseño está planificada para realizarse en un total de 54 días.

### 4.2.3.- Fase de Implementación

En la fase de Implementación se trata de realizar lo pensado en la fase de análisis y de diseño. Para ello se ha seguido un modelo en espiral, utilizando diferentes prototipos, que poco a poco se van complementando para acabar formando la aplicación final.

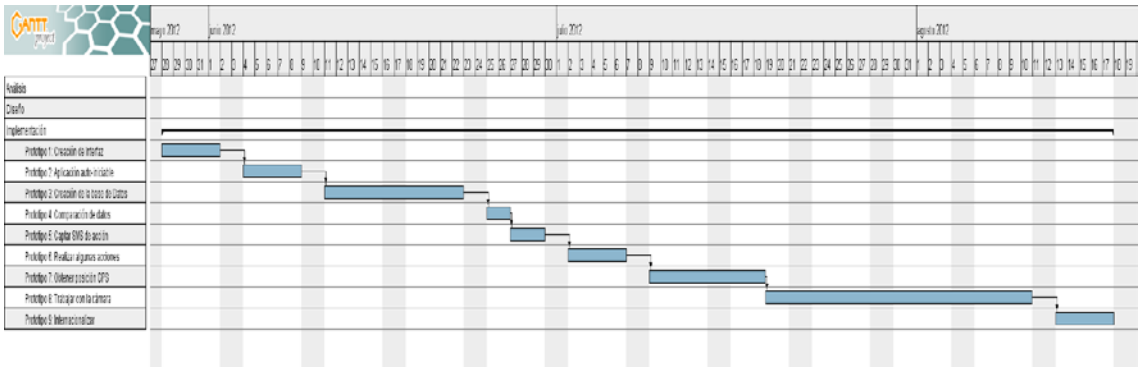


**Figura 30 Fase de implementación.**

Los prototipos utilizados son los siguientes:

**Prototipo1- Creación de la interfaz:** se crean las interfaces de la aplicación, buscando la máxima compatibilidad posible con la gran diversidad de pantallas que tienen los dispositivos móviles Android.

**Prototipo2- Aplicación auto-iniciable:** La aplicación se tendrá que iniciar cuando arranque el dispositivo móvil, por lo tanto, se buscará que el prototipo de aplicación se inicie cuando arranca el sistema operativo Android.

**Prototipo3- Creación de la base de datos:** se crea la base de datos (en adelante se comprueba si existe, y en caso negativo se crea) almacenando los datos más importantes tanto de la tarjeta SIM, como del dispositivo.

**Prototipo4- Comparación de datos:** en este prototipo se realiza la comparación entre los datos almacenados en la BD con los de la tarjeta SIM.

**Prototipo5- Captar SMS de acción:** se intenta controlar la llegada de SMS al dispositivo, discriminando entre SMS de acción y el resto. Los SMS de acción se capturan para evitar notificaciones.

**Prototipo6- Realizar algunas acciones:** se realizan las siguientes acciones:

- Alarm
- Blood
- Text
- Recovery
- Data
- Call

**Prototipo7- Obtener posición GPS:** se intenta obtener las coordenadas GPS para construir la dirección para crear el mapa.

**Prototipo8- Trabajar con la cámara:** se intenta realizar el modo para conseguir la captura de imagen mediante la cámara del terminal móvil.

**Prototipo9- Internacionalizar:** aunque la aplicación se ha ido desarrollando teniendo en cuenta la internacionalización, en esta última etapa se rellenan los datos para mostrar los mensajes en los dos idiomas, asegurándose de que no se olvida ningún mensaje.

La duración estimada de esta fase sería de 82 días.

Durante la realización de la aplicación se han cumplido los plazos la planificados en las fases de análisis, diseño e implementación. El único retraso sufrido fue al realizar las pruebas de la aplicación, que al estar de vacaciones no se pudo realizar la batería de pruebas serias a la aplicación, lo cual tuvo lugar en septiembre después de la finalización de los exámenes.

## *4.3.- Estimación de costes*

### 4.3.1.- Cálculo de los puntos función

Para calcular el esfuerzo, que va a requerir la realización de la aplicación, se va a utilizar el sistema de Puntos Función (PF), hallando los puntos función para proceder a calcular los costes. A continuación se describen los puntos que se han seguido para el cálculo de dicho esfuerzo.

Tipos de Puntos Función a contar:

**ILFs**: Internal Logical Files
**EIFs**: External Interface Files
**EIs**: External Inputs
**EOs**: External Outputs
**EQs**: External Inquirys

Para el cálculo de PF tendremos en cuenta que se trata de un proyecto de nuevo desarrollo, ya que sé trata de una nueva plataforma cuyas particularidades se desconocen.

**Internal Logical Files:** Conjunto de datos lógicamente relacionados, identificables por el usuario y mantenidos a través de procesos dentro de los límites de la aplicación.
En la aplicación serían los siguientes:

- Tarjeta SIM.
- Conexión Internet.
- Dispositivo GPS.
- SMS.
- E-mail.

La tabla 5 muestra los ILF detectados.

**External Interface Files:** grupo de datos relacionados lógicamente e identificables por el usuario, que se utilizan solamente para fines de referencia. Los datos residen enteramente fuera de los límites del sistema y se mantienen por las Entradas Externas de otras aplicaciones. En el caso de nuestro sistema no existen EIFs

| ILFS | | | |
|---|---|---|---|
| Nombre | DETs | RETs | Complejidad |
| Tarjeta SIM | 5 | 0 | Baja |
| Dispositivo (móvil) | 6 | 0 | Baja |
| SMS (enviar) | 2 | 0 | Baja |
| SMS (recibir) | 0 | 1 | Baja |
| e-mail | 1 | 0 | Media |

**Tabla 5 Internal Logical File detectados**

Antes de realizar el cálculo de los puntos función es necesario definir los términos External Input (EI), External Output (EO) y External Inquiries (EQ):

**External Input (EI):** Proceso elemental que introduce información en el sistema.
**External Output (EO):** Proceso elemental que introduce una salida de información en el sistema.
**External Inquiries (EQ):** Proceso elemental compuesto por una combinación de entrada-salida que obtiene una recuperación de datos internos.

| | Baja | | Media | | Alta | | Total |
|---|---|---|---|---|---|---|---|
| **EI** | 12 | **X3** | 0 | **X4** | 0 | **X6** | 36 |
| **EO** | 8 | **X4** | 0 | **X5** | 0 | **X7** | 32 |
| **EQ** | 0 | **X3** | 0 | **X4** | 0 | **X6** | 0 |
| **ILF** | 13 | **X7** | 1 | **X10** | 0 | **X15** | 101 |
| **ELF** | 0 | **X5** | 0 | **X7** | 0 | **X10** | 0 |
| | | | | | **PF Totales** | | **169** |

**Tabla 6 Puntos función detectados**

En External Input se añaden tres unidades al entender que el usuario debe introducir dos números de teléfono y una dirección de correo electrónico. Además habría que añadir los SMS de acción que podrían llegar (+9). Esto hace un total de 12.

En External Output habría que tener en cuenta las distintas salidas de datos que contiene la aplicación. Habría que añadir el envío de la posición GPS, el envío de los contactos, envío de los datos de la tarjeta SIM, y la cámara.

**Factor de ajuste y Puntos Función ajustados:**
Valoramos las distintas características del proyecto y definimos el valor de ajuste.

| Características | Grado de influencia |
|---|---|
| Comunicaciones de datos | 5 |
| Funciones distribuidas | 0 |
| Rendimiento | 5 |
| Configuraciones fuertemente utilizadas | 3 |
| Frecuencia de transacciones | 3 |
| Entrada de datos on-line | 2 |
| Eficiencia del usuario final (ayudas, scrolling, ventanas) | 2 |
| Actualizaciones on-line | 0 |
| Procesos complejos | 5 |
| Reutilización | 2 |
| Facilidad de instalación | 5 |
| Facilidad de operación (arranque, respaldo, recuperación) | 4 |
| Instalación de distintos lugares | 0 |
| Facilidad de cambios | 4 |
| **Grado total de influencia (TDI)** | 40 |

**Tabla 7 Valores para ajustar los puntos función**

**Valor del Factor de Ajuste**

El factor de ajuste es un coeficiente que sirve para adaptar el resultado de los puntos función extraídos anteriormente a las características del sistema diseñado en base a la importancia de éstas.

$$VAF = (TDI * 0.01) + 0.65 = (40 * 0.01) + 0.65 = 1.05$$

A partir del factor de ajuste, podemos hacer el cálculo de los Puntos Función Ajustados:

$$AFP = PF\ Totales * VAF = 166 * 1.05 = 177{,}45$$

### 4.3.2.- Cálculo de costes

A continuación se realizará una estimación del esfuerzo, con el que se estimará la duración aproximada en días que debería de tener la realización de la aplicación.

Productividad Estándar = 1PF/día-hombre
Esfuerzo = Tamaño * Productividad = 177 * 1 = 177

Plazo = Esfuerzo / Nº Recursos = 177 / 1 = 177 días

Con los Puntos Función se da una estimación de 177 días para realizar la aplicación.
Según el CONVENIO COLECTIVO DEL SECTOR DE LA INDUSTRIA SIDEROMETALÚRGICA DE LA PROVINCIA DE TERUEL PARA EL AÑO 2011 el sueldo base de un Técnico Titulado de Grado Medio (Ingeniero Técnico) es de 1582.75€ brutos. Esta cantidad multiplicada por 14 mensualidades hacen un sueldo bruto anual de 22158.5€.

El sueldo bruto diario de un Ingeniero Técnico en la Provincia de Teruel al día es de: 92.33€

Por lo tanto el coste total del proyecto sería de:

$$nº\ días\ x\ sueldo/día = 177 x 92.33 = 16342.41€$$

A estos costes de Mano de obra habría que sumarles los costes del equipamiento necesario para la realización del proyecto. El equipamiento con sus costes derivados serían los siguientes:

Ordenador Portátil: 700 €
Teléfono móvil Android: 400 €

Esto haría un total de:

16342.41+700+400=17442.41€

A esto habría que añadirle costes indirectos como pueda ser el alquiler de un local, agua, luz, calefacción/aire acondicionado.

$177 \approx 6 meses$ → 600€ alquiler de local →3600€
Costes indirectos aproximados → 1800€

Coste total= 17442.41+3600+1800=22842.41€

# 5.- Fase de diseño

A continuación se presentarán los diferentes diagramas utilizados durante la fase de diseño en la realización de este proyecto.

## *5.1.- Base de datos*

### 5.1.1.- Diseño conceptual

En la aplicación se utiliza una base de datos que almacena los datos de la tarjeta SIM y las del dispositivo, así como los datos de los números de teléfono y mail de contacto. Para ello, se utiliza una sencilla base de datos, compuesta por dos tablas, tal y como se muestra en el siguiente diseño conceptual:

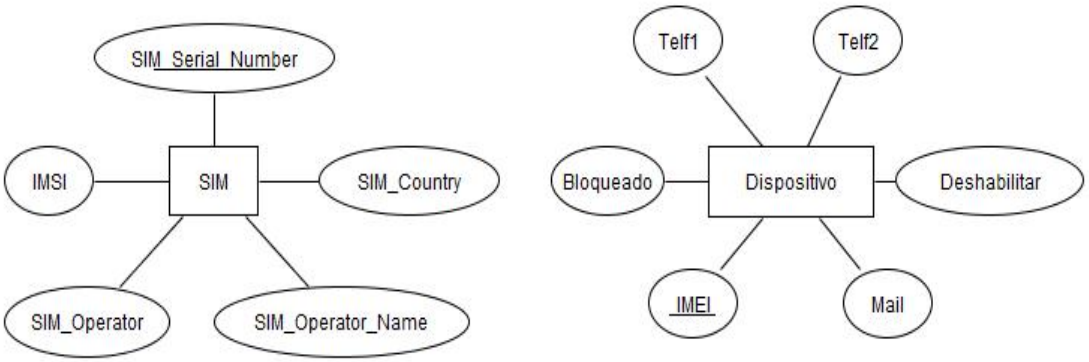



**Figura 31 Diseño conceptual de la base de datos utilizada en la aplicación.**

Dentro de la tabla SIM, se pueden encontrar los siguientes atributos:

- **SIM_Serial_Number:** numero único e inequívoco que representa e identifica a cada tarjeta SIM.
- **SIM_Country:** código que indica el país de procedencia de la tarjeta SIM.
- **SIM_Operator:** código numérico que indica la compañía a la que pertenece la tarjeta SIM.
- **SIM_Operator_Name:** compañía a la que pertenece la tarjeta SIM.
- **IMSI:** acrónimo de *International Mobile Subscriber Identity* (Identidad Internacional del Abonado a un Móvil) código que identifica de forma única para cada dispositivo de telefonía móvil, integrado en la tarjeta SIM, que permite su identificación a través de las redes GSM y UMTS.

Dentro de la tabla Dispositivo, se pueden encontrar los siguientes atributos:

- **Telf1:** representará al primer número de teléfono que el usuario inserta y al cual habrá que avisar en caso de que otra tarjeta SIM se introduzca en el dispositivo.
- **Telf2:** representará al segundo número de teléfono que el usuario inserta y al cual habrá que avisar en caso de que otra tarjeta SIM se introduzca en el dispositivo.

- **Deshabilitar:** opción que permitirá habilitar y deshabilitar la aplicación. En caso de estar esta opción activada, se deshabilita el aviso a los números de teléfono y al mail en caso de que otra tarjeta SIM sea introducida.
- **Mail:** representará a la dirección de correo electrónico que el usuario inserta y al cual habrá que avisar en caso de que otra tarjeta SIM se introduzca en el dispositivo.
- **IMEI:** número único e inequívoco que representa e identifica a cada dispositivo móvil. Puede ser utilizado para que la compañía telefónica lo bloquee.
- **Bloqueado:** si el dispositivo es bloqueado remotamente a través de la aplicación, esta opción se activará, bloqueando el dispositivo hasta que la tarjeta SIM válida sea insertada.

### 5.1.2.- Diseño lógico

El diseño lógico de la base de datos utilizada es el siguiente:

**SIM**(sim_country TEXT, sim_operator TEXT, sim_operator_name TEXT, sim_serial TEXT, IMSI TEXT)

CP:{sim_serial}

**Dispositivo**(imei TEXT, mail TEXT, telf1 INTEGER, telf2 INTEGER, deshabilitar INTEGER, bloqueado INTEGER)

CP:{imei}

### 5.1.3.- Diseño físico

Para la realización de la aplicación es necesario almacenar una serie de datos. La cantidad de datos a almacenar no es muy grande, y se podría haber optado por otros sistemas como por ejemplo los ficheros. Sin embargo un fichero podría ser fácilmente detectable, y por lo tanto, modificable. Utilizando una base de datos se oculta todo el proceso, provocando una detección de la aplicación más compleja. Esta es la justificación de la utilización de la una base de datos.

Para implementar la base de datos en la aplicación se ha utilizado SQLite, que es un sistema de gestión de bases de datos relacional, que implementa la mayor parte del estándar SQL-92.

SQLite es una tecnología que está integrada dentro de Android.

## *5.2.- Diagrama de casos de uso*

A continuación se expondrá el diagrama de casos de uso de la aplicación.

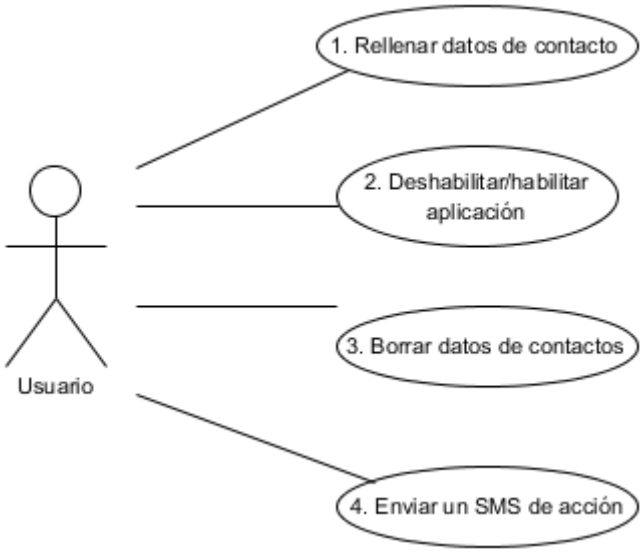



**Figura 32 Diagrama de casos de uso de la aplicación.**

El usuario podrá realizar los siguientes casos:

**CU 1. Rellenar datos de contacto:** el usuario deberá rellenar los datos de contacto que facilitará la localización del dispositivo.

**CU 2. Deshabilitar/habilitar aplicación:** deshabilitando la aplicación, se deshabilitan los avisos cuando otra tarjeta SIM es introducida.

**CU 3. Borrar datos de contactos:** el usuario podrá borrar los datos de contacto de la aplicación, ya sea para borrarlos, modificarlos, o porque desea vender el dispositivo.

**CU 4. Enviar un SMS de acción:** el usuario podrá enviar un SMS de acción para obtener los datos que estime oportuno.

## *5.3.- Diagrama de clases*

A continuación, se presenta el diagrama de clases de la aplicación realizada. En la primera imagen se puede observar las relaciones de las clases utilizadas. Habría que destacar que el diseño se ha realizado pensando en que la aplicación ha utilizado el paradigma de orientación a eventos, lo cual dificulta mucho la existencia de relaciones entre clases.

