UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

EASY PHONE MANAGER: Aplicación de gestión y automatización de dispositivos móviles

Proyecto Final de Carrera

INGENIERÍA INFORMÁTICA

Autor: Diego Panadero Roser

Director: David de Andrés Martínez

24 de marzo de 2015

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de códigos

1. Introducción

El mundo de la telefonía móvil ha evolucionado y está evolucionando a un ritmo vertiginoso desde la creación del primer dispositivo, el Motorola DynaTAC 8000X, en el año 1983 [1]. Por entonces, el dispositivo móvil se concibió con la clara finalidad de poder realizar llamadas desde cualquier parte y prescindir del cableado de los teléfonos fijos.

Dicha finalidad fue abordada con gran éxito. Diseñadores y desarrolladores se centraron entonces en la mejora estética del dispositivo, disminuyendo su tamaño para obtener una mejor portabilidad y creando diseños elegantes y modernos que pudieran despertar el interés de los usuarios, como la introducción del primer móvil con tapa.

Fue cuestión de tiempo, aproximadamente 17 años, cuando se decidió dotar a los móviles de una serie de funcionalidades que iban más allá de realizar llamadas y que supondría una revolución en el aspecto tecnológico. Estas funcionalidades serían la introducción de la cámara de fotos o la reproducción de audio, entre otras, que se han convertido en un estándar de todo móvil en la actualidad. Nacía así el "Smartphone" [2].

En el año 2007, aprovechando el tirón de ventas de la telefonía móvil y que aún era un terreno por explotar, la compañía Apple revoluciona el mercado con la creación de su propio terminal, el IPhone [3]. Este dispositivo supone una mejora tecnológica que marca un antes y un después en la era tecnológica, no sólo por sus increíbles mejoras en el diseño, como la incorporación de pantalla táctil, sensor de rotación, etc., sino por la apertura de un mercado de aplicaciones donde los programadores podían publicar sus propias aplicaciones y ser utilizadas por cualquier usuario, el APP Store.

A partir de entonces, surgen otros competidores de Apple que disponen de sus propios mercados de aplicaciones, como el Android Market de la compañía Android o el Marketplace de Windows Phone.

Así pues, con tal evolución en la telefonía móvil y con la posibilidad de ofrecer a los usuarios aplicaciones creadas por cualquier programador, surgen aplicaciones de todo tipo que intentan adaptarse a sus necesidades diarias. Desde aplicaciones que ayudan al usuario a cocinar todo tipo de comidas, hasta aplicaciones para ser utilizadas en el deporte que calculan el número de kilómetros recorridos, calorías consumidas, etc. o incluso aplicaciones de entretenimiento, como los tradicionales juegos de mesa llevados a la pantalla del dispositivo.

Gracias a la existencia de todas estas aplicaciones que facilitan la vida del usuario, el teléfono móvil se ha convertido en un compañero indispensable para el ser humano. A día de hoy casi todo el mundo posee su propio Smartphone y disfruta de todas sus funcionalidades hasta el punto de que, para muchas tareas en las que el uso de un ordenador era estrictamente necesario, ha dejado de serlo.

Los móviles son cada vez más potentes y más rápidos, lo que permite que se creen aplicaciones más complejas que requieran de esa potencia, entrando así en un círculo vicioso en el que a más potencia, mejores aplicaciones y más usuarios utilizando el Smartphone como compañero tecnológico preferido.

Es de hecho, este gran crecimiento e influencia de la tecnología móvil en el ser humano lo que motiva a desarrollar una aplicación móvil, y así poder aprovechar este tipo de mercado que tanto revuelo está causando en la sociedad actual.

Ahora bien, la tarea complicada sería encontrar una aplicación que pueda ser de interés y que no abunde en ese inmenso mercado de aplicaciones. Cierto es que los dispositivos móviles son capaces de ejecutar aplicaciones de gran complejidad, pero además de potencia, ¿No sería de gran utilidad poder automatizar nuestro dispositivo para que realice trabajos y tareas sin necesidad de interacción del usuario?

Un respuesta evidente a ello sería que sí. Sin duda, la realización automática de tareas que pueda realizar el dispositivo siempre es bienvenida ya que permite al usuario no tener que preocuparse por ellas, disminuyendo la dependencia de su dispositivo y otorgándole más tiempo para poder dedicarlo a otros asuntos de mayor interés.

Un ejemplo frecuente, ¿Cuántas veces hemos estado manipulando el dispositivo móvil con una red WIFI, por ejemplo dentro de casa, y cuando hemos salido de ella el WIFI se ha quedado conectado consumiendo batería de manera innecesaria?

O bien, ¿Cuántas veces mientras estamos conduciendo hemos recibido una llamada que no hemos podido contestar y nos gustaría notificar a la persona que ha llamado de que estamos conduciendo?

Por este tipo de situaciones cotidianas, entre muchas otras que se podrían automatizar mejorando nuestro día a día, surge la idea del desarrollo de una aplicación para dispositivos móviles de plataforma Android, **Easy Phone Manager**.

Easy Phone Manager busca ser una aplicación que, a pesar de la gran cantidad de aplicaciones ya existentes, pueda hacerse hueco en el mercado y pueda ser de utilidad para el usuario. Es decir, que pase a formar parte de su día a día, dejando atrás ese tipo de aplicaciones que se instalan y se dejan olvidadas.

Si bien es cierto que existen aproximadamente 1.300.000 aplicaciones en el mercado Android, también es cierto que un gran porcentaje pertenece a aplicaciones que no tienen influencia y ni siquiera son conocidas, ya sea porque el desarrollador no ha sabido establecer unas prestaciones acordes a la necesidades del usuario, la aplicación no ha tenido una interfaz atractiva o la aplicación no es intuitiva y la mayoría de usuarios no han sabido utilizarla.

Con Easy Phone Manager se desea evitar caer en esos errores, tratando de conseguir una aplicación compacta que responda fiablemente bajo cualquier situación a la que se pueda enfrentar el dispositivo. Será una aplicación sencilla e intuitiva, por lo que cualquier usuario la podrá manipular sin problemas gracias a su diseño “Dashboard” y al uso de notificaciones que informarán de lo que está ocurriendo en cada momento. Y sobre todo, cosa que se considera de importancia, buscará ser una aplicación útil y práctica que de verdad tenga una finalidad concreta y proporcione al usuario lo que se espera de ella.

Así pues, Easy Phone Manager será una aplicación versátil que ofrecerá diversas e interesantes funcionalidades, como la posibilidad de administrar automáticamente las conexiones del teléfono, como son el WIFI y los datos 3G. Dicha automatización se realizará atendiendo a diferentes situaciones que el usuario determinará según sus necesidades, como por ejemplo, la administración de las conexiones según el bloqueo de pantalla, según una localización GPS o un nivel bajo de batería, entre otras.

Otra función característica que se desarrollará en la aplicación será la de administrar la batería del terminal. El usuario podrá indicar un nivel de batería que se considerará crítico y escoger una serie de opciones disponibles. Cuando el dispositivo móvil alcance ese umbral de batería se activarán las opciones seleccionadas, como por ejemplo modificar el tiempo de bloqueo de la pantalla, así como desactivar el WIFI, el BLUETOOTH o cambiar el brillo.

Las funcionalidades explicadas corresponden con la parte de la aplicación que se encargará de administrar configuraciones del teléfono. Sin embargo, Easy Phone Manager ofrecerá también un par de funcionalidades extra, que resultará junto a las anteriores, la combinación perfecta para disponer de una aplicación completa que pueda resaltar de sus rivales en el mercado.

Una de esas funcionalidades sería el *modo coche*. Como su nombre indica, estará destinada a mejorar la experiencia de la conducción y brindará diferentes posibilidades, como por ejemplo poder enviar SMS automáticos a personas que hayan llamado mientras se conducía, rechazar llamadas entrantes o activar el BLUETOOTH.

La otra funcionalidad sería el *perfil de sonidos*, que ofrecerá la posibilidad de administrar los sonidos del teléfono de acuerdo a una franja horaria introducida por el usuario, además de poder mandar SMS predefinidos, entre otras opciones.

Con Easy Phone Manager no sólo se pretende crear una aplicación como ya se ha dicho, útil y fiable, sino una aplicación que destaque de las demás existentes. Obviamente existen aplicaciones similares, pero puede conseguir diferenciarse al ser una aplicación que incorporará las mejores funcionalidades de cada una de sus competidoras.

Esta diferenciación será explicada con mayor detalle en una sección futura, la sección **2. Estudio del mercado.**

Así mismo, las ideas iniciales, los primeros pasos a la hora de realizar la aplicación, la especificación de la misma, bocetos realizados y otras aspectos más, vienen contemplados en la sección **3. Especificación**.

Los recursos disponibles para el desarrollo de cada una de las partes de la aplicación, así como la elección y justificación de los mismos, se llevan a cabo en la sección **4. Diseño**.

Los pasos realizados a la hora de desarrollar la aplicación, así como la programación de la misma, los problemas que se hayan tenido que afrontar para ofrecer determinadas funciones o los cambios que se han tenido que efectuar en la aplicación (ya sea por problemas en el desarrollo, problemas de coherencia, etc.), se ven reflejados en la sección **5. Implementación.**

En la sección **6. Resultados** se muestra por una parte el resultado final que se ha logrado mediante la inclusión de capturas de pantalla de cada una de las partes de aplicación, explicando también los cambios realizados respecto a la idea inicial. Por otra parte se incluye una serie de resultados obtenidos mediante la comprobación de su funcionamiento en diferentes dispositivos.

Por último, en la sección **7. Conclusiones** se lleva a cabo un pequeño repaso de todo lo que ha consistido la aplicación desarrollada así como posibles mejoras o propuestas de cara a una versión futura.

2. Estudio del mercado

En esta sección se van a estudiar las diferentes plataformas móviles disponibles en la actualidad y se va proceder a la elección de una de ellas como plataforma destino para la aplicación a desarrollar de acuerdo a una serie de criterios que también se explicarán a lo largo de la sección.

Por otra parte se procederá a analizar las diferentes aplicaciones en el mercado que puedan ser una competencia directa de la aplicación a desarrollar y se hará una comparativa entre todas ellas para valorar que funcionalidades resultan de interés y qué otras puede ser prescindibles.

2.1 Introducción

Al disponer de unos teléfonos Smartphone con capacidades tan potentes (acceso a internet, correo, juegos, etc.) así como componentes muy evolucionados (teclado táctil, acelerómetro, etc.), sería lógico pensar que existiera un software capaz de aprovechar todo este potencial. Es por ello que en la actualidad oímos hablar de tiendas de aplicaciones móviles como el App Store de la compañía Apple o el Play Store de Google.

Estos servicios destacan por ofrecer una cantidad inmensa de aplicaciones, que dependiendo del desarrollador, pueden ser de pago o gratuita y que aportan un valor añadido a nuestro dispositivo móvil, ya sea explotando alguna de sus cualidades o añadiendo nuevas que no tenía.

Algunas muestras de estas aportaciones: el retoque fotográfico. Con nuestro dispositivo éramos capaces de realizar fotografías pero no disponíamos la posibilidad de poder editarlas. Actualmente existen centenares de aplicaciones que permiten realizar este retoque de una manera casi profesional. Otro ejemplo sería la navegación GPS. Los Smartphone disponían de un sistema de posicionamiento GPS, pero gracias a aplicaciones que han sabido aprovechar ese recurso podemos viajar hacia cualquier destino deseado, siendo guiados en todo momento por el terminal.

Estos ejemplos advierten de la inmensa cantidad de aplicaciones que existen hoy día para los dispositivos móviles. Por ello, desde el punto de vista del desarrollador existe una serie de cuestiones a plantearse antes de ponerse a trabajar en cualquier aplicación ya que, de no tenerse en cuenta, podría suponer un fracaso a nivel de ventas y reputación.

La primera de las cuestiones sería: ***¿Qué se puede ofrecer que sea diferente o innovador de lo ya existente?***

Cuando existe una competencia tan elevada se puede enfocar las aplicaciones de diversas maneras. O bien crear una aplicación que sea totalmente diferente de lo ya existente, es decir, que sea completamente novedosa en el mercado y que no haya ninguna que se le parezca, o de lo contrario, crear una aplicación que mejore algo conocido, por medio de la combinación de varias aplicaciones o simplemente ofreciendo un servicio mejor que las aplicaciones que tienen la misma finalidad que la que se quiere diseñar.

En el caso de la aplicación que se va a abordar en este proyecto, aunque ya se ha comentado en líneas generales en la *introducción*, se ha optado por la creación de una aplicación que tiene una competencia similar en el mercado, pero a diferencia de estas, se le ha dotado de una serie de funcionalidades extra que la hacen algo más particular, mucho más potente y completa.

Otra de las cuestiones de importancia y que se va a tratar en el siguiente punto, sería: ***¿En qué plataforma o a qué tipo de mercado queremos ofrecerla?***

2.2 Estudio de las plataformas móviles actuales

En este apartado se va hacer un repaso de las características más importantes que ofrecen las diferentes plataformas móviles existentes en el mercado: *IOS* [4], *Android* [5], y *Windows Phone* [6]. De esta manera se podrá determinar con un mejor criterio cuál de ellas resulta más conveniente para la aplicación a desarrollar.

A continuación se expone en *Tabla 2.1*, de manera rápida y visual, los diferentes aspectos de cada una de las plataformas que se van a tener en cuenta para la decisión final de la misma:

Tabla 2.1 : Comparativa entre las diferentes plataformas móviles

	PLATAFORMA		
CARACTERÍSTICA	**Windows Phone 8.1**	**Android 4.4**	**IOS**
Adobe Flash	*No - Disponible en app de pago*	*Si*	*No*
Compartir información en redes sociales	*Si*	*Si*	*SI*

Sistema multi-tarea	*Si - Limitada a ciertas aplicaciones*	*Si*	*SI*
Navegador Web	*Si*	*Si*	*Si*
Suite Ofimática	*Si*	*Si*	*Si*
Notificaciones	*Si*	*Si*	*Si*
Bluetooth	*Si*	*Si - Compatibilidad universal*	*Si - Sólo con dispositivos compatibles*
NFC	*Si*	*Si*	*Si – Limitado a ciertas aplicaciones*
Personalización de la interfaz	*Si - Limitado*	*Si*	*Si - Limitado*
Mapas con navegación GPS gratuita	*Si - Maps*	*Si - Navigation*	*No*
Tienda de aplicaciones	*Si - Windows Phone Store*	*Si - Play Store*	*Si - App Store*
Gestor de correo	*Si*	*Si*	*Si*

A primera vista, podría decirse que no existe mucha diferencia entre una plataforma u otra y eso puede deberse a que, existe tal competencia entre ellas que les lleva a intentar evolucionar y plantar cara a sus principales competidores. De manera que, si una de ellas ofrece un servicio que está teniendo mucha acogida entre los usuarios, al poco tiempo sus competidores tratan de ofrecer el mismo o mejor servicio.

A pesar de que ofrezcan prácticamente las mismas características, siempre hay detalles que hacen que un usuario, ya sea usuario común o desarrollador de aplicaciones, se decante por una de las plataformas.

Desde el punto de vista del desarrollador se va a tener en cuenta la acogida del usuario final, ya que el objetivo principal de una aplicación es que sea útil y sea utilizada por la mayor cantidad de usuarios posible.

Un usuario estándar lo más probable es que se decante por una plataforma móvil que ofrezca una gran variedad de aplicaciones, como pueden ser aplicaciones de ocio (gran catálogo de juegos, entretenimiento, etc.), aplicaciones de interés social (redes sociales, periódicos, mensajería, etc.) o aplicaciones específicas para una determinada necesidad. Además de ello, si la mayoría de aplicaciones anteriormente mencionadas pueden encontrarse de manera gratuita, es un aliciente más que suficiente para la elección del usuario.

Por tanto, las plataformas que podrían incluirse dentro de esa serie de requisitos serían las dos grandes competidoras en el mercado hasta el momento, ***Android*** e ***IOS***, ya que Windows Phone a pesar de estar progresando, necesita una mayor propagación para poder estar a la altura de sus competidoras puesto que su mercado de aplicaciones, *Windows Phone Store,* cuenta con unas cifras aproximadas de 300.000 aplicaciones frente a las 1.300.000 que ofrece *Google Play* de Android o las 1.200.000 que ofrece *Play Store* de Apple, como puede comprobarse en la *Figura 2.1.*

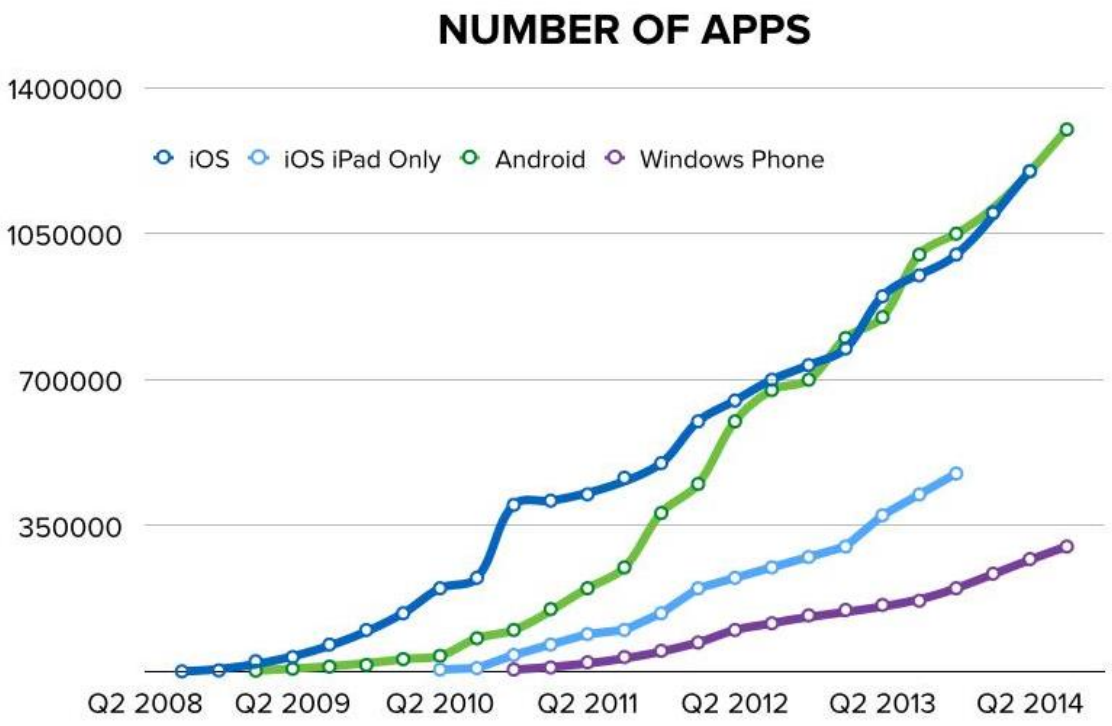

Figura 2.1: Crecimiento de las aplicaciones según la plataforma móvil

Otra cuestión de importancia para la elección de la plataforma suele ser el grado de personalización que ofrece cada una ellas sobre los dispositivos móviles. Por una parte, Android ofrece la posibilidad de personalizar casi la totalidad de la interfaz gráfica de su sistema operativo, es decir, poder cambiar iconos, fondos de escritorio, pantalla de bloqueo etc.

Por otra parte, iOS no ofrece apenas personalización ya que se trata de una plataforma más limitada y restringida, hecho que se queda en el olvido y se contrarresta de alguna manera gracias a su interfaz de aspecto muy conseguido y atractivo. En la *Figura 2.2* se muestra el aspecto de cada una de las interfaces.

Figura 2.2: Interfaces de usuario, de izq. a der. IOS8, Android 4.4, Windows Phone 8.1

Teniendo ahora en cuenta el punto de vista del desarrollador, se ha visto que ambas plataformas, tanto Android como iOS, podrían ser perfectas candidatas además de que poseen millones de usuarios a su disposición.

