Proyecto Fin de Máster

Título del proyecto:	Implantación de Plataforma de Visualización de Worldsensing en Dispositivos con Sistema Operativo Android
Titulación:	Máster Universitario de Software Libre
Autor:	Felipe Rodríguez da Silva
Tutor:	Rubén Mondéjar Andreu
Fecha:	6 de Enero de 2011

Índice de contenido

1. Introducción

En este documento se pretende presentar en detalle la elaboración de un proyecto de desarrollo software de la empresa WorldSensing.

Este proyecto de desarrollo se realiza como Proyecto Fin de Máster del Máster en Software Libre cursado por el autor de este documento en WorldSensing a modo de prácticas externas.

WorldSensing ha desarrollado un novedoso sistema para facilitar y reducir el tráfico en las ciudades además de proporcionar una rápida y eficaz localización de plazas de estacionamiento públicas que se encuentren libres a los usuarios de sus servicios.

Este sistema está basado en una red inalámbrica de sensores implantados en el suelo de las plazas de aparcamiento. Estos sensores se encargan de detectar si la plaza se encuentra libre u ocupada por algún vehículo.

La forma en que una persona podrá ver esta información en su teléfono móvil será subscribiéndose al servicio de WorldSensing y descargándose la aplicación de visualización. Esta aplicación de visualización será la que se pretenda desarrollar en este proyecto.

La calidad y prestaciones de los teléfonos móviles actuales permite a los usuarios disponer de gran cantidad de aplicaciones y funcionalidades que hasta hace pocos años sería imposible.

Para un buen manejo y gestión de todas las aplicaciones que pueden correr en un teléfono móvil de última generación es necesario el uso de Sistemas Operativos más potentes y estables.

Con el desarrollo de esta aplicación de visualización WorldSensing pretende ampliar el número de posibles clientes de sus servicios, ya que cada día aumenta el número de empresas de telefonía móvil que están implantando este Sistema Operativo en sus terminales y por lo tanto cada vez habrá más usuarios que puedan gozar de la aplicación.

2. Antecedentes.

2.1. Historia

2.1.1. GNU/Linux en sistemas embebidos.

Cuando se habla de un sistema embebido se está haciendo referencia a un dispositivo que, a diferencia de un ordenador cotidiano, se le prepara para un único uso. Suelen estar diseñado para realizar un conjunto específico de tareas y operaciones.

Por lo general, se les podría considerar como ordenadores de una sóla placa o tarjeta, conocidos como SBC (Single Board Computer).

Estos dispositivos se caracterízan por funcionar con un sistema operativo guardado en la memoria interna de la propia placa y en la mayor parte de los casos son desarrollados por los propios fabricantes del dispositivo.

Esta clase de dispositivos son más usuales de lo que la mayoría de la gente se puede imaginar, buenos ejemplos son los routers, switchs, telefonos móviles, PDAs, microcontroladores, etc

En el pasado, el desarrollo de sistemas embebidos fué llevado a cabo en su mayoría utilizando código propietario escrito en ensamblador. Buena parte del trabajo de los desarrolladores se centraba en escribir el código de los controladores hardware en ensamblador y las interfaces desde cero.

El núcleo Linux, combinado con un conjunto de algunas otras utilidades de software libre, puede ajustarse dentro del limitado espacio hardware de los sistemas embebidos. Estas distribuciones no suelen superar los 2 megaBytes.

El entorno gráfico estándar usado por las distribuciones de Linux es el sistema X-Window, cuya arquitectura Cliente/Servidor permite que cualquier aplicación Linux sea visualizada en un terminal X. Este servidor está compilado para soportar diversos chipsets y múltiples copias de las funciones de dibujo. Además suele tener un tamaño grande y complejo que requiere muchos megabytes.

En la mayoría de los dispositivos embebidos se carece de disco duro y RAM suficiente para ejecutar un servidor X estándar. Fué cuando entonces nació el proyecto Microwindows. Este sistema fué diseñado para permitir el desarrollo de aplicaciones con gráficos de calidad con el mínimo esfuerzo de programación.

En definitiva, GNU/Linux encaja perfectamente en el cambiante mercado de los sistemas embebidos. La avanzada tecnología, la disponibilidad del código fuente, la ausencia de royalties y su amplio soporte lo convierten en una elección de futuro para estos dispositivos.

2.1.2. Sistemas de posicionamiento geográfico.

En la actualidad existen sistemas de posicionamiento geográfico como el GPS que cada vez son más accesibles para todos los públicos.

El GPS es un sistema global de navegación por satélite que permite determinar en todo el mundo la posición de una persona, vehículo, objeto, etc con gran precisión. Se le atribuye su invención a los gobiernos francés y belga aunque fué desarrollado y operado por el Departamento de Defensa de los Estados Unidos.

El GPS funciona mediante una red de 32 satélites en órbita sobre la esfera terrestre que se mueven con trayectorias sincronizadas para cubrir toda la superficie global. En el momento en que se pretende obtener la posición, el receptor (P.ej: teléfono móvil o el del coche) localiza automáticamente un mínimo de tres satélites de la red, de los que se recibe unas señales indicando la identificación y la hora del reloj de cada uno de estos. Teniendo como base estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo y así mide la distancia al satélite mediante el método de triangulación, basado en la medición de las distancias de cada satélite al punto deseado. Con esto los satélites obtienen una posición relativa a sí mismos. Para obtener la posición absoluta del receptor GPS basta con calcularla frente a las posiciones de los satélites.

Uno de los usos más extendidos de los GPS portátiles ha sido para el sector del automóvil. Se han desarrollado GPSs que en realidad son mapas interactivos que muestran rutas y carreteras. Dado un punto de inicio y un punto final el GPS te calcula un recorrido y te va dando información detallada de como alcanzar tu destino. Este dispositivo se ha convertido en una gran herramienta para profesionales del transporte, taxistas o para cualquier persona evitando la necesidad de consultar un mapa en papel durante la conducción o parando para ello.

Existen muchos de estos dispositivos que no sólo ofrecen información sobre rutas, sino que además ofrecen localización de restaurantes, cines, gasolineras cercanas. La situación de estos lugares se guardan en la memoria del receptor, junto con los mapas de carreteras y planos de las calles. Así puede avisar al usuario si está cerca de uno. Los mapas y estos puntons de interés se actualizan a través de los medios que el fabricante del receptor ponga a disposición para ello, como en CD-ROM o en su sitio web.

2.2. Situación actual.

2.2.1. Sistemas de posicionamiento

Hoy en día ya se pueden ver GPSs que además de localizaciones de interés también ofrecen el estado del tráfico. Muchos de los GPS actuales pueden comunicarse con Internet para recibir información actualizada sobre el tráfico o incidencias en la ruta. La comunicación se realiza en generál a través del teléfono móvil. El receptor GPS se conecta con un servidor para recibir datos en tiempo real. Así puede avisar de una calle cortada, atascos en hora punta y poder buscar una ruta alternativa.

Existen fabricantes de programas de cartografía para GPS que disponen de servicios de información de tráfico. Con algunos de ellos, como la empresa holandesa TomTom, es necesario realizar suscripción y pagar una cuota anual. Otros ofrecen el mismo servicio sin necesidad de cuotas, esta estaría ya incluída en el precio de compra del programa de navegación.

2.2.2. Android

En lo que a sistema operativo se refiere, han sido muchos los fabricantes de teléfonos móviles los que han utilizado, como sistema operativo, una versión de Linux embebido. Empresas como Samsung, Motorola, HTC, Nokia, LG, Siemens han confiado en las prestaciones y potencia que este sistema ofrecía.

También cabe recalcar que gracias al enorme avance tecnológico sufrido por el mercado de los dispositivo móviles, desembocaría en la necesidad de desarrollar sistemas operativos más especializados en esta clase de arquitecturas y es así como hoy en día existen diversos Sistemas Operativos especiales para teléfonos móviles como el Symbian, Windows Mobile, IOS o el Android.

Será este último el elegido como plataforma que soportará la aplicación de visualización de WorldSensing.

Este Sistema Operativo está basado en GNU/Linux por lo que goza de toda su estabilidad y seguridad. Grandes empresas como HTC, Samsung, Motorola o Google están detras de su desarrollo, mejora y promoción.