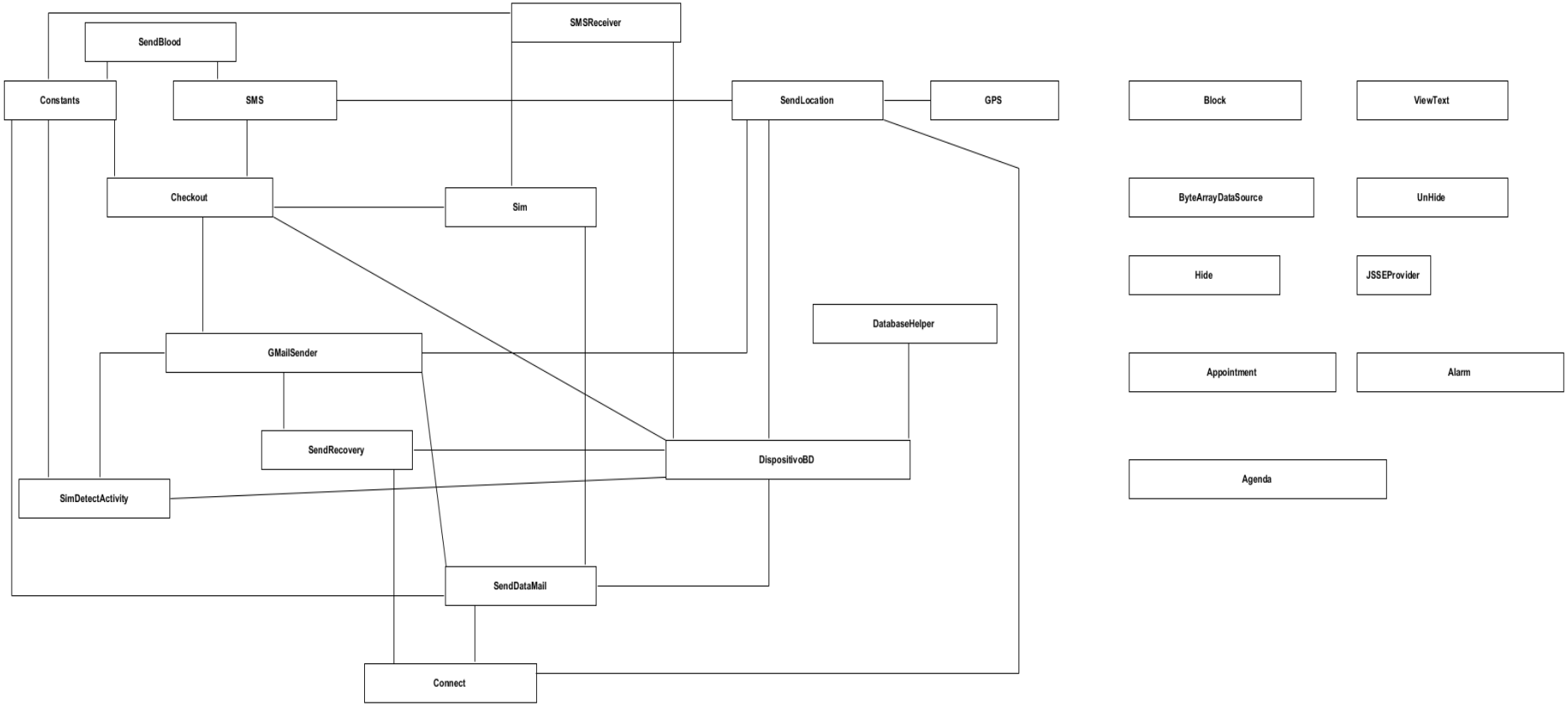
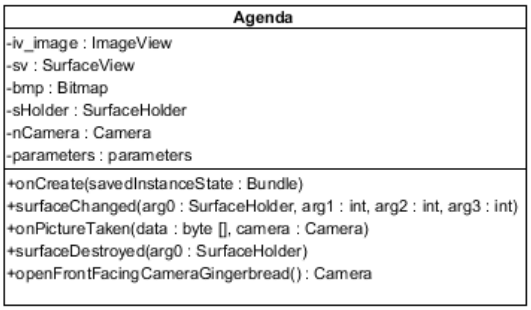



**Figura 33 Diagrama de clases de la aplicación.**

### 5.3.1.- Clase Agenda

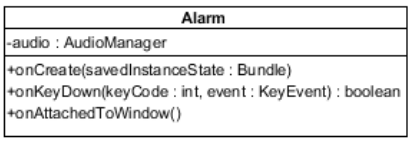



Esta clase contiene un nombre que podría parecer el de una aplicación que gestiona citas, a forma de agenda, pero nada más lejos de la realidad. Esta clase será la encargada de realizar la fotografía mediante la cámara frontal. Se le ha dado este nombre para apoyar al engaño en la interfaz. La fotografía será almacenada en memoria secundaria, para enviarla posteriormente mediante e-mail a la cuenta de correo electrónico de confianza insertada por el usuario.

El método onCreate() se ejecutará cuando se inicie el hilo de ejecución, creando una superficie que será la que se muestre por pantalla. Al detectar esta nueva superficie se ejecutará el método surfaceChanged(), la cual primeramente seleccionará la cámara frontal a través del método openFrontFacingCameraGingerbread() y posteriormente se realizará la fotografía. Este es un proceso totalmente instantáneo. Cuando la fotografía se haya realizado, se capturará el evento, mandando ejecutar el método onPictureTaken() el cual almacenará en almacenamiento secundario.

Esta clase ha sido una de las más problemáticas en la realización de este TFC, ya que la cantidad y diversidad de cámaras fotográficas que hay hoy en día dificulta mucho la utilización de este gadget. Hay que tener en cuenta que la API que facilita Android no permite la realización de fotografías sin que se muestre por pantalla lo que la cámara fotográfica está captando, lo cual ha dificultado mucho esta tarea, además de mostrar una gran incompatibilidad.

### 5.3.2.- Clase Alarm



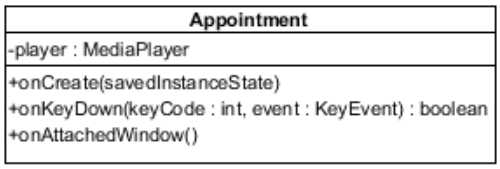

Cuando se reciba un SMS de acción con el código “alarm” para intentar localizar el móvil en un lugar cercano, se lanzará un Activity que ejecutará esta clase. Esta clase reproducirá un sonido a modo de alarma para permitir su localización. Esto lo hace gracias a un objeto de la clase AudioManager.

Cuando se ejecute el hilo, y por lo tanto el método onCreate(), se realizará una activación de captura de las teclas pulsadas del dispositivo. Posteriormente, obtiene el

control del volumen del dispositivo, poniéndolo al máximo volumen que admita, para facilitar que se pueda escuchar la alarma. La alarma sonará aunque el teléfono se encuentre en “modo silencioso”.

Con el método onKeyDown() se capturarán los eventos producidos al pulsar teclas del teléfono móvil. Se ha modificado el comportamiento standard de las teclas, para dificultar que no se pueda apagar la alarma. Hay dos teclas que Android no permite bloquear por seguridad, la tecla HOME, y la de apagar el terminal. La primera se ha conseguido bloquear mediante la utilización del método onAttachedToWindows().

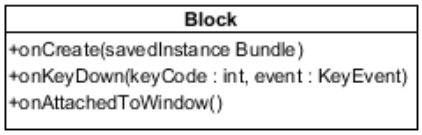
### 5.3.3.- Clase Appointment




Esta clase será la encargada de actuar cuando se reciba un SMS de acción con el código de acción “camera”. Cuando este código llegue, se lanzará un activity que ejecutará la clase Appointment, ejecutándose el método onCreate(). Este método muestra por pantalla un icono de una agenda mostrando un mensaje por pantalla, imitando una notificación de la agenda. Además, se notifica de la supuesta notificación/cita mediante avisos sonoros. Al igual que en la clase anterior, se capturan las teclas pulsadas, impidiendo que se pueda anular la notificación con el método onKeyDown(), y con el método onAttachedWindows() se bloquea el uso de la tecla HOME.

En la interfaz se muestra un botón para cancelar la notificación. Cuando se pulse, se lanzará un activity de la clase Agenda, que será el encargado de hacer la foto.

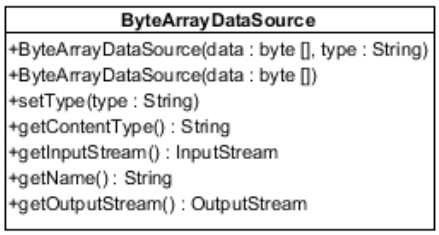
### 5.3.4.- Clase Block




Esta clase será la encargada de actuar cuando se reciba un SMS de acción con el código de acción “block”, que bloqueará el terminal impidiendo su utilización. Cuando se lance el activity al llegar el SMS el método onCreate() lanza una interfaz de pantalla en negro, y al igual que en las clases anteriores, bloquea el teclado, impidiendo que se pueda utilizar el terminal. Comentar que este bloqueo es permanente, aunque se reinicie el móvil, éste quedará bloqueado, impidiendo su utilización.

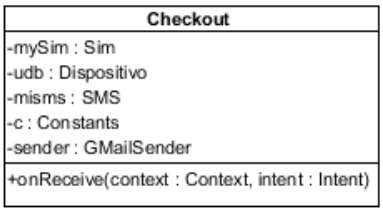
El terminal se desbloqueará cuando se inserte la tarjeta SIM del usuario del terminal.

### 5.3.5.- Clase ByteArrayDataSource




Esta clase se utiliza como apoyo a la hora de enviar correos electrónicos. Cuando ya se ha construido todo el mail, a la hora de enviarlo es necesario convertirlo a unos y ceros. Este es el cometido de esta clase.

### 5.3.6.- Clase Checkout



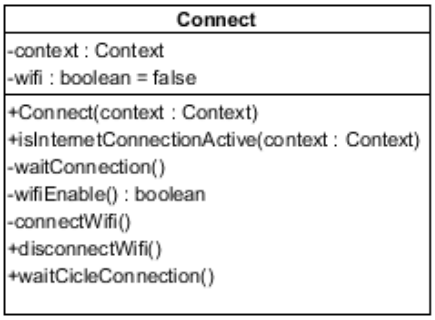

Esta clase será la encargada de realizar la comprobación inicial cuando se arranca el dispositivo. Esta clase captura cuándo se ha iniciado el sistema operativo, y comprueba si la tarjeta SIM es la correcta. Si es la correcta, se finaliza la aplicación para evitar que la aplicación consuma recursos.

En caso de no ser la misma tarjeta SIM envía una notificación a los contactos de confianza para informar al usuario. En caso de estar bloqueada se lanza un activity de la clase Block, para bloquear el terminal móvil.

Siempre que la tarjeta SIM introducida no sea válida se ocultará la aplicación de la lista de aplicaciones, para evitar que la aplicación pueda ser vista por el usuario sustractor.

En caso de iniciarse el sistema operativo y no haber rellenado los datos de los contactos de confianza se ejecutará un nuevo activity lanzando la clase SimDetectActivity, para que el usuario no se olvide de rellenar los datos.

### 5.3.7.- Clase Connect




Clase que gestionará las conexiones a Internet. Cuando se tenga que enviar un dato a través de Internet se comprobará siempre, que existe conexión a Internet disponible. Como no se pretende que esta clase actúe como un activity no incluye el método onCreate(), aunque al acceder a datos es necesario acceder a ciertas estructuras de los activitys, es por eso que en el constructor es necesario un parámetro del tipo Context, que es, al fin y al cabo, el context de un activity, con el cual se accederá a los recursos.

El método isinternetConnectionActive() comprueba si hay una conexión a Internet activa, ya sea a través de WiFi, 3G, GPRS…

El método waitConnection() para la comprobación durante 15 segundos. La utilidad de esto se verá en el método waitCicleConnection().

El método wifiEnable() comprobará el estado del WiFi, comprobando si está habilitado o no.

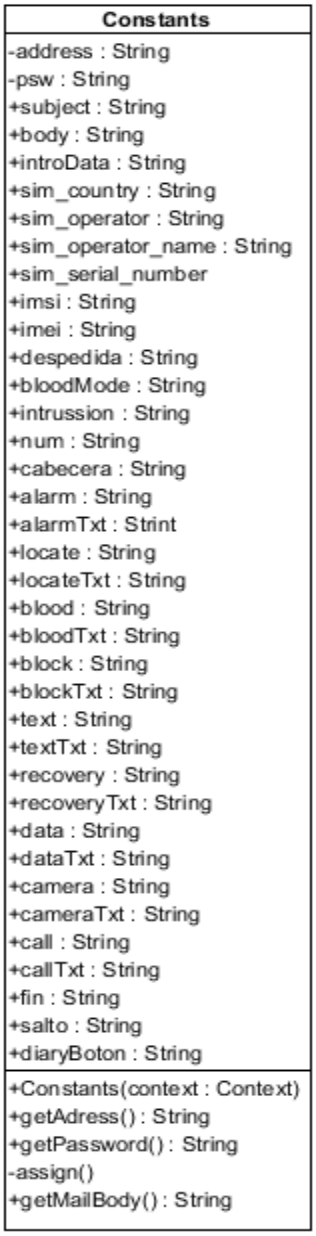
El método connectWifi() es un método que habilita el WiFi del dispositivo. Comentar que esto es algo que no está permitido por Android. Para realizar esta función se ha utilizado un agujero de seguridad descubierto.

El método disconnectWifi() deshabilita el WiFi del dispositivo. Al igual que en el anterior caso, no está permitido por Android. Para realizar esta función se ha utilizado un agujero de seguridad descubierto.

El método waitCicleConnection() se utiliza, para que en caso no existir ninguna conexión a Internet libre, haga una espera cíclica en busca de conexión. El primer paso es forzar la habilitación del WiFi, y después comprueba si existe conexión. En caso de no encontrar ninguna conexión, utiliza de forma cíclica una conexión mediante el método waitConnection(), es decir, comprueba si tiene conexión, duerme 15 segundos y vuelve a comprobar, así hasta que encuentre alguna conexión.

Esto se realiza para facilitar el caso de que se pueda encontrar una red abierta o alguna almacenada en el terminal y que se conecte, facilitando la labor de enviar datos a través de la red.

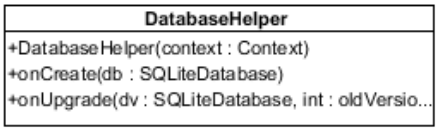
### 5.3.8.- Clase Constants




La clase Constants se ha utilizado para facilitar el uso a la hora de formar los mensajes que se van a enviar, ya sean mediante SMS o mediante correo electrónico.

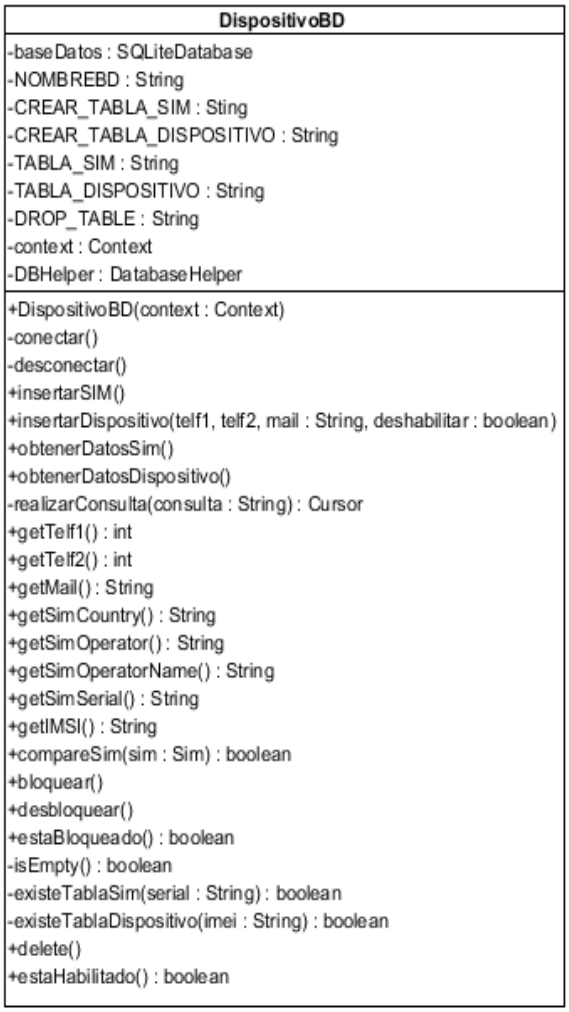
Por decisión de diseño se ha decidido realizar la aplicación en dos idiomas, inglés y español. No siendo imprescindible esta clase, facilita, encapsula y hace más higiénico y visible la formación de estos mensajes en ambos idiomas.

### 5.3.9.- Clase DatabaseHelper




La clase DatabaseHelper es una clase de tipo API, que hay que sobrescribir y cuya función, es facilitar el uso de las bases de datos con SQLite en Android.

### 5.3.10.- Clase DispositivoBD

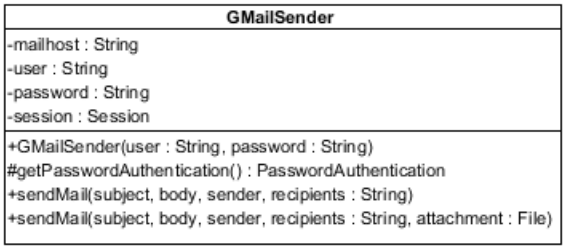



Esta clase será la encargada de trabajar con la base de datos utilizada en la aplicación. Se trabaja con SQLite.

En esto caso, al no interesar que se ejecute como un activity, no tiene el método onCreate(), y al necesitar recursos del sistema es necesario pasarle el contexto de otro activity. El resto de parámetros son Constantes que contienen las tablas de la base de datos, o las sentencias SQL que facilitan la inserción, actualización o eliminación de datos en la base de datos.

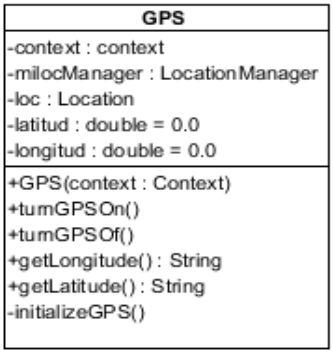
El método insertarSIM() introducirá en la base de datos, los datos a almacenar de la tarjeta SIM. insertarDispositivo() almacena los del dispositivo. El método realizarConsulta() realiza una consulta en SQL a la base de datos. El resto de métodos son métodos GET/SET.