Sin embargo, existen una serie de razones que podrían decantar la balanza a favor de una plataforma u otra, como por ejemplo, **¿cuál de las plataformas está teniendo un mayor éxito en el mercado?** Para ello, se analizará la cuota de mercado actual a partir del siguiente gráfico, *Figura 2.3,* que recoge información sobre la cuota de mercado actualizada a Septiembre del año 2014.

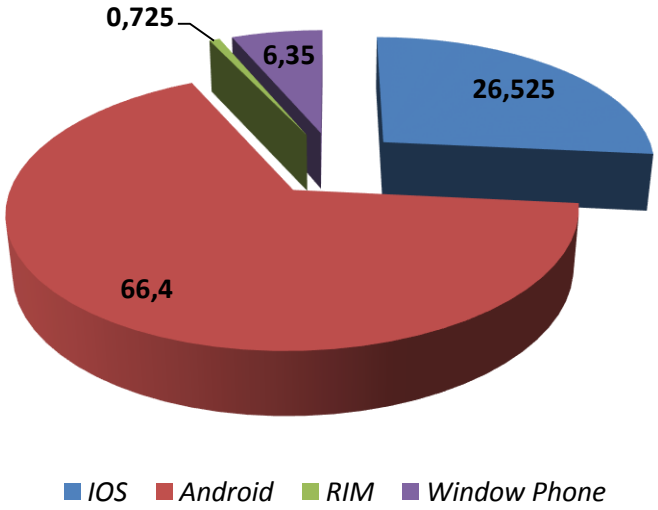

Figura 2.3: Cuota de mercado de Smartphone a primer cuatrimestre de 2014

A partir del gráfico de la *Figura 2.3*, se podrían obtener las conclusiones de que en número de ventas de dispositivos Android está muy por encima de iOS. Esto no es un hecho aislado o una casualidad, pues bien es cierto que Android, desde que se dio a conocer en el mundo de la tecnología móvil, ha ido creciendo año tras año tanto en número de ventas como en número de distribuciones de su sistema operativo en gran cantidad de marcas de dispositivos móviles.

Los siguientes gráficos, *Figura 2.4* y *Figura 2.5,* muestran a pequeña escala, la proyección comentada anteriormente del sistema operativo Android, durante los años 2012 y 2013 respectivamente.

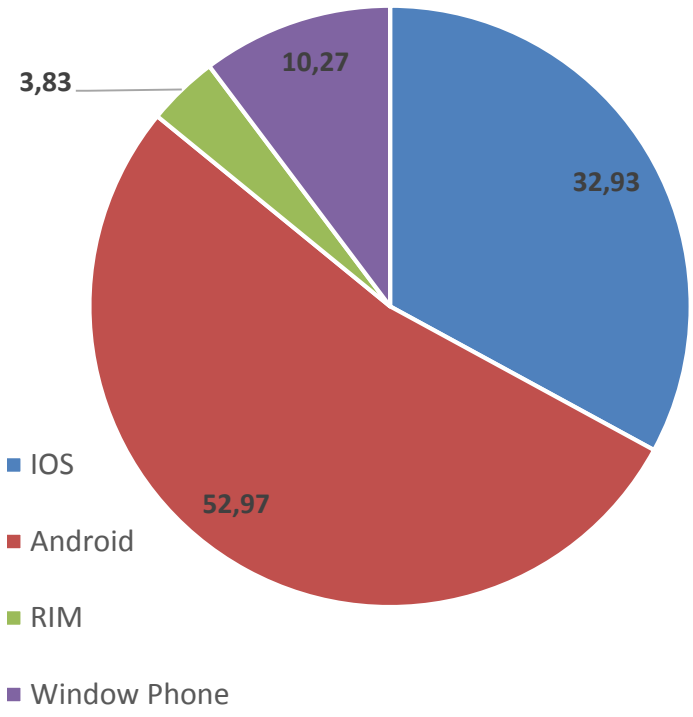

Figura 2.4: Cuota de mercado a junio de 2012

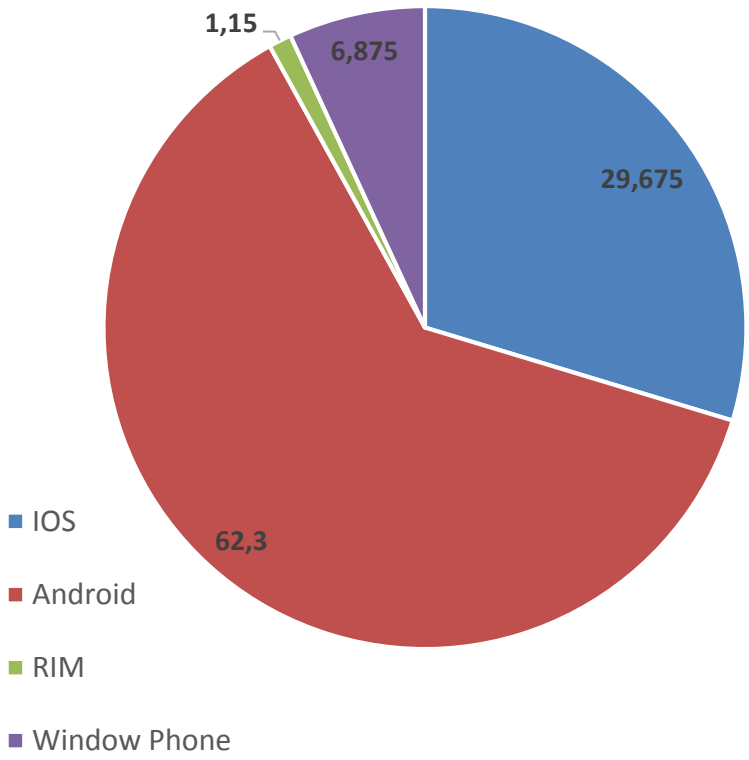

Figura 2.5: Cuota de mercado a agosto de 2013

Los datos de los gráficos anteriores pertenecen a estadísticas recogidas por Kantar Worldpanel [7], y recogen información de todo Estados Unidos, Europa, China y Australia. A pesar de no englobar estadísticas a nivel mundial, son datos más que suficientes para poder extraer las conclusiones que se han necesitado.

Así pues, desde un punto de vista comercial de la aplicación, la plataforma Android sería la elegida ya que, habiendo casi el doble de usuarios utilizando dispositivos con esta plataforma, el número de usuarios finales a los que se les puede hacer llegar la aplicación o aplicaciones desarrolladas debería ser mayor, aunque sólo en términos estadísticos, dado que en la práctica influyen muchos otros factores para que una aplicación resulte exitosa: una buena publicidad, que sea una aplicación útil, cosmopolita, bonita, fácil de usar, etc.

Otra consideración a tener en cuenta a parte de todo lo mencionado anteriormente, sería las facilidades que ofrece cada una de las plataformas a los desarrolladores para la creación de aplicaciones. Tanto Android como IOS, cuentan con una serie de herramientas de desarrollo

muy completas, una gran cantidad de APIs, librerías, etc., además de información detallada y documentada con ejemplos de todo tipo.

Sin embargo a nivel personal, Android resulta una plataforma más libre y de fácil acceso que iOS. Esto es así ya que, para la programación de aplicaciones Apple se requiere de un software específico. No es estrictamente necesario disponer de un ordenador Apple para su desarrollo pero de lo contrario, habría que recurrir a la utilización de máquinas virtuales para poder simular el entorno iOS y por tanto el rendimiento y productividad se verían notablemente afectados.

Por el contrario, la programación Android es multiplataforma (Mac, Windows, Linux) y permite poder desarrollar aplicaciones con varios software de desarrollo, (Android Studio, Eclipse, etc.) Además de ello, cualquier dispositivo físico con sistema operativo Android a nivel de costes siempre va a ser notablemente más bajo que cualquier dispositivo de la marca Apple.

A parte del tema económico y la accesibilidad de las herramientas para el desarrollo de aplicaciones también resulta un punto importante el lenguaje de programación mediante el cual se desarrollan las mismas.

Normalmente, también debido al apartado económico mencionado anteriormente, los programadores que se inician en el mundo del desarrollo de aplicaciones móviles lo hacen con mayor frecuencia para la plataforma Android. El lenguaje de programación de las apps de este sistema operativo es Java [8]. Se trata de un lenguaje muy común y conocido que se suele enseñar en las universidades por su amplia utilización, no sólo en el desarrollo de apps.

Por otro lado, el lenguaje de programación de apps para iOS es el Objective-C [9], que resulta más peculiar y cuya finalidad reside únicamente en el desarrollo de aplicaciones por lo que normalmente es algo más desconocido para los desarrolladores.

Con todo lo mencionado en esta sección sobre la elección de la plataforma móvil, ha quedado claro que Android ha sido la elegida para abordar este proyecto.

A modo de resumen, las razones que han influido a tomar esa decisión han sido:

- Mayor número de usuarios (cuota de mercado).
- Costes de desarrollo más bajos.
- Lenguaje de programación conocido previamente.

2.3 Comparativa con otras aplicaciones

Es difícil que hoy día, con la cifra aproximada de 1.300.000 aplicaciones en el Play Store de Android, no existan algunas aplicaciones que se asemejen bastante a la desarrollada en este proyecto, *Easy Phone Manager*.

En esta sección se va a realizar una comparativa entre las aplicaciones más influyentes que tengan características similares a *Easy Phone Manager* y que han podido servir de inspiración para la elaboración de la misma.

2.3.1 Green Power Free Battery Saver

La primera aplicación a analizar es *Green Power Free Battery Saver*. Se encuentra disponible de forma gratuita en el mercado de aplicaciones Play Store.

Link de descarga de la aplicación:

https://play.google.com/store/apps/details?id=org.gpo.greenpower&hl=es

Green Power Free Battery Saver es una aplicación destinada al ahorro de energía del dispositivo móvil, con la clara finalidad de aumentar la vida útil del terminal entre carga y carga. Este ahorro de batería lo consigue mediante la gestión automática de las conexiones WIFI, de los datos móviles y del BLUETOOTH. También dispone de un modo de ahorro de batería predeterminado, nombrado como *modo día,* que posee una serie configuraciones iniciales que el usuario puede modificar si lo desea.

Green Power Free Battery Saver ofrece al usuario una versión de pago llamada *Green Power Premium,* disponible en el Play Store por 3,90 €, que incluye una serie de características que no están disponibles en su versión gratuita.

Algunas de esas características son:

- Modo “Nocturno”.
- Widget de la aplicación.
- Gestión del WIFI según localización.
- Eliminación de los anuncios.

Así pues, atendiendo únicamente a la parte gratuita de la aplicación, las principales funciones que ofrece son:

- Indicador de vida útil de la batería en la barra de notificaciones.
- Gestión WIFI automática según el bloqueo de pantalla del terminal.
- Gestión automática de los datos móviles según el bloqueo de pantalla del terminal.
- 20 idiomas disponibles.
- Compatible con todas las versiones de Android.

A continuación en la *Figura 2.6* se puede apreciar el aspecto de la interfaz gráfica de la aplicación así como alguna de sus funcionalidades.

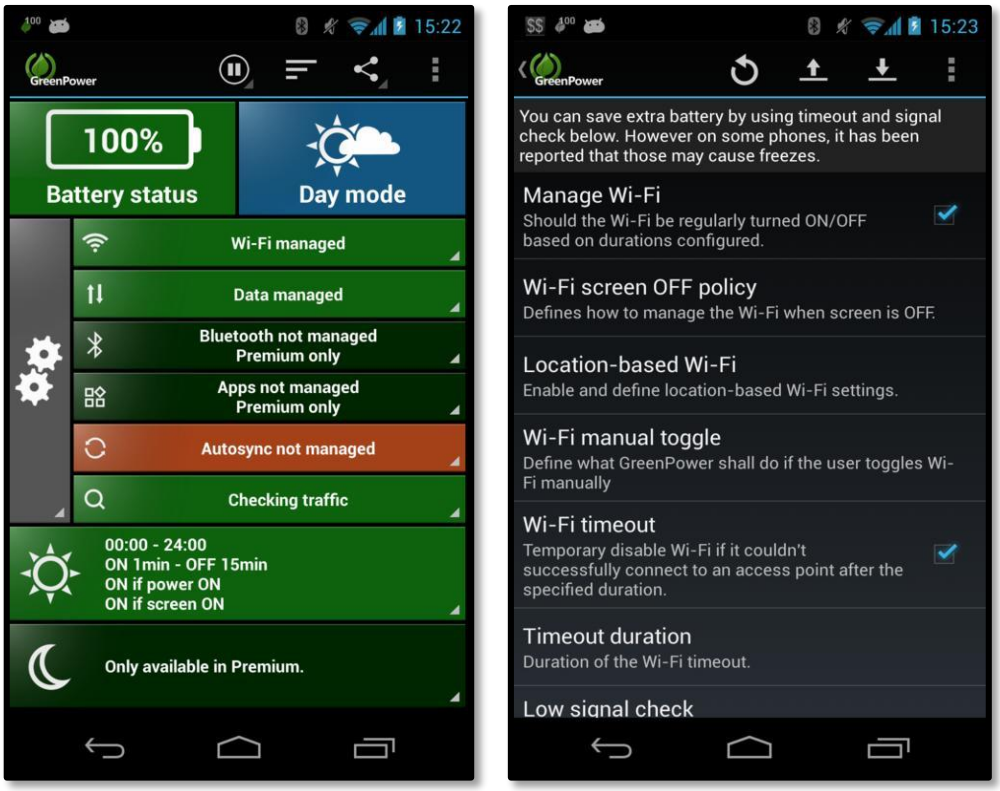

Figura 2.6: A la izq., interfaz de usuario de de la app Green Power Save Free y la derecha funciones para administrar el WIFI

2.3.2 Agent

Agent es una aplicación que pretende dotar de inteligencia a nuestro dispositivo para que pueda realizar de forma automática una serie de funcionalidades destinadas a mejorar la calidad de vida del usuario.

Se puede encontrar en el Play Store de forma gratuita, link de la aplicación:

https://play.google.com/store/apps/details?id=com.tryagent&hl=es

Agent sorprende sobre todo con su interfaz, puesto que ofrece un diseño muy atractivo y sencillo, que permitirá al usuario moverse con facilidad y fluidez por el amplio abanico de funcionalidades que dispone. Puede verse su aspecto en la Figura 2.7, donde se muestra a parte de su interfaz de usuario, una de las funcionalidades que ofrece: el modo coche.

Así pues, las diferentes características que ofrece *Agent* están divididas en cinco bloques claramente diferenciados:

- **Ahorro de batería:** destinado a alargar la duración de la batería del dispositivo. Ofrece la posibilidad de escoger un nivel de carga y elegir el comportamiento que

adoptará el WIFI, el BLUETOOTH, el brillo, etc. cuando el terminal se encuentre por debajo de ese nivel.

- **Un modo coche:** destinado a facilitar determinadas acciones durante la conducción. Como por ejemplo, leer SMS a través del altavoz, responder a llamadas dentro de una lista de contactos personalizable, mandar SMS predefinidos, etc.
- **Modo aparcamiento:** con la finalidad de guardar en un mapa la posición de donde se ha estacionado el coche.
- **Modo reunión:** posibilidad de silenciar o poner en vibración el teléfono durante ciertos días definidos por el usuario mediante un calendario, así como la posibilidad de enviar SMS a determinadas personas.
- **Modo dormir:** pensado para que el terminal no emita ningún tipo de sonidos que puedan molestar al usuario mientras duerme. Permite definir una lista de personas a las que el modo dormir no le afecta.

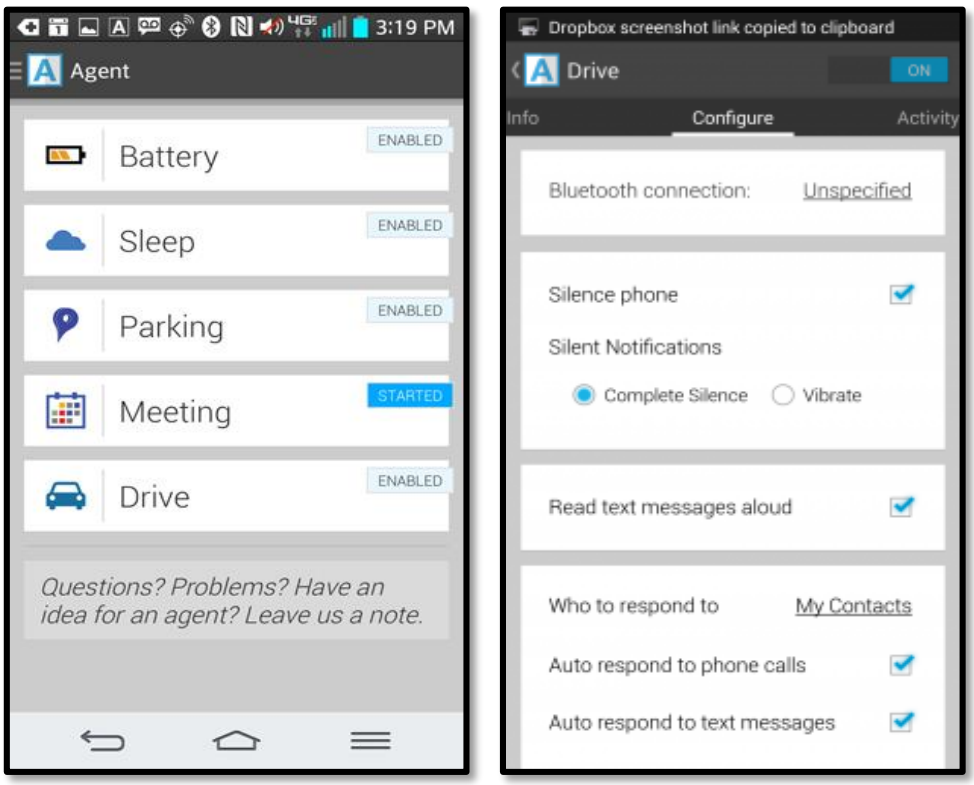

Figura 2.7: A la izq., intefaz de usuario de la app Agent y la der. su función de "modo coche"

2.3.3 GO Battery Saver & Power Widget

Se trata de una aplicación que tiene gran influencia en el mercado, en concreto posee entre 10.000.000 - 50.000.000 descargas, gracias a la popularidad de su launcher para Android, "*GO Launcher*". Link de descarga:

https://play.google.com/store/apps/details?id=com.gau.go.launcherex.gowidget.gopowermaster&hl=es

GO Battery Saver & Power Widget surge como complemento o como aplicación adicional del launcher comentado, proporcionando una estética muy atractiva y un widget muy cómodo que facilita al usuario la configuración y uso de la aplicación en unos pocos pasos.

Esta aplicación se centra puramente en el ahorro de batería del dispositivo, a diferencia de otras aplicaciones que pueden añadir alguna funcionalidad extra al margen del ahorro de batería, y ofrece diversas configuraciones o planes de ahorro de energía que se adaptan a la situación del usuario.

Estos planes de ahorro se dividen según la necesidad de ahorro, es decir, son modos de ahorro que van desde una configuración muy simple para ahorrar levemente la batería, hasta planes muy estrictos que desactivan multitud de procesos y servicios para ahorrar al máximo, (*general save*, *super power*, *ultimate power*).

Así pues, las características principales que ofrece esta aplicación son:

- Widget atractivo e intuitivo, con una interfaz personalizable.
- Proporciona una estimación del tiempo de batería restante con el plan de ahorro actual, así como una estimación en caso de escoger cualquier otro.
- Un abanico de planes de ahorro que se adaptan a diferentes situaciones que pueda necesitar el usuario.

Se muestra a continuación dos capturas de pantalla de la aplicación, en la *Figura 2.8*, donde se aprecia la interfaz de usuario y los modos de ahorro de batería.

Figura 2.8: A la izq., interfaz de usuario de la app Go Power Saver y a la der. función "ahorro de batería"

2.3.4 Easy Battery Saver

Es también, junto a muchas otras, una de esa aplicaciones que centra todo su funcionamiento en conseguir alargar la vida útil de la batería del terminal entre carga y carga. Al igual que las demás comentadas hasta el momento, es de disfrute gratuito.