Android nació como un proyecto de la ya desaparecida, Android Inc. Esta pequeña empresa fué adquirida por Google en 2005. Los fundadores de esta empresa fueron contratados por Google como desarrolladores, los cuales crearon un sistema operativo basado en Linux que mostró a los operadores de telefonía móvil como un sistema flexible y actualizable.

En diciembre de 2006 tanto la BBC como el Wall Street Journal publicaron que Google estaba apostando fuerte para que su buscador y sus aplicaciones funcionasen en los teléfonos móviles.

Google fué dando pasos hasta que en noviembre de 2007 la [1]Open Handset Alliance se marcó el objetivo de definir los estándares abiertos para los dispositivos móviles. Google liberó la mayoría del código de Android bajo licencia Apache, la cual es una licencia libre y de código abierto.

En octubre de 2008 sale al mercado el primer teléfono que corre sobre Android. Era el G-1 de T-Mobile, sólo distribuido en EEUU y Reino Unido.

Aunque han existido versiones previas, es a partir de la 1.5 que se empieza a tener en cuenta a Android como un sistema operativo realmente potente. Las versiones más conocidas han sido:

- Android 1.5, conocida como ***Cupcake***. Debido a las grandes mejoras introducidas en la tercerca versión de Android, de abril de 2009, se saltó directamente a la versión 1.5. Esta versión poseía el núcleo Linux 2.6.27 y mostraba novedades como un completo rediseño del interfaz gráfico, transiciones animadas entre ventanas, mejoras en la velocidad de la cámara, reducción del tiempo de búsqueda de los satélites GPS, inclusión de intérprete de JavaScript, etc.

- Android 1.6, conocida como ***Donut***, se lanzó en septiembre de 2009 y se la considera una actualización menor debido a que se basaba fundamentalmente en la corrección de errores presentes en la anterior. También es cierto que las correcciones propiciaron la aparición de algunas novedades como una nueva pantalla para controlar la batería, que permitía comprobar qué aplicaciones y servicios son los que más consumen, o un nuevo motor de texto a voz.

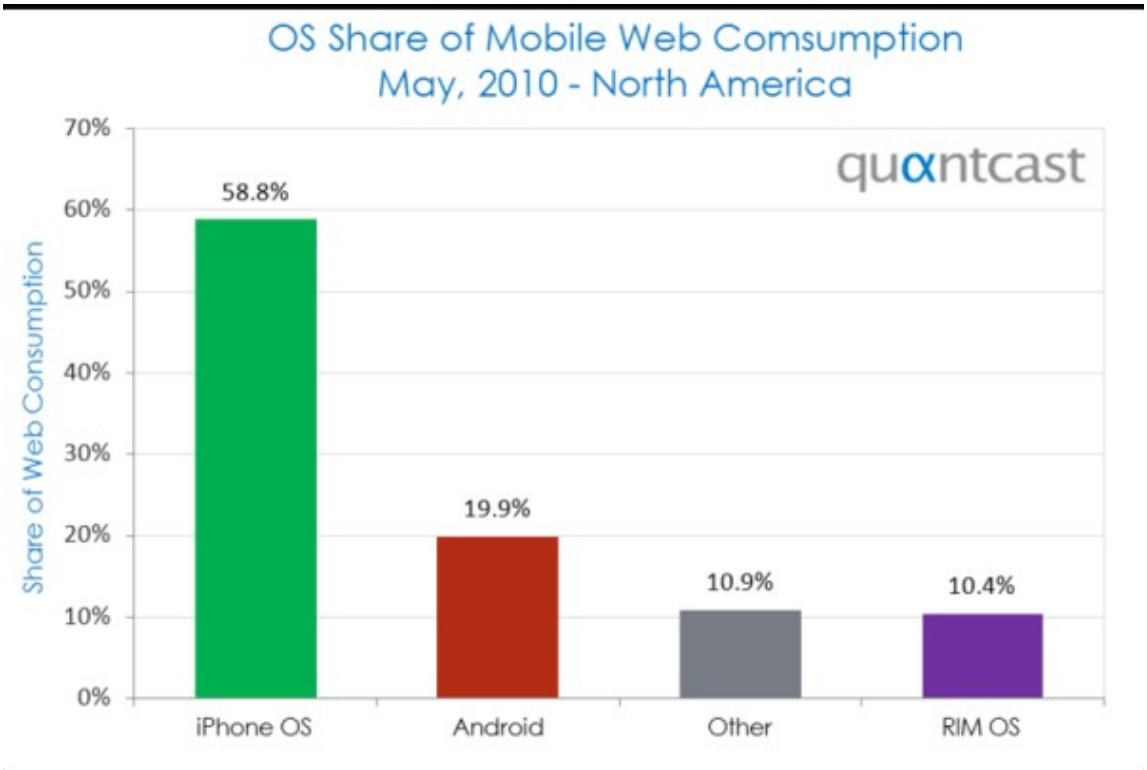
- Android 2.0, conocida como ***Éclair***, y publicada en noviembre de 2009 cuyas novedades se centraban en el rediseño del interfaz del navegador, soporte nativo de flash para la cámara, mejoras del teclado virtual, soporte nativo de Facebook, etc.

1 La Open Handset Alliance es una alianza comercial de más de 70 compañías para desarrollar estándares abiertos para dispositivos móviles. Algunos miembros son Google, HTC, Dell, Intel, Motorola, Texas Instruments, etc

- Android 2.2, conocida como ***Froyo***, fué lanzada en junio de 2010 y cuyos cambios más novedosos son actualizaciones automáticas para aplicaciones, funcionalidad de Radio FM, compatibilidad con Wi-Fi IEEE 802.11n, compatibilidad con Adobe Flash 10.1, compatibilidad con la API gráfica de OpenGL Embedded Systems 2.0, etc.

- Android 2.3, conocida como ***Gingerbread***, es la última versión estable. Esta versión ha sido lanzada recientemente, el 6 de diciembre de 2010. El primer móvil comercializado que tiene esta versión de fábrica es el Nexus S y los dispositivos previstos para su lanzamiento en los primeros trimestres de 2011. Algunas de las novedades que se muestran son mejoras en las funcionalidades de cortar, copiar y pegar, mejora la interconexión con las redes sociales, reconocimiento facial, gestor de descargas, etc.

La cuota de mercado de los sistemas operativos para dispositivos móviles sigue estando liderada por el IOS de Apple. La empresa dedicada a seguimientos en internet, QuantCast, publicó en mayo de 2010 un estudio sobre la cuota de mercado actual de sistemas operativo para móviles en Estados Unidos.



Pese a que se pueda observar un amplio dominio por parte del IOS, con un 60% de cuota de mercado, esta es una cifra engañosa, puesto que no se trata del mucho o poco terreno que tenga cada sistema operativo en el mercado, sino de lo rápido que ha cambiado el panorama en los últimos meses. El verdadero ganador en este caso es Android.

La empresa QuantCast está comparando la cuota de mercado de sistemas operativos para móviles, y eso también supone incluir la navegación en Internet que se hace desde el Ipod Touch y el Ipad. Con todo esto se puede llegar a contabilizar hasta 18'3 millones de dispositivos funcionando con iOS, aunque si sólo se tienen en cuenta teléfonos esta cifra se reduce a 10'7 millones.

En términos de crecimiento, el ganador es Android. El motivo es que este sistema operativo ha aumentado rápidamente sobre los meses anteriores, al pasar de cerca de 5% en enero de 2009 a un 20% en mayo de 2010, habiendo un total de 8'7 millones de dispositivos móviles utilizándolo, cifra que se encuentra muy cercana a la de móviles que utilizan iOS. Este crecimiento ha propiciado una reducción del 75% al 59% de la cuota de mercado del sistema operativo de Apple.

El mercado de las aplicaciones móviles crece año tras año. Se espera que el número de aplicaciones descargadas crezca de 7 mil millones, en 2009, a 50 mil millones en 2012.

Los beneficios obtenidos en 2009 por aplicaciones móviles se estimaron en 4,1 mil millones de dólares y se espera que en 2011 ya haya superado los 18 mil millones.

3. Estudio de Viabilidad.

La empresa WorldSensing promueve la innovación para el desarrollo de aplicaciones basadas en redes de sensores inalámbricos embebidos.

Esta empresa hace de integrador y desarrolladores de las tecnologías más actuales en el sector emergente de las TIC, inversión en desarrollo y promoción de soluciones innovadores con alto capital tecnológico.