### 5.3.11.- Clase GMailSender




Esta clase será la encargada de la creación de correos electrónicos para enviar datos al usuario. Con el constructor se crea un objeto de tipo GMailSender, y para enviarlo se utiliza el método sendMail(). Existen dos métodos con el mismo nombre, la diferencia está en si se quiere enviar un fichero adjunto.

### 5.3.12.- Clase GPS



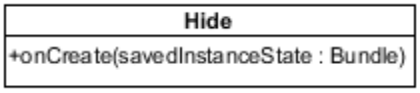

La clase GPS será la encargada de utilizar el GPS del dispositivo móvil, para facilitar su localización. Al igual que en otras clases comentadas, se pasa el context de otro activity para acceder a recursos del sistema en el constructor.

El método turnGPSOn(), habilita el GPS del terminal móvil. Comentar que esto no está permitido por Android. Para realizar esta función se ha utilizado un agujero de seguridad descubierto.

El método turnGPSOf(), deshabilita el GPS del terminal móvil. Comentar que esto no está permitido por Android. Para realizar esta función se ha utilizado un agujero de seguridad descubierto.

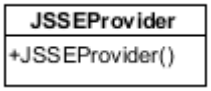
El método initializeGPS() comienza la búsqueda de satélites para obtener las coordenadas en las que se encuentra el dispositivo móvil.

### 5.3.13.- Clase Hide



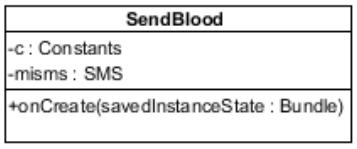

Cuando se detecte que la tarjeta SIM no es válida, aparte de notificar al usuario, se lanza el activity Hide. Cuando se lanza, se ejecuta el método onCreate() que lo oculta la aplicación, eliminándola de la lista de aplicaciones. La aplicación se volverá a mostrar cuando se inserte la tarjeta SIM correcta.

### 5.3.14.- Clase JSSEProvider



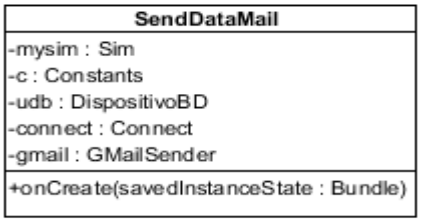

Esta clase facilita el envío de los correos electrónicos, realizando la conexión con los servidores que los gestionan.

### 5.3.15.- Clase SendBlood




Esta clase será la encargada de actuar cuando se reciba un SMS de acción con el código de acción “blood”. Cuando se lanza el activity, el método onCreate() obtiene los números de los contactos de la tarjeta SIM introducida, y se les envía un SMS informando de que el usuario sustractor está utilizando un teléfono móvil robado.

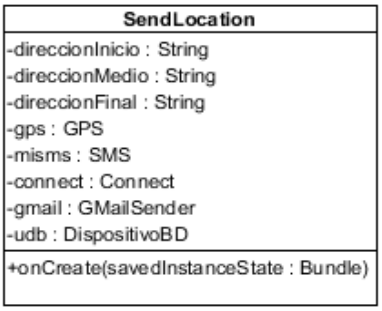
### 5.3.16.- Clase SendDataMail




Esta clase es la encargada de enviar los datos de la tarjeta SIM mediante correo electrónico. Para obtener los datos, utiliza un objeto de la clase Sim. El mensaje que se envía se construye utilizando la clase Constants ya comentada con anterioridad. Con la clase DispositivoBD se obtendría la dirección de correo electrónico a la cual enviar el e-mail. La clase Connect es necesaria para comprobar la conexión a Internet, y con la clase GMailSender se crea el correo electrónico y se envía.

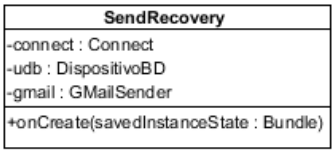
Cuando se lance el activity, el método onCreate() obtiene los datos de la tarjeta SIM y del dispositivo, se crea el mensaje que se va a enviar, se comprueba si se tiene conexión a Internet; en caso negativo hace la comprobación cíclica ya comentada, y cuando se tenga Internet se envía el correo electrónico.

### 5.3.17.- Clase SendLocation



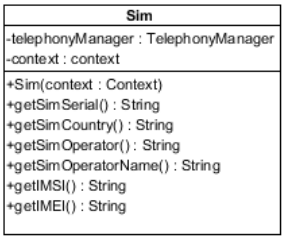

La clase SendLocation es la encargada de localizar el dispositivo obteniendo las coordenadas mediante el GPS del terminal móvil para posteriormente enviarlas. Para ello se necesita: (i) un objeto del tipo GPS, para habilitar/deshabilitar el GPS, y obtener las coordenadas, (ii) un objeto del tipo SMS para enviar los datos mediante SMS, y (iii) objetos del tipo DispositivoBD, Connect y GMailSender, para obtener la dirección de correo electrónico a enviar los datos, comprobando la conexión a Internet y enviarlos.

### 5.3.18.- Clase SendRecovery



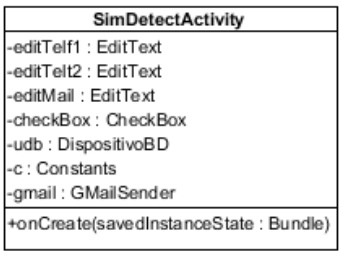

Esta clase será la encargada de enviar los datos de los contactos de la agenda del dispositivo móvil cuando se reciba un SMS de acción con el código “recovery”. Cuando el activity se ejecute, el método onCreate() obtiene los datos de los contactos del dispositivo, se comprueba la conexión a Internet y se envía utilizando un objeto del tipo GMailSender.

### 5.3.19.- Clase Sim



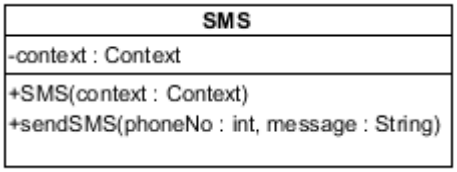

La clase Sim es la encargada de obtener los datos de la tarjeta SIM y del dispositivo. Al no interesar que la clase sea un activity, al igual que en casos anteriores, al acceder a recursos del dispositivo es necesario pasar el context de un activity.

### 5.3.20.- Clase SimDetectActivity



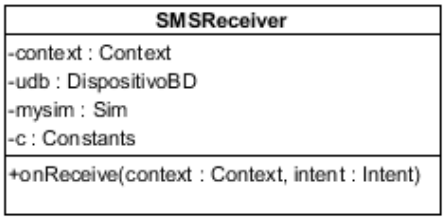

La clase SimDetectActivity es la encargada de mostrar la interfaz que carga el formulario para rellenar los datos de los contactos seguros. Comprueba que han sido rellenados, y los almacena en la base de datos. Por último, se envía un correo electrónico a la dirección introducida, informando de los códigos a enviar en los SMS de acción.

### 5.3.21.- Clase SMS




Esta clase es utilizada para enviar SMS. Para ello, es necesario eñ context para acceder al recurso del sistema, y a la hora de enviarlo, el texto y el número de teléfono del destinatario.

### 5.3.22.- Clase SMSReceiver

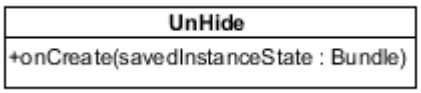



Esta clase es la que captura el evento cuando se recibe un SMS. Está diseñada de tal manera que captura el evento de nuevo SMS recibido antes que la aplicación del propio Sistema Operativo realice la notificación, obtiene el texto del mensaje y comprueba si es un SMS de acción.

En caso de no ser un mensaje de acción se le da el control del SMS al Sistema Operativo para que realice la notificación.

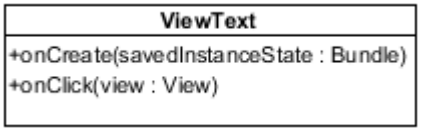
En caso de ser un SMS de acción, obtiene el código de acción para realizar la acción pertinente.

### 5.3.23.- Clase UnHide




Cuando la aplicación se haya escondido, debido a la introducción de una tarjeta SIM no válida, no será visible al usuario poseedor del teléfono. Cuando se vuelva a insertar la tarjeta SIM correcta, la aplicación se volverá a mostrar. Esta es la función de esta clase, que hará visible la aplicación en la lista de aplicación del dispositivo móvil.

### 5.3.24.- Clase ViewText




Cuando se reciba un SMS de acción con el código “text”, el usuario podrá escribir en el propio SMS un mensaje para que el poseedor del dispositivo lo lea. Esta clase sacará una interfaz por la pantalla mostrando el mensaje que el usuario haya introducido. Para ello cuando se lance el activity, cargará el método onCreate(), que mostrará la interfaz con el mensaje del usuario.

## *5.4.- Diagrama de actividades*

A continuación se expondrán y se comentarán los diferentes diagramas de actividad realizados durante la fase de diseño, y utilizados durante la fase de implementación.

### 5.4.1.- Comprobación inicial

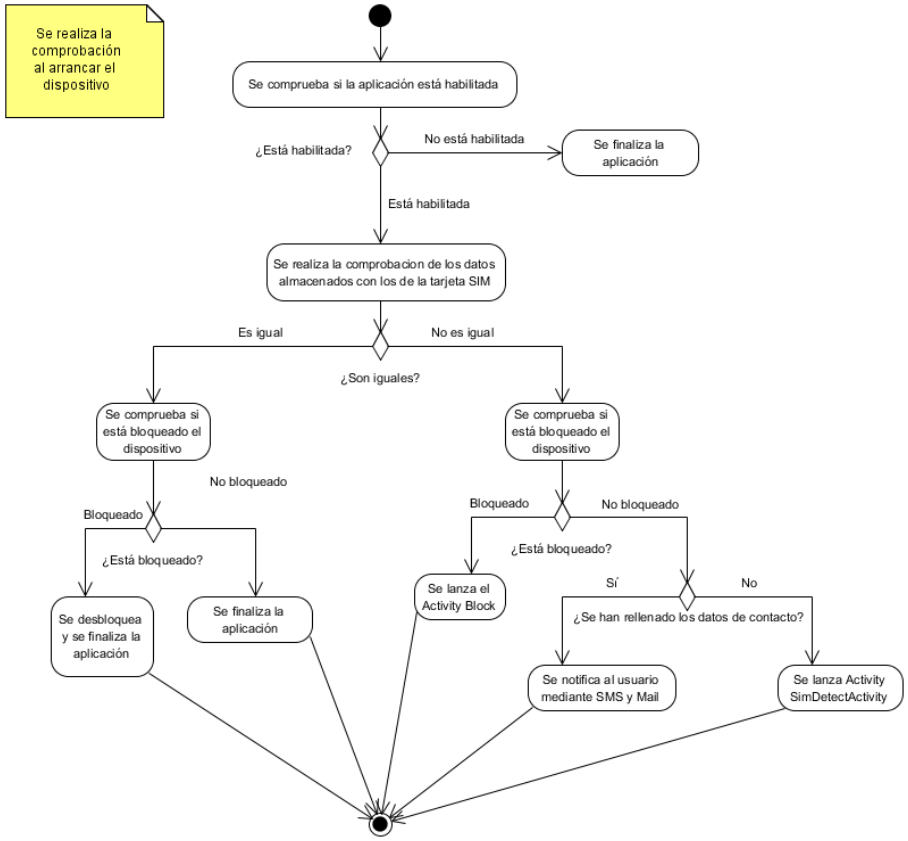



**Figura 34 Diagrama de actividad de la comprobación inicial.**

Lo primero que realizará la aplicación es comprobar si la aplicación está habilitada o deshabilitada. Si la aplicación no está habilitada finaliza, en caso contrario realiza la comprobación de la tarjeta SIM.

1. Si la SIM es la misma, se comprueba si se ha bloqueado la aplicación. En caso de estar bloqueada, se desbloquea. En ambos casos finaliza la aplicación.
2. En caso de no ser la SIM válida, se comprueba si se ha bloqueado, En caso de estarlo se lanza el Activity que bloquea el dispositivo móvil.

   En caso de no estar bloqueado se realiza la comprobación de si se han rellenado los datos de contacto. Si no se han rellenado, se lanza el Activity, mostrando la interfaz de los contactos, en caso contrario se supone que alguien ha introducido una tarjeta SIM no válida, y se avisa a los números de teléfono y mail de contacto.

### 5.4.2.- Llega un SMS de acción

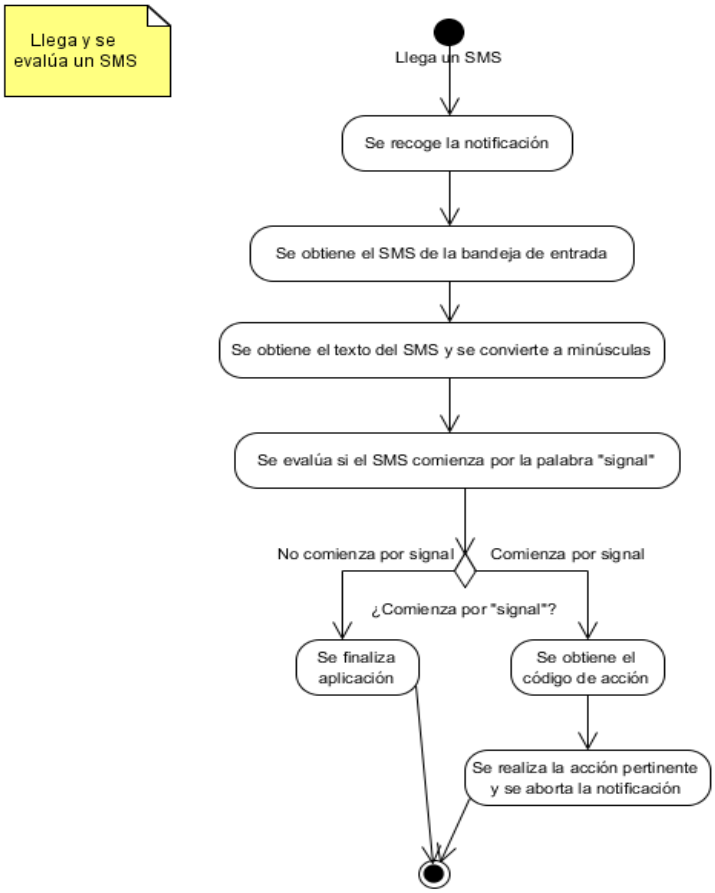



**Figura 35 Diagrama de actividad de la llegada de un SMS.**

Cuando se recibe un SMS, se carga una parte de la aplicación, capturando el evento y dándole más prioridad que al propio sistema operativo. Se obtiene el SMS y se evalúa el cuerpo del mensaje, buscando un código de acción.

Si el SMS no contiene la palabra ‘signal’ se finaliza la aplicación dando el control del SMS al sistema operativo y por lo tanto lanzando la notificación de nuevo mensaje recibido. Si el SMS contiene la palabra ‘signal’ se entiende que es un SMS de acción, y según su código se realiza una acción u otra (se verán a continuación); por último se aborta la notificación para que el dispositivo no notifique al usuario ni muestre el SMS.

### 5.4.3.- Llega un SMS de acción Alarm

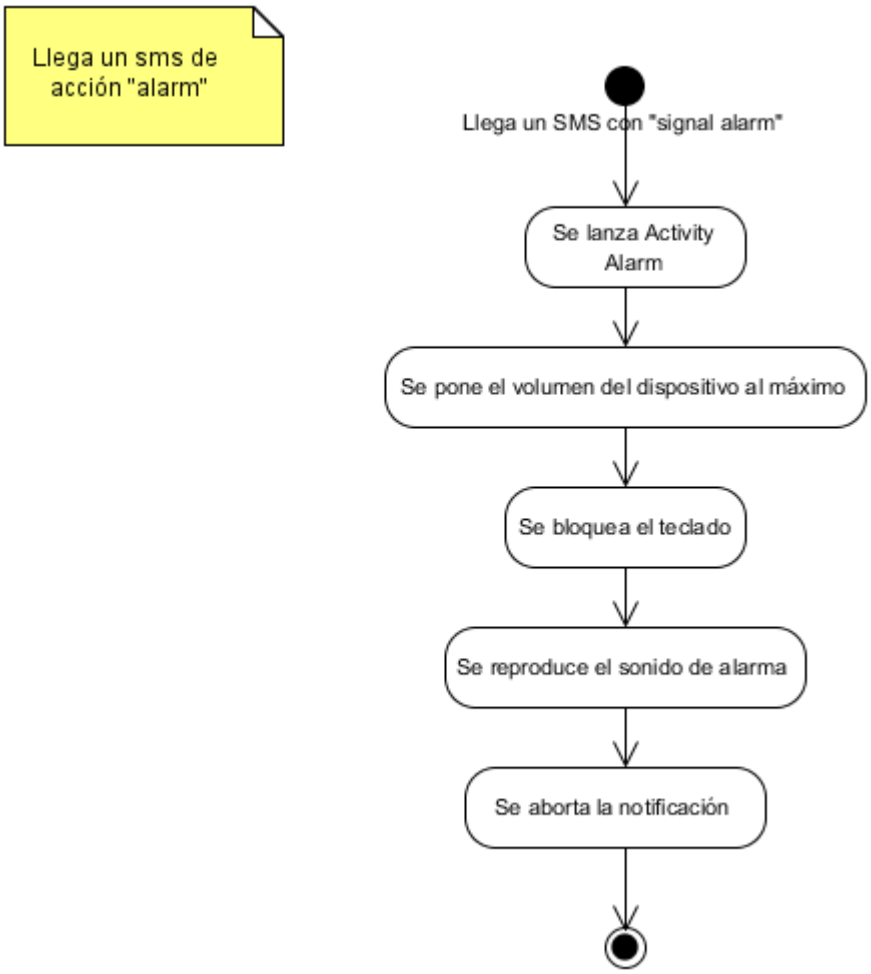



**Figura 36 Diagrama de actividad de la llegada de un SMS con código "signal alarm".**

Cuando llegue un SMS de acción con el código Alarm se lanza un nuevo Activity, que será el encargado de hacer sonar la alarma. Se pone el volumen del dispositivo al máximo, y se bloquea el teclado, para evitar que se pueda deshabilitar. Se reproduce el sonido y se aborta la notificación.