Link de descarga de la aplicación:

https://play.google.com/store/apps/details?id=com.easy.battery.saver

Easy Battery Saver utiliza un diseño muy sencillo e intuitivo, basado en cinco modos diferentes de ahorro de energía organizados de menos a más, según su influencia en la optimización.

La interfaz también incorpora un visualizador para prever la duración de la batería en caso de seleccionar cualquiera de los modos de ahorro, hecho que se puede observar en la *Figura 2.9*.

Resulta una aplicación ligera y muy sencilla, pero con gran potencial en la tarea a la que está orientada. También ha tenido mucha influencia en el mercado, ya que tiene entre 10.000.000 – 50.000.000 descargas con votos muy positivos de los usuarios.

Resumiendo, las características principales de *Easy Battery Saver son*:

- Buen funcionamiento y rendimiento.
- Interfaz clara y sencilla.
- Varios modos de ahorro de energía, permitiendo al usuario escoger el que más necesite.
- Previsión de la duración de la batería con cada modo de ahorro

Figura 2.9: A la izq., interfaz de usuario de la app Easy Battery Saver y a la der. perfiles de ahorro de batería

2.4 Conclusiones obtenidas

Estas cuatro aplicaciones que se han expuesto brevemente son un ejemplo de las más representativas (número de descargas, opiniones…) en cuanto a ahorro de batería y funcionalidades extra orientadas a la gestión y automatización del dispositivo móvil.

A continuación se muestra una tabla-resumen comparando de manera breve y concisa las cuatro aplicaciones, según las características y funcionalidades que ofrecen.

Tabla 2.2 : Comparativa de las aplicaciones Green Power Saver, Agent, Go Battery Saver & Power Widget y Easy Battery Saver

	APLICACIONES			
FUNCIONALIDADES Y CARACTERISTICAS	**Green Power Saver**	**Agent**	**GO Battery Saver & Power Widget**	**Easy Battery Saver**
Interfaz clara e intuitiva	*No*	*Si*	*Si*	*Si*
Widget	*No*	*No*	*Si*	*No*
Gestión wifi y datos según bloqueo	*Si*	*No*	*No*	*No*
Gestión wifi según conexión disponible	*Si*	*No*	*No*	*No*
Ahorro de batería	*Si*	*Si*	*Si*	*Si*
Varios perfiles de ahorro de batería	*No*	*No*	*Si*	*Si*
SMS automáticos	*No*	*Si*	*No*	*No*
Administrar sonidos según día y hora	*No*	*Si*	*No*	*No*
Responder llamadas automáticamente	*No*	*Si*	*No*	*No*
Seleccionar una lista de contactos en los que influirán ciertas acciones (permitir llamadas, responder llamadas, mandar SMS, etc.)	*No*	*Si*	*No*	*No*

Así pues, echando un vistazo a la *Tabla 2.2* se puede observar que existen tres "grupos" diferenciados de aplicaciones.

En primer lugar, *Green Power Saber* estaría en el grupo de aplicaciones que ofrecen una mejor gestión de las conexiones, tanto del WIFI como de los datos móviles ya que permite automatizarlas según el bloqueo de pantalla o según si existe alguna red disponible. Son automatizaciones interesantes ya que permiten al usuario poder elegir la que más se adapte a sus necesidades y además reducir el consumo de batería.

En segundo lugar, estarían las aplicaciones *GO battery saver & Power Widget* y *Easy Battery Saver*. Ambas son aplicaciones destinadas estrictamente el ahorro de energía y por lo tanto tienen características muy similares. Permiten la modificación de las conexiones (WIFI y datos) atendiendo únicamente a los diferentes perfiles y niveles de batería, pero no permite una administración tan completa como la que ofrece *Green Power Saver*.

Por último, el tercer grupo lo cerraría la aplicación *Agent*. Se diferencia de las demás al ser una aplicación que no solo contempla una serie de opciones, por escasas que sean, de ahorro de batería, sino que también añade una serie de funcionalidades realmente interesantes como el "modo coche", con el envío de SMS predefinidos, posibilidad de aceptar llamadas automáticamente, etc. Son una serie de funciones extra que se convierten en candidatas perfectas para ser complementarias del ahorro de batería y poder formar una aplicación más completa, que no se centre únicamente en el ahorro.

Analizando la tabla anterior surge la idea de crear una aplicación que intente englobar lo mejor de cada una de ellas y así crear una aplicación que pueda ser "única" y tenga cabida en el mercado de aplicaciones.

A continuación se plantea cuáles de las características analizadas resultan de interés para el desarrollo de este proyecto, cuáles no lo resultan y qué otras funciones se podrían añadir que no se han visto hasta el momento.

Todas las aplicaciones analizadas presentan una función de ahorro de batería que además es similar, el terminal realiza una serie de cambios cuando el nivel de batería está por debajo de un cierto nivel de carga, como desconectar el WIFI, los datos móviles o bluetooth. Aunque se trate de una característica que ya existe y que todas las aplicaciones incorporan, la propuesta a desarrollar, ahora en adelante *Easy Phone Manager*, no puede quedarse atrás y deberá incluir esa característica ya que es básica en este tipo de aplicaciones. De no ser así probablemente muchos de los usuarios que busquen un ahorro de batería decidirían rechazar *Easy Phone Manager* de inmediato.

Por otra parte, todas las aplicaciones en general, y aún más este tipo de aplicaciones donde hay tantas opciones seleccionables, deben de presentar una interfaz de usuario muy sencilla e intuitiva, ya que de lo contrario puede resultar muy cargante y pesada a la vista de los usuarios. Todas las anteriores excepto *Green Power Saver* han tenido en cuenta esta consigna. *Easy Phone Manager* también deberá incluir una interfaz de estas características.

Se analizan ahora funcionalidades más particulares y peculiares, como por ejemplo la gestión automática de las conexiones WIFI y datos móviles. Este tipo de característica sólo la incluye *Green Power Saber* y cierto es que resulta realmente interesante dar la posibilidad al usuario de que pueda, por ejemplo, activar y desactivar el WIFI dinámicamente según si el móvil se encuentra bloqueado o no. Este tipo de funciones ayudan además a la reducción del consumo de energía, por lo que se convierten en unas candidatas perfectas para dotar a *Easy Phone Manager* de un mayor abanico de posibilidades.

Es precisamente este tipo de opciones relacionadas con la conexión WIFI, lo que inspira a añadir unas funcionalidades que no ofrecen ninguna de las aplicaciones vistas y que podrían ser muy útiles y necesarias. Una de ellas es añadir la posibilidad de habilitar el WIFI cuando se llega a cierta localización geográfica, mediante el uso controlado del GPS. Este tipo de función la incluye *Green Power Saver* pero en su versión de pago.

La otra funcionalidad también relacionada con la conexión WIFI consiste en poder deshabilitarlo cuando no existe conexión alguna, como por ejemplo cuando el usuario sale de su casa y se aleja perdiendo la conexión con su router.

Este par de funcionalidades notablemente novedosas y útiles de cara a los usuarios, dotarían a *Easy Phone Manager* de una mayor potencia, que junto a otros factores podrían hacer que se diferenciara de sus competidoras.

Como se ha comentado anteriormente, la aplicación *Agent* innova añadiendo una serie de características ajenas al ahorro de batería, pero que son un complemento interesante para hacer una aplicación más compacta y que pueda llegar a un mayor número de usuarios. Así pues, haciendo acopio de esa innovación, resulta de gran interés dotar a *Easy Phone Manager* de una mayor versatilidad, añadiéndole funciones similares a “modo coche” y al “modo dormir” que incluye *Agent*.

Una de las cosas que no se han considerado influyentes para tener en cuenta es la posibilidad de elegir entre varios modos de ahorro de energía. Por una parte, *Easy Phone Manager* no sólo se va a centrar en el ahorro de energía, por lo que se ha decidido que no se debe hacer excesivo hincapié en ese apartado. Por otra parte, si se dispone de un modo de ahorro de energía completamente personalizable por los usuarios, resulta poco útil ofrecer otros modos más en los que las opciones son prácticamente idénticas. Por tanto, *Easy Phone Manager* dispondrá de un solo modo de ahorro de energía que el usuario podrá modificar con las diferentes opciones de optimización.

Esas son las características que se consideran de interés por lo que *Easy Phone Manager* conseguirá unificar cada una de ellas, además de poseer una interfaz gráfica sencilla e intuitiva que no resulte complicada y pesada a los usuarios finales.

Con todo esto, *Easy Phone Manager* se convertirá en una aplicación que no pasará inadvertida en el mercado ya que engloba las mejores funcionalidades de las aplicaciones que son del mismo tipo.

3. Especificación

En la actual sección se va a presentar la especificación de la aplicación. Se desarrolla un modelo a tres capas mediante el cual se muestran las diferentes funcionalidades que posee la aplicación (capa de negocio), así como la gestión de la información que circula por la misma (capa de datos) y el aspecto que visualizan los usuarios para su interacción y consiguiente obtención de resultados (capa de presentación).

3.1 Capa de negocio

La capa de negocio de la aplicación es donde reside toda la computación de la misma. Esta capa es la encargada de recibir las peticiones por parte del usuario y hacer los cálculos pertinentes para poder devolverle una respuesta [10].

Para exponer la capa de negocio se presentará a continuación los casos de uso de la misma, donde se verán reflejadas las diferentes funcionalidades que los actores de la aplicación podrán desempeñar.

3.1.1 Actores de la aplicación

Los actores representan un tipo de usuario del sistema. Se entiende como usuario cualquier cosa externa que interactúa con el sistema demandando una funcionalidad. No tiene por qué ser un ser humano, puede ser otro sistema informático o unidades organizativas o empresas [11].

En la aplicación desarrollada existe un único tipo de usuario:

- **Usuario estándar:** podrá hacer uso de la aplicación mediante su dispositivo móvil. Tendrá acceso a la totalidad de las funcionalidades ofrecidas sin excepción alguna. La aplicación se desarrolla de manera que cualquier usuario pueda utilizarla sin diferenciar entre ellos, por lo que se considera que sólo existe un actor que interactúa con la aplicación.

3.1.2 Casos de uso

Un caso de uso es una descripción de los pasos o las actividades que deberán realizar los actores para llevar a cabo algún proceso interactuando con el sistema [12].

Atendiendo a los actores que participan en la aplicación desarrollada, un caso de uso general sería el representado en la *Figura 3.1*:

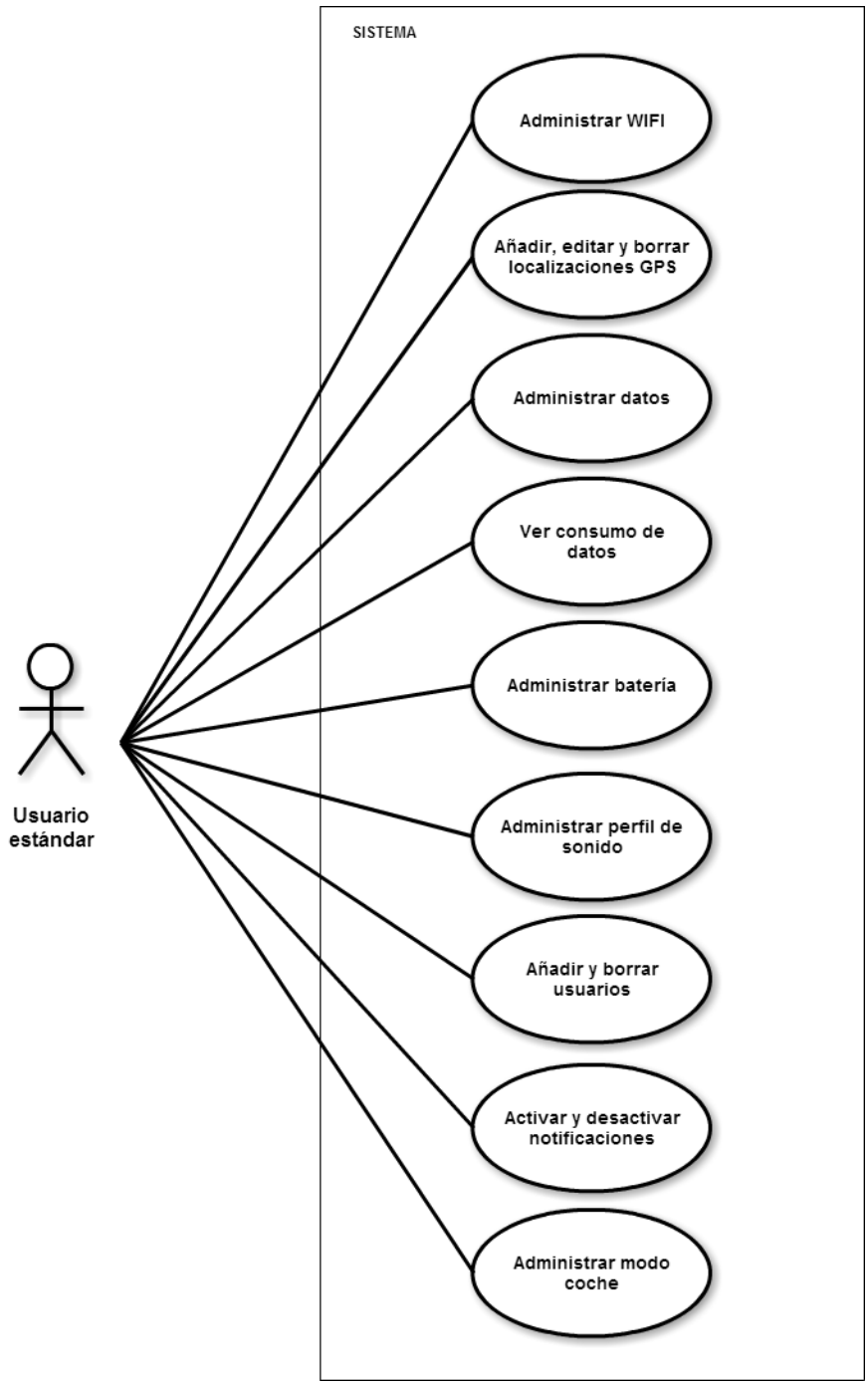

Figura 3.1: Caso de uso general

A continuación se plantearán los casos de uso más relevantes que el usuario estándar podrá realizar interactuando con la aplicación.

3.1.2.1 Administrar el WIFI

Cualquier usuario podrá desplazarse a la pantalla de administración del WIFI. Una vez allí podrá seleccionar, de entre una serie de opciones, el comportamiento que quiera que adopte el WIFI del dispositivo.

Las opciones de administración y gestión WIFI disponibles son:

- **Administrar WIFI según bloqueo:** los usuarios podrán seleccionar si activar o no esta opción, la cual activará el WIFI cuando el terminal esté desbloqueado y lo desactivará cuando pase a estar bloqueado.
- **Administrar WIFI según nivel de batería:** los usuarios podrán decidir si activar o no la opción que desactivará el WIFI cuando el nivel de batería del dispositivo esté por debajo de un nivel crítico seleccionado previamente.
- **Administrar WIFI según conexiones disponibles:** los usuarios tendrán la posibilidad de elegir entre una lista de rangos temporales. Así pues, una vez seleccionada la opción y pasado el rango temporal, si el dispositivo no dispone de una red WIFI a la que poder conectarse automáticamente, el WIFI del mismo se desconectará.
- **Administrar según GPS:** los usuarios podrán activar la opción para poder automatizar el WIFI al encontrarse cerca de un punto geográfico que deberá haber sido insertado previamente.

3.1.2.2 Añadir, editar y borrar localizaciones GPS

Como se acaba de mencionar, cualquiera de los usuarios puede administrar el WIFI según una localización GPS que debe haber sido añadida previamente.

Así pues, los usuarios podrán añadir una localización desde la pantalla de administración de localizaciones.

Para poder agregar una localización, será necesario que los usuarios rellenen dos campos:

- **Nombre de la localización:** indica el alias o nombre que el usuario le quiera dar a esa localización GPS con el fin de que sea algo intuitivo en el caso de tener varias localizaciones. Por ejemplo: *mi casa*.
- **Radio:** indica el radio de acción en el cual la localización por GPS surgirá efecto. El valor mínimo y máximo están predefinidos y variará entre cero y mil metros.

Respecto a la modificación de localizaciones, si es que hay alguna añadida, los usuarios tendrán la posibilidad de modificar los dos campos nombrados anteriormente: el nombre o alias de la localización y el radio de acción.

El borrado de localizaciones está también disponible para todos los usuarios. Cuando una localización es eliminada desaparecerá de la aplicación toda información relativa a la misma.

3.1.2.3 Administrar datos

De forma análoga a la administración de WIFI, los usuarios pueden administrar los datos móviles, es decir, la cantidad de megabytes que se consumen por el uso de las aplicaciones.

Las opciones que los usuarios disponen son:

- **Administrar datos según bloqueo:** los usuarios podrán seleccionar si activar o no esta opción la cual activará los datos móviles cuando el terminal está desbloqueado y los desactivará cuando pase a estar bloqueado.
- **Administrar según nivel bajo de batería:** los usuarios podrán decidir si activar o no la opción que desactivará los datos móviles cuando el nivel de batería esté por debajo de un nivel crítico seleccionado.

3.1.2.4 Ver consumo de datos

En la pantalla de administración de datos móviles los usuarios podrán acceder a una vista de gestión. Esta vista es la proporcionada por defecto con el sistema operativo de Android y la información que se muestra contiene los siguientes datos:

- **Ciclo de uso:** indica la franja temporal de la que se mostrará el consumo de datos. Los usuarios podrán variar la franja temporal como deseen.

- **Grafica de datos:** gráfica que muestra de forma visual el consumo producido por los usuarios durante el ciclo de uso.
- **Lista de aplicaciones:** muestra una lista detallada de aplicaciones instaladas en el terminal, ordenadas de mayor a menor consumo (las aplicaciones que no consumen no aparecen en la lista). Los usuarios podrán desde esta lista detener cualquier aplicación al instante.

3.1.2.5 Administrar batería

Los usuarios podrán administrar el uso de la batería y de esta manera lograr alargar su tiempo de vida antes de tener que ponerlo a cargar de nuevo.

Para poder conseguir dicho ahorro dispondrán de una serie de opciones:

- **Nivel de batería:** los usuarios indicarán el nivel de batería que se considerará como nivel crítico a partir del cual el ahorro de batería comenzará a hacerse efectivo. El valor que puede tomar el nivel de batería variará entre cero y cien, ya que es el rango de batería que dispone cualquier terminal.
- **Bloqueo de pantalla:** los usuarios seleccionarán entre una lista de tiempos que variará de entre quince segundos a cinco minutos. Ese tiempo será lo que el terminal tarde en bloquearse una vez que esté por debajo del nivel crítico de batería.
- **Brillo:** los usuarios indicarán el porcentaje de brillo que el dispositivo adoptará cuando la batería del mismo esté por debajo del nivel crítico seleccionado.
- **Bluetooth:** los usuarios indicarán si quieren tener activo o inactivo el bluetooth cuando el terminal esté por debajo del nivel de batería deseado.

3.1.2.6 Administrar perfil de sonido

Los usuarios podrán elegir de entre de una serie de opciones para poder personalizar un perfil telefónico que afectará a la configuración de los sonidos.
Para que el perfil se configure correctamente los usuarios tendrán que rellenar dos campos obligatorios:

- **Desde:** indica la fecha inicial en formato *horas/minutos* en la cual el perfil de sonidos se activará.

- **Hasta:** indica la fecha final en formato *horas/minutos* en la que el perfil de sonidos dejará de estar activo.