WorldSensing ha desarrollado un servicio denominado FastPrk con el cual pretende facilitar a sus usuarios la localización de plazas de estacionamiento en las ciudades. Además de aliviar esta tediosa tarea para los conductores también se reducirá el tiempo de conducción innecesario que se realiza durante la búsqueda de aparcamiento consiguiendo una reducción de las emiciones de $CO_{2.}$

El FastPrk ofrece una plataforma completa de las TIC para mejorar la movilidad urbana y reducir su impacto sobre el medio ambiente, sobre la base de información de alta precisión en tiempo real de plazas de aparcamiento y tráfico en las calles. Con FastPrk lo que se pretende es poner a disposición de los ciudadanos toda esta información por medio de aplicaciones para teléfono móvil y/o paneles en la calle.

Este novedoso sistema incluye detección, seguimiento y control de espacio de infraestructuras de estacionamiento para entornos urbanos. En adición de plataformas para gestionar la información obtenida y proporcionar esta información a los usuarios finales y autoridades de gestión.

La arquitectura de detección de FastPrk incluye pequeños sensores de bajo consumo de energía instalados. Cada nodo sensor detecta la ocupación de la zona de aparcamiento.

Permiten un seguimiento detallado y el control de disponibilidad de plazas de aparcamiento urbano. El sistema es aplicable a cualquier tipo de espacio de estacionamiento al aire libre. Los sensores están conectados en una red inalámbrica que recoge datos de los sensores e informa a los ciudadanos.

La comunicación se aprovecha al máximo de los paradigmas de redes de sensores inalámbricos adaptada a las necesidades de los escenarios urbanos.

Estas características hacen que la comunicación FastPrk de redes de sensores sea robusta, estable y aplicable a los entornos urbanos de forma realista.

Hoy en día las operadoras de telefonía móvil cada vez ofrecen más facilidades para poder conseguir teléfonos móviles de última generación, conocidos como SmartPhones. Estos SmartPhones tienen cada vez más funcionalidades similares a las de un PC genérico y por lo tanto sus Sistemas Operativos son cada vez más complejos.

Uno de los Sistemas Operativos para dispositivos móviles más extendidos es el Android. Este sistema operativo es una adaptación de GNU/Linux para dispositivos móviles. Su desarrollo lo lleva a cabo una alianza formada por distintos fabricante de móviles, como HTC o Motorola, y de software como Google.

Aunque existen kits de desarrollo para hacer aplicaciones en C o C++, el principal lenguaje de desarrollo de aplicaciones para Android es Java.

El desarrollo de este proyecto nace a través del interés de la empresa WorldSensing de ampliar su abánico tecnológico, proporcionando servicios a usuarios de SmartPhones con Sistema Operativo Android.

3.1. Contexto.

Este proyecto nace de la necesidad de mejorar la circulación de los vehículos en las calles de las ciudades. En muchas ocasiones las personas suelen pasar mucho tiempo recorriendo calles o dando vueltas a manzanas en busca de una plaza de aparcamiento.

Cuando se está en esta situación se suele mover con su vehículo de forma más lenta para poder obtener un mayor margen de maniobra en el momento en que se vea una plaza. Esta conducción lenta entorpece el tráfico haciéndolo denso y poco fluído, llegando en ocasiones formar caravanas.

Además del impacto en la fluidez del tráfico se puede observar un gran impacto en términos medioambientales, ya que esta conducción con marchas cortas, conocidas como marchas de fuerza, conlleva un mayor consumo de combustible y por lo tanto el efecto contaminante de la conducción es todavía mayor.

3.2. Fundamentos teóricos.

El principal avance que se quiere mostrar en este proyecto frente a otros ya conocidos es que se va más allá de la información del tráfico y también se muestra dónde se puede aparcar.

Cada día hay más coches en las ciudades y el número de plazas de aparcamiento no aumenta por lo que encontrar una plaza cada vez se convierte en una tarea más ardua.

Con el sistema que WorldSensing propone, conectándose a un servidor de internet a través del teléfono móvil, podrá obtener información precisa de qué plazas de aparcamiento hay libres cerca, al rededor suya, evitando el dar vueltas y realizar recorridos inncesarios.

Una red de sensores inalámbricos instalados instalados en las ciudades se podrá saber dónde hay plazas libres. Se podrá encontrar un sensor por cada plaza de aparcamiento y este estará emitiendo una señal para notificar su estado, ocupado o libre.

Una gran ventaja es que estos sensores están basados en tecnologías inalámbricas por lo que supone un gran ahorro en el coste de instalación. En sistemas conectados por cable supone un gran coste de instalación debido a la necesidad de habilitar el medio urbano para introducir el cable y toda la obra que es necesaria llevar a cabo. Los sensores inalámbricos suponen un ahorro en coste de instalación, material y tiempo.

Estos sensores se comunican entre ellos hasta que alcanzan la puerta de enlace. Llegado a este punto un servidor recibe toda esta información y la procesa, almacena y muestra. A través de algoritmos de control se envía esta información ya procesada a paneles en las calles o a los terminales móviles de los conductores que se haya suscrito al servicio.

Una de las mayores ventajas es que tan sólo será necesario ser usuario de un Smartphone y por lo tanto no será necesaria la adquisición de ningún otro dispositivo adicional. Descargándose una sencilla aplicación el usuario suscrito podrá ver en un mapa su localización y la de las plazas que se encuentren más cerca o también consultar las plazas en una zona determinada a la que se quiere dirigir.

Con todo esto los ciudadanos se podrán ver beneficiados, tanto directamente aquellos que estén suscritos ya que verán un ahorro en dinero y tiempo, como indirectamente para el resto de personas con una menor contaminación y una reducción del precio del crudo.

4. Justificación del proyecto.

La elaboración de este proyecto se fundamenta en los siguientes puntos:

- **Aptitud del alumno**: el primer objetivo que tiene la elaboración de este proyecto es demostrar la capacidad del alumno, autor de este documento, a la hora de elaborar y desarrollar proyectos de índole software una vez finalizado el Máster en Software Libre por la Universitat Oberta de Catalunya. Este trabajo se realizará como prácticas externas en la empresa WorldSensing, con el que se pretende trasladar la plataforma de visualización que utiliza la empresa a dispositivos móviles con sistema operativo Android.

- **Innovación**: uno de los puntos más importantes es la novedad que supone el desarrollo de este proyecto. Hasta el momento se conocían sistemas que daban información acerca del tráfico en las ciudades o la disponibilidad de plazas de aparcamiento en parkings privados. Con este proyecto se pondrá a disposición de los ciudadanos un sistema que proporciona al usuario información sobe la localización de plazas de estacionamiento públicas.

- **Evolución**: haciendo uso de los avances tecnológicos más actuales se ha podido desarrollar un sistema que provee a los ciudadanos de información sobre tráfico y la localización de plazas de estacionamiento públicas que se encuentren disponibles. Este sistema se basa en una red de sensores inalámbricos de bajo consumo instalados en las ciudades. Por otro lado, aprovechando el auge de la telefonía móvil, la aparación de los SmartPhones y la mejora de los sistemas operativos para móviles, con este proyecto se pretende desarrollar una aplicación que corra en estos dispositivos y de esta manera ofrecer a los usuarios de estos móviles el poder acceder a la información obtenida de la red de sensores.

- **Economía**: como se ha mencionado en el punto anterior, la red de sensores que darán la información sobre las plazas de estacionamiento, está basada en sensores inalámbricos de bajo consumo. Esto evita la necesidad de grandes obras en las urbes, con la necesidad de levantar el suelo para soterrar el cableado, etc, evitando los costes que conlleva y el empeoramiento del tráfico. Por otro lado el software que se pretende desarrollar en este proyecto consiste en una sencilla aplicación para Android que mostrará información de tipo geográfico por pantalla con ayuda del API de Google Maps para Android por lo que también los usuarios se podrán ahorrar el coste económico de licencias software. La aplicación a desarrollar estará basada en código libre y utilizará el API de Google Maps para Android que Google proporciona de forma gratuita. Además se liberará finalmente como libre por lo que se estará colaborando con el mundo del Software Libre ofreciendo una solución basada en estándares.WorldSensing se dedica a ofrecer un servicio a través del cual los usuarios podrán subscribirse para obtener la información sobre la localización de los sensores. Sus beneficios no nacen directamente de la venta de licencias de software sino prestando sus servicios al mayor número de usuarios posibles.