### 5.4.4.- Llega un SMS de acción Blood

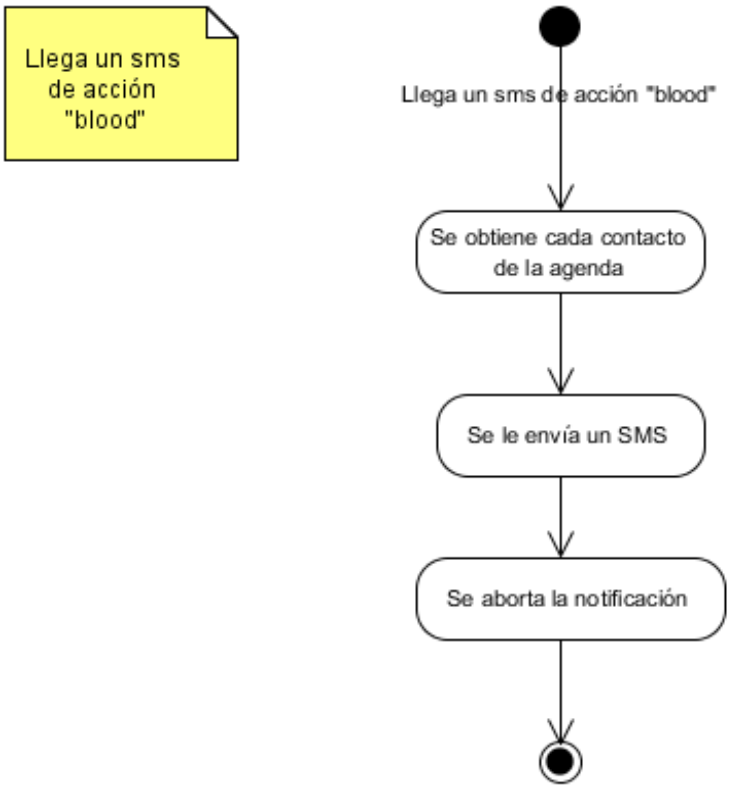



**Figura 37 Diagrama de actividad de la llegada de un SMS con código "signal blood".**

Se accede a la agenda de contactos del teléfono, se va accediendo a cada contacto enviándole un SMS a cada usuario, notificándole que el usuario (usuario sustractor) está utilizando un dispositivo robado.

Por último se aborta la notificación.

### 5.4.5.- Llega un SMS de acción Call

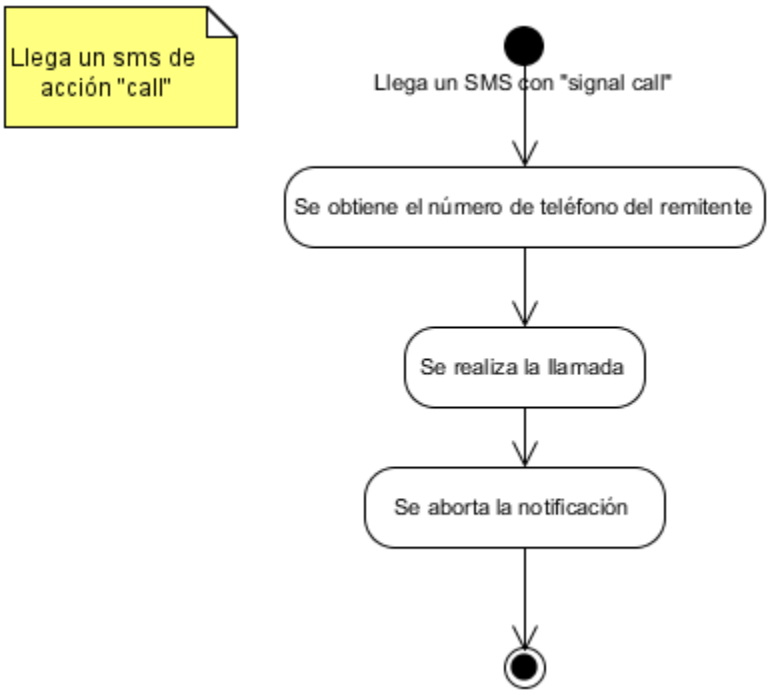



**Figura 38 Diagrama de actividad de la llegada de un SMS con código "signal call".**

Se obtiene el número de teléfono del emisor del SMS de acción. Se realiza la automáticamente la llamada telefónica al número de teléfono obtenido con anterioridad.

Por último se aborta la notificación.

### 5.4.6.- Llega un SMS de acción Camera

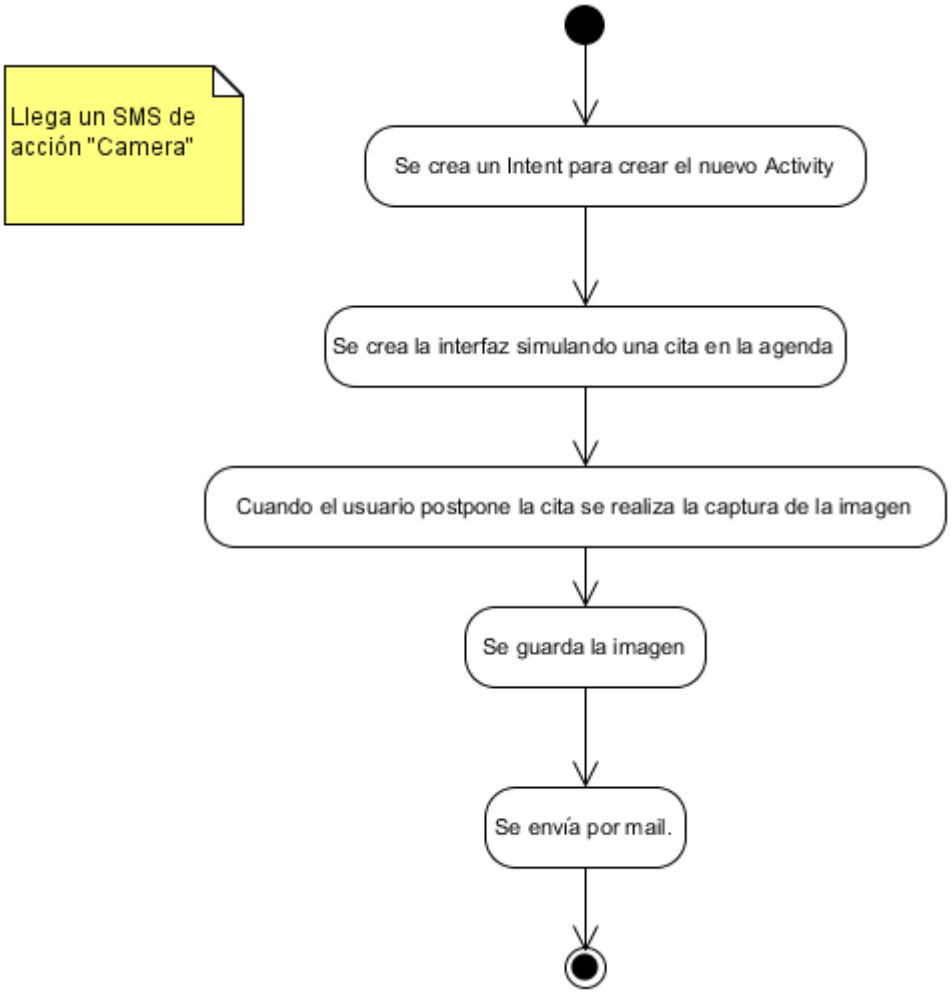



**Figura 39 Diagrama de actividad de la llegada de un SMS con código "signal camera".**

Se crea un intent para iniciar la nueva actividad. Esta nueva actividad creará una interfaz simulando una cita en la agenda. Cuando el usuario posponga la cita, la aplicación realizará una captura con la cámara frontal del dispositivo móvil. Esta imagen se almacena y se envía por correo electrónico utilizando la dirección de confianza.

### 5.4.7.- Llega un SMS de acción Data

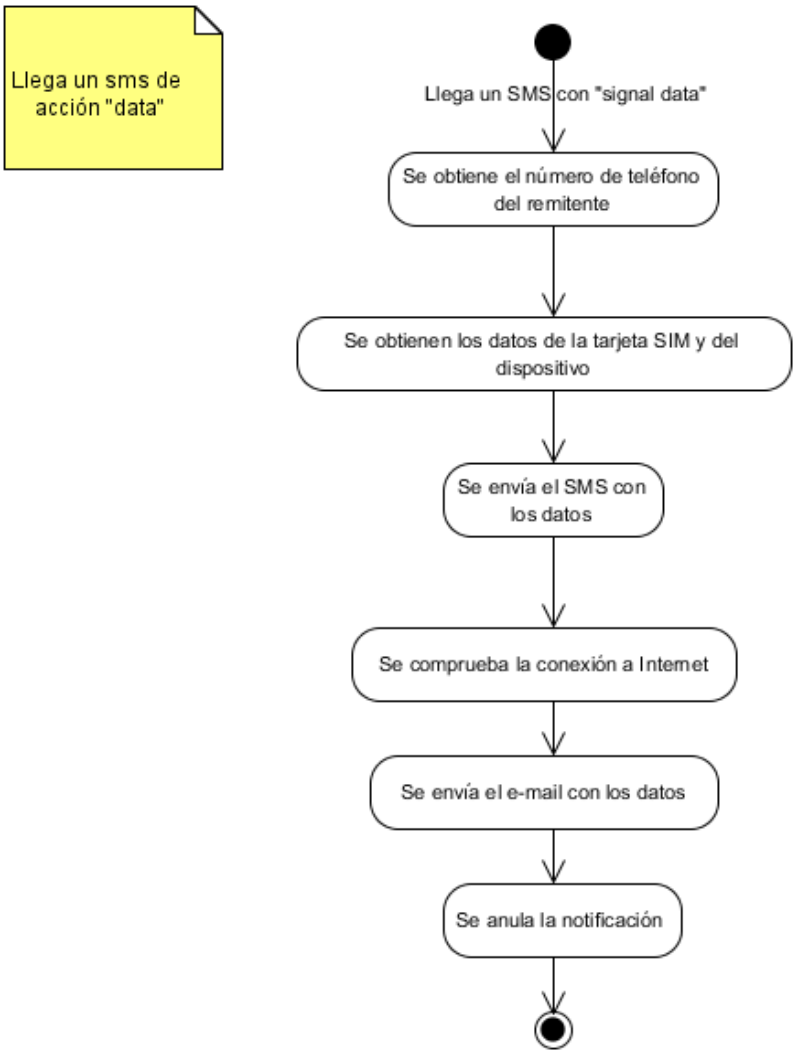



**Figura 40 Diagrama de actividad de la llegada de un SMS con código "signal data".**

Se obtiene el número de teléfono del emisor del SMS de acción. Se obtienen los datos de la tarjeta SIM y del dispositivo. Se envían los datos por SMS, y posteriormente se reenvían por correo electrónico.

Por último se aborta la notificación.

### 5.4.8.- Llega un SMS de acción Locate

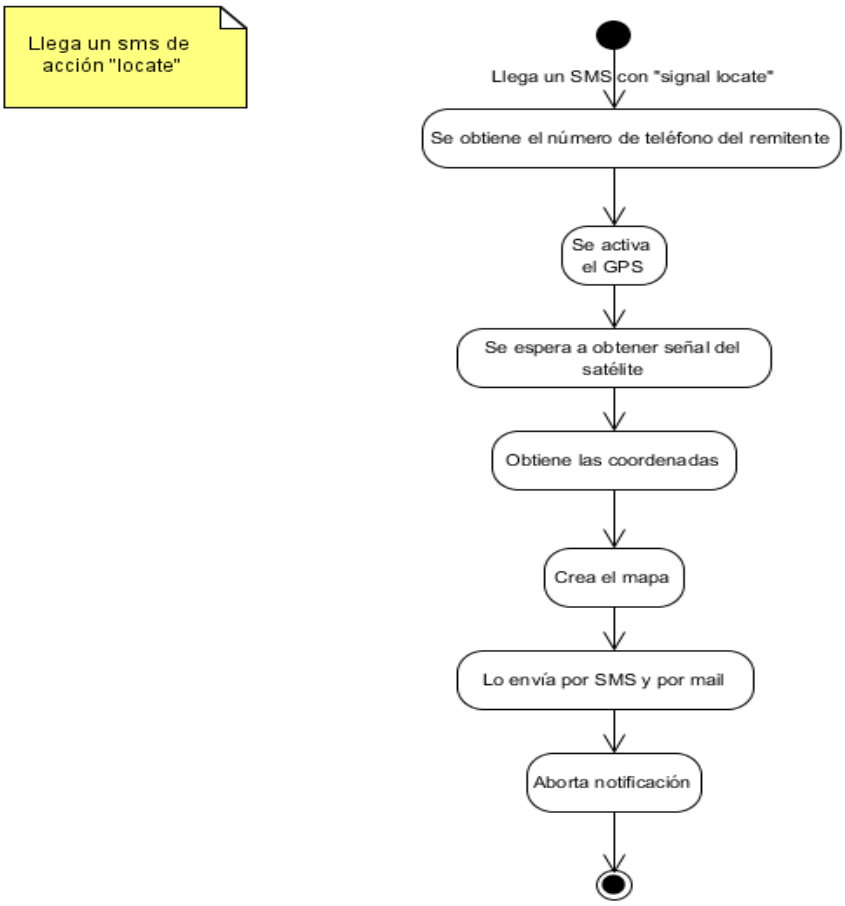



**Figura 41 Diagrama de actividad de la llegada de un SMS con código "signal locate".**

Cuando se llega un SMS de acción con el código 'locate' se obtiene el número de teléfono del remitente, se activa el GPS, se espera hasta obtener una señal de satélite válida. Posteriormente obtiene las coordenadas, crea el mapa y lo envía por SMS y por mail.

Por último se aborta la notificación.

### 5.4.9.- Llega un SMS de acción Recovery

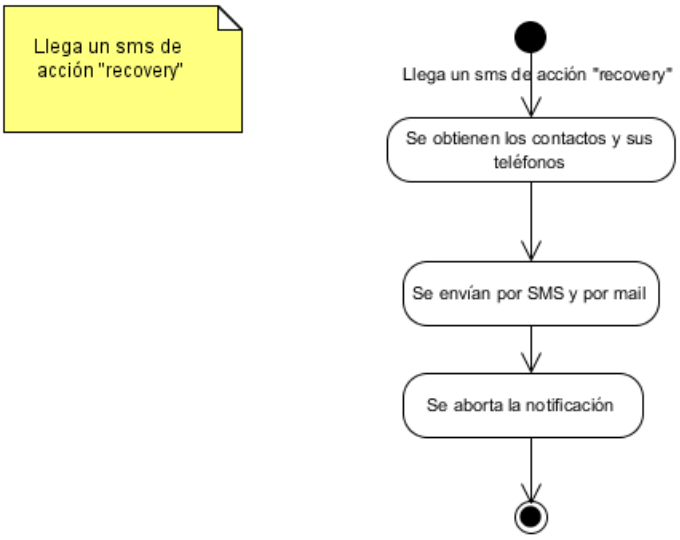



**Figura 42 Diagrama de actividad de la llegada de un SMS con código "signal recovery".**

Cuando llegue un SMS de acción con el código 'recovery' la aplicación accede a la agenda del dispositivo, accede a los contactos obteniendo tanto su nombre como su número de teléfono.

Posteriormente se envían tanto por SMS como por mail.

Por último se aborta la notificación.

### 5.4.10.- Llega un SMS de acción Text

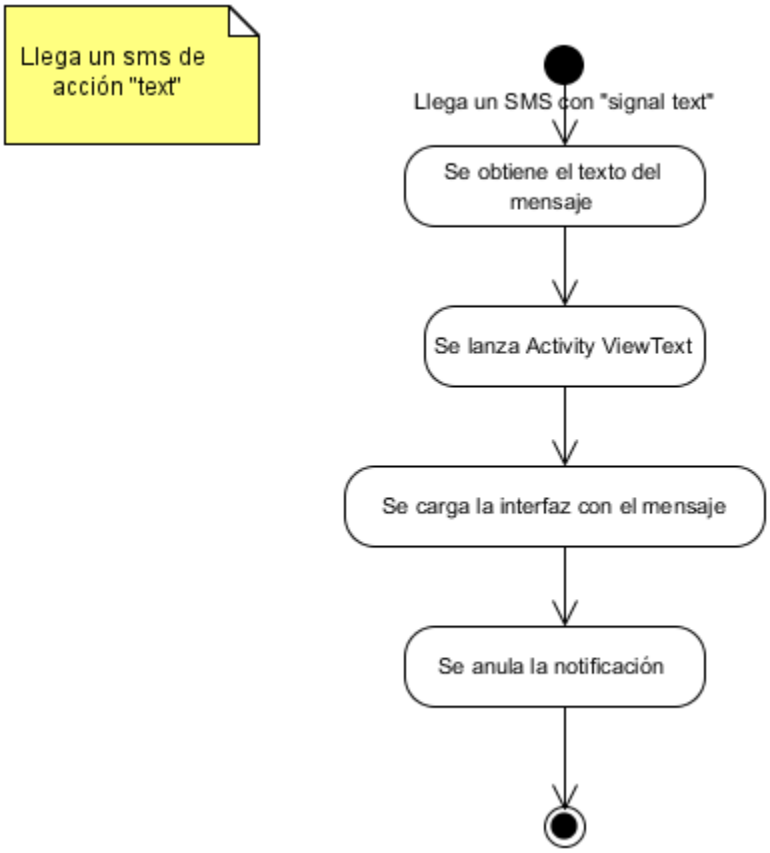



**Figura 43 Diagrama de actividad de la llegada de un SMS con código "signal text".**

Se obtiene el texto del mensaje a enseñar, del cuerpo del SMS recibido. Se lanza un nuevo Activity que carga la interfaz mostrando el mensaje. Para finalizar se anula la notificación.