Las opciones que los usuarios dispondrán para su personalización son las siguientes:

- **Sonidos de llamadas:** indica el volumen que emitirá el terminal al recibir llamadas telefónicas. El usuario seleccionará entre una lista de opciones que contemplará desactivar sonidos o activarlos al nivel mínimo, medio o máximo.
- **Vibración:** indica el estado de la vibración del terminal. Los usuarios podrán elegir si activar o no esta opción.
- **Filtro de llamada:** indica la persona o personas a las que no les afectará la configuración del perfil. Es decir, si el perfil está configurado para que esté en silencio y llama cualquier persona que se encuentre en el filtro, el terminal emitirá sonidos de llamada.
- **SMS automático:** indica el envío automático de un SMS predefinido que el usuario podrá modificar en cualquier momento. El envío se realizará a todas aquellas personas que hayan realizado una llamada mientras el perfil de sonidos esté activo. Los usuarios podrán elegir si activar o no esta opción.

3.1.2.7 Añadir y borrar usuarios

Como se acaba de explicar en el caso de uso anterior, *administrar perfil*, una de las opciones permite al usuario la posibilidad de añadir un filtro de llamada para que la configuración del perfil no afecte a los contactos seleccionados.

Así pues, los usuarios podrán añadir contactos al filtro desde su terminal, por medio de la agenda tradicional que incorpora Android, a la que serán redirigidos desde la misma aplicación.

Siempre y cuando existan contactos añadidos en el filtro los usuarios tendrán la posibilidad de eliminarlos, borrando así toda la información relativa a los mismos.

3.1.2.8 Activar y desactivar notificaciones

Los usuarios podrán, desde la pantalla de ajustes de la aplicación, activar y desactivar las diferentes notificaciones que ésta incorpora:

- **Notificaciones de GPS:** indicará que el usuario se encuentra dentro de un área de localización GPS guardada y que por lo tanto, la aplicación procederá a la activación automática del WIFI.
- **Notificaciones de ahorro de batería:** indicará que el nivel de batería se encuentra por debajo del nivel crítico seleccionado, por lo que el plan de ahorro de energía se activará.
- **Notificaciones del modo coche:** indicará el número de teléfono al cual se le ha rechazado la llamada mientras el modo coche está activo.
- **Notificaciones de SMS automáticos:** indicará el número de teléfono al cual se le ha mandado un SMS automático, ya sea si la opción activada es la disponible en el modo coche o la del perfil de sonidos.

3.1.2.9 Administrar modo coche

Los usuarios podrán elegir entre una serie de opciones que les ofrecerán una mejor experiencia al volante:

- **Bluetooth:** determina el estado del bluetooth, el usuario decidirá si activar o no el bluetooth cuando el modo coche esté en funcionamiento.
- **Responder llamadas:** el usuario podrá rechazar las llamadas entrantes que se reciban cuando el modo coche esté activo.
- **Silenciar llamadas:** permitirá silenciar la llamada, lo cual no la rechazará, sino que desconectará el sonido de la misma.
- **Contestar llamadas:** la llamada entrante será aceptada automáticamente.
- **Altavoz de llamada:** indica el estado del altavoz cuando se acepta una llamada entrante. El usuario seleccionará si lo desea activar automáticamente.
- **SMS automático:** al igual que en el caso de uso – *administrar perfil* – indica el envío automático de un SMS predefinido por el usuario. El envío se realiza al número de teléfono (móvil) que llame mientras el modo coche esté en activo.

3.2 Capa de presentación

La capa de presentación corresponde a la capa con la que el usuario va a interactuar para poder comunicarse y mandar información a la capa de negocio. A esta capa se le denomina comúnmente interfaz gráfica.

Puesto que va a estar expuesta a los usuarios de la aplicación, debería tener la característica de ser intuitiva y amigable, de modo que no presente una gran dificultad al ser manipulada.

En esta sección se presentará el diseño de la aplicación mediante los bocetos o *mockups* [13] que fueron desarrollados en primera instancia para dar forma a la aplicación final.

3.2.1 Pantalla inicial a la aplicación

Desde un primer momento, el diseño que se planteó para la aplicación se realizó con la finalidad de facilitar su uso por parte de los usuarios. Una interfaz sencilla e intuitiva era lo que se buscaba principalmente. Por ello, se pensó en un diseño *dashboard* para la interfaz gráfica.

Un diseño *dashboard* es un patrón de diseño en el cual se ubica en una misma pantalla todas las funcionalidades principales que dispone la aplicación. De esta manera, el usuario puede conocer de un vistazo las funcionalidades disponibles y acceder a ellas con un solo *click*.

Así pues, una vez decidida la propuesta de diseño, el resultado del boceto sería el presentado en la *Figura 3.2*.

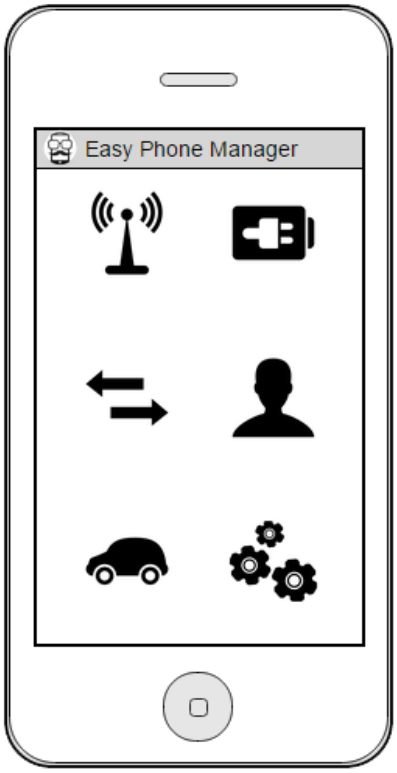

Figura 3.2: Pantalla inicial de la aplicación

Aparecen seis iconos claramente diferenciados asociados a las funcionalidades que representan para que no haya confusión alguna. La relación icono - funcionalidad es la siguiente:

- **Icono de antena** (parte superior izquierda): redirige a la pantalla de administración del WIFI.
- **Icono de batería** (parte superior derecha): redirige a la pantalla de administración del ahorro de batería.
- **Icono de flechas** (parte central izquierda): redirige a la pantalla de administración de los datos móviles.
- **Icono de avatar** (parte central derecha): redirige a la pantalla de administración del perfil de sonidos.
- **Icono de coche** (parte inferior izquierda): redirige a la pantalla de administración del modo coche.
- **Icono de engranajes** (parte inferior izquierda): redirige a la pantalla de los ajustes de la aplicación.

3.2.2 Pantalla de administración WIFI

Una vez pulsado el icono pertinente en la pantalla inicial el usuario será redirigido a la pantalla de administración WIFI, mostrada en la *Figura 3.3*.

El usuario dispondrá de una lista con opciones para poder administrar el WIFI de su terminal de la manera que más se adapte a sus necesidades. Dichas funcionalidades han sido explicadas en la sección anterior, por lo que este apartado se centrará primordialmente en el diseño y navegación de la aplicación.

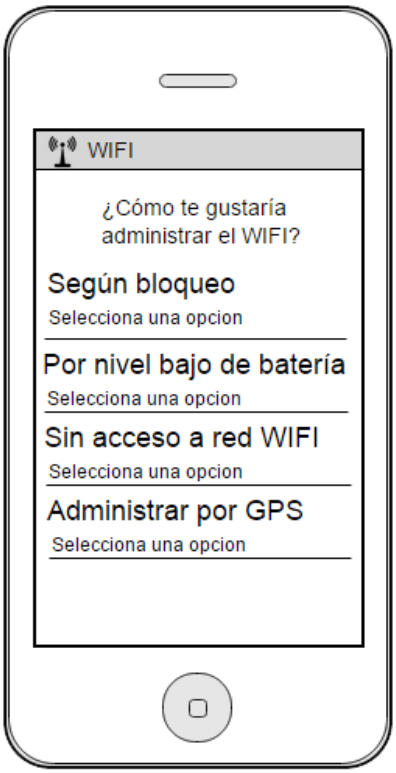

Figura 3.3: Opciones de administración del WIFI

Así pues, el usuario podrá seleccionar cualquiera de las opciones haciendo *click* en ellas. Esto desplegará una ventana de diálogo con información más detallada de la funcionalidad y con la lista de los posibles valores que pueda tomar. Véase el ejemplo en la *Figura 3.4*.

El comportamiento de las demás opciones sería idéntico al que se acaba de mostrar en la *Figura 3.4,* a excepción de la administración WIFI según una localización GPS, que se explicará con más detalle en la sección *3.2.3 Administrar WIFI por GPS*.

Figura 3.4: Ventana de valores disponibles para la opción "Sin acceso a red WIFI"

3.2.3 Administrar WIFI por GPS

Como ya se ha ido explicando a lo largo de esta memoria, el usuario podrá administrar el WIFI automáticamente dependiendo de si está cerca de una localización GPS. La *Figura 3.5* muestra la pantalla a la que el usuario será redirigido cuando active la localización GPS, que es la pantalla de administración de localizaciones.

Figura 3.5: Pantalla de administración de localizaciones GPS

En esta pantalla es donde aparecerán las localizaciones que el usuario haya guardado previamente. Para poder agregar una localización el usuario deberá pulsar el icono con el signo '**+**'. Una vez pulsado, será redirigido a otra pantalla en la cual aparecerá el mapa geográfico que ofrece los servicios de Google para que el usuario pueda elegir el punto GPS que desee.

Ya redirigido a la pantalla del mapa geográfico el usuario podrá utilizar una barra de búsqueda situada en la parte superior de la aplicación para buscar la localización que se desee (como ya se ha comentado, el mapa utiliza los recursos de Google, por lo que la búsqueda se realiza con el mismo patrón que se haría en la web del famoso buscador).

Una vez desplazado sobre la zona deseada el usuario podrá pulsar durante un breve periodo de tiempo sobre cualquier parte del mapa, lo cual creará un marcador rojo indicando la zona sobre la que se acaba de pulsar. Así pues, si se pulsa ese marcador aparecerá una ventana con la información del punto GPS que se desea guardar.

Figura 3.6: Mapa y marcador con la información de la localización

En la *Figura 3.6* aparece la información que la aplicación muestra de la localización a guardar. Dicha información contiene la dirección, la localidad y el código postal. Además, como se ha visto en la *capa de negocio,* existen dos campos que el usuario deberá rellenar, que son el nombre de la localidad (a modo de alias) y el radio de acción del GPS.

Una vez que el usuario considere que la localización es la correcta y sus campos están completados correctamente, podrá guardar los cambios realizados y añadirla a la lista de localidades confirmando la selección.

En la *Figura 3.7* se muestra la pantalla de administración de localizaciones con la localización recientemente añadida. El usuario podrá ver, mediante una pulsación simple sobre la localización, la información de la misma y poder modificar los dos campos editables: nombre y radio.

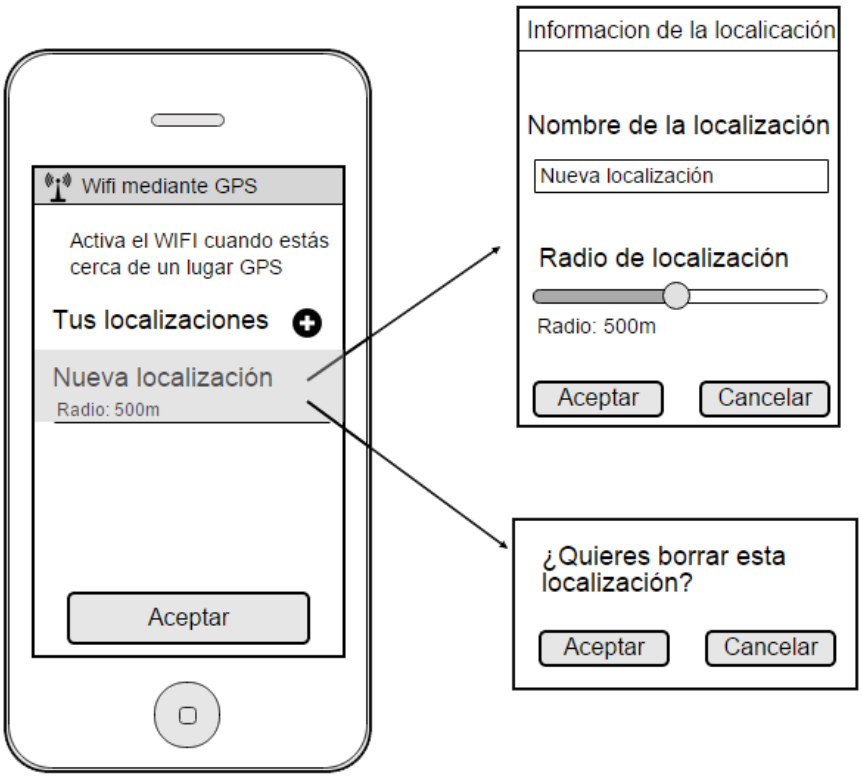

Figura 3.7: Modificación y borrado de un localización GPS

Por otra parte, mediante el uso de una pulsación prolongada sobre cualquier localización la aplicación dará la opción al usuario de poder borrarla, eliminado así todo el contenido referente a la misma, véase también en la *Figura 3.7*.

3.2.4 Pantalla de administración de datos

Desde la pantalla inicial, si el usuario pulsa el icono pertinente (el símbolo de las flechas) será redirigido a la pantalla de administración de los datos móviles, que puede observarse en la *Figura 3.8*.

El mecanismo de uso es el mismo que en la pantalla de administración WIFI vista en una sección anterior. El usuario dispondrá de diversas opciones de administración que podrá seleccionar pulsando sobre cualquiera de ellas. Una vez pulsada alguna de las opciones aparecerá una ventana con los valores que pueda tomar.

La opción "ver consumo de datos" redirigirá al usuario a la ventana de gestión de datos móviles que incorpora por defecto Android en su sistema operativo.

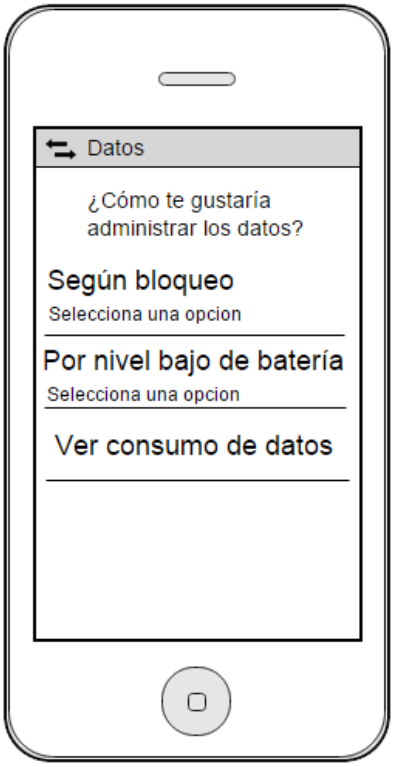

Figura 3.8: Opciones para administrar los datos móviles

3.2.5 Pantalla de ahorro de batería

La pantalla de administración de la batería cuenta con varias opciones dispuestas en una lista, distribución que se está siguiendo en todo el diseño de la aplicación.

Si el usuario pulsa para seleccionar tanto el nivel de batería como el brillo, aparecerá una ventana con un selector en forma de barra (seekbar), de manera que el usuario podrá escoger de manera rápida y sencilla el valor que desea para cada una de las opciones. El ejemplo de la Figura 3.9 muestra la opción pulsada para cambiar el brillo del dispositivo.

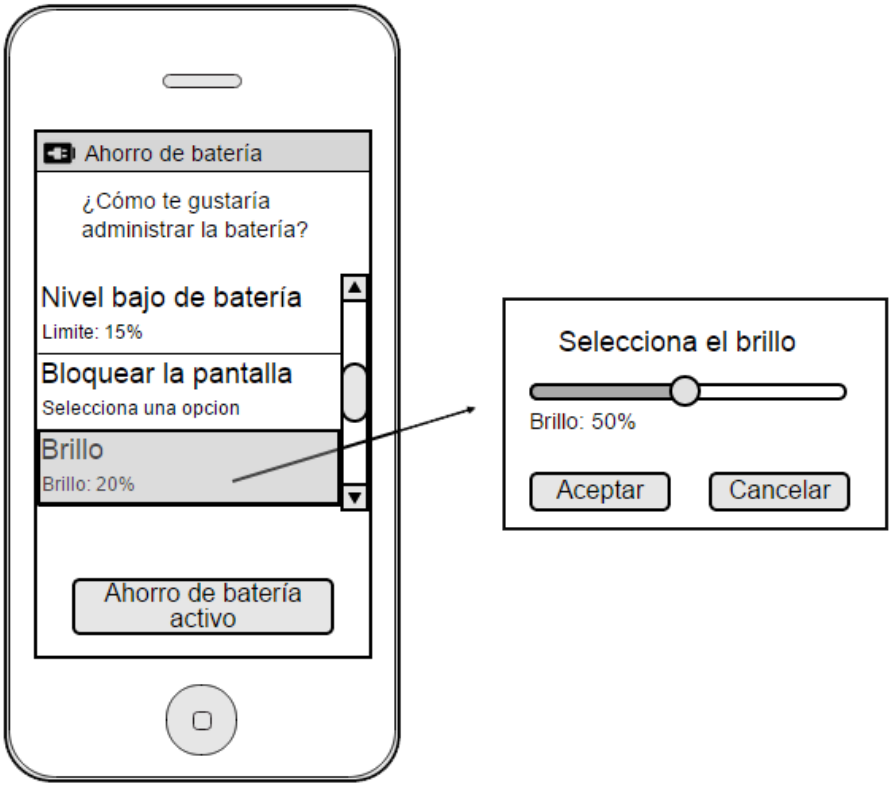

Figura 3.9: Opciones para administrar la batería

Además de contar con las diferentes opciones en forma de lista, el modo de ahorro de batería cuenta con un botón de encendido que permitirá al usuario seleccionar cuando quiere que esté en funcionamiento.

3.2.6 Pantalla de administración del perfil

Si se observa la *Figura 3.10,* lo primero que aparece a la vista del usuario son los campos con las horas de inicio y fin que se deberán rellenar para que el perfil pueda entrar en funcionamiento. Cuando el usuario pulse sobre cualquiera de los campos temporales la aplicación los completará por defecto con la hora actual del dispositivo, a fin de ahorrarle tiempo y prestarle facilidad en la selección.

Como ya es habitual en el diseño de esta aplicación aparece en forma de lista las opciones que dispone el usuario para la personalización del perfil, siguiendo el mecanismo de pulsar sobre ellas para seleccionar el valor deseado.

Figura 3.10: A la izquierda, opciones de personalización del perfil de sonidos; a la derecha, administración del filtro de llamadas

Si el usuario pulsa la opción "añadir filtro de llamada", que como ya se explicó es una opción para añadir contactos a los cuales no les afectará el perfil de sonidos, será redirigido a la pantalla de administración de los contactos, que puede verse en la *Figura 3.10*.

Desde esta pantalla, de una forma siempre muy intuitiva, los usuarios podrán tanto añadir nuevos contactos como borrar aquellos que hayan sido añadidos. Con sólo pulsar el botón con el signo '**+**' los usuarios serán redirigidos a la agenda del teléfono, donde podrán seleccionar el contacto que deseen para añadirlo en la lista.

Mediante una pulsación prolongada sobre alguno de los contactos añadidos se dará la opción a los usuarios de efectuar su borrado, apareciendo siempre una ventana informativa para que el usuario confirme la acción.

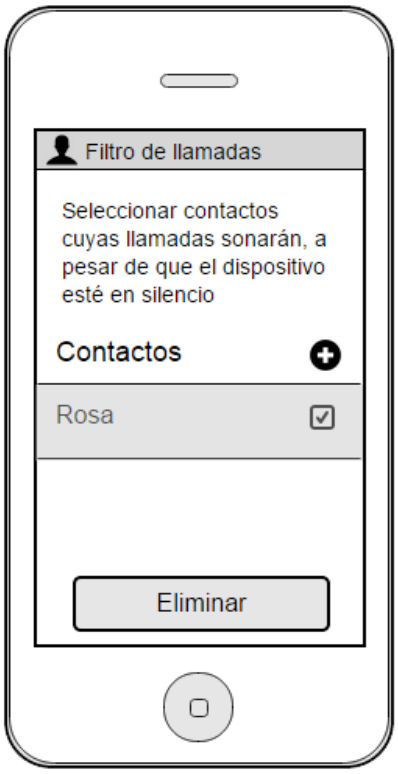

Figura 3.11: Eliminación de un contacto

En la *Figura 3.11* se muestra el ejemplo de la pantalla de administración de contactos en la cual existe un sólo contacto al que se le ha pulsado prolongadamente para poder ser borrado. Tras seleccionar el mismo y confirmar la selección, será eliminado de la lista.