- **Necesidad**: este proyecto nace de la constante problemática que se topan los conductores con el tráfico diario en las grandes urbes y la dificultad de localizar plazas de estacionamiento. Con el sistema implantado por WorldSensing el usuario sabrá dónde localizar una plaza o dónde no debe buscar y evitar los tiempos muertos de la búsqueda. Por otro lado el hecho de desarrollar una aplicación de tanta utilidad para el sistema operativo Android, hará que este sea más atractivo para los usuarios frente a otros sistemas operativos propietarios. Gracias a los avances conseguidos por el equipo de desarrollo de Android y a la elaboración, por parte multitud de empresas y programadores, de aplicaciones para este sistema operativo, está consiguiendo que una solución basada en software libre esté creciendo en el mercado de la telefonía móvil a un ritmo mucho mayor que las soluciones de código propietario.

5. Análisis de requisitos.

El servicio FastPrk de WorldSensing sólo se servirá en aquellas ciudades en las que se encuentre instalada toda la infraestructura necesaria para el servicio.

WorldSensing ofrecerá el servicio a todo aquel que se lo solicite, siendo responsabilidad de este asegurarse que el servicio se ofrece en la ciudad dónde quiere hacer su uso.

Con el desarrollo de esta aplicación, WorldSensing pretende ampliar su "target" y poder alcanzar un mayor número de clientes, ya que está orientada a usuarios de teléfonos móviles con Sistema Operativo Android.

La elección de Android como plataforma sobre la que correrá la aplicacion tiene dos razones fundamentales:

- Es un Sistema Operativo basado en GNU/Linux y se presenta como una buena alternativa basada en software libre frente a otros sistemas propietarios como el IOS de Apple. Del mismo modo el código de la aplicación, desarrollado con Java, una vez finalizado será licenciado, en un principio, bajo licencia LGPL para poder contribuir así a la comunidad de Software Libre.

- Android cada vez está más implantado en dispositivos móviles. En un principio era una "insignia" de HTC, una de las empresas de teléfonos móviles que colaboran en el desarrollo y mejora de Android, pero actualmente otras marcas como Motorola y Samsung también lo están implantando en sus dispositivos como muestra de confianza a la estabilidad y seguridad de este Sistema Operativo.

5.1. Requisitos de Usuario.

Es necesario que el usuario disponga de un dispositivo móvil con sistema operativo Android. Además el móvil, como mínimo deberá tener conexión con internet, bien sea por wifi o 3G, ya que será a través de internet que la aplicación obtendrá información, tanto de su propia localización, el mapa de Google Maps o la localización de las plazas de aparcamiento.

Para obtener la localización del dispositivo móvil, cierto es que bastaría con tener conexión a internet, pero sería conveniente, para una localización más precisa, tener incluído un sistema de GPS.

Pese a que la aplicación estará hecha con código libre, no será suficiente con descargarse la aplicación en el móvil, sino que también será necesario suscribirse al servicio que WorldSensing ofrece.

La información que la aplicación proporcione será la localización de plazas de aparcamiento libres en zonas de aparcamiento públicas y el estado del tráfico por medio de la red de sensores inalámbricos que la empresa WorldSensing tiene instalada en las ciudades.

WorldSensing a través de estos medios ha querido ofrecer una solución a la tediosa tarea de buscar aparcamiento en una ciudad proporcionando un servicio que ofrezca información detallada de la localización de plazas de aparcamiento y estado del tráfico.

Una vez que los usuarios del servicio tienen información de dónde puede estacionar su vehículo acudirán de forma más rápida al lugar indicado evitando los tiempos de búsqueda y pudiendo alcanzar su destino a través de rutas más liberadas de tráfico denso. Con esto se reducirá el tiempo que el motor esté en marcha consumiendo combustible innecesariamente, por lo que el impacto medioambiental será menor.

No sólo el medioambiente se verá beneficiado, si no que también lo podrán notar los transeuntes que se encuentren por la calle ya inhalarán una menor cantidad de gases tóxicos.

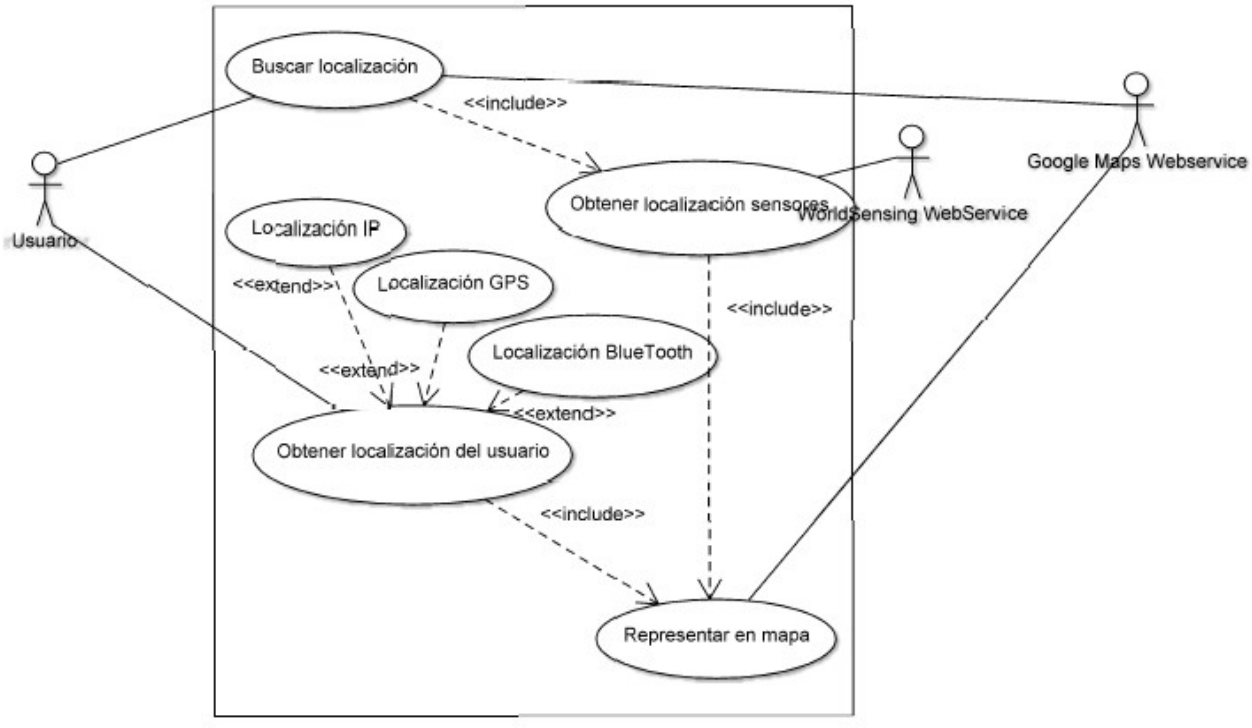
5.2. Casos de Uso.

En el aspecto software, la aplicación constará de un interfaz que ocupará toda la pantalla del dispositivo móvil. Esta pantalla estará ocupada principalmente por el mapa, dejando una barra en la parte superior que ocupará todo el ancho de la pantalla.

Esta barra constará de tres elementos:

- Un campo de texto dónde poder introducir la localización que se desea buscar en el mapa.
- Un botón para efectuar las búsquedas.
- Un botón GPS que situaría la localización del usuario del dispositivo móvil sobre el mapa.

Es necesario que a medida que el usuario se mueva y cambie su localización su posición se vaya actualizando en el mapa. Además se deberá redibujar la localización de los sensores de WorldSensing en el mapa a medida que vayan apareciendo o desapareciendo del encuadre realizado por el zoom.



El API de Google Maps proporciona una serie de servicios web como una interfaz para solicitar servicios externos de este API y utilizarlos en aplicaciones de Google Maps. Los servicios se han diseñado para utilizarse junto a un mapa, tal y como se establecen en las Restricciones de Licencia de las condiciones del servicio del API de Google Maps.

Las búsquedas se limitan a resultados que puedan ser obtenidos en Google Maps, esto implica que no se representarán elementos que Google Maps no identifique o no localice.

Por otro lado la función de localización del dispositivo en el mapa sólo funcionará en zonas y áreas para los que esté habilitado el GPS, es decir, dónde funcione la licencia.

6. Especificación de Requisitos.

A continuación se definirá la especificación de requisitos de sistema para la aplicación a desarrollar. Estos requisitos se dividen fundamentalmente en: Requisitos Funcionales y Requisitos no Funcionales.