## *5.5.- Diagrama de secuencia*

A continuación se expondrán los diagramas de secuencia, correspondientes a los diagramas de actividad mostrados y comentados con anterioridad.

### 5.5.1.- Comprobación inicial 1

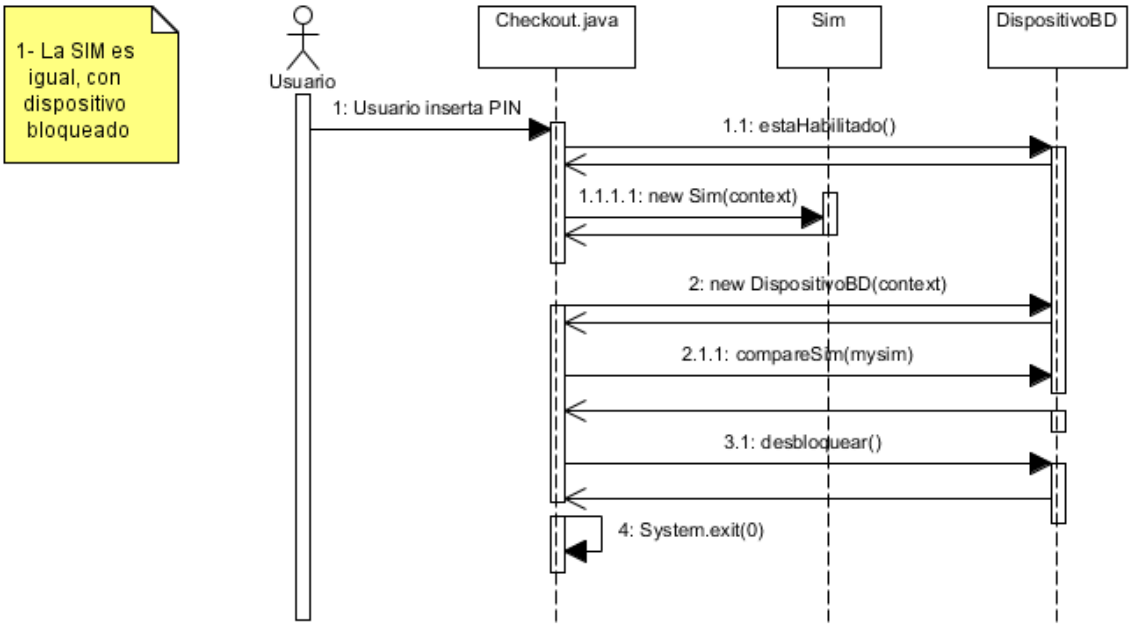



**Figura 44 Diagrama de secuencia de la comprobación inicial de la aplicación.**

Cuando el usuario inserta el código PIN, se lanza la aplicación para comprobar si la tarjeta SIM introducida en el dispositivo es válida. La primera comprobación que hace es comprobar si la aplicación está habilitada. Si no está habilitada la aplicación finaliza, en caso contrario hace la comprobación de la tarjeta SIM.

En este diagrama se asume que la tarjeta SIM es válida, pero con el dispositivo bloqueado. Así que obtiene los datos de la base de datos, los compara, se desbloquea y por último se finaliza la aplicación.

### 5.5.2.- Comprobación inicial 2

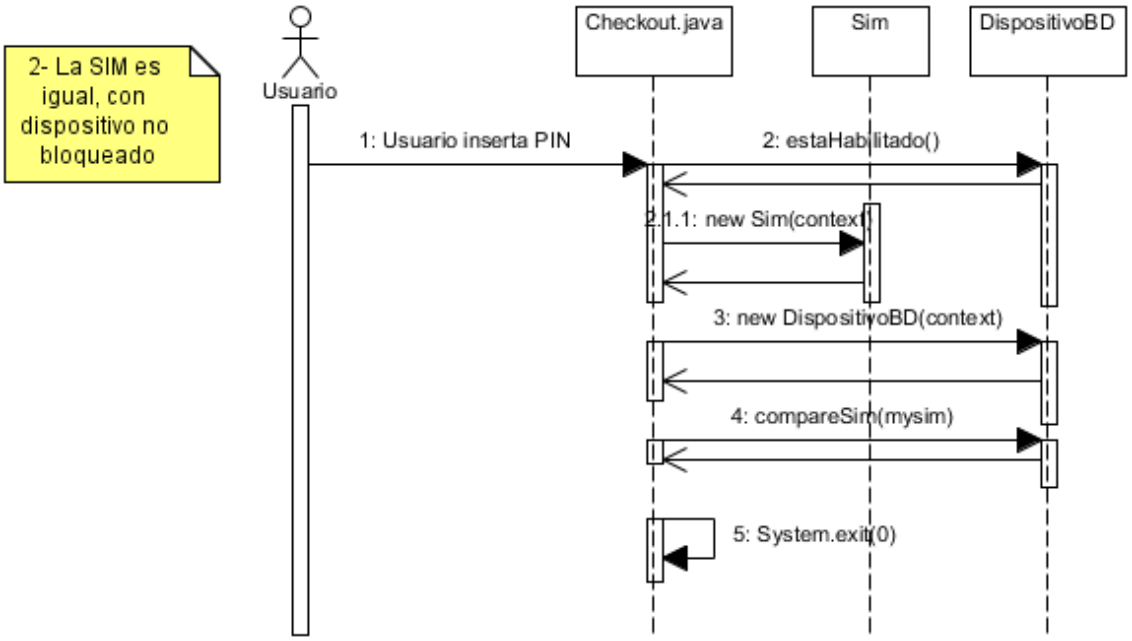



**Figura 45 Diagrama de secuencia de la comprobación inicial de la aplicación.**

En este diagrama se asume que la tarjeta SIM es válida con un dispositivo no bloqueado. La primera comprobación como en el caso anterior y también en los posteriores es comprobar si la aplicación está habilitada. Se obtienen los datos de la SIM y se comprueban con los almacenados, se comparan y al no estar bloqueada la aplicación finaliza.

### 5.5.3.- Comprobación inicial 3

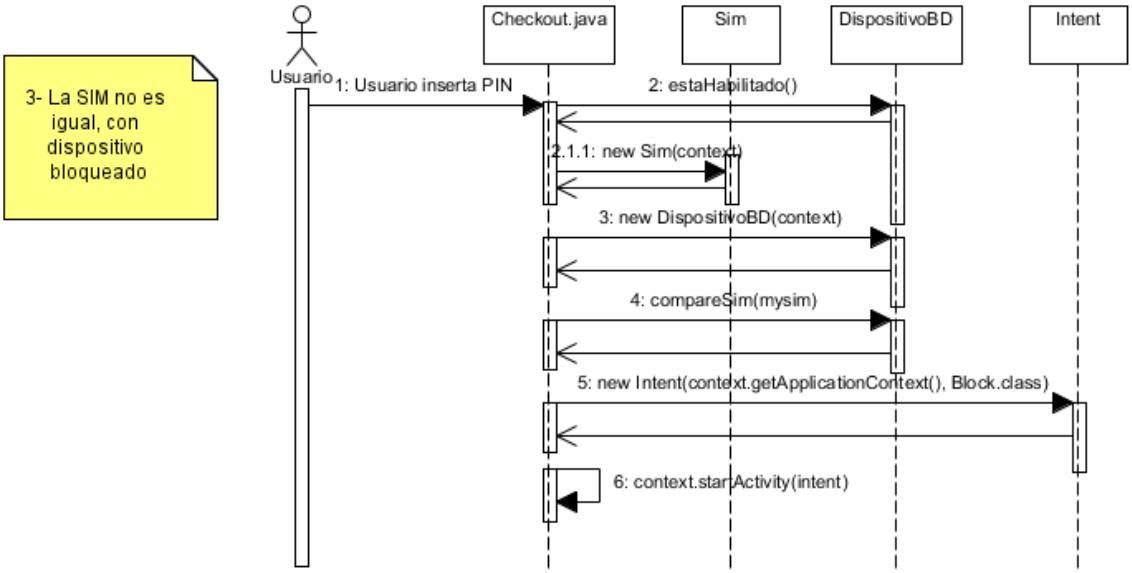



**Figura 46 Diagrama de secuencia de la comprobación inicial de la aplicación.**

En este diagrama se asume que la tarjeta SIM no es igual, teniendo el dispositivo bloqueado. Se comprueba que la aplicación está habilitada, se obtienen los datos de la tarjeta SIM y se comparan con los almacenados. En este caso la comprobación daría false, y al estar bloqueado se crearía un intent para lanzar un nuevo Activity que bloqueara el dispositivo. Este Activity bloquea todas las teclas del dispositivo, impidiendo que el usuario pueda realizar alguna acción.

### 5.5.4.- Comprobación inicial 4

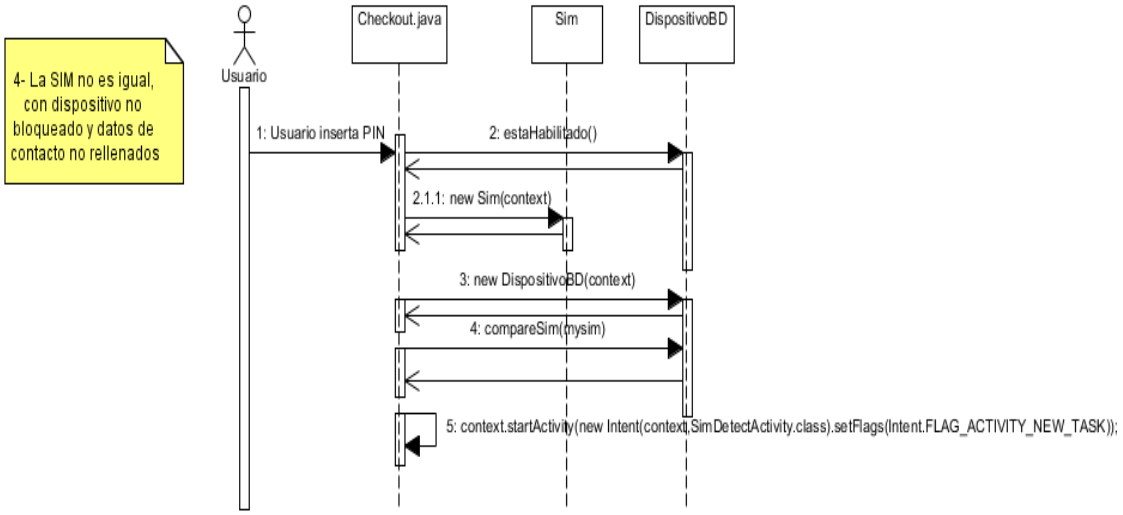



**Figura 47 Diagrama de secuencia de la comprobación inicial de la aplicación.**

Este diagrama asume que la tarjeta SIM no es válida, con un dispositivo no bloqueado y con los datos de contacto no válidos.

Comprueba que la aplicación está habilitada, obtiene los datos de las SIM y los comprueba con los almacenados. Se detecta que los datos no coinciden al no estar rellenados los datos, por lo que se crea una actividad, que mostrará la interfaz para que el usuario rellene los datos de contactos seguros. Esta interfaz se mostrará todas las veces que el usuario arranque el dispositivo móvil, hasta que los datos sean rellenados.

### 5.5.5.- Comprobación inicial 5

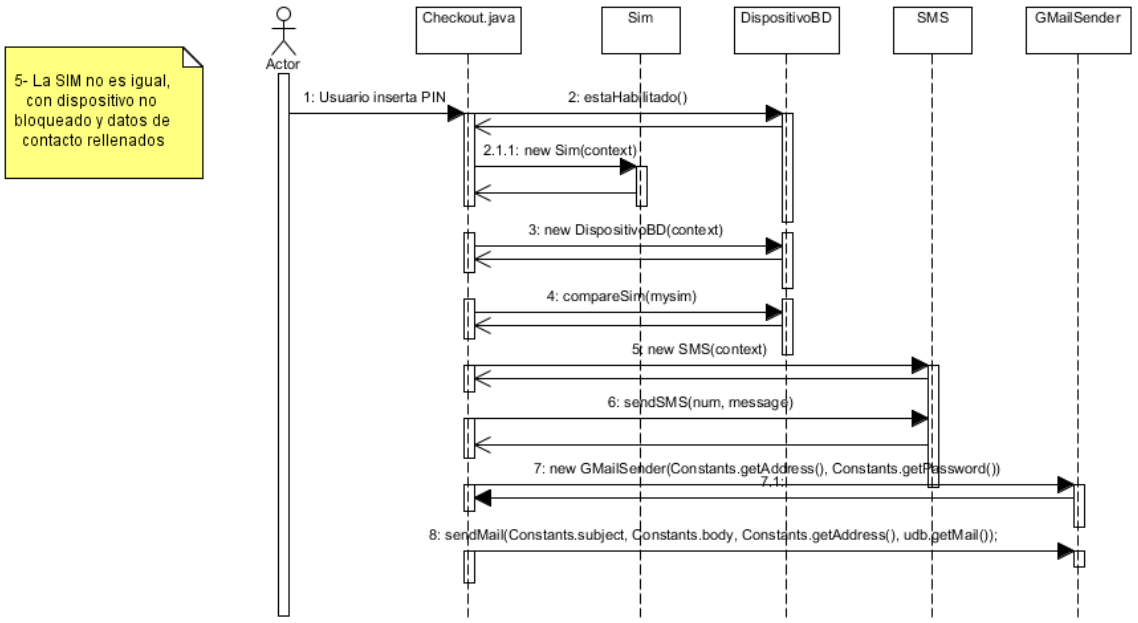



**Figura 48 Diagrama de secuencia de la comprobación inicial de la aplicación.**

En este diagrama se asume que la tarjeta SIM no es válida, con un dispositivo que no está bloqueado y los datos de contacto rellenados.

Se comprueba si la aplicación está habilitada; posteriormente se obtienen los datos de la tarjeta SIM y los compara con los almacenados. Comprueba que no son iguales, por lo que se supone que se ha insertado una tarjeta SIM no válida. Se avisa a los números de contacto mediante SMS por lo que el usuario podrá obtener el número de teléfono de la tarjeta SIM introducida. También se realiza una notificación mediante correo electrónico.

### 5.5.6.- Llega un SMS de acción

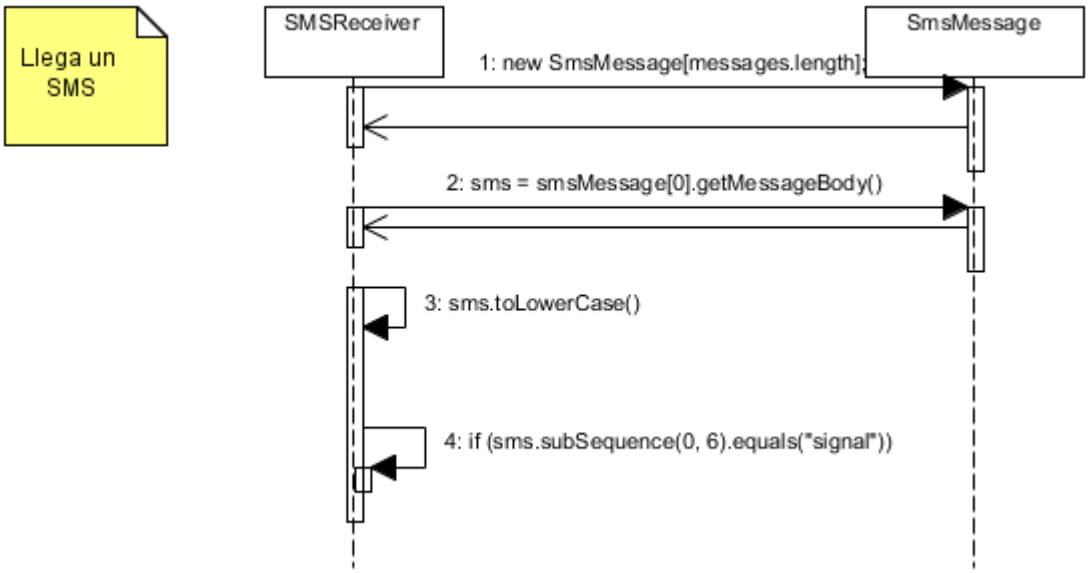



**Figura 49 Diagrama de secuencia de la llegada de un SMS de acción.**

Cuando llega un SMS de acción se crea un objeto de tipo SmsMessage, el cual obtiene el mensaje del buzón de entrada.

Se obtiene el cuerpo del mensaje recibido.

Se pasa todo el mensaje a minúsculas y se busca si el SMS comienza con el código 'signal'. Si lo contiene, evaluará el mensaje, realizando la acción pertinente. Si no contiene ningún código se acaba la aplicación dándole el control al sistema operativo.