Otra de las opciones que puede activar el usuario desde la pantalla de administración de perfil es el envío de SMS automáticos. Cuando esta opción es pulsada, se redirige al usuario a la pantalla de edición del mensaje a enviar.

Como puede observarse en la *Figura 3.12*, la aplicación contiene un SMS predefinido que sirve de ejemplo y de guía para facilitar a los usuarios algo que poner en caso de que no se les ocurriera nada.

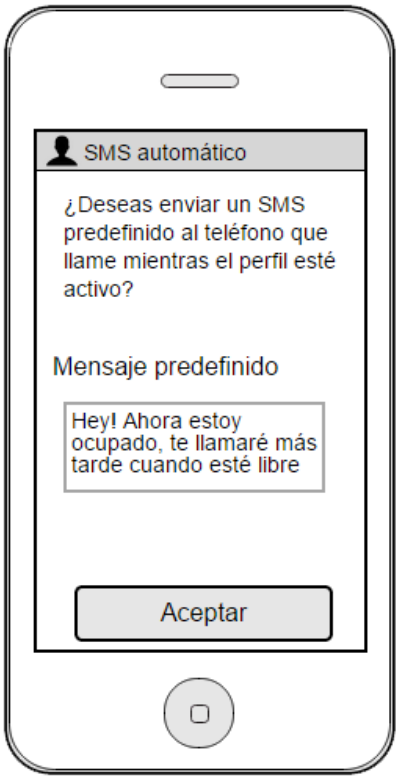

Figura 3.12: Edición de SMS predeterminado

El mensaje viene contenido en un cuadro de texto editable y podrá ser modificado sin problemas por el usuario siempre y cuando no supere la cantidad de 160 caracteres, ya que es el tamaño estándar permitido para el envío de un solo SMS.

Una vez introducido el SMS deseado y confirmada la selección, el SMS habrá sido guardado y estará listo para ser enviado.

3.2.7 Pantalla de administración del modo coche

Accediendo desde el icono correspondiente en la pantalla principal el usuario será redirigido a la pantalla de administración del modo coche. En ella aparecerá, como ya es habitual, una lista ofreciendo al usuario las diferentes posibilidades que dispondrá para hacer su trayecto al volante más confortable.

En la *Figura 3.13*, además de mostrarse el aspecto que tiene dicha pantalla de administración, aparece pulsada la opción “responder llamadas”, que, como se ha visto con más

profundidad en la capa de negocio, incluye las opciones de poder rechazar, aceptar o silenciar la llamada.

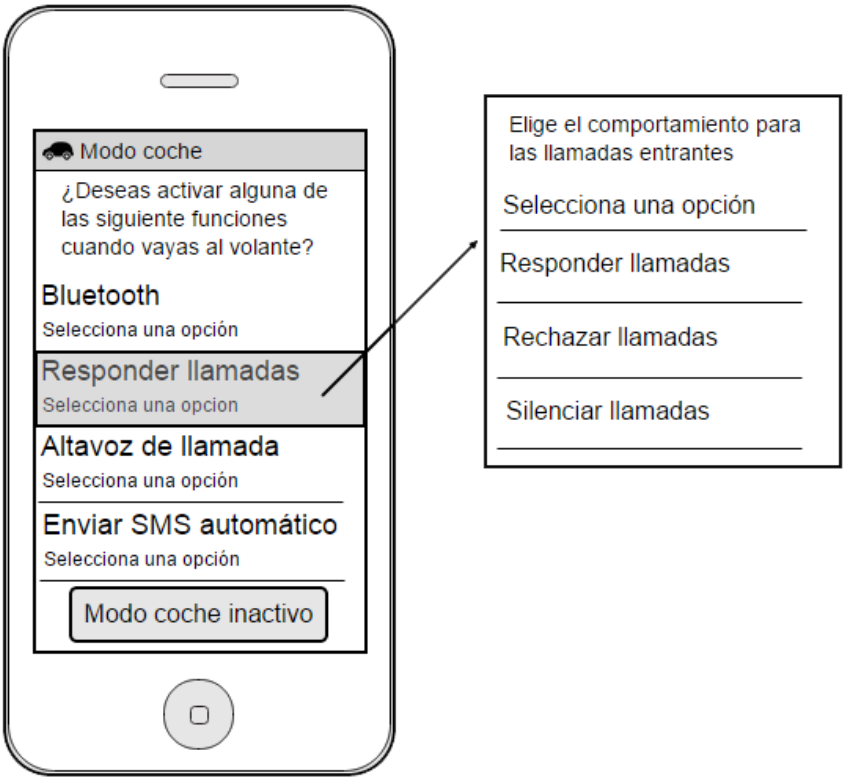

Figura 3.13: Opciones del modo coche

A su vez, el modo coche dispone también de una función de envío de SMS automáticos que enviará un SMS predefinido a cualquier número que llame mientras el modo coche esté activo.

Si el usuario activa dicha opción, será redirigido a la pantalla de edición del mensaje, que como puede apreciarse en la *Figura 3.14*, es prácticamente la misma pantalla que la vista en la administración del perfil de sonidos, véase *Figura 3.12.*

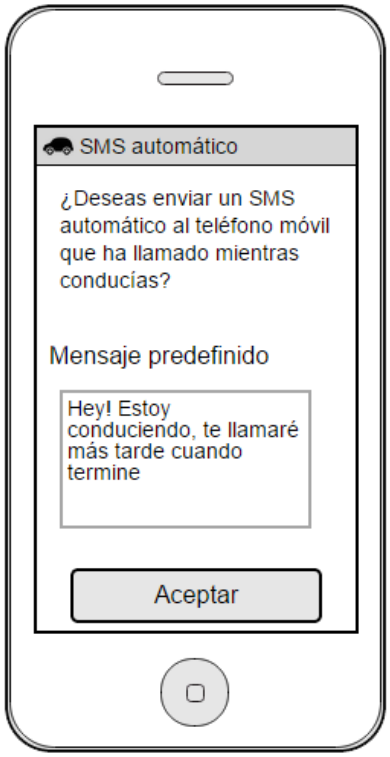

Figura 3.14: Edición de SMS automático en el modo coche

En ella el usuario podrá cambiar el SMS sin superar los 160 caracteres (tamaño permitido para el envío de un SMS). Por defecto, la aplicación incluye un SMS de ejemplo.

Al confirmar el SMS introducido (botón de aceptar), será guardado y listo para ser enviado. Si el usuario volviera a editar otra vez el SMS aparecerá en la pantalla la última edición que se haya realizado.

3.2.8 Pantalla de ajustes

La pantalla de ajustes está pensada básicamente para la administración de las notificaciones de la aplicación. Como puede verse en la *Figura 3.15* se trata de un diseño muy sencillo e intuitivo. El usuario dispondrá de una serie de botones a modo de interruptores para activar o desactivar las diferentes notificaciones que ofrece la aplicación.

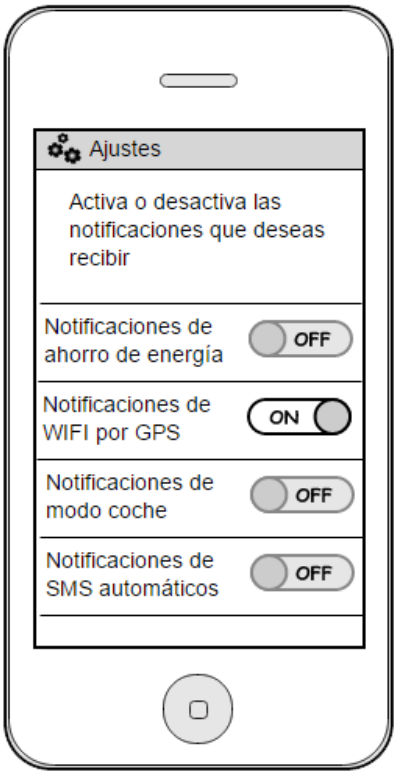

Figura 3.15: Pantalla de ajustes

3.3 Capa de datos

La capa de datos recoge la información que se ha considerado relevante para ser almacenada y para ser mostrada cuando sea necesario, a los usuarios de la aplicación.

Como esta aplicación se basa en la gestión y automatización de funciones y procesos de nuestro terminal, la cantidad de datos que se necesita almacenar es relativamente pequeña. Por ello, se ha optado por el uso de una base de datos relacional SQlite, la cual será más que suficiente para poder almacenar esta información.

Una base de datos relacional, es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. Permiten establecer interconexiones entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas. [14]

Así pues, en esta aplicación se ha requerido la creación de dos tablas de información no relacionadas entre sí, que son las que se presentan a continuación.

3.3.1 Tabla LocationInfo

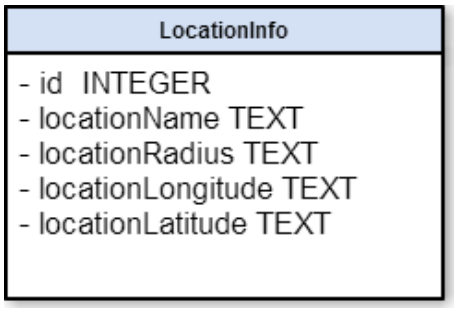

Figura 3.16: Tabla LocationInfo

La tabla *LocatioInfo* mostrada en la *Figura 3.16* almacena toda la información relativa a las localizaciones GPS introducidas por el usuario. Así pues, esa información está compuesta por los siguientes campos:

- **id:** identificador de la localización.
- **locationName:** Nombre de la localización.
- **locationRadius:** Radio de acción de la localización.
- **locationLongitude:** Parte de la coordenada geográfica, en este caso, la longitud de la localización.
- **locationLatitude:** Parte de la coordenada geográfica, en este caso, la latitud de la localización.

3.3.2 Tabla ContactsTable

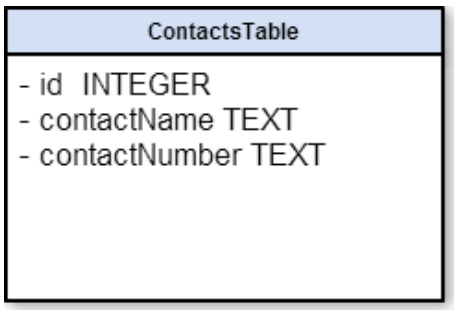

Figura 3.17: Tabla ContactsTable

En la *Figura 3.17* se puede apreciar la tabla *ContactsTable* recoge la información necesaria para la gestión de los contactos que el usuario ha añadido en el filtro de llamadas, opción ofrecida en el perfil de sonidos de la aplicación. Dicha información se compone de los siguientes campos:

- **id:** Identificador del contacto.
- **contactName:** Nombre del contacto.
- **contactNumber:** Número de teléfono del contacto.

4. Diseño

En este apartado se llevará a cabo un estudio de las diferentes herramientas disponibles para la elaboración del proyecto así como la que finalmente se ha escogido. Se presentará también las reglas y convenios que se han seguido para su desarrollo.

4.1 Capa de presentación

La capa de presentación, como se ha visto en la sección de especificación, se trata de la interfaz gráfica que el usuario manipula para su interacción y comunicación con la capa de negocio.

En este caso la interfaz, que es la misma aplicación, viene desarrollada en Java, lenguaje propio para la creación de aplicaciones Android y complementado con el kit de desarrollo de aplicaciones SDK [15] (Software development Kit) de Android, el cual ha sido necesario para poder realizar tanto la interfaz como la totalidad proyecto en sí.

4.1.1 Android Development Kit: Interfaces

El SDK de Android incorpora gran variedad de elementos para permitir a los desarrolladores la creación de interfaces [16].

Gran parte del SDK dedicado a interfaces se programa en XML [17], ver sección futura 4.1.1.1, donde se comentará con más detalle este tipo de lenguaje.

Así pues, alguno de los ejemplos de la gran variedad de recursos que ofrece el SDK de Android para la creación de interfaces y que se han utilizado en el desarrollo de este proyecto son:

- **Métodos de entrada:** son componentes interactivos que permiten al usuario la introducción de datos, ya sea en forma de texto (cuadros de inserción de texto), de selección (radio buttons, check boxes, spinners, etc.), o de botón.
- **Eventos de entrada:** se trata de funciones ya implementadas para detectar la interacción de los usuarios con la aplicación. Algunos ejemplos de estas funciones:

  - **onClick():** es una función que se utiliza para saber cuándo el usuario está pulsando sobre algún elemento de la aplicación.
  - ***onLongClick():*** realiza la misma función que onClick(), pero en este caso, estará detectando cuando el usuario mantiene pulsado un elemento o parte de la aplicación .
  - **onTouch():** detectaría el movimiento del usuario sobre la aplicación, como por ejemplo un movimiento slide para hacer un scroll down/up y moverse por la misma.

- **Elementos para notificar:** son componentes cuya función primordial es notificar o informar al usuario de algo que está sucediendo o ha sucedido en la aplicación. En este apartado de elementos tendría cabida:

  - **Notificaciones:** son mensajes informativos que pueden aparecer aunque la aplicación no se encuentre en primer plano. Es un elemento meramente informativo, aunque en algunos casos el usuario podrá desde la notificación realizar alguna pequeña operación.
  - **Toast**: son elementos que proporcionan información sobre una operación que acaba de tener lugar. Por ejemplo: cuando se manda un mensaje, aparecería un toast con la información, "*mensaje enviado"*.

- **Ventanas:** se trata de ventanas emergentes, las cuales se suelen utilizar como complemento de alguna parte de la aplicación, normalmente para que el usuario tome alguna decisión. Son los comúnmente llamadas "*dialogs"* en la programación de aplicaciones móviles.
- **Elementos para organizar:** el SDK de Android también dispone de elementos para poder organizar la información que se va a mostrar y definir la estructura de la interfaz de usuario. Son los llamados "layouts", y algunos de ellos son:

  - **Linear layout:** es el diseño que alinea la información, es decir los elementos de la interfaz, en una sola dirección ya sea horizontal o vertical.
  - **Relative layout:** en este diseño la información mostrada se organiza atendiendo a los demás elementos de la interfaz. Cada elemento se colocará en una posición relativa respecto a los demás.
  - **List View:** diseño en forma de lista de elementos. La información se muestra en una lista en la que se permite hacer scroll (desplazar los elementos horizontalmente o verticalmente para poder ver aquellos que no caben en la pantalla del terminal).

4.1.1.1 XML

XML (en inglés, Extensible Markup Language) es un lenguaje de etiquetas, es decir, cada paquete de información está delimitado por dos etiquetas, como se hace también en el lenguaje HTML, pero XML separa el contenido de la presentación.

XML se plantea como un lenguaje estándar para el intercambio de información entre diferentes programas de una manera segura, fiable y libre, ya que no pertenece a ninguna compañía.

Características principales del lenguaje XML:

- XML es un subconjunto de SGML que incorpora las tres características más importantes de este:
  - Extensibilidad
  - Estructura
  - Validación
- Basado en texto.
- Las etiquetas se definen para crear los documentos, no tienen un significado preestablecido.

Ejemplo XML

Un ejemplo de código XML sería:

"*Mateo nació el 15.10.2012 en la ciudad de Madrid con un peso de 3.1 kg y una estatura de 45 cm*."

Si se quisiera pasar ese ejemplo a código XML y con la premisa de que los campos: nombre, fecha, ciudad, peso y estatura forman la estructura persona, el código XML correspondiente que surgiría de ello sería algo como:

```
<Persona>
      <Nombre>Mateo</Nombre>
      <Fecha>15.10.2012</Fecha>
      <Ciudad>Madrid</Ciudad>
      <Peso>3.1Kg</Peso>
      <Estatura>45cm</Estatura>
   </Persona>
```

4.2 Capa de negocio

La capa de negocio, al igual que la capa de presentación, ha sido desarrollada mediante el SDK de Android. De hecho, la totalidad de la aplicación se ha desarrollado utilizando íntegramente el kit de desarrollo que proporciona Android.

4.2.1 Android Development Kit (SDK)

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

Los sistemas operativos soportados son Linux, Mac y Windows. La plataforma de desarrollo soportada oficialmente es Android Studio, ofrecida por Google y disponible gratuitamente en su página web oficial, basado en el entorno de desarrollo IntelliJ IDEA.

Se trata de un software de código abierto, con un previo acuerdo con Google a la hora de poder publicar los proyectos en el mercado de aplicaciones.

Así pues, como se acaba de comentar, este kit contiene todos los componentes que han sido necesarios para el desarrollo de esta aplicación. Algunos de los componentes más relevantes [18], que son los que forman el cuerpo, estructura y comportamiento de cualquier aplicación Android son:

Vistas

Las vistas son los elementos que componen la interfaz de usuario de una aplicación. Son por ejemplo un botón, una entrada de texto, etc. Todas las vistas van a ser objetos descendientes de la clase View, y por tanto, pueden ser definidos utilizando código Java. Sin embargo, lo habitual va a ser definir las vistas utilizando un fichero XML y dejar que el sistema cree los objetos por nosotros a partir de este fichero.

Layout

Un layout es un conjunto de vistas agrupadas de una determinada forma. Se dispone de diferentes tipos de layout para organizar las vistas de forma lineal, en cuadrícula o indicando la posición absoluta de cada vista.

Los layout también son objetos descendientes de la clase View, igual que las vistas. La forma habitual de definirlos en utilizando código XML.

Actividad (Activity)

Una aplicación Android está formada por un conjunto de elementos básicos de visualización, coloquialmente conocidos como pantallas de la aplicación. En Android cada uno de estos elementos o pantallas se conoce como actividad.

Su función principal es la creación de la interfaz de usuario. Una aplicación suele necesitar varias actividades para crear la interfaz de usuario. Las diferentes actividades creadas serán independientes entre sí, aunque todas trabajarán para un objetivo común. Toda actividad ha de pertenecer a una clase descendiente de Activity.

Servicio (Service)

Un *servicio* es un proceso que se ejecuta en segundo plano, es decir, "detrás" de la aplicación, sin la necesidad de una interacción con el usuario, lo que significa que un servicio podrá estar ejecutando operaciones aunque el usuario esté utilizando una aplicación diferente.

Intent

Un intent representa la voluntad de realizar alguna acción, como realizar una llamada de teléfono o visualizar una página web, entre otras.

Se utiliza cada vez que se quiera:

- Lanzar una actividad.
- Lanzar un servicio.

- Enviar un evento de tipo broadcast.
- Comunicarse con un servicio.

Los componentes lanzados pueden ser internos o externos a nuestra aplicación. También se utilizarán los intent para el intercambio de información entre estos componentes.

Broadcast receiver

Un broadcast receiver reacciona ante eventos de tipo broadcast. Los eventos broadcast pueden ser originados por el propio sistema o por las aplicaciones. Algunos tipos de eventos originados por el sistema son: batería baja, una llamada entrante, un cambio de hora, etc.

Las aplicaciones también pueden crear y lanzar nuevos eventos broadcast. Los broadcast receiver no disponen de interfaz de usuario, aunque pueden iniciar una actividad si lo estiman oportuno.

4.3 Métodos de implementación

La elaboración del proyecto se ha llevado a cabo siguiendo una serie de normas y reglas de forma que, en caso de que fuera retomado en un futuro ya sea por el mismo programador u otro diferente, pueda hacerse con la mayor facilidad posible. Es decir, que permita su mantenibilidad en la medida de lo posible y no se convierta en un código engorroso que sea ininteligible.