A continuación se detallará cada uno de ellos.

6.1. Requisitos Funcionales.

Para la aplicación a desarrollar el usuario dispondrá de un interfaz gráfico, el cúal está formado por una sencilla barra de herramientas en la parte superior y en el resto del interfaz habrá un mapa, obtenido de Google Maps.

En la barra de herramientas tendrá un campo de texto para introducir una dirección o localización, un botón para efectuar la búsqueda de la localización y un botón GPS que se situará en el mapa la localización del usuario.

Cuando se realiza la búsqueda de una localización, a su vez la aplicación solicitará la posición y estado exactos de los sensores que tenga por la zona presentada a través de un servicio web de WorldSensing.

Cuando se pulsa el botón GPS, la aplicación mostrará la localización del usuario sobre el mapa. Para esto recurrirá al mejor proveedor de localizaciones que disponga el dispositivo móvil. Es necesario hacer incapié en que para que esta función tenga efecto, el móvil donde correrá la aplicación debería tener al menos un proveedor de localización, como Wifi, 3G, GPS, etc.

Ambas acciones, GPS y búsqueda de localizaciones, interactuarán con los servicios proporcionados por Google Maps para poder representar la información en el mapa.

6.2. Requisitos no Funcionales.

De esta aplicación se espera que:

- Al estar visualizando la localización del usuario en el mapa y este se mueva, dicha modificación se refleje en tiempo real en el mapa mostrado por la aplicación.

- Si se realiza un zoom del mapa, que incluya o excluya la localización de los sensores en el marco del mapa, según sea un zoom de alejamiento o de acercamiento.

- Muestre como mínimo la localización del usuario en el mapa, sea cual sea el método para obtenerla.

- Para obtener la localización y estado de los sensores de aparcamiento es necesario solicitar la suscripción al servicio FastPrk a WorldSensing. Las condiciones de esta suscripción son competencias exclusivas de WorldSensing.

- El servicio FastPrk de WorldSensing sólo se servirá en aquellas ciudades en las que se encuentre instalada toda la infraestructura necesaria para el servicio.

- WorldSensing ofrecerá el servicio a todo aquel que se lo solicite, siendo responsabilidad de este asegurarse que el servicio se ofrece en la ciudad dónde quiere hacer su uso.

- Las búsquedas se limitan a resultados que puedan ser obtenidos en Google Maps, esto implica que no se representarán elementos que Google Maps no identifique o no localice.

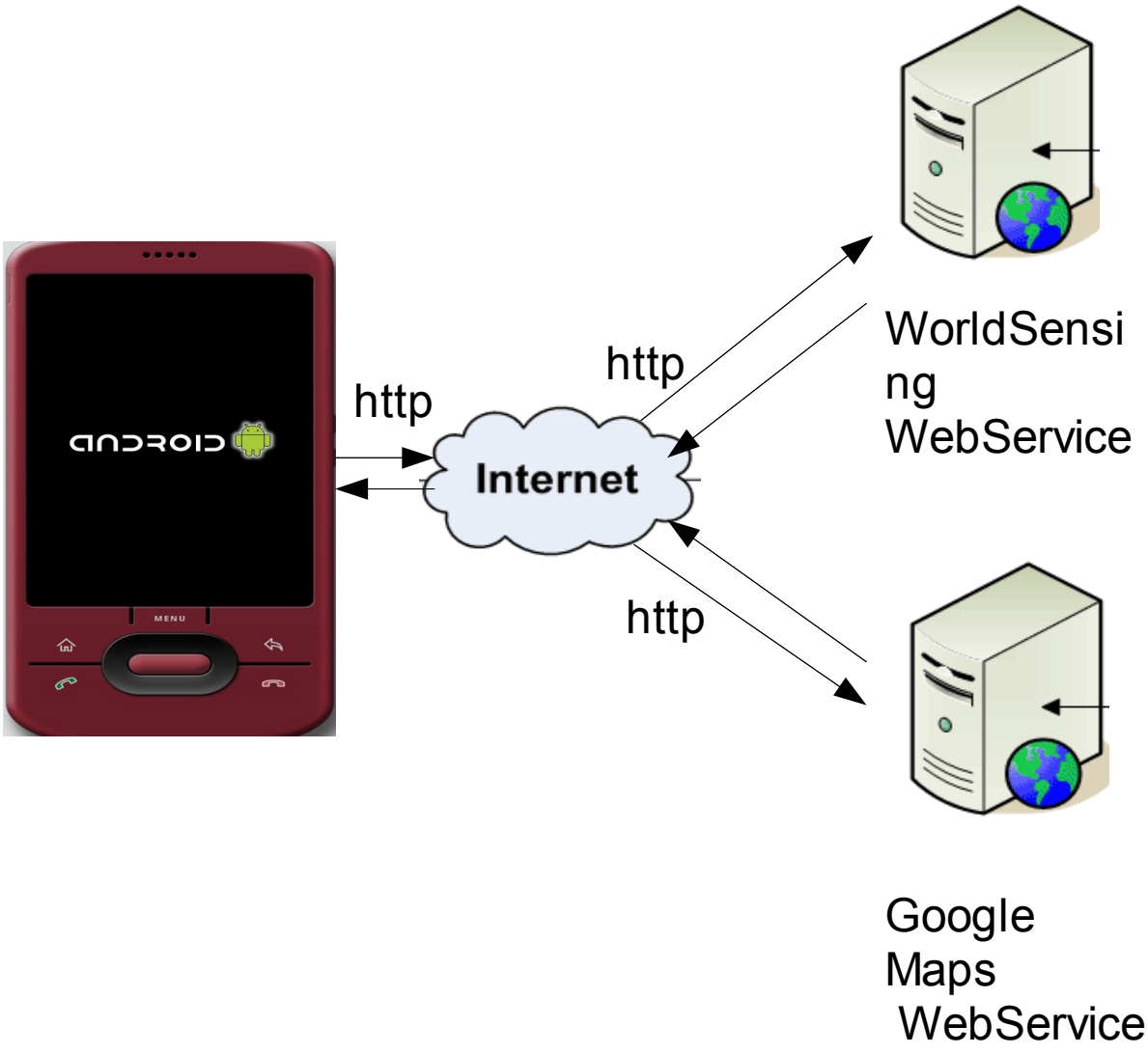
- La función de localización del dispositivo en el mapa, para el caso de que esta se obtenga mediante GPS, sólo funcionará en zonas y áreas para los que esté habilitado el GPS, es decir, dónde funcione la licencia.

- La aplicación estará hecha con código libre, por lo que su liberacíón también será bajo licencias libres. Su descarga será gratuita. WorldSensing obtendrá beneficios de los servicios que preste a los usuarios.

7. Diseño.

7.1. Arquitectura y comunicación de la aplicación con el exterior.

La arquitectura de la aplicación funcionaría de la siguiente manera:



Los WebServices o servicios web eran conocidos como un sistema softwarediseñado para soportar operaciones e interacciones maquina a maquina sobre una red. Así mismo se les considera como un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

Estas aplicaciones, que pueden estar escritas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma, pueden hacer uso de los servicios web para intercambiar datos en redes de ordenadores tal y como es Internet. Esta interoperabilidad se consigue gracias a la adopción de estándares abiertos.

Al apoyarse en HTTP, los Webservices pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado. Además permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedas ser combinados facilmente para proveer servicios integrados.

Aunque muchos servicios web son de pago, los webservices de Google Maps son gratuitos, disponibles para todo el mundo. Tan sólo es necesario solicitar una clave para poder utilizar la aplicación.

Estos servicios web envían solicitudes HTTP a URL específicas, transmitiendo a los servicios parámetros de URL como argumentos. Normalmente, estos servicios devuelven los datos de la solicitud HTTP en formato JSON o XML para que la aplicación los analice y procese.

Por otra parte la aplicación hará solicitudes, de manejo similar a los servicios web de Google Maps, a un servidor, provisto por WorldSensing, del estado de los sensores de aparcamiento. El servidor devolverá la información en ficheros de formato kml. El KML (Keyhole Markup Language) es un lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones.

Los ficheros KML especifican una característica como un lugar, imagen, etc, para Google Earth. Contiene título, una descripción básica del lugar, información adicional (en este caso el estado de los sensores) y sus coordenadas por latitud y longitud.