### 5.5.7.- Llega un SMS de acción Alarm

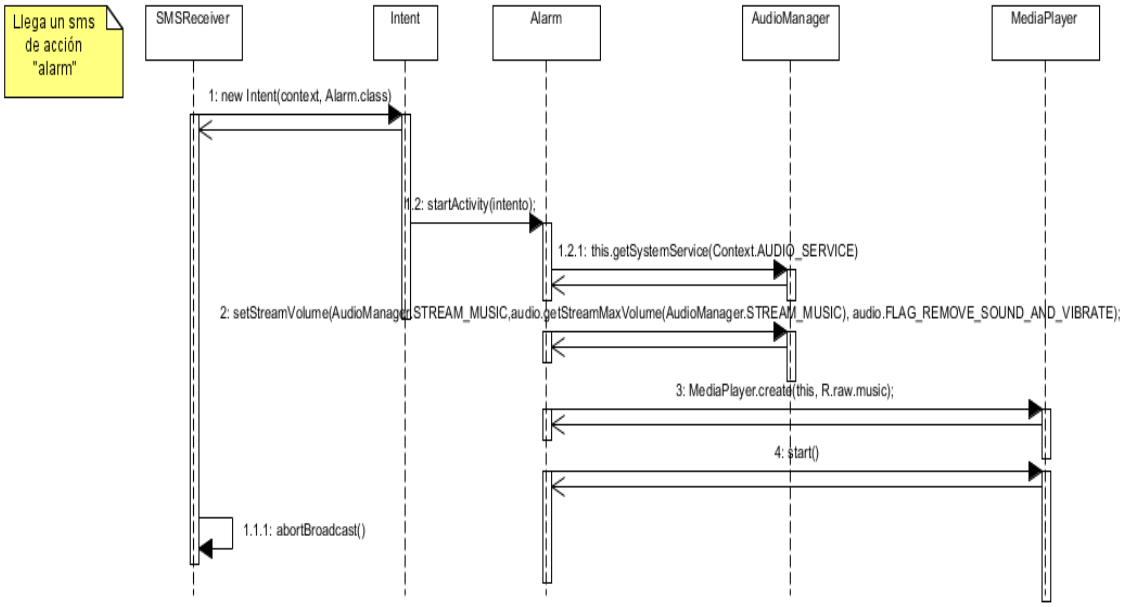



**Figura 50 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Alarm".**

Cuando llegue un SMS con el código Alarm, se creará un objeto de tipo Intent, el cual se utilizará para crear el nuevo Activity, que se creará con la instrucción starActivity.

Dentro del nuevo Activity creado, se accede al sistema de control del volumen, poniéndolo al máximo.

Posteriormente se crea un objeto del tipo MediaPlayer, con el cual se reproducirá la alarma. Se inicia el reproductor.

Por último se aborta la notificación.

### 5.5.8.- Llega un SMS de acción Block

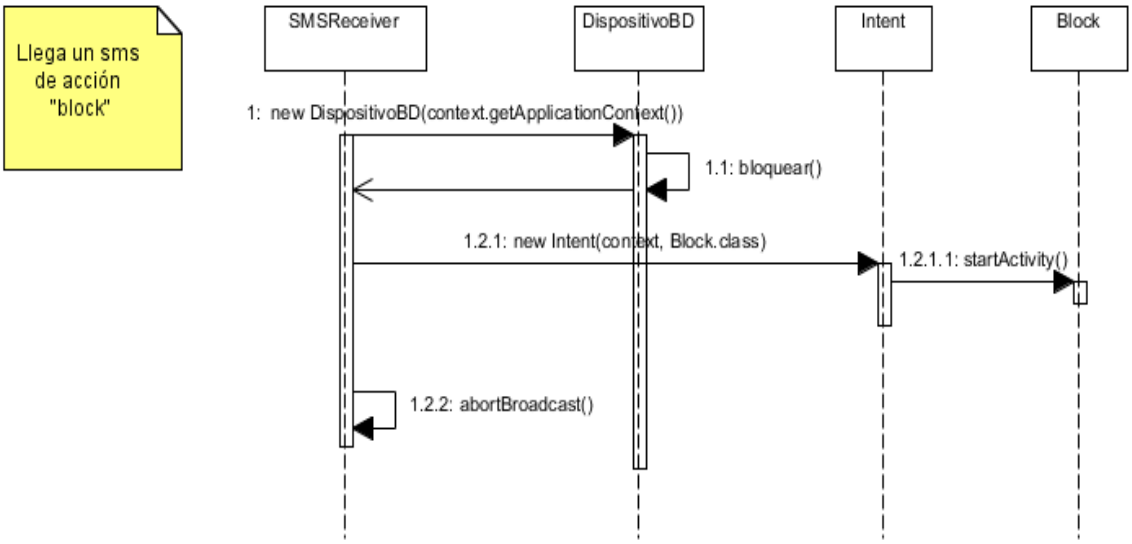



**Figura 51 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Block".**

Cuando llegue un SMS con el código 'Block' se creará un objeto de tipo DispositivoBD, con el cual se indicará en la base de datos que el dispositivo está bloqueado.

Posteriormente se crea un nuevo Intent con el cual se iniciara el Activity encargado de bloquear el dispositivo.

Por último se aborta la notificación.

### 5.5.9.- Llega un SMS de acción Blood

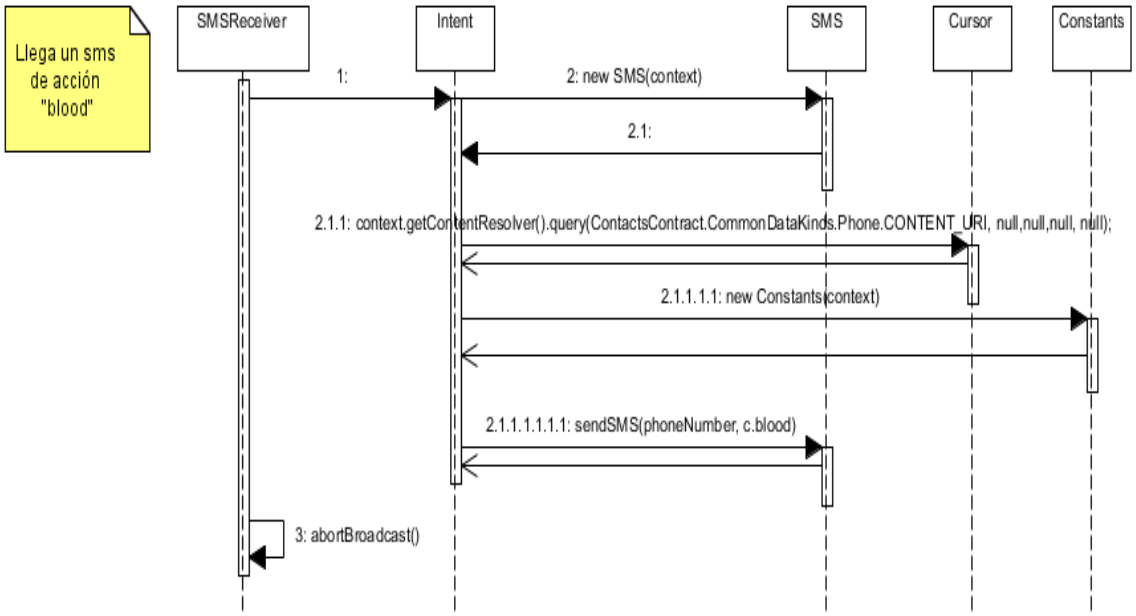



**Figura 52 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Blood".**

Cuando se reciba un SMS con el código 'Blood', se creará un objeto de tipo SMS.

Se añadirán los contactos del dispositivo a un objeto Cursor (objeto similar al ArrayList) y se recorren enviando un SMS a cada usuario. El cuerpo del mensaje se obtiene de la clase Constants.

Por último se anula la notificación.

### 5.5.10.- Llega un SMS de acción Call

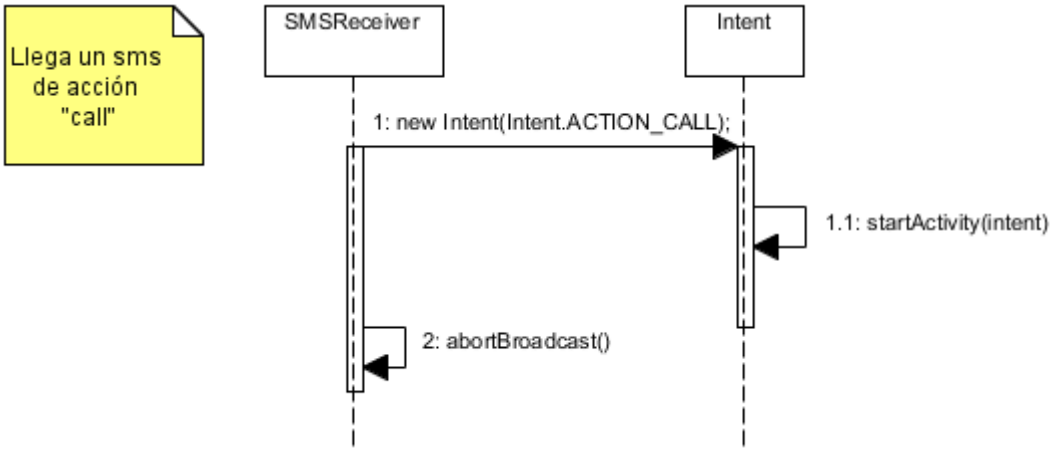



**Figura 53 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Call".**

Cuando se reciba un SMS de acción con el código 'Call' se creará un objeto Intent, que se utilizará para la creación de un nuevo Activity, con el que se realizará la llamada.

Por último se aborta la notificación.

### 5.5.11.- Llega un SMS de acción Camera

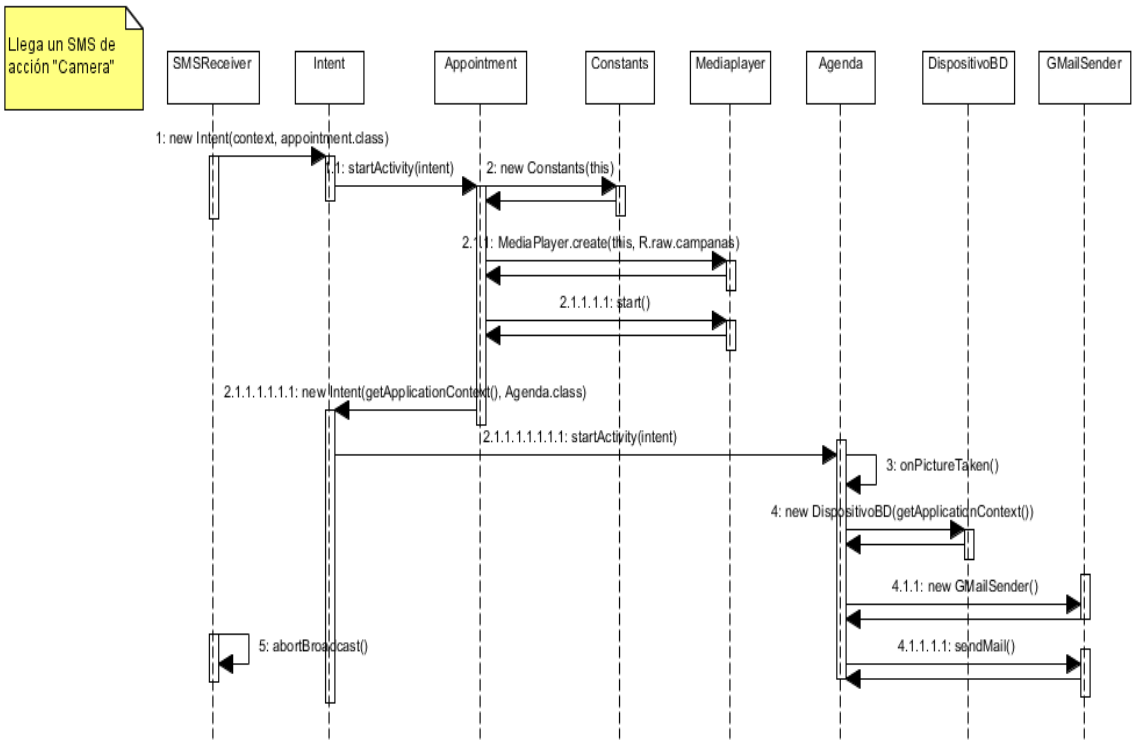



**Figura 54 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Camera".**

Cuando se reciba un SMS de acción con el código 'Camera' se creará un objeto Intent, que se utilizará para la creación de un nuevo Activity. Este nuevo Activity, lanzará una interfaz que simulará ser un recordatorio de la agenda, sonando un sonido de alerta. Cuando el usuario posponga la cita, se captura el evento, lanzando un nuevo Activity que realizará una captura del rostro del usuario mediante la cámara frontal del dispositivo móvil, almacenándola y enviándosela al usuario propietario mediante correo electrónico, utilizando la dirección de confianza.

### 5.5.12.- Llega un SMS de acción Data

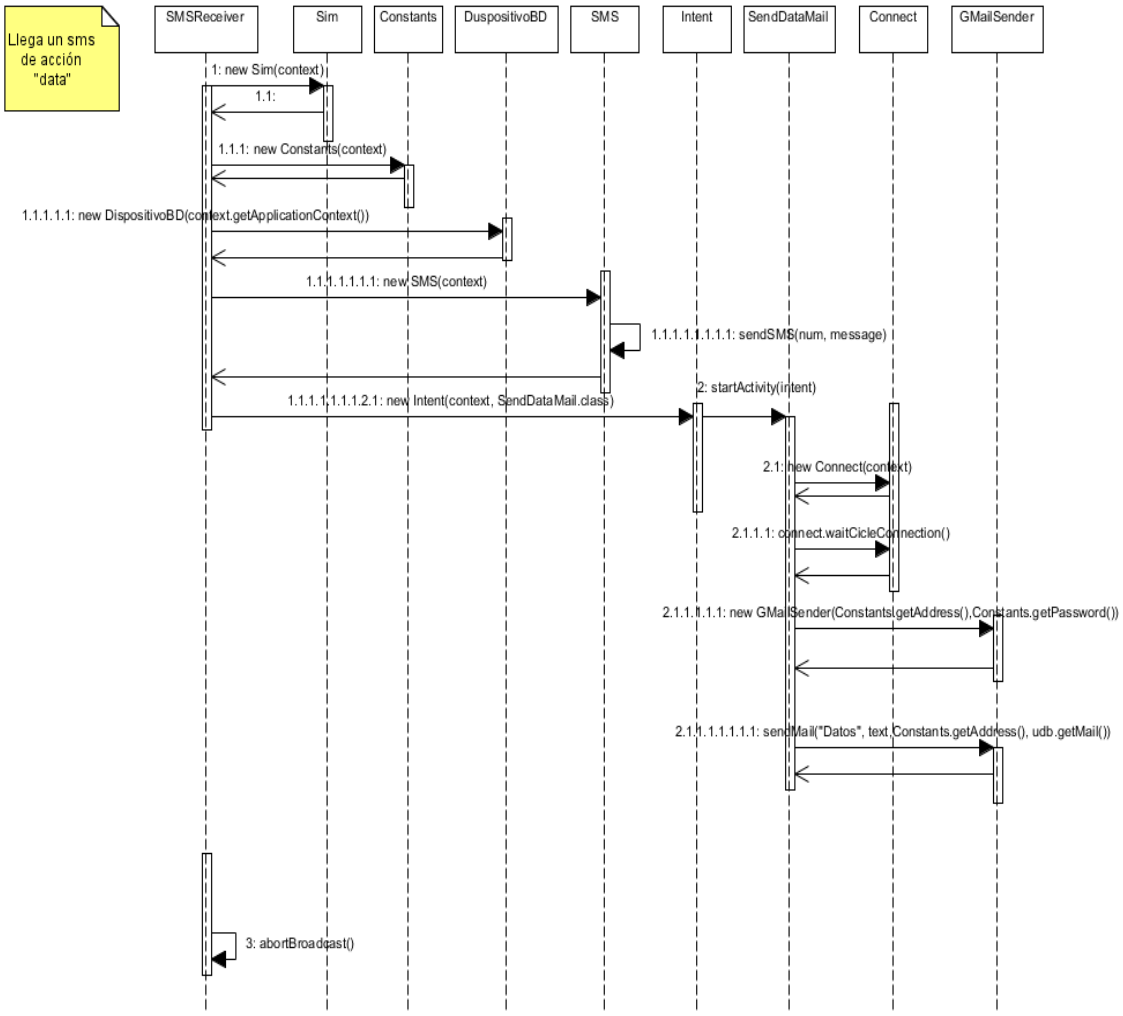



**Figura 55 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Data".**

Cuando se reciba un SMS de acción con el código 'Data' se creará un objeto de la clase SIM, que facilitará el acceso de los datos de la tarjeta SIM y del dispositivo.

Se construye un objeto de la clase Constants que contendrá los textos a mostrar. Es necesario crear otro objeto de la clase DispositivoBD para obtener la dirección de correo electrónico a enviar la información.

Se crea un objeto de la clase SMS, y se llama al método sendSMS(), que enviará la información obtenida al número de teléfono que envió el SMS de acción.

Posteriormente se crea un objeto de la clase Connect, que comprobará si hay conexión a Internet. En caso de no haberla, esperará comprobándola cíclicamente.

Cuando se disponga de conexión, se creará un objeto de tipo GMailSender que enviará el mail a la dirección de correo electrónico almacenada en la base de datos.

Por último se aborta la notificación.