Así pues, las normas que se han tenido en cuenta son las siguientes:

- Los comentarios de código explicando algunas situaciones que se han considerado convenientes se han realizado en inglés ya que, al ser un idioma universal, permitirá dotar de internacionalidad a la aplicación.
- Las variables locales que aparecen siguen todas el convenio de estar escritas, además de en inglés, en minúsculas y que tenga relación con el nombre que posee. En caso de ser una variable con más de una palabra, se unirá cada una de las palabras con el signo “_”. Un ejemplo de ello, si queremos referirnos a

una variable para el nombre de la localización GPS, el resultado sería: "*location_name".*

- Las variables globales están escritas en inglés y todas ellas en mayúsculas. Al igual que en las variables locales, si estuviera compuesta por varias palabras irían unidas por "_". Ejemplo: booleano para saber si el modo de ahorro de batería está o no activo: "*BATTERY_MODE"*.
- Los métodos o funciones vienen definidos por un texto descriptivo en inglés. La primera letra de la primera palabra siempre será en minúscula. En caso de tener varias palabras se escribirá todo junto, siendo la primera letra de las siguientes palabras en mayúsculas. Por ejemplo, un método para devolver el nombre de una localización GPS sería: "getLocationName()";

4.4 Control de versiones

Como es lógico en cualquier proyecto desarrollado, se parte de una elaboración desde cero y se va construyendo poco a poco. A medida que se va desarrollando, el autor puede que se percate de que lo que había hecho no era lo que él esperaba y por tanto necesite retroceder hasta cierto punto del proyecto para poder retomarlo desde ahí.

Por otro lado, resulta de interés guardar avances implementados y no perder ningún tipo de configuración o información en caso de que surgiera algún problema: de repente se estropea el ordenador, no funciona el programa de edición, etc.

Es por estos motivos por los cuales se ha llevado a cabo en este proyecto un control de versiones. Actualmente existen varias propuestas que ofrecen este tipo de servicio, entre ellas Dropbox y GitHub, que serán analizadas a continuación. En este caso, se optado por el uso de ***Dropbox***, decisión que se argumentará al analizar dicha propuesta.

4.4.1 Dropbox

Dropbox se trata de un servicio de almacenamiento de archivos de todo tipo (videos, fotos, documentos, etc.) en la nube, lo que quiere decir que cualquier usuario puede acceder a sus datos desde cualquier dispositivo a través de su cuenta [19] [20]. Véase su logotipo en la *Figura 4.1.*

Permite creación, modificación y borrado de carpetas, además de poder organizar los datos siguiendo la misma estructura de carpetas que se tendría en el sistema operativo Windows.

Se trata de un servicio gratuito, aunque también dispone de unas versiones de pago llamadas *Premium,* en las cuales se ofrece un servicio ampliado, como por ejemplo una mejor recuperación de los archivos en caso de pérdida y un mayor espacio de almacenamiento.

Aunque Dropbox no se trate en sí de un programa específico de control de versiones, permite ver una traza de los archivos subidos así como ver la fecha en la que se realizó la sincronización. Además, en el caso de que se haya producido una eliminación errónea de alguno de los archivos ofrece la posibilidad de volver a recuperarlos.

Figura 4.1: Logotipo de Dropbox

Principales características de Dropbox:

- Sistema multiplataforma compatible con Windows, Linux y Mac.
- Sólo sincroniza archivos modificados.
- Tiene cliente de ordenador, aparte de su versión web.
- Puede trabajar offline (Sincroniza los archivos una vez se restaura la conexión).
- Guarda el historial de archivos modificados (30 días versión gratuita).

4.4.2 Alternativa: Git & GitHub

A continuación se estudian los sistemas de control de versiones Git y GitHub por separado, (a pesar de GitHub es in sistema que requiere de Git) incluyendo las características que ambos poseen y los servicios que son capaces de ofrecer, para valorar finalmente, que sistema de control de versiones sería el adecuado para el desarrollo de este proyecto.

4.4.2.1 Git

Git [21] se trata de un sistema de control de versiones distribuido que fue impulsado por Linus Torvalds y el equipo de desarrollo del Kernel de Linux. Surgió porque en un principio estaban utilizando otro sistema de control de versiones que al tiempo pasó a ser un sistema que no era de código abierto, sino privado. Por tanto, entraba en conflicto que una plataforma como Linux, que es en su totalidad de código abierto, utilizara un sistema de control de versiones privado. Entonces el equipo de Linux creó el sistema Git desde cero.

Es un sistema de código abierto y relativamente nuevo que ofrece las mejores características en la actualidad, pero sin perder la sencillez y que a partir de entonces no ha parado de crecer y de ser usado por cada vez más desarrolladores en el mundo.

Git es multiplataforma, por lo que puede usarse y crear repositorios locales en todos los sistemas operativos más comunes, Windows, Linux o Mac. Se muestra su logotipo comercial en la *Figura 4.2*.

Figura 4.2: Logotipo de Git

Características principales de Git [22]:

- Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal.
- Gestión distribuida. Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera que se hace con la rama local.
- Gestión eficiente de proyectos grandes.
- Todas las versiones previas a un cambio determinado implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio.

4.4.2.2 GitHub

GitHub [23] es un servicio de alojamiento de repositorios de software gestionados por el sistema de control de versiones Git. Por tanto, Git es algo más general que sirve para controlar

el estado de un desarrollo a lo largo del tiempo, mientras que GitHub es algo más particular: un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software. Su logotipo mostrado en la *Figura 4.3.*

En definitiva, GitHub es un sitio web pensado para hacer posible el compartir el código de una manera más fácil.

Cabe destacar que GitHub es un proyecto comercial, a diferencia de la herramienta Git que es un proyecto de código abierto. No es el único sitio en Internet que mantiene ese modelo de negocio, pues existen otros servicios de alojamiento similares.

Figura 4.3: Logotipo de Github

Características de GitHub [24]:

- Wiki para cada proyecto.
- Página web para cada proyecto.
- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto.
- Bueno para el trabajo colaborativo entre programadores.
- Funcionalidades como si se tratase de una red social, como por ejemplo: seguidores.

4.4.3 ¿Por qué Dropbox?

Como se ha introducido en la sección *control de versiones*, se ha escogido Dropbox a pesar de no ser una herramienta estrictamente dedicada para este tipo de servicios por los siguientes motivos:

- **Experiencia de uso:** se trata de una herramienta con la cual se había trabajado con anterioridad y se conocía cada de una las funciones que era capaz de ofrecer además de haber obtenido siempre una experiencia de uso satisfactoria.
- **Facilidad y sencillez:** Dropbox destaca por su gran sencillez de uso. Apenas hay que documentarse para poder manejar esta herramienta con gran fluidez y en tiempo escaso, gracias a su interfaz de usuario amigable e intuitiva.
- **Proyecto individual:** es unos de los principales motivos por los que se ha decidido escoger esta herramienta frente a Github. Al tratarse de un proyecto en el que sólo trabaja una persona no hay problema para subir la información necesaria al repositorio online cada vez que se crea conveniente. De haber sido un proyecto colectivo se habría descartado completamente esta posibilidad ya que Github es una herramienta que funciona realmente bien para controlar proyectos grupales. Sin embargo, Dropbox resultaría muy ineficiente para poder trabajar colectivamente sobre un mismo proyecto.
- **Multi acceso:** permite tener acceso a la información desde todo tipo de dispositivos: móviles, tablet, ordenadores, etc. siendo ésta una opción muy práctica, ya que en alguna ocasión se ha necesitado acceder desde varios dispositivos diferentes.

5. Implementación

En esta sección se verá cómo, mediante el uso de las herramientas explicadas anteriormente, se ha desarrollado la idea final del proyecto así como los posibles problemas que han surgido durante su elaboración y las medidas que se han adoptado para poder solucionarlos.

5.1 Consumo creciente de RAM

A lo largo de esta memoria se ha podido ver que esta aplicación se dedica básicamente, a la gestión automática de algunas funciones del dispositivo móvil que de por sí no lo son. Para poder realizar esta gestión, ha sido imprescindible el uso de *broadcast receivers* y *services*.

Los *broadcast receiver* (se explicó con detalle en la sección de *diseño*), es una funcionalidad que ofrece Android la cual permite a la aplicación reaccionar ante eventos que puedan suceder.

Por otra parte, un *service* es un proceso que se ejecuta en el fondo o "*background*" de la aplicación, aunque esta no esté en primer plano.

Tiene sentido que para gestionar automáticamente funciones como ahorro de batería, activar-desactivar WIFI dependiendo del bloqueo de la pantalla o silenciar el terminal en una determinada franja horaria, entre otras, se haya hecho uso tanto de los *broadcast receiver* como de los *services*.

El problema surgió al probar la aplicación. En primera instancia, el consumo de RAM que estaba experimentando era lo esperado para cualquier aplicación. Sin embargo, al cabo del poco tiempo de estar en ejecución, el consumo de RAM se disparaba de una manera fuera de lo común, casi exponencial.

Lo que sucedía era que, cada vez que un broadcast receiver reaccionaba a cualquiera de los eventos que estaba esperando, (cambio de nivel de batería, llamada entrante, etc.), se lanzaba un *service* para que ejecutara la acción o acciones pertinentes. Por tanto, la cantidad de *service* que creaba la aplicación estaba creciendo continuamente y esto elevaba de una manera brutal el consumo de la aplicación.

La solución para este problema fue relativamente sencilla. En la actividad en la que se creaban los *service* había que realizar una comprobación para ver si no había ya uno existente.

```
public void callService(Activity activity){
	intentService = new Intent(activity, managementService.class);
	//	Call service if there is no service already running
		SharedPreferences servicePref =
activity.getSharedPreferences("servicePref", Context.MODE_PRIVATE);
		serviceRunning = servicePref.getBoolean("serviceRunning",
false);

	if(!serviceRunning)
		activity.startService(intentService);
}
```

Código 5.1: Comprobación y creación de services

El *Código 5.1* muestra el ejemplo de dicha comprobación. Se utiliza la gestión de datos con los *sharedPreferences* por comodidad, ya que los flags para realizar las comprobaciones, como *serviceRunning* en este caso, es utilizado por actividades diferentes.

De esta manera se pudo asegurar que sólo hubiera un *service* en ejecución y que no creciera en número constantemente.

Se pudo comprobar utilizando posteriormente la aplicación que el consumo se encontraba dentro de la normalidad, ya que consumía como cualquier otra aplicación instalada en el terminal, oscilando siempre entre los mismos valores de consumo sin experimentar un crecimiento abusivo.

5.2 Problema con el colapso de la aplicación

En el sistema operativo Android la gestión de las aplicaciones en ejecución se hace de la siguiente manera. Según la RAM disponible en el terminal, los usuarios pueden abrir aplicaciones móviles hasta abarcar casi su totalidad. Es entonces cuando Android gestiona las aplicaciones y pasa a eliminar aquellas que no se están usando para dejar hueco a las nuevas que se vayan a abrir.

No obstante, que vaya eliminando aplicaciones no significa que mate procesos, de hecho sólo se eliminan aplicaciones que no tienen actividad.

En el caso de este proyecto, como se ha ido viendo, es una aplicación que sí tiene actividad ya que se compone de un *service* que se va ejecutando en el *background*. Sin embargo, se experimentó que cuando la RAM del terminal estaba casi al completo, la aplicación dejaba de funcionar con normalidad, es decir, el service se suspendía sin razón aparente.

Para saber un poco más sobre el comportamiento que adoptan los service véase la información incluida en la documentación oficial para desarrolladores Android [25]. Los *service* tienen tres tipos de valor de retorno:

- Service.START_STICKY es el comportamiento estándar, en este caso el método onStartCommand() será invocado cada vez que el servicio sea reiniciado tras ser terminado por la máquina virtual. Nótese que en este caso al reiniciarlo el Intent que se le pasa por parámetro será null. Service.START_STICKY se utiliza típicamente en servicios que controlan sus propios estados y que se inician y terminan de manera explícita con startService y stopService. Por ejemplo, servicios que reproduzcan música u otras tareas de fondo.
- Service.START_NOT_STICKY se usa para servicios que se inician para procesar acciones específicas o comandos. Normalmente utilizarán stopSelf() para terminarse una vez completada la tarea a realizar. Cuando la máquina virtual termine este tipo de servicios, éstos serán reiniciados sólo si hay llamadas de startService pendientes, de lo contrario el servicio terminará sin pasar por onStartCommand. Este modo es adecuado para servicios que manejen peticiones específicas, tales como actualizaciones de red o polling de red.
- Service.START_REDELIVER_INTENT es una combinación de los dos anteriores de manera que si el servicio es terminado por la máquina virtual, se reiniciará sólo si hay llamadas pendientes a startService o bien el proceso fue matado antes de hacer la llamada a stopSelf(). En este último caso se llamará a onStartCommand() pasándole el valor inicial del Intent cuyo procesamiento no fue completado. Con Service.START_REDELIVER_INTENT nos aseguramos de que el comando cuya ejecución se ha solicitado al servicio, sea completada hasta el final.

Así pues, en la aplicación desarrollada se optó por un valor de retorno "*STAR_REDELIVER_INTENT"* para que, en caso de que el sistema necesitara "matar" la aplicación por falta de memoria RAM, ésta reactivase sus procesos con los mismos parámetros con los que se habían iniciado.

Pues bien, ese tipo de solución no fue suficiente, ya que, a pesar de lo que se ha explicado, cuando el terminal se encontraba con escasa RAM disponible, los procesos no se volvían a poner en funcionamiento correctamente.

La solución que se adoptó para resolver este tipo de problema fue añadir una función característica que poseen los *services* que, pese a no ser recomendable para todas las aplicaciones, resulta indispensable para el tipo de aplicaciones cuya fiabilidad y robustez es lo más importante.

Esta característica consiste en dotar al service de *"inmunidad"* frente a la liberación de RAM por parte de Android ante un nivel bajo. Cabe destacar que en cualquier momento si fuera necesario, Android podría eliminar la aplicación, pero sería bajo una situación crítica.

Así pues, la función que incluye el service es la siguiente:

```
startForeground(NOTIFICATION_ID, notification);
```

En el *Código 5.2* se muestra parte del código referente al service que gestiona los diferentes broadcast receivers que se han utilizado en la aplicación

```
public static class managementServiceextends Service {
	Notification notification = new Notification();
	@Override
	public void onCreate() {
		super.onCreate();
		// REGISTER RECEIVER THAT HANDLES SCREEN ON AND SCREEN OFF LOGIC
		IntentFilter screenFilter = new
	IntentFilter(Intent.ACTION_SCREEN_ON);
		screenFilter.addAction(Intent.ACTION_SCREEN_OFF);
		screenReceiver = new ScreenReceiver();
		registerReceiver(screenReceiver, screenFilter);
. . .
		startForeground(2314, notification)
}
. . .
```

Código 5.2 Ejemplo del service que gestiona los boadcast receivers

5.3 Gestión de las localizaciones GPS

En la sección de *diseño* se ha mostrado que los usuarios pueden añadir localizaciones GPS utilizando el mapa que proporciona Google. Estas localizaciones son almacenadas en una lista donde el usuario podrá en todo momento, tanto modificar los valores permitidos como eliminar la localización en sí.

Para visualizar esta información, se ha optado por utilizar una lista de elementos *item* aprovechando la estructura de los elementos que forman cada una de las listas de opciones, formada por dos campos de tipo *string*.

Por otra parte, para poder gestionar la información recibida por el marcador añadido en el mapa, como se trata de varios campos que se van a tener que guardar de alguna forma para poder comprobar si el usuario está en alguno de los puntos GPS, se ha decido usar una pequeña base de datos local utilizando SQLite, que viene incluida en el paquete de herramientas para desarrolladores SDK de Android.

Se ha optado por una base de datos SQLite [26] puesto que se va a manejar información en una aplicación Android a la cual se va a necesitar acceder desde varias actividades diferentes.

Además, siendo una cantidad de información considerable, la opción de la base de datos se trata de una elección más que acertada, ya que cubre ampliamente las necesidades encontradas.

Así pues, se ha creado una tabla con un código incremental como clave primaria y con cuatro campos de texto: el nombre de la localización, el radio, la longitud y la latitud, quedando el código que la compone de la manera que se muestra en el *Código 5.3*.

```
public class locationInformation extends SQLiteOpenHelper {

	// SQL sentence that creates the DB
	String sqlCreate = "CREATE TABLE LocationInfo (codigo INTEGER PRIMARY
KEY AUTOINCREMENT NOT NULL,"
		+ "locationName TEXT, locationRadius TEXT,”
		+ "locationLongitude TEXT, locationLatitude TEXT)";

	public locationInformation(Context context, String name,
				CursorFactory factory, int version) {
		super(context, name, factory, version);
	}

	@Override
	public void onCreate(SQLiteDatabase db) {
		// TODO Auto-generated method stub
		// Executes the sentence to create the DB
		db.execSQL(sqlCreate);
	}

	@Override
	public void onUpgrade(SQLiteDatabase db, int versionAnterior,
			int versionNueva) {

		// The last version of the DB is deleted
		db.execSQL("DROP TABLE IF EXISTS LocationInfo");

		// A new DB is created
		db.execSQL(sqlCreate);
	}

}
```

Código 5.3: Gestión de las localizaciones utilizando SQlite

Se trata de una tabla sencilla que ha servido de gran ayuda para poder almacenar toda esa información que se necesitaba de las localizaciones.

De esta manera, tanto la modificación como el borrado de localizaciones se han podido desarrollar sin problemas. En el caso del borrado de localizaciones, ha bastado con borrar de la tabla la entrada que coincidiera con el nombre de la localización a borrar.

Ejemplo de borrado mostrado en el *Código 5.4*:

```
String[] args = new String[] { locationNameDeleted };

      db.execSQL(

      "DELETE FROM LocationInfo WHERE locationName=?",

      args);
```

Código 5.4: Borrado de localizaciones

En el caso de la modificación de localizaciones, existe un primer paso idéntico al mencionado anteriormente y un segundo paso añadiendo una entrada nueva con la información actualizada. Véase el *Código 5.5* donde se muestra el ejemplo de inserción de una localización, con toda su información pertinente.

```
// Insert values to the data base
if (db != null) {
db.execSQL("INSERT INTO LocationInfo ( locationName, locationRadius,
locationLongitude, locationLatitude) "
                                        + "VALUES ('"
                                        + locationName
                                        + "', '"
                                        + locationRadius
                                        + "', '"
                                        + locationLongitude
                                        + "' ,'"
                                        + locationLatitude + "')");
}
```

Código 5.5: Inserción de una localización

5.4 Problema de cálculo GPS

Una de las funcionalidades que ofrece la aplicación es la de activar el WIFI según si el usuario está dentro del área de un punto GPS introducido previamente.

En realidad se trata de un cálculo sencillo, aunque en primera instancia, la inclusión de un radio que podía variar hizo que se tardara algo más en llegar a la solución.

Para dar con la solución bastó con plantear un boceto de lo que se quería calcular y rápidamente se pudo intuir la solución de forma gráfica.

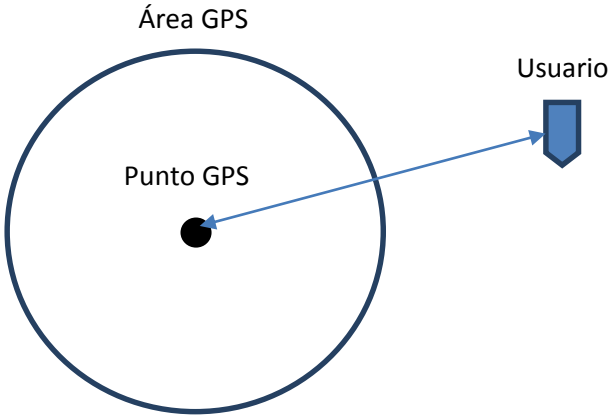

Figura 5.1: Situación de uso de localización GPS

Viendo la *Figura 5.1* se puede deducir que si la distancia del usuario al punto GPS es mayor que el radio del área establecido en el punto GPS, el usuario estará fuera de la localización GPS. De lo contrario, si la distancia es menor al radio del área, se encontrará en su interior y por lo tanto, se dará la condición para que se encienda el WIFI automáticamente.