Mediante el empleo de la latitud y longitud del fichero KML se podrá situar en el mapa la localización de los sensores.

7.2. Arquitectura básica de aplicaciones Android.

Una aplicación en Android está definida por el contenido de su manifiesto. Cada aplicación Android declara todas sus actividades, puntos de entrada, interfaces de comunicación, permisos e intenciones a través del AndroidManifest.xml. Además se podrían mencionar 4 bloques principales en los que se divide la aplicación:

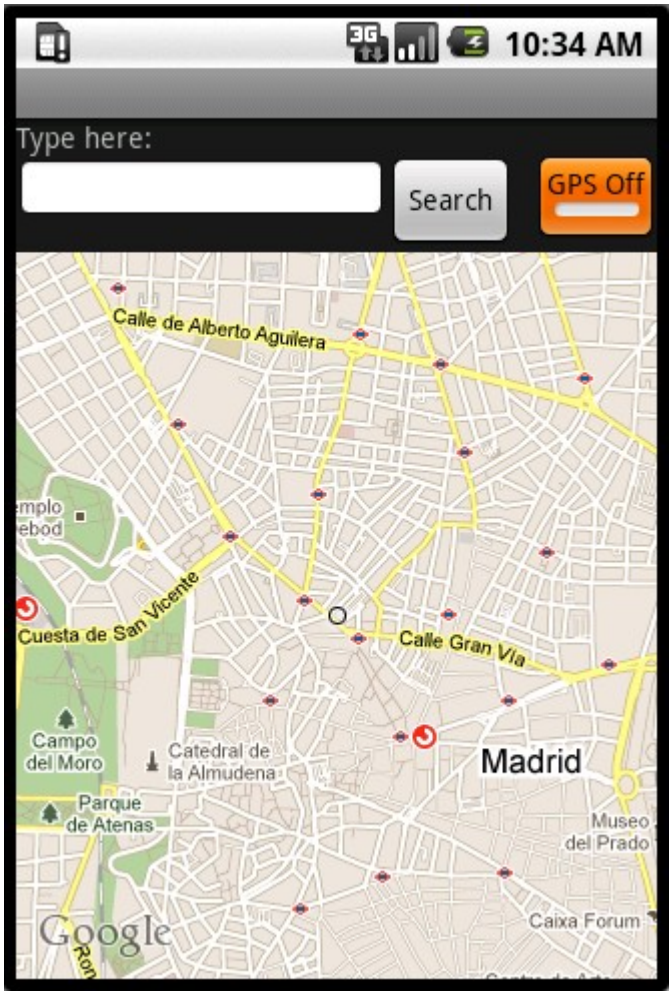
- Activity: Es el bloque básico de la aplicación. Una actividad usa una serie de funciones para interactuar con su entorno. Especificamente, una actividad debe sobrecargar el método "onCreate". También es recomendable sobrecargar los métodos "onStop", "onPause", "onResumen" y "onKeyDown". Con estas funciones se consigue manejar de manera más precisa la aplicación.

- Intent Receiver: Un objeto "reactivo" lanzado para manejar una tarea específica. Se encuentra a la espera de intentos de acciones para poder manejarlas cuando son recibidas.

- Service: Un proceso lanzado en segundo plano y por lo tanto no tendrá ninguna interacción con el usuario.

- Content Provider: Framework para el manejo y almacenamiento de datos.

7.3. Interfaz gráfico.

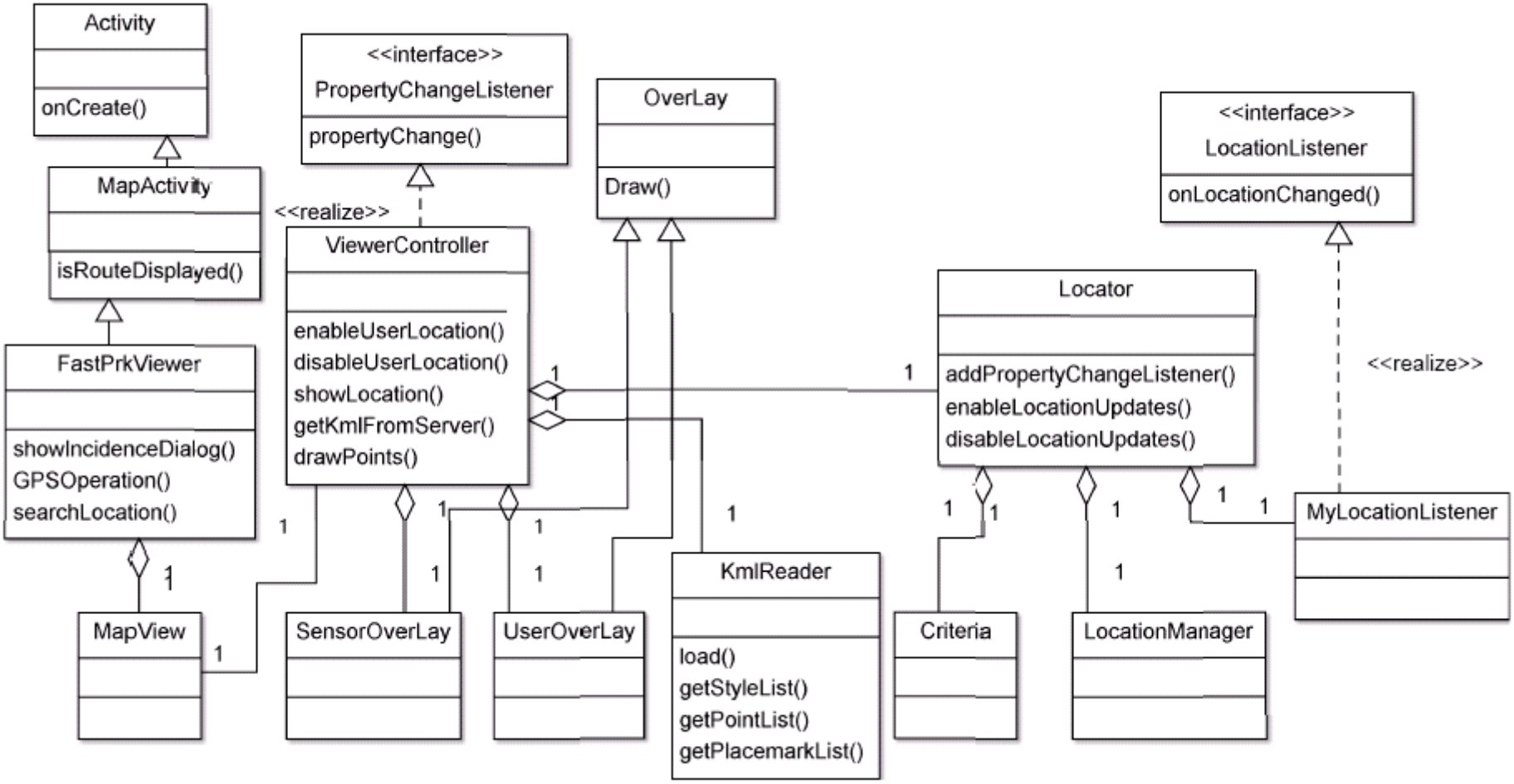


Este será el aspecto de la aplicación. Como se ha indicado en apartados anteriores en la zona superior se situarán el campo de texto y los botones para manejar las posibles acciones y el resto de la pantalla lo ocupará el mapa.

Como se puede observar el mapa tiene su configuración urbana.

7.4. Diagrama de clases

El diagrama de relación entre clases que tendrá la aplicación será el siguiente:



A continuación se explicará el cometido de cada una de estas clases:

- **Activity**: como bien se ha dicho más arriba es el bloque básico de la aplicación en Android. Una actividad usa una serie de funciones para interactuar con su entorno. Por lo general sobrecargará el método "onCreate" el cual se utiliza comunmente para construir todo el interfaz gráfico. También es recomendable sobrecargar los métodos "onStop", "onPause", "onResumen" y "onKeyDown". Con estas funciones se consigue manejar de manera más precisa la aplicación.