### 5.5.13.- Llega un SMS de acción Locate

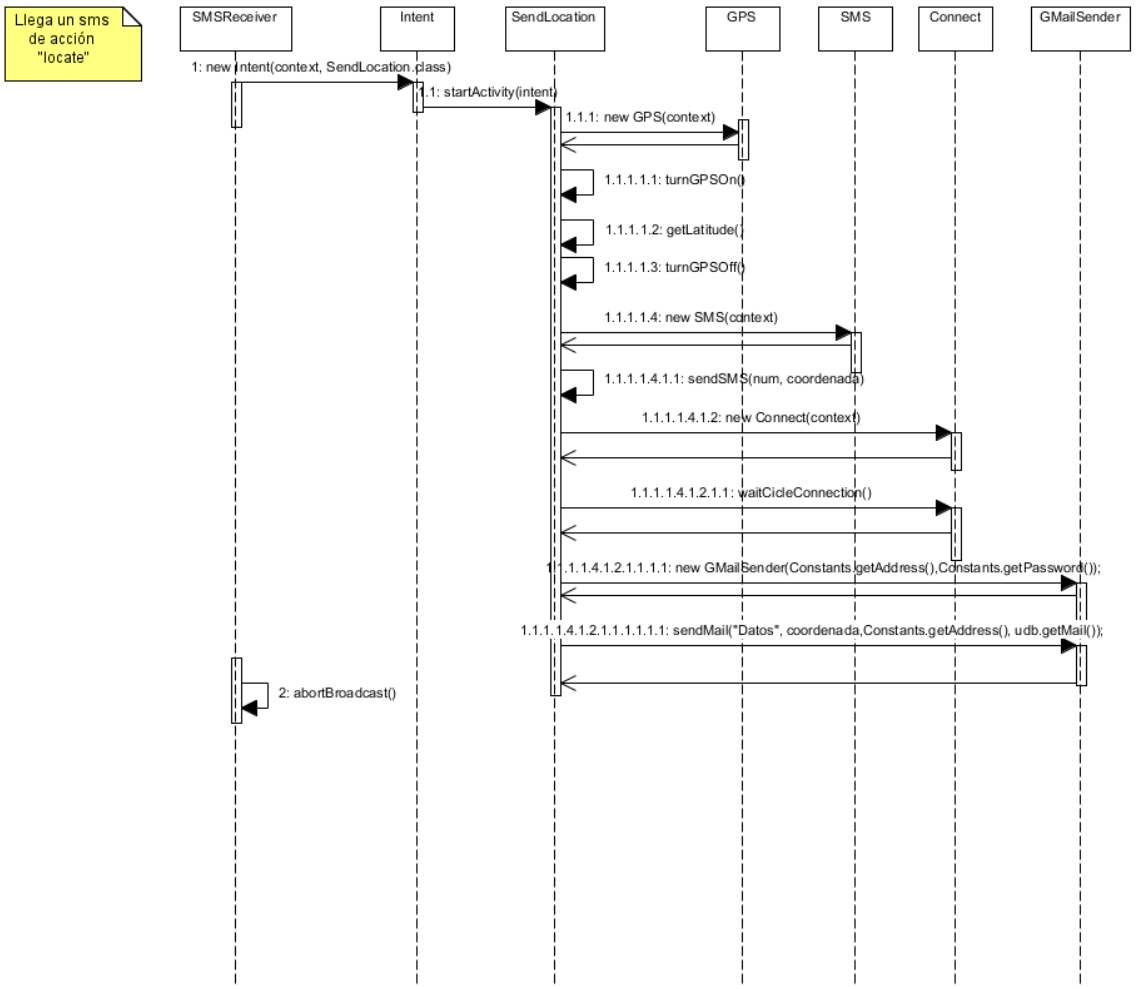



**Figura 56 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Locate".**

Cuando se reciba un SMS con el código de acción 'Locate' se creará un objeto de la clase GPS, se activará el GPS del dispositivo y se obtendrá la longitud y la latitud. Posteriormente se construye el mapa y se envía mediante SMS.

Además, también se comprueba la conexión a Internet mediante la clase Connection, se construye un objeto de tipo GMailSender y se envía el mapa por correo electrónico con la dirección almacenada en la base de datos.

Por último se aborta la notificación.

### 5.5.14.- Llega un SMS de acción Recovery

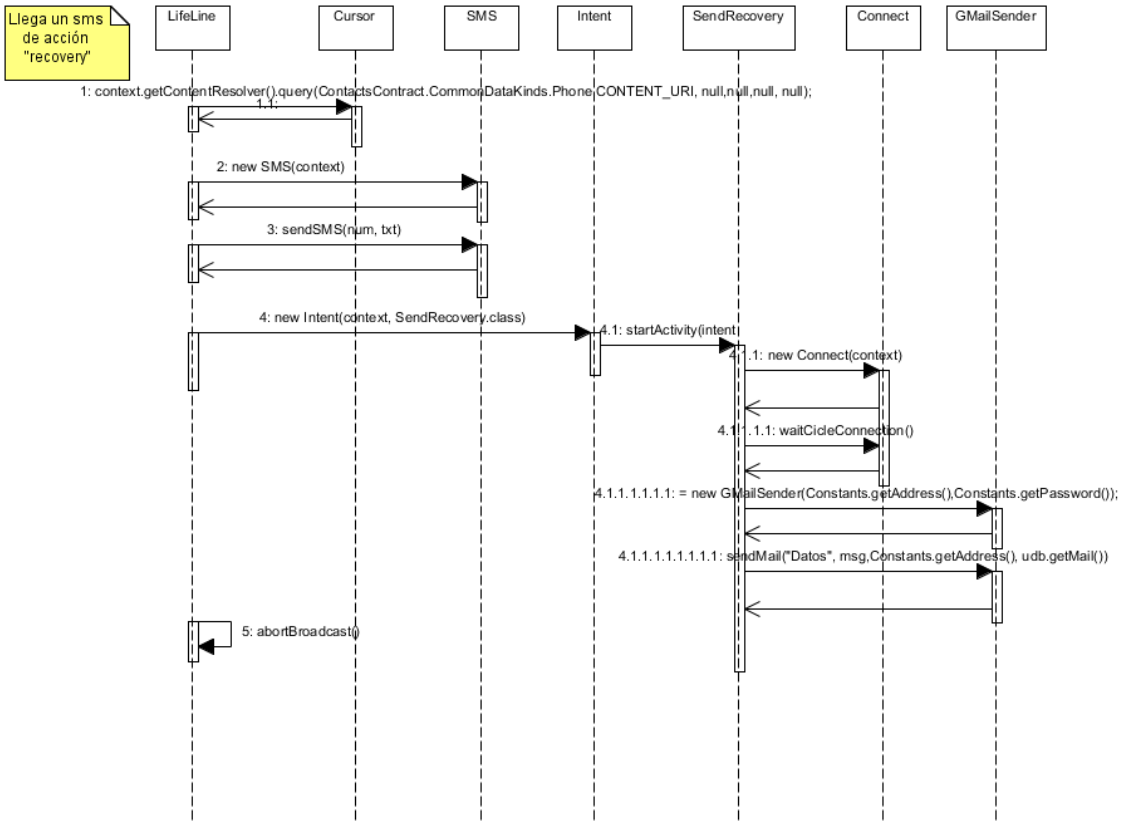



**Figura 57 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Recovery".**

Cuando llegue un SMS de acción con el código 'Recovery' se accederá a la agenda del dispositivo. Se creará un objeto de tipo SMS, y se enviarán los contactos mediante SMS.

Para enviarlas también vía correo electrónico, se comprueba si hay conexión mediante la clase Connection, en caso de no encontrarla esperará hasta que la encuentre, y las enviará mediante un objeto de tipo GMailSender.

Por último se aborta la notificación.

### 5.5.15.- Llega un SMS de acción Text

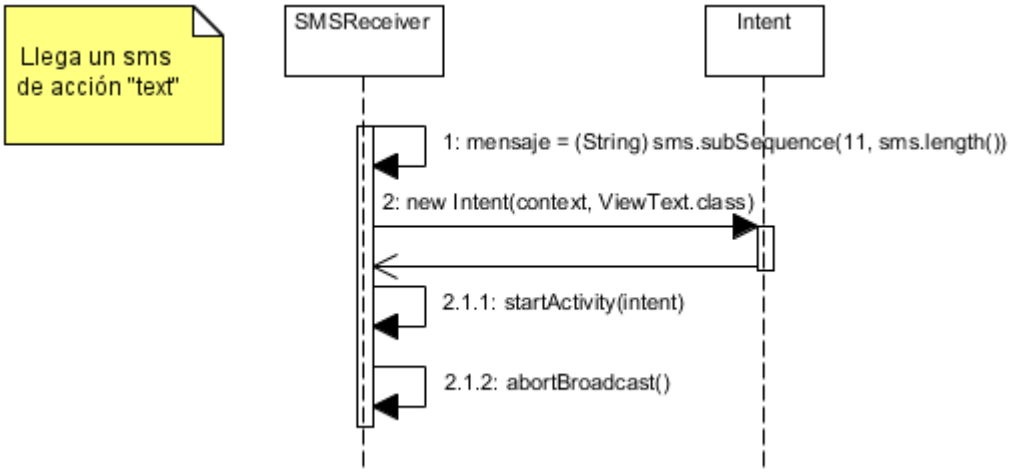



**Figura 58 Diagrama de secuencia de la llegada de un SMS de acción con código "Signal Text".**

Cuando llegue un SMS de acción con el código 'Text', lo primero que se hará será obtener el mensaje a mostrar por pantalla. Posteriormente se creará un Intent para la creación del Activity que mostrará el mensaje por pantalla.

Se carga la interfaz mostrando el mensaje, y se aborta la notificación.

# 6.- Fase de implementación

En este apartado se comentarán algunos aspectos de la fase de implementación de la aplicación.

## *6.1.- Selección de herramientas*

Uno de los aspectos más importantes a la hora de realizar aplicación es el software que se utiliza. En principio había pensado utilizar una herramienta ya presentada con anterioridad, llamada App inventor. Esto no ha podido ser posible debido a dos cosas, App Inventor es una herramienta destinada a hacer software de propósito general, y no de propósito tan específico como lo es la aplicación de este TFC. Y la segunda y más importante debido a la nula flexibilidad de dicha herramienta. Dicho esto, se procede a comentar las herramientas utilizadas en la implementación.

**Eclipse versión 3.7:** IDE utilizado para desarrollar la aplicación.

**SDK v17 Android:** versión del SDK utilizada para desarrollar la aplicación.

**Argo UML:** aplicación para modelizar mediante diagramas UML, los aspectos a cumplir por la aplicación.

**Microsoft Word 2010:** procesador de textos utilizado para realizar la memoria.

**Microsoft PowerPoint 2010:** aplicación utilizada para realizar las diapositivas de la presentación.

**Google Maps:** para mostrar los mapas.

## *6.2.- Interfaz de usuario*

Esta aplicación no se distingue por la cantidad de interfaces que posee, sino más bien por lo contrario, por todo lo que hace de manera interna, oculta al usuario. Hay que comentar aun así, que una de las cosas que más complejidad tiene a la hora de realizar aplicaciones Android es el aspecto de las interfaces, ya que hay toda una jungla de dispositivos con características tan distintas que resulta difícil realizar interfaces que se adapten bien a todos los dispositivos del mercado.

Otro aspecto que se ha buscado en la realización de las interfaces es el de la internacionalización de la aplicación, logrando realizar una aplicación multi-idioma, lo cual amplía el mercado de futuros clientes que tendría la aplicación.

La interfaz principal de la aplicación es la siguiente:

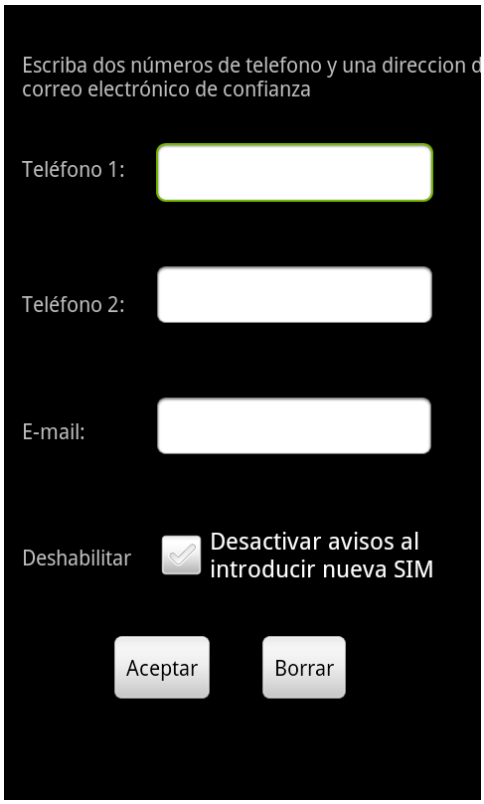



**Figura 59 Interfaz principal de SimDetect.**

Se espera que se inserten los datos de los contactos de confianza, lo que incluye dos números de teléfono y una dirección de correo electrónico. Estos datos se almacenarán y se podrán modificar cuando se desee.

También se podrá desactivar la aplicación si se va a insertar una tarjeta SIM distinta voluntariamente. Desactivándola, se evitará que se envíe la notificación por insertar una tarjeta SIM no válida.

Si se pulsa el botón de "Borrar" se borrarán los datos almacenados y se podrá introducir unos nuevos.

Cuando se pulse el botón de "Aceptar", se comprobará si existe conexión a Internet; en caso de no existir se mostrará un aviso de que para completar el paso es necesario conexión a Internet.

En caso de estar conectado, se enviará un mail a la dirección introducida y se almacenarán los datos introducidos.

La segunda interfaz más importante de la aplicación es la que se puede observar a la derecha de estas líneas, y en la que se busca capturar la foto de la persona que posee el teléfono.

En esta interfaz lo que se hace es simular un recordatorio de la agenda, para buscar la intervención de la persona que posee el teléfono.

Cuando la persona busque posponer la cita para quitar el aviso, se realizará la foto de la persona, enviándola por correo electrónico a la dirección introducida en la aplicación.


**Figura 60 Simulación de recordatorio de una cita.**

## *6.3.- Licencia*

La aplicación realizada en este trabajo se ha licenciado mediante licencia Apache o Apache License, que es una licencia de software libre creada por la Apache Software Foundation (ASF). Esta licencia está aprobada por OSI (Open Source Initiative) y es compatible con la licencia GPLv3.

Las condiciones de la licencia son las siguientes:

1. La Licencia Apache permite al usuario del software la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de ese software.

2. La Licencia Apache no exige que las obras derivadas (versiones modificadas) del software se distribuyan usando la misma licencia, ni siquiera que se tengan que distribuir como software libre / open source.

3. La Licencia Apache sólo exige que se notifique a los receptores que en la distribución se ha usado código con la Licencia Apache.

# 7.- Manual de usuario

SimDetect es una aplicación sencilla de utilizar, y sobre todo, no le interferirá en su día a día y no le provocará una reducción de la batería de su dispositivo. SimDetect le facilitará encontrar su dispositivo móvil en caso de pérdida o robo.

A continuación el usuario podrá leer el manual de usuario, en el que se explicará cómo proceder en caso de necesitar localizar su dispositivo.

## *7.1.- Primeros pasos*

Una vez instalada la aplicación, podrá ejecutarla para rellenar los datos que le facilitarán encontrar su dispositivo móvil, en caso de pérdida o robo. Si decide no realizar esta acción, cuando reinicie el dispositivo, la aplicación le recordará que debe de rellenarlos.

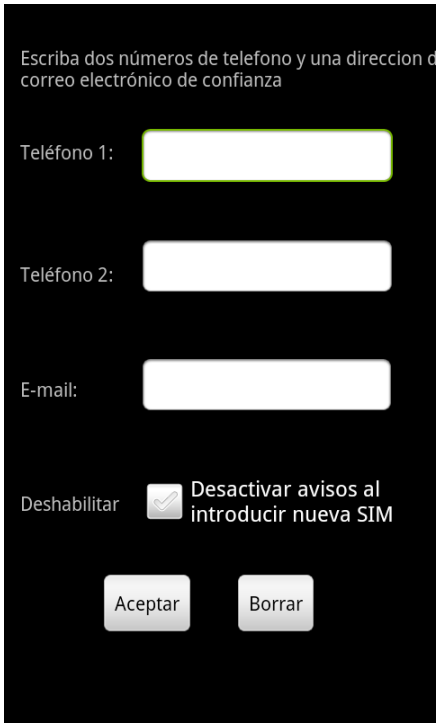



**Figura 61 Interfaz principal de SimDetect.**

Éstos son de vital importancia, ya que en caso de que se inserte otra tarjeta SIM se notificará a esos datos de contacto. Cuando los datos estén rellenados solo tendrá que pulsar el botón de aceptar para que los datos se almacenen. Es necesario que esté conectado a Internet para que SimDetect le permita realizar este paso, ya que le enviará a la dirección de correo electrónico introducida un pequeño manual con la forma de utilizar la aplicación.

Una vez se haya cumplido este paso no se tendrá que preocupar más, SimDetect no le requerirá más su atención, no recibirá molesta publicidad, no aparecerán pantallas que molesten, simplemente estará oculto esperando su momento de actuar.

Simdetect no aumentará prácticamente el consumo de batería de su dispositivo en el día a día ya que está muy optimizado y no notará que está instalado.

No se preocupe si alguien quita su tarjeta SIM e introduce otra, SimDetect seguirá funcionando y le enviará un SMS a los números de teléfono insertados en la aplicación. Así podrá obtener el número de teléfono al que podrá enviar los SMS para intentar localizarlo.

En caso de pérdida o robo, usted podrá obtener los datos de su agenda y/o intentar localizar e móvil con el envío de SMS sin que el poseedor del móvil se entere. Tan solo deberá enviar un SMS que comience por los siguientes códigos:

**Signal data:** en caso de robo y de que alguien inserte otra tarjeta SIM, enviando este SMS usted podrá obtener los datos de la tarjeta SIM insertada. Con estos datos podrá ir a la policía a realizar una denuncia si usted lo desea, o a su operador telefónico para que bloque el dispositivo de manera irremediable.


**Figura 62 Datos obtenidos con un "Signal data".**

**Signal text:** escriba este código junto con el texto que usted desee. El texto se mostrará en la pantalla del teléfono. Si alguien tiene o encuentra el móvil podrá leer el texto que usted escriba.


**Figura 63Mensaje mostrado en el teléfono.**

**Signal call:** si se envía este mensaje se realizará una llamada desde su móvil (siempre y cuando tenga una tarjeta SIM insertada) al móvil desde el que envío el SMS. Así podrá escuchar lo que esté hablando la persona que tenga su dispositivo móvil.


**Figura 64 Llamada realizada por la aplicación.**

**Signal alarm:** si usted no encuentra el teléfono móvil y piensa que está o puede estár muy cerca, cuando envíe este mensaje sonará una alarma que le facilitará conocer donde está o quién posee su dispositivo móvil.