5.5 Gestión de los contactos del filtro de llamadas

En el apartado de la administración del perfil de sonidos una de las opciones que se le ofrece al usuario es la de poder crear una lista con los contactos que desee, a los cuales no les afectará ninguna de las opciones del perfil.

Esta situación en su desarrollo, ha resultado ser idéntica a la gestión de localizaciones GPS, vista en una de las secciones anteriores. Por lo tanto, se ha optado también por el uso de una pequeña base de datos SQLite como utilidad para poder manipular la información de forma cómoda y sin riesgos de perderla.

En este caso la tabla ha resultado más sencilla que la de las localizaciones GPS, ya que sólo ha sido necesario almacenar dos campos: el nombre del contacto y su número de teléfono. Ambos se han guardado como formato *string* y en el momento que se ha necesitado el número de teléfono como un *int* se ha realizado la conversión pertinente.

5.6 Flags de control

En este proyecto, como se ha visto a lo largo de esta memoria, se ha desarrollado una aplicación que ofrece multitud de servicios y funciones.

La aplicación está compuesta por una gran cantidad de listas de opciones seleccionables por los usuarios. Además, existen multitud de situaciones como una llamada entrante, un cambio en el nivel de batería, un transcurso del tiempo, etc., que desencadenan alguna función automática.

Para poder tener un control de todas las situaciones y que no entren en conflicto unas con otras, cabe destacar que se han empleado una gran cantidad de flags de control.

5.7 Bloquear llamadas entrantes

El bloqueo de las llamadas entrantes, función ofrecida en el apartado modo coche de la aplicación, se trata de una acción que no está disponible de manera sencilla para los programadores, al tratarse de una funcionalidad que requiere una comunicación más profunda con el terminal.

En el SDK no se incorpora la opción de poder rechazar llamadas y para poder ofrecer este tipo de servicio se ha necesitado de un paquete externo muy conocido en el mundo de los desarrolladores, llamado "*IThelephony.aidl*", que al agregarlo al proyecto a desarrollar, desempeña la función de ofrecer al desarrollador una serie de funciones más privilegiadas que le permiten una comunicación más interna con el terminal.

Este paquete es utilizado con frecuencia por los desarrolladores de ROMs (Read-only memory), que son distribuciones no oficiales de Android, a las cuales se le han añadido nuevas características, funcionalidades o se le han optimizado procesos, etc. para conseguir una versión de Android personalizada, acorde con las preferencias del desarrollador o desarrolladores.

Con *IThelephony.aidl* agregado al proyecto se ha podido realizar la comunicación pertinente gracias a los métodos que vienen incluidos en este paquete y así poder desempeñar las funciones privilegiadas que se quería ofrecer a los usuarios finales de la aplicación.

Ejemplo de rechazo de una llamada utilizando las funciones del paquete ITelephony.aidl en el *Código 5.6*:

```
import com.android.internal.telephony.ITelephony;

...

TelephonyManager telephony = (TelephonyManager) context

      .getSystemService(Context.TELEPHONY_SERVICE);
      try {
            Class<?> c = Class.forName(telephony.getClass().getName());

            Method m = c.getDeclaredMethod("getITelephony");

            m.setAccessible(true);

            telephonyService = (ITelephony) m.invoke(telephony);

            telephonyService.endCall(); // Reject the call

      }

...
```

Código 5.6: Rechazo de una llamada utilizando el paquete ITelephony.aidl

6. Resultados

En esta sección se presentará el resultado final de la aplicación desarrollada incluyendo algunas capturas de pantalla que se han considerado relevantes. Además, se hablará de los resultados obtenidos en diferentes dispositivos tanto físicos como virtuales y bajo diversas versiones de Android.

6.1 Capturas del resultado final

A continuación se muestra una serie de capturas de pantalla con el resultado final obtenido. Se exponen aquellas que se han considerado relevantes o con cambios significativos y se realiza una escueta comparativa respecto al modelo inicial presentado en la sección 3. Especificación.

6.1.1 Pantalla inicial

Al desarrollar la aplicación, una de las ideas principales que se quería llevar a cabo era la de ofrecer una interfaz de usuario que no fuera demasiado pesada y cargante ya que se trata de una aplicación en la que básicamente todo lo que se muestra de cara al usuario es texto plano con todas las posibles opciones a elegir.

Se conservó hasta el final la idea de la interfaz con diseño *dashboard*, ya que muestra de forma muy sencilla y eficaz el acceso a las diferentes posibilidades. De esta forma, el usuario puede desplazarse con un solo *click* por todas las características que ofrece la aplicación.

Se pensó en un diseño un tanto "minimalista" basado en iconos muy sencillos y tonos de aplicación suaves, que no cansaran la vista de los usuarios.

Así pues, tras probar varias combinaciones de colores se comprobó, por medio de su utilización y opiniones de gente desinteresada que ofreció su colaboración, que una combinación de blancos y negros resultaba la más acertada. Dicho aspecto puede verse en la *Figura 6.1*, donde se muestra una captura de pantalla de la aplicación final.

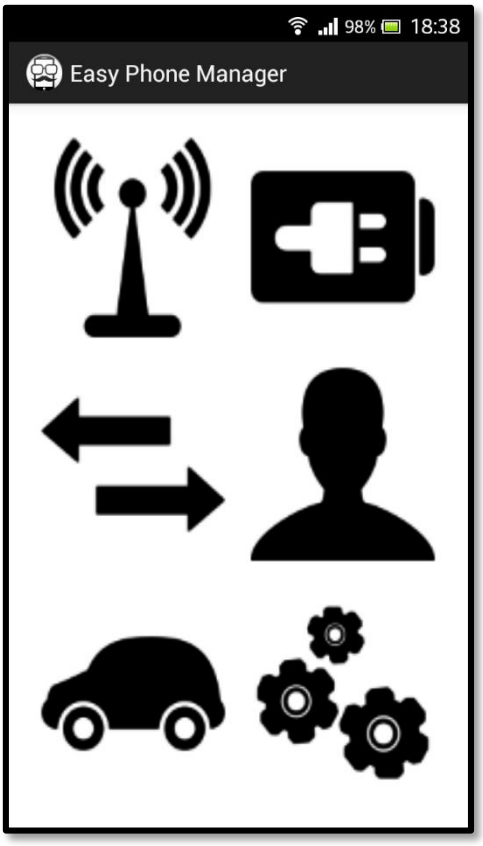

Figura 6.1: Pantalla inicial

6.1.2 Añadiendo una nueva localización

Como se ha visto en la sección de diseño, el usuario podrá añadir nuevas localizaciones GPS para automatizar el funcionamiento del WIFI al estar dentro de su área. La pantalla de administración de localizaciones no ha variado respecto a la idea y bocetos originales. En la *Figura 6.2* se observa el resultado de la misma.

Figura 6.2: Gestión de localizaciones

Aparece una localización añadida, introducida con el nombre "*mi casa"* y con un radio de 300 metros. Si el usuario pulsa sobre el icono de añadir una nueva localización, símbolo '**+**', será redirigido al mapa geográfico de Google.

Por su parte, en la pantalla de añadir la localización sí que ha habido algún pequeño cambio o complemento respecto a la idea original. En primer lugar, cuando el usuario es redirigido al mapa, aparece mostrada por defecto la localización en la que se encuentra mediante un marcador de color azul, como puede verse en la *Figura 6.3.*

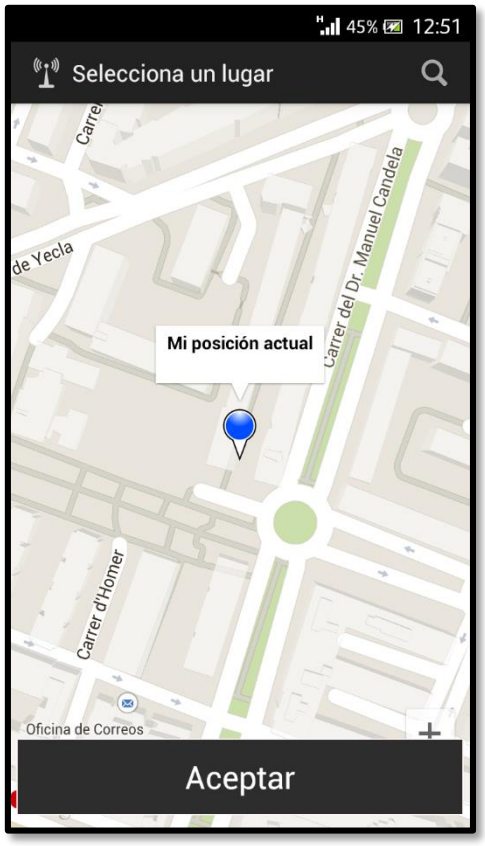

Figura 6.3: Posición actual en el mapa GPS

Esta información puede variar, dependiendo de si el terminal disponía de conexión o no, ya que en realidad almacena la última posición que se pudo descargar de un punto de referencia GPS, y aunque normalmente sí suele ser la posición actual, puede que no lo sea. En tal caso, el usuario deberá volver a atrás en la aplicación y volver a entrar en el mapa para forzar al dispositivo a buscar su posición GPS.

El usuario podrá seleccionar la posición actual para añadirla a su lista de localizaciones GPS o añadir cualquier otro marcador (sólo uno al mismo tiempo) mediante una pulsación prolongada. De esta forma, aparecerá en el mapa un nuevo marcador de color rojo. Véase la figura de ejemplo, *Figura 6.4*.

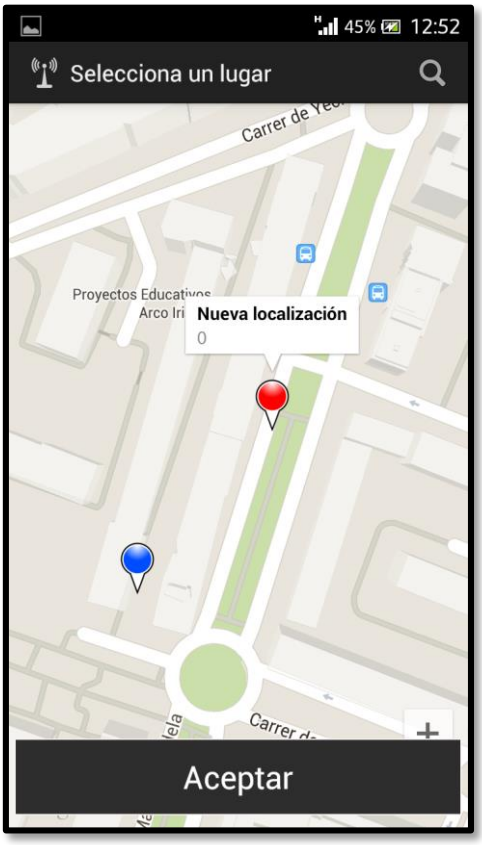

Figura 6.4: Marcador con la nueva localización

Si el usuario pulsa sobre el nuevo marcador fijado aparecerá la información de la localización junto con la información a completar: el nombre de localización y el radio de acción. Se sigue el mismo esquema que la idea inicial, como puede verse en la *Figura 6.5*.

Figura 6.5: Información de la localización

Llegados a este punto es dónde se ha introducido una pequeña mejora para facilitar al usuario la inserción de la localización. Esta mejora viene relacionada con el radio de acción de la misma.

Cuando el usuario introduce los datos necesarios y confirma la selección aparece en el mapa, sobre el marcador de la localización, un círculo con un sombreado gris que permite al usuario comprobar qué partes del mapa abarca aproximadamente ese punto GPS y de esta manera, poder establecer con más exactitud el marcador donde le sea necesario. En la *Figura 6.6* se muestra una captura de pantalla con un ejemplo de inserción de punto GPS.

Si el usuario vuelve a pulsar sobre el marcador y modifica el radio, cada vez que se confirma la selección, el círculo se redibujará mostrando su área actualizada.

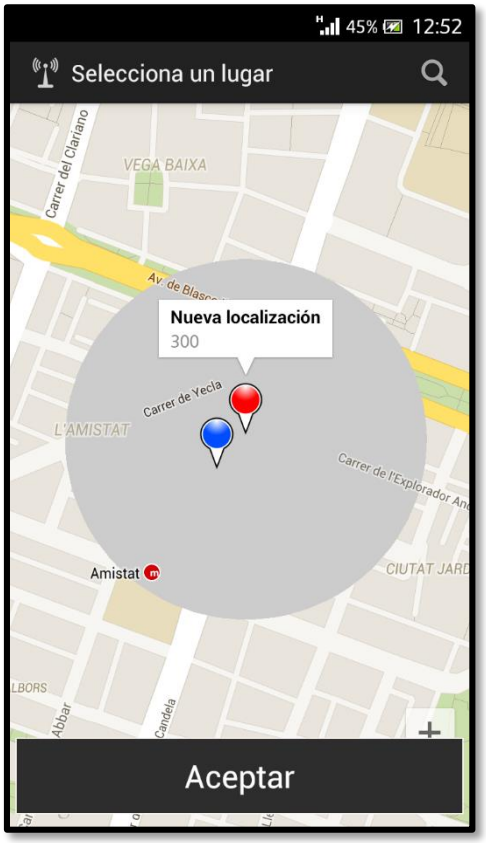

Figura 6.6: Área de acción de una localización GPS

6.1.3 Pantalla de ahorro de batería

En modo de ahorro de batería desarrollado en la aplicación también ha seguido con fidelidad el modelo inicial, como puede verse en la *Figura 6.7*.

Se ha introducido un pequeño cambio. En la parte superior de la figura ejemplo puede verse el icono con el dibujo de una batería. Este icono va variando dinámicamente según el porcentaje de la batería del terminal, cambiando de imagen de acuerdo al siguiente criterio:

- **Batería por encima del 75% de carga:** icono de batería completa, es decir, tres “barras” de carga.

- **Batería entre un 25% y un 75% de carga:** icono de batería con dos “barras” de carga.
- **Batería con menos de un 25% de carga:** icono de batería con una sola “barra”.

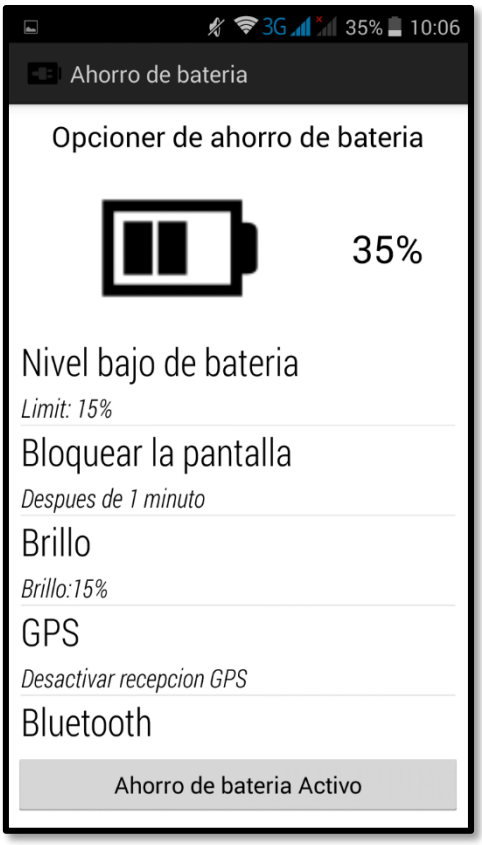

Figura 6.7: Ahorro de batería

6.1.4 Pantalla de perfil de sonidos

Se ha visto en la sección de *diseño* que para que el perfil de sonidos pueda ser activado el usuario deberá escoger una franja horaria. Tendrá dos campos, uno de hora inicial y otro de hora final. Por comodidad cuando el usuario pulsa sobre cada uno de ellos, la aplicación seleccionará la hora actual.

El aspecto que tiene esta parte de la aplicación es el mostrado en la *Figura 6.8*.

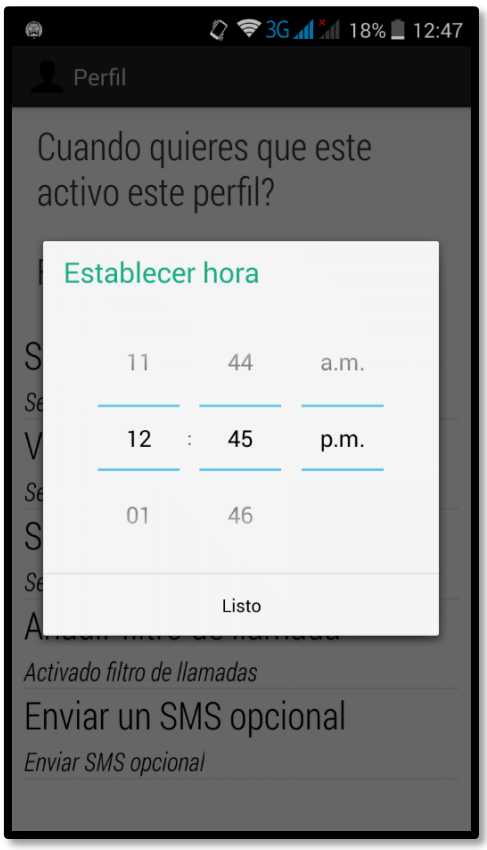

Figura 6.8: Selección de hora en el perfil de sonios

Por otra parte, en la pantalla de administración de los contactos para el filtro de llamadas, (recordar que el usuario podrá añadir cuantos contactos desee de su agenda telefónica para que no se vean afectados por el perfil de sonidos), se ha modificado el método de eliminación de los contactos añadidos.

La modificación ha sido leve y no afecta al comportamiento inicial. En un principio, el usuario debía pulsar prolongadamente sobre un contacto para que pudiera ser eliminado, uno por uno.

En este caso, se ha decantado por un diseño con eliminación múltiple para ofrecer mayor comodidad. Cuando el usuario pulse prolongadamente sobre los contactos aparecerá un cuadro de selección a modo de *checkbox* al lado de cada uno.

De esta manera se podrá seleccionar todos aquellos contactos que se desean eliminar y eliminarlos todos al mismo tiempo al confirmar la selección. Véase el ejemplo en la *Figura 6.9*.

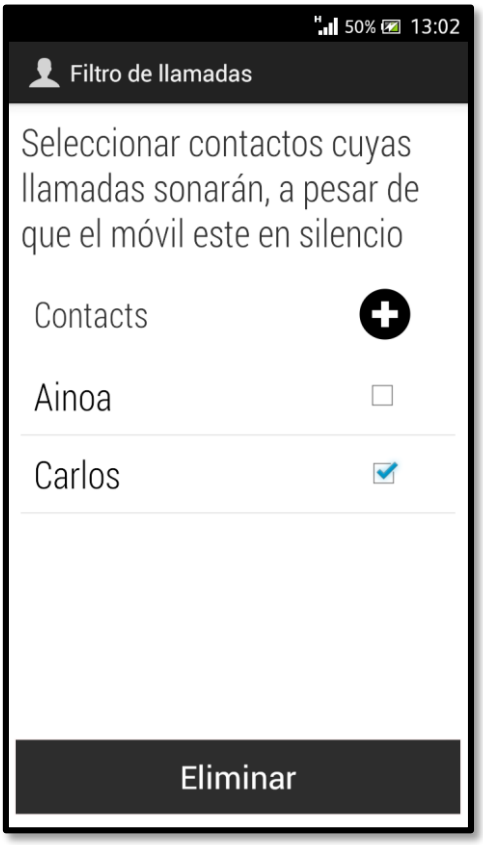

Figura 6.9: Eliminación de contactos en el filtro de llamada

6.1.5 SMS automáticos

El envío de SMS automáticos, tanto en la opción ofrecida en el *modo coche* como en la del *perfil de sonidos,* ha quedado como se muestra en la *Figura 6.10*.

No ha habido ninguna modificación. Se ha añadido un *toast* informativo que muestra al usuario cuando ha llegado a la totalidad de caracteres permitidos en el SMS. Recordar que el tamaño es de 160 caracteres.