- **MapActivity**: Es una clase básica de Android que se extiende de Activity en la cual se puede incluir un mapa. Esta clase maneja el ciclo de vida de la aplicación y gestiona los servicios en background para poder mostrar el mapa por pantalla. Es necesario que MapActivity contenga un atributo de la clase MapView

- **FastPrkViewer**: Clase básica de la aplicación. Esta clase extiende de MapActivity. Se encarga de cargar el interfaz gráfico definido en el fichero main.xml e instanciar el resto de clases necesarias para la aplicación. Dos de sus métodos, GPSOperation y searchLocation son acciones asociadas a los botones del interfaz gráfico.

- **MapView**: Clase original de Android. Esta clase será la que permita el manejo del mapa, es decir, será la clase utilizada para mostrar por pantalla una localización, un punto, etc

- **ViewerController**: Clase que contiene la lógica para el manejo del mapa y las distintas fuentes de localizaciones. Esta clase se implementa de la interfaz PropertyChangeListener. Exactamente esta clase se encuentra a la escucha de cambios en la localización del usuario a través de la clase Locator. Cada vez que reciba un cambio este será representado en el mapa con la nueva localización.

- **OverLay**: Clase original de Android utilizada para situar una imagen sobre el mapa.

- **SensorOverLay**: Clase que extiende de OverLay. Se utiliza para situar sobre el mapa los iconos que indiquen la localización de un sensor de aparcamiento.

- **UserOverLay**: Clase que extiende de OverLay. Se utitiliza para situar sobre el mapa el icono que represente la localización del usuario del móvil.

- **KmlReader**: Clase que se encarga de leer y realizar un parser sobre un fichero kml. Con su método load() se encarga de realizar un recorrido por el fichero kml, registrando los "estilos", <Style>, de los puntos y buscando las etiquetas <Placemark> que son las que definen un punto. A su vez se asegura que estas etiquetas contengan realmente la etiqueta <Point>. Con su método getPointList se obtiene un listado de puntos con sus coordenadas y la url donde se encuentra la imagen que representará el overlay.

- **Locator**: Clase que contiene la información para obtener las coordenadas de localización del usuario con los métodos disponibles en el móvil. Además contendrá un listado de listeners de tipo PropertyChangeListener. Cuando se detecte un cambio de localización se lo notificará a todos los listeners suscritos, en este caso será la clase ViewerController.

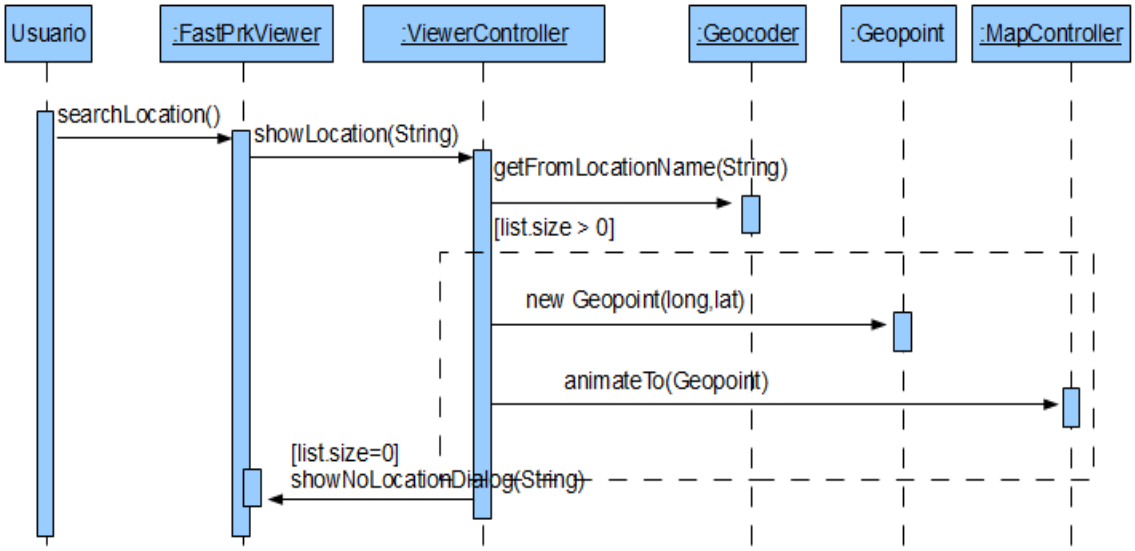
- **Criteria**: Clase original de Android. Se utiliza para definir una serie de requisitos para la elección de un proveedor de localizaciones. Estos requisitos pueden ser una mayor o menor precisión en la localización, consumo de batería, etc.

- **LocationManager**: Clase original de Android utilizada para manejar los servicios basados en localización. Aquí se puede definir qué proveedor de localización se quiere utilizar como el GPS, red, etc, o bien utilizar la clase anterior mencionada, Criteria, para buscar el mejor proveedor de localización que se adapte a los requisitos que se le han impuesto.

- **MyLocationListener**: Esta clase extiende del interfaz LocationListener de Android, la cual, entre otros métodos, obliga a implementar el método onLocationChanged(). Con este método se conseguirá actualizar la posición del usuario.

7.5. Diagramas de secuencia.

7.5.1. Acción "Search".



Cuando se define el botón Search en el fichero xml de definición del layout, main.xml, se establece que el método asociado al evento [2]*onClick* del botón será *searchLocation()*.

Una vez dentro del método searchLocation() se recoge el contenido del cuadro de texto para poder realizar la búsqueda. Para esto se llama al método *showLocation()* y se le pasa como parámetro la cadena de texto obtenida del cuadro de texto del interfaz gráfico.

Una vez dentro de la clase ViewerController, utilizando el atributo Geocoder, se llama a su método *getFromLocationName(String, Integer)*, pasándole como parámetro la cadena de texto del cuadro de texto y el número máximo de resultados que se desea. Esta función devuelve una lista de direcciones.

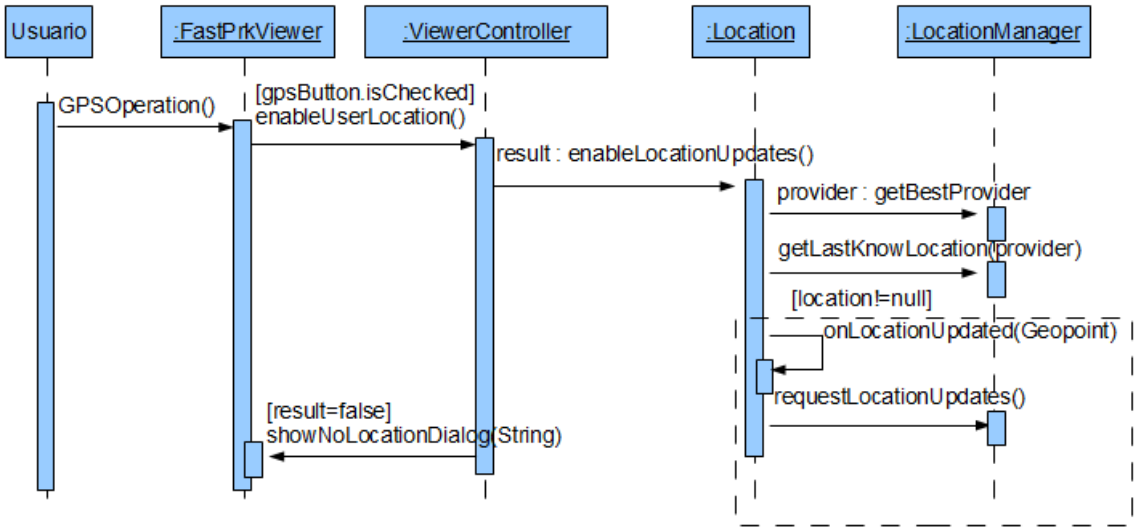
Si el tamaño de la lista de direcciones es superior a 0, se instancia un objeto de tipo Geopoint, pasándo la longitud y latitud del primer elemento de la lista.

2 Evento generado cuando se pulsa un botón, a este evento se le pueden asociar métodos para ejecutar diferentes acciones.

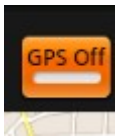
Una vez obtenido el punto geográfico se sitúa en el mapa con la ayuda del método *animateTo(GeoPoint)* del MapController.

En caso de que el tamaño de la lista que devuelve el método getFromLocationName sea 0, se procedería a llamar al método *showNoLocationDialog* de la clase FastPrkViewer, el cual mostrará al usuario un mensaje con notificando la incidencia ocurrida.