**Signal blood:** con este código se enviará un SMS a cada contacto de la agenda informando de que está utilizando un teléfono móvil robado. Utilice este comando solo si se ha insertado otra tarjeta SIM.

**Signal block:** enviando este SMS se bloqueará el teléfono, impidiendo que se pueda utilizar. Este bloqueo continuará aunque se reinicie el teléfono. Se desbloqueará cuando se introduzca la tarjeta SIM válida. Este bloqueo puede ser inutilizado por un experto.

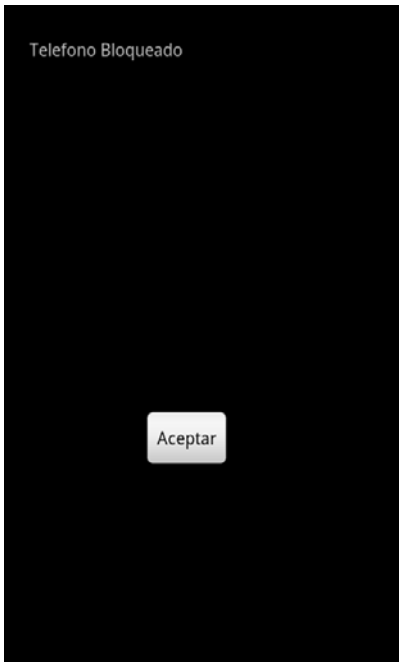



**Figura 65 captura de un teléfono bloqueado.**

**Signal locate:** este código obtendrá las coordenadas GPS donde se encuentre el teléfono móvil. Enviará al remitente una dirección de Internet que le facilitará localizar el móvil mediante un mapa. Se recomienda tener el GPS activado para esta función. Si la aplicación no encontrase conexión con los satélites GPS continuará buscando hasta que la encuentre.

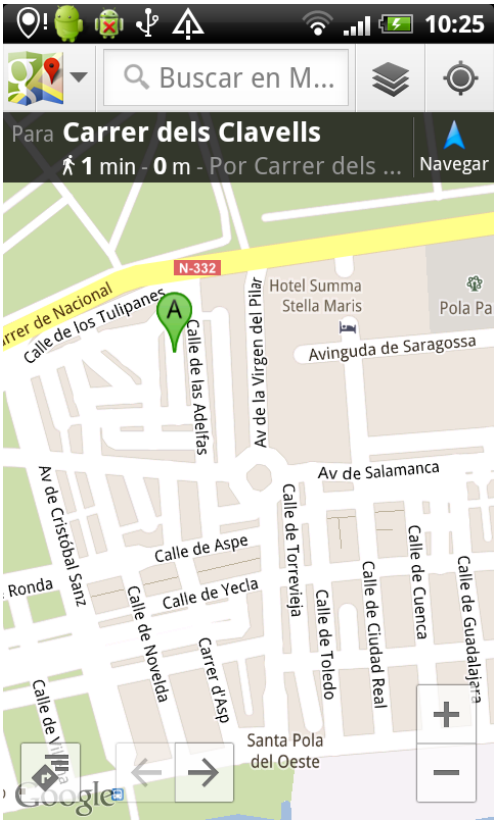



**Figura 66 Aplicación localizando el teléfono.**

**Signal camera:** esta función buscará obtener la cara de la persona que tiene su teléfono móvil. La enviará a la dirección de correo electrónico introducida en la aplicación. Para obtenerla simulará un recordatorio de la agenda de una cita con el médico. Cuando se vaya a posponer la cita se realizará la fotografía con la cámara frontal del dispositivo móvil.


**Figura 66 Aplicación localizando el teléfono.**

**Signal recovery:** obtendrá los nombres y los números de teléfono de la agenda, para que en caso de pérdida pueda recuperarlos. No utilice este comando si se ha insertado otra tarjeta SIM en su teléfono.

A la hora de escribir los comandos no se preocupe por las mayúsculas o minúsculas, SimDetect entenderá los códigos de ambas maneras.

## *7.2.- Cómo proceder en caso de pérdida en interiores*

Si usted no recuerda dónde dejó su dispositivo móvil dentro de su propia casa, pero lo dejó en "modo silencioso", podrá localizarlo fácilmente enviando un SMS con el código "**Signal alarm"**. Aunque esté en "modo silencioso" sonará una alarma que le permitirá localizarlo fácilmente.

## *7.3.- Cómo proceder en caso de pérdida en exteriores*

En caso de haber perdido el dispositivo móvil en exteriores se deberá de proceder de diferente manera.

El primer paso sería localizar la zona en la que se encuentra el dispositivo enviando un SMS con el código "**Signal locate"** que mostrará la situación del móvil en un mapa.

Una vez esté en el sitio, podrá localizarlo enviando un SMS con el código "**Signal alarm"**, aunque es recomendable asegurarse de que el dispositivo sigue en el mismo sitio.

Si se desea, como medida de precaución se puede enviar un SMS con el código "**Signal text"** e indicando que si alguien encuentra el teléfono que llame a un número de teléfono en particular.

En cualquier caso, si se desea recuperar los datos de los contactos o realizar una copia de seguridad de los mismos se podrá realizar enviando un SMS con el código "**Signal recovery**".

## *7.4.- Cómo proceder en caso de robo*

En caso de que el dispositivo móvil haya sido sustraído ilegalmente, el modo a operar será semejante al caso anterior, salvo que se haya introducido una tarjeta SIM distinta, en ese caso se le notificará mediante SMS y correo electrónico a los números de contacto introducidos en la aplicación. En este caso los pasos a seguir son los siguientes.

El primer paso a realizar será obtener los datos de la tarjeta SIM introducida. Esto se realizará mediante el envío de un SMS con el código "**Signal data**" Con estos datos se podrá realizar una denuncia en un futuro.

Enviando un SMS con el código "**Signal camera**" facilitará la obtención de la cara de la persona poseedora del dispositivo. Otro dato que se podría utilizar en una eventual denuncia.

En cualquier caso, siempre es recomendable obtener los datos de la agenda realizando una copia de seguridad antes de que puedan ser eliminados.

Para localizar el dispositivo, tal y como se ha comentado con anterioridad, se realizaría enviando un SMS con el código "**Signal locate"**. Una vez en la zona, para encontrarlo bastará con enviar un SMS con el código "**Signal alarm"**.

Si se desea escuchar a la persona poseedora del dispositivo, habrá que enviar un SMS con el código "**Signal call**", el teléfono llamará de forma automática al teléfono desde el que se envió el SMS.

Siempre podrá enviar un mensaje utilizando el código "**Signal Text**".

Enviando un SMS con el código "**Signal Blood**" podrá presionar al poseedor del dispositivo a que le devuelva el teléfono móvil. Se recomienda no abusar de este comando.

El último paso a realizar será en caso de no poder obtener el dispositivo, bloquearlo para impedir su utilización. Esto se podrá realizar enviando un SMS con el código "**Signal Block**". El dispositivo quedará bloqueado impidiendo su utilización hasta que se introduzca la tarjeta SIM del usuario.

# 8.- Conclusiones y trabajo futuro

El objetivo principal, aparte de realizar el TFC, era el de realizarlo con el objetivo de aprender algo más, de salir al mercado con más conocimientos que me puedan facilitar encontrar trabajo en este maltrecho mercado laboral.

Puedo decir con gran orgullo que considero que el objetivo ha sido cumplido con creces, ya que podido aprender dos cosas que considero pueden facilitarme el acceso al mercado laboral como son aprender la programación en una plataforma de gran auge como es Android, y aprender un nuevo paradigma de programación como es la programación orientada a eventos.

Respecto a la aplicación, el objetivo de realizar una aplicación Android que facilitase el encontrar el dispositivo móvil en caso de pérdida o de robo. La aplicación tiene exactamente esa función, proporciona una serie de opciones que pueden facilitar que en caso de pérdida o de robo faciliten encontrar el dispositivo.

También me gustaría comentar la gran cantidad de problemas y de quebraderos de cabeza que la propia plataforma Android me ha dado, y que ha provocado que un proyecto con una dificultad media/alta se convierta en una dificultad muy alta impidiéndome realizar muchas cosas, dándome errores en muchas otras y obligándome a romperme la cabeza y destripar Android para encontrar agujeros de seguridad en el propio sistema para realizar algunas de las funciones que realiza la aplicación. Estas dificultades son las siguientes:

- **Versiones de Android:** cada versión de Android lleva su propia API, lo cual dificulta la programación, ya que en según qué versiones de Android no se tiene en cuenta el hardware que los teléfonos ya disponen. Por ejemplo hasta la versión 2.3 Android no tiene en cuenta en su API la cámara fotográfica frontal, cuando los teléfonos ya la incluían con anterioridad, o antes de la versión 2.2 no se podían mover ficheros entre memoria interna y memoria externa, a pesar de que prácticamente todos los dispositivos disponían de ambas.

- **Obtener el número de teléfono:** los operadores de teléfono españoles no incluyen el número de teléfono como dato en la tarjeta SIM, lo cual imposibilita que se pueda obtener dicho dato. Es por esto que aplicaciones como whatsapp te pide el número de teléfono.

- **API con carencias:** en la realización de la aplicación me he encontrado con una API no tan completa como me hubiese gustado, encontrando algunas lagunas, como por ejemplo en el trabajo con la base de datos, obligándome a trabajar en bajo nivel para cubrirlas. También me encontré con lo que he bautizado como API abierta, que es una API que Google ofrece pero que cada compañía implementa como le conviene, lo cual puede provocar una aplicación funcione en teléfonos de una marca, pero no en otra. Esto me ocasionó serios problemas a la hora de trabajar con la cámara fotográfica.

- **Bloquear teclas del dispositivo:** aunque se pueden capturar los eventos de las pulsaciones de las teclas del dispositivo, y modificar su comportamiento, hay ciertas teclas que Android no permite modificar o bloquear, para impedir que una aplicación pueda bloquear el dispositivo. Esta medida de seguridad ha sido un serio problema ya que uno de los objetivos era poder bloquear el dispositivo. Al final se consiguió bloquear el dispositivo aprovechando un agujero de seguridad en Android.

- **Capturar SMS:** aunque no supuso una gran dificultad, la aplicación necesitaba capturar el evento de la llegada de un nuevo SMS antes de que el propio Sistema Operativo lo notificara.

- **Habilitar/deshabilitar Hw:** otra cosa que Android no permite es la activación del WiFi, 3G o del GPS sin la acción directa del usuario. Solo se permite la activación de estos dispositivos, sin acción humana, a aquellas aplicaciones que vienen de fábrica instaladas en los dispositivos móviles. Es otra medida de seguridad que Android ofrece a sus usuarios. Para poder activar el WiFi y el GPS en la aplicación se ha explotado un agujero de seguridad encontrado durante la realización de la aplicación.

- **Permisos:** para acceder a los recursos del dispositivo se requiere utilizar permisos en la aplicación. El problema es que existen tres niveles de permisos en Android, y no todos son accesibles a los programadores, lo cual imposibilita que una aplicación pueda realizar funciones sobre el sistema.

- **Problemas a l a hora de obtener las coordenadas GPS:** Android permite obtener las coordenadas del GPS para mostrarlas sobre un mapa sin ninguna dificultad, el problema reside cuando quieres obtenerlas sin mostrar ninguna interfaz o mapa, ocultándola al usuario. Esto ha ocasionado más de un quebradero de cabeza.

- **Problemas con los Activity:** permiten ejecutar en un hilo una parte de la aplicación, pero accediendo a dispositivos o al propio sistema resultan algo problemáticos si no se utilizan con interfaces.

- **Problemas de compatibilidad con la cámara:** como se ha comentado con anterioridad trabajar con la cámara del dispositivo ha sido todo un reto, ya que contiene API personalizables que no todas las marcas implementan de la misma manera, lo cual provoca que una aplicación pueda no funcionar todos los teléfonos, sobre todo si esa fotografía no se muestra por pantalla, algo que Android de nuevo no permite hacer por motivos de seguridad. En la aplicación se ha tenido que engañar al sistema para poder obtener la fotografía.

Aunque existen aplicaciones similares en el mercado, me gustaría comentar que en mi opinión estas aplicaciones no hacen un uso correcto de la gestión de memoria, ya que la aplicación está constantemente en memoria y/o ejecutándose, lo cual provoca una reducción considerable en la duración de la batería, algo que en mi aplicación ha sido optimizado al máximo siendo uno de los objetivos principales de la aplicación.

Como trabajos futuros o posibles mejoras, comentaría el de buscar una mejor compatibilidad, algo que ya de por si es bastante complicado debido a la filosofía y a las restricciones que el propio sistema operativo tiene.

Otra posible mejora a realizar en la aplicación sería incluir localización mediante de NFC (Near Field Communication), que permitiera localizar el dispositivo cuando se introduzca en un local que utilice esta tecnología.

# Bibliografía


[A3ADC] Kyle Merrifield Mew, “Android 3.0 Application Development Cookbook”, PACKT Publishing 2011

[APPI] Página web de la aplicación App Inventor.
http://appinventor.mit.edu/
Última consulta: 31 de Octubre de 2012

[ARQAN] Arquitectura utilizada en Android. Disponible en:
http://lh6.ggpht.com/-qFnfvkYMHkM/ToSl3Rh0BII/AAAAAAAAAPU/LZIyyqaykcw/Arquitectura%252520del%252520sistema%252520operativo%252520Android%25255B6%25255D.jpg

[AVA] Dirección Web de la aplicación Avast! Disponible en: http://www.avast-antivirus.es/avast-free-mobile-security.html
Última consulta: 31 de Octubre de 2012

[B4A] Página web de la aplicación Basic4Android.
http://www.basic4ppc.com/
Última consulta: 31 de Octubre de 2012

[CORSDK] Dirección Web de la aplicación Corona SDK. Disponible en:
http://www.coronalabs.com/products/corona-sdk/
Última consulta: 31 de Octubre de 2012

[CSDK] Página web de la aplicación corona SDK.
http://www.coronalabs.com/products/corona-sdk/
Última consulta: 31 de Octubre de 2012

[DDAA] *Joan Ribas Lequerica*, “Desarrollo de aplicaciones para Android”, Anaya Multimedia, 2011

[DIRAVG] Dirección Web de la aplicación AVG Antivirus. Disponible en:
http://free.avg.com/es-es/antivirus-for-android
Última consulta: 31 de Octubre de 2012

[DIRLOO] Dirección Web de la aplicación LookOut. Disponible en:
https://www.mylookout.com/
Última consulta: 31 de Octubre de 2012

[DIRMAC] Dirección Web de la aplicación McAfee WaveSecure. Disponible en:
https://www.wavesecure.com/
Última consulta: 31 de Octubre de 2012

[DIRNMS] Dirección Web de la aplicación Norton Mobile Security. Disponible en:
http://us.norton.com/norton-mobile-security/
Última consulta: 31 de Octubre de 2012

[DIRSD] Dirección Web de la aplicación Seek Droid. Disponible en:
https://seekdroid.com/v2/
Última consulta: 31 de Octubre de 2012

[ECLI] Página web de la aplicación Eclipse.
http://www.eclipse.org/
Última consulta: 31 de Octubre de 2012

[FGM11] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Analysis of the most representative factors affecting Warning Message Dissemination in VANETs under real roadmaps," in19th annual meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Singapore, July 2011, pp. 197–204.

[IARPC] Invasión Androide. Revista Personal Computer & Internet, nº 104 pp 32-34, 2011.

[IEEE1830] IEEE Recommended Practice for Software Requirementes Specifications, asignatura Ingeniería del Software, Escuela Universitaria Politécnica de Teruel.

[MCC09] F. J. Martinez, J.-C. Cano, C. T. Calafate, P. Manzoni, and J. M. Barrios. "Assessing the feasibility of a VANET driver warning system". In Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks (PM2HW2N'09). ACM, New York, NY, USA, 2009, pp. 39-45. DOI: 10.1145/1641913.1641919

[MOD] Página web de la aplicación MonoDroid.
http://xamarin.com/monoforandroid
Última consulta: 31 de Octubre de 2012

[MTC12] F. J. Martinez, C.-K. Toh, J.-C. Cano, C. T. Calafate, and P. Manzoni. "Determining the Representative Factors affecting Warning Message Dissemination in VANETs". Wireless Personal Communications, 2012. Springer, Netherlands, vol. 67, no. 2, pp. 295-314. DOI: 10.1007/s11277-011-0379-3.

[NETB] Página web de la aplicación Netbeans.
http://netbeans.org/
Última consulta: 31 de Octubre de 2012

[PHLOC] Dirección Web de la aplicación PhoneLocatorPro. Disponible en:
http://phonelocatorpro.com/
Última consulta: 31 de Octubre de 2012

[RHR] Página web de la aplicación Rhomobile Rodes.
http://www.motorola.com/Business/US-EN/RhoMobile+Suite/Rhodes
Última consulta: 31 de Octubre de 2012

[RUB] Página web de la aplicación Ruboto.
http://ruboto.org/
Última consulta: 31 de Octubre de 2012

[SOESP] Gráfico que expresa la utilización de los diferentes sistemas operativos móviles en España. Disponible en:
http://gs.statcounter.com/#mobile_os-ES-monthly-200901-201202
Última consulta: 23 de marzo de 2012.

[SOMUN] Gráfico que expresa la utilizacion de los diferentes sistemas operativos móviles en el mundo. Disponible en:
http://gs.statcounter.com/#mobile_os-as-monthly-200901-201202
Última consulta: 23 de marzo de 2012.

[UVAND] Dirección web que proporciona los datos estadísticos que Google ofrece sobre Android. Disponible en:
http://developer.android.com/resources/dashboard/platform-versions.html
Última consulta: 21 de Octubre de 2012

[WASL] *D. Wolber, H. Albenson, E. Spertus, L. Looney* "App Inventor-Create your Own Android Apss", O'Reilly, 2011