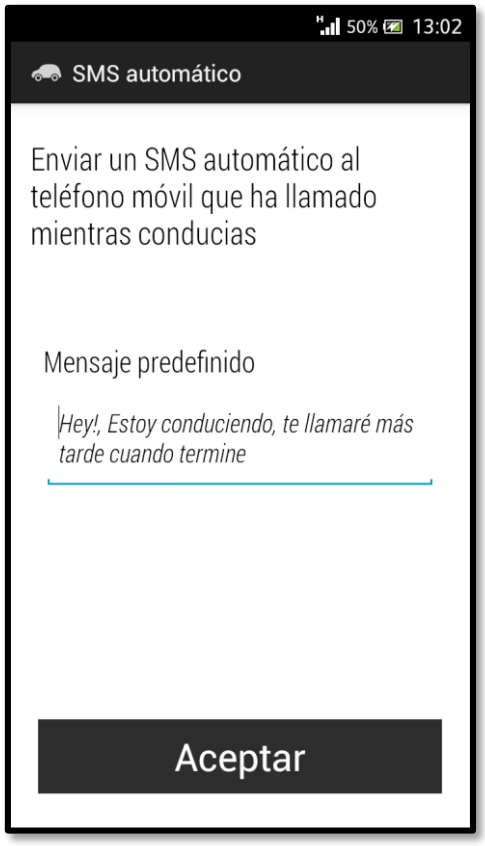

Figura 6.10: Envío de SMS automático

6.1.6 Notificaciones

El aspecto de las notificaciones de la aplicación desarrollada sigue la misma estructura que las notificaciones de cualquier otra aplicación. En ellas aparece el nombre de la aplicación, el icono de la misma y un título informando del proceso en cuestión que la ha ejecutado junto con un subtítulo añadiendo más información u ofreciendo un mayor detalle.

En esta aplicación existen cuatro tipos de notificaciones, que como ya se vio en el apartado de *diseño* son:

- Notificaciones de WIFI según la localización GPS.
- Notificaciones de aviso de ahorro de batería.
- Notificaciones de envío de SMS automáticos, (tanto en el *modo coche* como en el *perfil de sonidos*).
- Notificaciones de rechazo de llamadas en el *modo coche*.

A modo de ejemplo se muestra en la *Figura 6.11* la notificación que aparece cuando el usuario se encuentra dentro de un área de localización GPS, opción ofrecida en la sección de administración wifi de la aplicación.

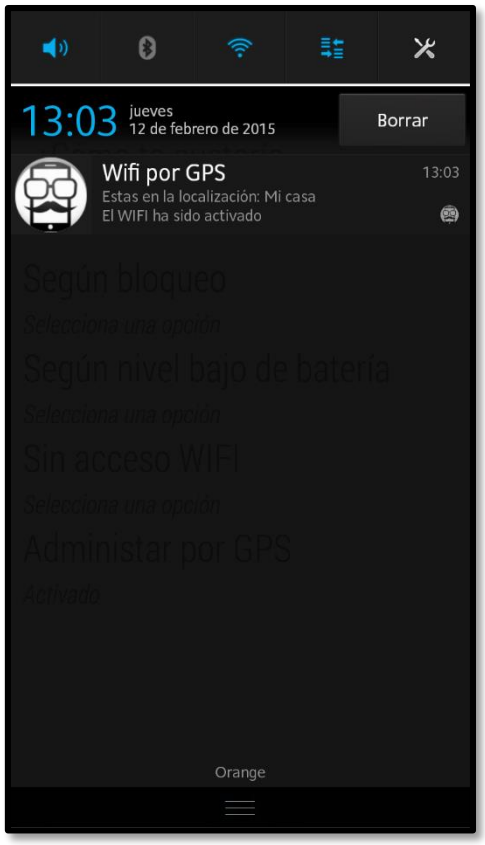

Figura 6.11: Notificación de wifi según localización GPS

Las notificaciones no realizarán acción alguna al ser pulsadas (por ser meramente informativas) y podrán ser eliminadas de la misma manera que se elimina cualquier otra notificación del terminal.

Como se ha comentado, todas ellas poseen el mismo aspecto, tan sólo varía la información mostrada y todas siguen la misma estructura y diseño: título de la aplicación, información del proceso en cuestión que la ha activado y si fuera conveniente, información adicional.

6.2 Pruebas realizadas

En esta sección se presentarán las pruebas llevadas a cabo que demuestran que la aplicación funciona con la fidelidad y eficacia con la que fue diseñada y desarrollada. Estas pruebas corresponden, entre otras, al uso de la aplicación en diferentes versiones de Android, así como en diferentes terminales.

6.2.1 Versiones Android

La solución final de la aplicación ha sido instalada y testeada en diferentes versiones [27] Android, tanto físicas como virtuales. En concreto, las versiones en las que ha sido probada son las siguientes:

- Android *Ice Cream Sandwich* version 4.0. Bajo un entorno virtual (emulador).
- Android *Jelly Bean version* 4.2. Bajo un entorno físico y real (dispositivo móvil).
- Android *Kitkat* versión 4.4. Bajo un entorno físico y real (dispositivo móvil).

Se ha decidido probar la aplicación en estas versiones por ser, cada una de ellas, versiones de Android diferenciadas entre sí en las que ha habido cambios importantes. Además de ello, corresponden con versiones que actualmente están siendo utilizadas por la mayoría de usuarios Android, como puede apreciarse en la figura *Tabla 6.1*.

Tabla 6.1: Uso de las diferentes versiones Android

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%

Se trata de datos recogidos por Google, que realizó un estudio de un periodo de siete días observando desde que terminales se accedía a la tienda de aplicaciones Android *Google Play*.

Así pues, el resultado que se obtuvo fue satisfactorio para todas las versiones mencionadas. La versión *Gingerbread 2.3* no fue siquiera testeada ya que, el desarrollo de la aplicación se realizó pensando en usuarios con terminales cuya versión fuese igual o superior a la *Ice Cream Sandwich* (API 15). Por lo tanto, cualquier usuario con una versión inferior no podrá hacer uso de la aplicación.

Puesto que la versión de Android 2.3 es utilizada por un número considerablemente reducido de usuarios (7.8% atendiendo a la *Figura 6.12*), se optó por desarrollar la aplicación absteniéndose de los mismos. De haber realizado una aplicación apta para todas las versiones Android, el desarrollo de la misma hubiera sido mucho más engorroso y complicado ya que, para poder dotar de determinadas funcionalidades se requería el uso de componentes de compatibilidad y programar con mucha más cautela de la que se necesitaba para conseguir una aplicación compatible con versiones más modernas.

Ejemplos de funcionalidades que se verían afectadas en su desarrollo serían la implementación de notificaciones o el uso del mapa geográfico utilizando la API de Google.

6.2.2 Dispositivos móviles

Al igual que se ha comprobado que el proyecto desarrollado se comporta de manera fiable y eficaz en diferentes versiones de Android, se ha probado su estabilidad y funcionamiento bajo dispositivos móviles con diferentes características de hardware: número de núcleos, RAM, tamaño de pantalla, etc.

De las pruebas realizadas se han podido obtener las siguientes conclusiones:

- **Pantalla:** el tamaño de la pantalla no influye en la experiencia de uso de la aplicación. Se adapta al tipo de pantalla que se dispone. Si la pantalla es grande, los iconos se distribuyen uniformemente y si por el contrario es reducida, la aplicación permite hacer scroll para que el usuario pueda acceder a todas las funcionalidades disponibles.
- **RAM:** la RAM del terminal móvil sí que podría afectar al funcionamiento de la aplicación en ciertos casos. Por las pruebas realizadas, se ha comprobado que si el terminal dispone

de una RAM un tanto escasa, digamos 512 MB, la aplicación podría verse afectada y ser detenida bajo un consumo elevado. A pesar de que la aplicación está dotada con la característica de no ser eliminada por el sistema Android, (como se vio en la sección de *diseño*), no quiere decir que nunca lo hará ya que, bajo sobrecarga de RAM, el comportamiento de Android respecto a la eliminación de aplicaciones en uso puede ser impredecible.

- **Velocidad del procesador:** no se trata de una aplicación pesada y que requiera excesivos recursos, por lo que se comporta mostrando un buen rendimiento con procesadores tanto de uno, dos o cuatro núcleos. Obviamente la transición de las pantallas y el uso, sobre todo del mapa geográfico, mejora cuanto mejor sea la velocidad y el número de procesadores.

6.2.3 Comportamiento inesperado

Todas las funcionalidades de la aplicación han sido testeadas y probadas una a una, comprobando que trabajan de forma correcta y cumplen la finalidad para la que han sido desarrolladas.

Sin embargo, al instalar la aplicación en un terminal y someterla a un uso indefinido, se detectó un problema que provocaba la detección inmediata la de aplicación. Este problema se producía cuando el usuario establecía una llamada telefónica.

Tras revisar el código, se detectó que había un pequeño problema con el *broadcast receiver* que atiende a los eventos, precisamente, relacionados con la recepción de llamadas.

6.2.4 WIFI por GPS y modo coche

La administración de WIFI según las localizaciones GPS, al ser una funcionalidad que ha requerido un mayor tiempo de desarrollo y dedicación, se ha querido comprobar que funcionara correctamente no sólo en el emulador.

Para ello, se han introducido varias localizaciones GPS y se ha desplazado físicamente el terminal hasta estar dentro del área de cada una de ellas para comprobar que, efectivamente el WIFI se activa automáticamente.

El consumo de batería y datos móviles para poder hacer uso de esta funcionalidad no han sido un problema ya que, la comprobación GPS se realiza cada quince minutos y el consumo que realiza apenas se percibe en la vida útil de la batería.

Por otra parte el modo coche, como se ha ido viendo a lo largo de la memoria, incluye una serie de funcionalidades de las que se espera una fidelidad impecable, como son el envío automático de SMS y la aceptación o rechazo de llamadas.

Para realizar esta prueba se ha utilizado también dispositivos físicos además del emulador Android que incorpora el SDK, por una parte para asegurar el buen comportamiento y por otra por la facilidad que supone probarlo en un medio físico, ya que en el emulador Android simular llamadas y envío de SMS a algún terminal en el que se pueda comprobar es un poco más costoso.

De esta manera, se han utilizado dos terminales físicos y se ha comprobado cada una de las funcionalidades del modo coche, obteniendo en todos los casos un resultado satisfactorio.

7. Conclusiones

Como sección final se muestra un resumen del trabajo realizado, repasando todo el desarrollo a lo largo del proyecto así como las tecnologías que se han empleado para la elaboración del mismo. Se indica también una serie de mejoras y propuestas que se podrían tener en cuenta en una versión futura, con la intención de conseguir una aplicación mucho más completa y funcional.

7.1 Resumen del proyecto realizado

El trabajo desempeñado que se ha descrito en esta memoria corresponde con el diseño y desarrollo de una aplicación móvil de gestión y automatización para dispositivos Android, compatible con versiones superiores o iguales a la versión Ice Cream Sandwich 4.0.

Dicha aplicación concederá a los usuarios la posibilidad de automatizar una serie de funciones y características del terminal que por defecto, en cualquier versión de Android, no lo son. Así pues, se podrá automatizar la conexión WIFI y los datos móviles atendiendo a diferentes factores, como por ejemplo: un nivel bajo de batería, el bloqueo de la pantalla o una localización GPS (en el caso de la gestión WIFI).

La aplicación también ofrecerá a los usuarios varias funcionalidades diferenciadas entre sí y destinadas a mejorar la experiencia de uso con sus terminales.

Entre estas funcionalidades se encuentra el *modo coche*, que permitirá a los usuarios mejorar su experiencia al volante por medio de características como el envío de SMS automáticos a aquellas personas que llamen, o la recepción/rechazo de las llamadas entrantes.

Otra de las funcionalidades sería el modo de *ahorro de batería*, que se compone de una serie de opciones que se ejecutan cuando el terminal se encuentra por debajo de un nivel crítico de batería, como reducir el brillo, desactivar el bluetooth, etc.

Como última funcionalidad a mencionar será el *perfil de sonidos*, que permitirá a los usuarios establecer una franja horaria, de manera que todas las opciones que se hayan seleccionado se activarán automáticamente cuando el terminal se encuentre dentro de esa franja horaria establecida.

Para poder desarrollar la aplicación se ha utilizado el kit para desarrolladores de Android (SDK) y ha sido imprescindible la utilización y comprensión del lenguaje Java. Además, ha sido necesario el manejo de SQLite para crear una pequeña base de datos en la que insertar, modificar y eliminar elementos. Para poder acceder a funciones más “privilegiadas” a las que no

se podía acceder únicamente con el SDK de Android, ha sido necesario añadir al proyecto el paquete ITelephony.aidl y utilizar las funciones implementadas en el mismo, que permiten la interacción con una capa más interna del terminal.

Este proyecto, al tratarse de una aplicación para automatizar funciones del terminal, como el WIFI, los datos móviles, gestionar llamadas, etc. ha necesitado de constantes búsquedas de información e investigación para saber cómo se accedía a cada una de los componentes, cómo se activaban, cómo se desactivaban, etc. así como información detallada sobre los servicios de que se ejecutan en segundo plano y de los broadcast receiver que están a la espera de eventos, en cada caso, de lo que se ha ido necesitando.

7.2 Posible trabajo futuro

La aplicación desarrollada no es para nada una aplicación definitiva y que no admita mejoras y ampliaciones que la conviertan en una aplicación todavía más completa. Todo lo contrario, se trata de una aplicación abierta a la que se le pueden ir añadiendo funcionalidades extra de diversos tipos así como mejorar las ya presentes. Así pues, unas cuantas mejoras que podrían considerarse serían:

- **Visualización de localizaciones GPS:** en la sección de automatización del WIFI según una localización geográfica, actualmente se ofrece la posibilidad de agregar en el mapa un marcador con la localización deseada y añadirla a la lista de localizaciones. Sin embargo, una vez introducidas todas las localizaciones deseadas (una a una), no existe la posibilidad de visualizar todas ellas en el mapa. Podría ser de gran ayuda y comodidad disponer en la pantalla de administración de localizaciones de un botón para poder visualizar las localizaciones introducidas hasta el momento. Otra pequeña mejora relacionada con lo mismo podría ser que, en el momento de añadir una nueva localización, el mapa recuerde las localizaciones introducidas, y así evitar introducir dos veces la misma o poner alguna demasiado cerca de otra existente.

- **Widget de aplicación:** ya que la aplicación tiene diferentes modos y métodos de automatización, sería de gran utilizad para los usuarios poder disponer de un widget de la misma. De esta manera, sin tener que abrirla explícitamente, los usuarios tendrían la posibilidad de acceder a una serie de funciones de forma cómoda y sencilla. El widget podría contener, a modo de ejemplo, accesos a la activación/desactivación del modo de *ahorro de batería*, del *modo coche* o del *perfil de sonidos*, además de algún componente visual para saber qué función o funciones están activas de la automatización del WIFI y de datos móviles.
- **Avisar del consumo de datos:** en la parte de automatización de datos móviles, podría ser de interés para los usuarios incluir una opción para poder controlar el consumo que

se está realizando. Por ejemplo, el usuario podría introducir una cifra (en megabytes) de consumo y el terminal lanzaría una notificación avisando de que se ha consumido tal cantidad. En Android existe la posibilidad de limitar los datos móviles, pero en este caso, más que limitar sería una opción para informar.

- **Estructura del perfil de sonidos:** la estructura actual del perfil de sonidos está diseñada de tal forma que el usuario debe introducir la franja horaria en la cual se activará cualquiera de las opciones seleccionadas. Una posible modificación podría ser la de ofrecer directamente al usuario varios tipos de perfiles predeterminados (siempre configurables), que le faciliten la elección de sus necesidades. Por ejemplo, si el usuario seleccionara un perfil nocturno, por defecto se desactivaría los sonidos desde las 10 p.m. hasta las 8 a.m. Y de esta forma añadir diferentes perfiles: diario, nocturno, reunión, etc.
- **Mejorar el envío de SMS:** tanto en el *modo coche* como en el *perfil de sonidos* existe la posibilidad de enviar un SMS predeterminado a cualquiera de los teléfonos móviles que han llamados cuando el perfil estaba activo. Sería un gran avance y una utilidad poder incorporar la opción de enviar el texto informativo a través de la conocida y expandida aplicación *WhatsApp*. De esta manera, el usuario no gastaría dinero por el envío de SMS, además que, actualmente la mayoría de usuarios utilizan la mencionada aplicación como aplicación predeterminada para comunicarse entre sí.
- **Mejorar la estética:** a pesar de que se ha buscado ofrecer un diseño sencillo e intuitivo, el diseño actual posee una interfaz gráfica no demasiado estética. Se podría mejorar mediante la inserción de combinaciones de colores e iconos más llamativos y estilosos, que no hagan perder la esencia de su diseño: ser una aplicación que no sea pesada y cargante a la vista de los usuarios.

8. Bibliografía

[1] Wikipedia, «Telefonía móvil,» 8 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Telefonía_móvil.

[2] Wikipedia, «Teléfono inteligente,» 1 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Teléfono_inteligente.

[3] Wikipedia, «Iphone,» 25 febrero 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/IPhone.

[4] Wikipedia, «Plataforma iOS,» 8 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/IOS#Caracter.C3.ADsticas.

[5] Wikipedia, «Plataforma Android,» 3 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Android.

[6] Wikipedia, «Plataforma Windows Phone,» 3 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Windows_Phone.

[7] Wikipedia, «Sistemas operativos móviles,» 28 febrero 2015. [Online].
Disponible: http://en.wikipedia.org/wiki/Mobile_operating_system.

[8] Wikipedia, «Java,» 4 marzo 2015. [Online].
Disponible: http://en.wikipedia.org/wiki/Java_%28programming_language%29.

[9] Wikipedia, «Objective-C,» 6 marzo 2015. [Online].
Disponible: http://en.wikipedia.org/wiki/Objective-C.

[10] Wikipedia, «Programación por capas,» 17 noviembre 2014. [Online].
Disponible: http://es.wikipedia.org/wiki/Programación_por_capas.

[11] Wikipedia, «Actor,» 22 febrero 2014. [Online].
Disponible: http://es.wikipedia.org/wiki/Actor_(UML).

[12] Wikipedia, «Casos de uso,» 10 febrero 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Caso_de_uso.

[13] R. F. Dam, «Mock-ups,» [Online].
Disponible: https://www.interaction-design.org/encyclopedia/mock-ups.html.

[14] Wikipedia, «Base de datos relacional,» 7 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Base_de_datos_relacional.

[15] Wikipedia, «Kit de desarrollo software,» 3 marzo 2015. [Online].
Disponible: http://en.wikipedia.org/wiki/Android_software_development.

[16] Android Developers, «Interfaces de Usuario,» [Online].
Disponible: http://developer.android.com/guide/topics/ui/index.html.

[17] N. Walsh, «A Technical Introduction to XML,» [Online].
Disponible: http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58.

[18] J. Tomás, «Componentes de una aplicación Android,» [Online].
Disponible: http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion.

[19] Wikipedia, «Dropbox,» 6 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Dropbox.

[20] Dropbox, «Dropbox,» [Online].
Disponible: https://www.dropbox.com/tour/1.

[21] Wikipedia, «Git,» 6 marzo 2015. [Online].
Disponible: http://es.wikipedia.org/wiki/Git.

[22] Git, «Características de Git,» [Online].
Disponible: http://git-scm.com/about.

[23] Wikipedia, «GitHub,» 7 marzo 2015. [Online].
Disponible: http://en.wikipedia.org/wiki/GitHub.

[24] GitHub, «Características de GitHub,» [Online].
Disponible: https://github.com/features.

[25] Android Developers, «Services,» [Online].
Disponible: http://developer.android.com/guide/components/services.html.

[26] L. Vogel, «Android SQLite database and content provider,» 19 agosto 2014. [Online].
Disponible: http://www.vogella.com/tutorials/AndroidSQLite/article.html.

[27] Android Developers, «Versiones Android,» [Online].
Disponible: https://developer.android.com/about/dashboards/index.html.

[28] Wikipedia, «Widget,» 24 diciembre 2014. [Online].
Disponible: http://es.wikipedia.org/wiki/Widget.