7.5.2. Acción GPS On



La acción de GPS On se inicia cuando se pulsa el botón de GPS del interfaz gráfico cuando este se encuentra desactivado, como se ve en la imagen:



El método asociado al evento de pulsado del botón es GPSOperation(), definido en el fichero xml de definición del layout, main.xml.

Para saber si cuando se ha pulsado el botón ha pasado de estado Off a On se comprueba el método booleano isChecked, del ToggleButton. Si este método nos devuelve **True** se hace una llamada al método *enableUserLocation()* de ViewerController.

Dentro de enableUserLocation, haciendo uso de un atributo de tipo Locator, se llama al método *enableLocationUpdates()*. Este es un método booleano, que en caso de devolver un valor **False** se ejecutaría el método *showNoLocationDialog()* de FastPrkViewer, el cual le mostrará al usuario de que no se ha podido obtener su localización.

En el interior del método enableLocationUpdates() se hace uso de un atributo de tipo LocationManager.

Con este atributo se hace una llamada al método *getBestProvider(Criteria). E*ste método devuelve una cadena de texto con el nombre del mejor proveedor de localizaciones encontrado en el móvil. Para que este método pueda saber qué proveedor es mejor que otro es necesario pasarle como parámetro un objeto de la clase **Criteria**. Este objeto, definido como atributo de la clase Locator, es parametrizado con valores que definen el uso de batería del dipositivo, precisión deseada del localizador, etc.

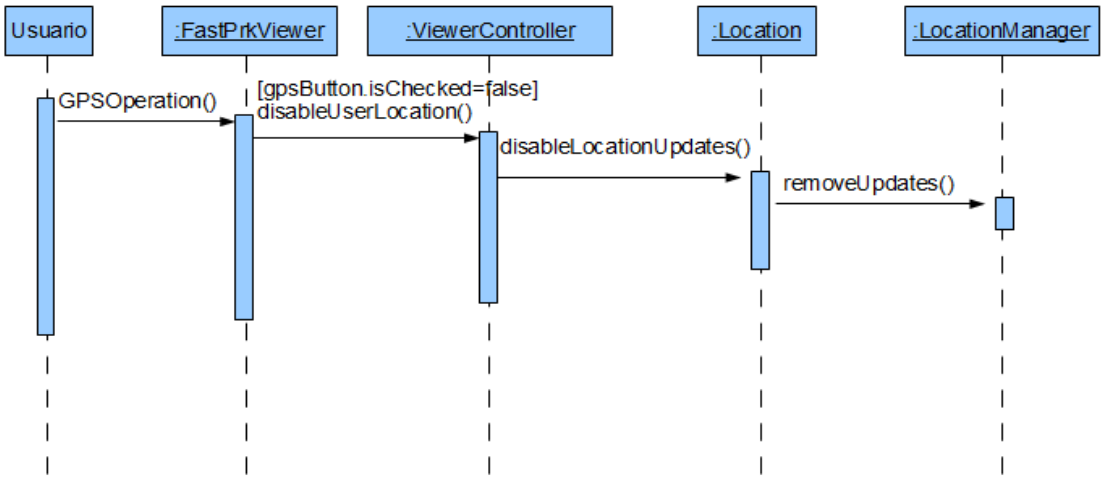
Utilizando la cadena de texto con el nombre del proveedor se llama al método *getLastKnowLocation()* de la clase LocationManager. Se llama a este método para poder obtener una localización más fiable. Este método devuelve un objeto de tipo Location. Si este objeto resultara tener un valor **null**, se sale del método dando **False** como valor de retorno.

Si el valor de location es distinto de null, se precede a convertir ese valor de un objeto Location a Geopoint, para acto seguido llamar al método privado *onLocationUpdated()*. Este método hace una llamada a los listeners suscritos y ejecuta el método firePropertyChanged(), de la clase PropertyChangeSupport. A los listeners suscritos les llega el objeto de tipo Geopoint.

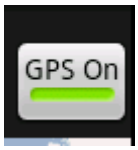
En este caso el listener suscrito es ViewerController el cual situará el punto Geopoint sobre el mapa.

Una vez obtenida la primera localización y enviada al mapa, se suscribe a LocationManager un listener para que permanezca a la escucha de los cambios de localización detectados. La suscripción se realiza con el método *requesLocationUpdates()* de LocationManager. Con cada actualización se llamal método privado citado anteriormente, *onLocationUpdated()*.

7.5.3. GPS Off



La acción de GPS Off se inicia cuando se pulsa el botón de GPS del interfaz gráfico cuando este se encuentra activo, como se ve en la imagen:



El método asociado al evento de pulsado del botón es GPSOperation(), definido en el fichero xml de definición del layout, main.xml.

Para saber si cuando se ha pulsado el botón ha pasado de estado On a Off se comprueba el método booleano isChecked, del ToggleButton. Si este método nos devuelve **False** se hace una llamada al método *disableUserLocation()* de ViewerController.

En el iterior de este método se procede a llamar al método *disableLocationUpdates()* de la clase Locator.

Por último se ejecuta el método *removeUpdates()* pasando como parámetro el atríbuto de la clase que hace de Listener de localización.

Finalmente con este proceso se hace que deje de actualizarse en el mapa de la aplicación los cambios de localización que realice el usuario.

8. Implementación.

8.1 .Recursos

En la web de Android se puede obtener el SDK para el desarrollo de aplicaciones para este Sistema Operativo, el cual funcionará con el IDE Eclipse.

El interfaz de la aplicación permitirá introducir datos, concretamente de localizaciones geográficas, para realizar las búsquedas.

La forma en que se obtendrá la información será a través del google maps. Para implementar estas acciones será necesario utilizar el API de Google Maps para Android dónde se pueden encontrar numerosas funcionalidades.

En la web de "Google Code" se puede ver toda la familia del API de Google Maps. De entre las opciones se hará uso de los servicios web para Google Maps.

Estos servicios web son un conjunto de interfaces HTTP de los servicios de Google con las que se pueden obtener datos geográficos para aplicacione con mapas.

Además se deberá mostrar por pantalla, en el mapa, la localización del dispositivo móvil dónde corra la aplicación.

Para esto se utilizará el GPS del própio móvil, si lo tiene, de lo contrario deberá realizar la acción mendiante IP.

Normalmente para obtener la ubicación geográfica de una dirección IP se consulta una base de datos de información de las direcciones asignadas a un DNS pasándole la dirección que pretende buscar. De esta base de datos se obtendrá un DNS en respuesta a la IP pasada. Esto derivará en uno o más direcciones de sitios web provenientes de este DNS. Escaneando cada una de las direcciones de los sitios web derivados se podría determinar la información de la dirección geográfica.

Por otro lado el API de Google Maps también dispone de métodos que sitúan en el mapa la localización en función de la IP del dispositivo.

Por otro lado para obtener la localización del usuario se utilizarán métodos que analicen cual es el mejor proveedor de localizaciones, pero estos métodos sólo tienen validez sobre un teléfono móvil real y no sobre el emulador de Android para PC. Con el emulador tan sólo se puede simular el GPS pero es necesario que el usuario, a través de herramientas del propio emulador, le proporcione las coordenadas de la localización, tal y como lo haría el satélite.

La empresa WorldSensing se ha compremetido a facilitar servicios web para la obtención de las localizaciones de los sensores que esta tenga implantados además del estado de estos.

8.2. Generación del proyecto.

A la hora de crear el proyecto se utiliza el IDE Eclipse con el SDK de Android previamente instalado. Se deberán seguir los pasos para crear un nuevo proyecto con el Eclipse con la diferencia de que se deberá seleccionar la opción de Android Project.

En el diálogo que se lanza se define el nombre del proyecto, su dominio, el API que utilizará, la actividad principal, etc. Como se va a hacer uso de Google Maps se debe incluir el API de Google más actual, en este caso la versión 2.3.

Cuando se crea un proyecto Android, se genera automaticamente una serie de ficheros, que los diferencian de los demás proyectos Java. A continuación se explica para qué sirven estos ficheros:

Fichero	Propósito

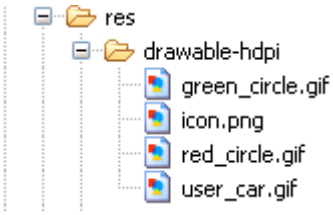
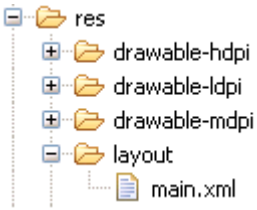
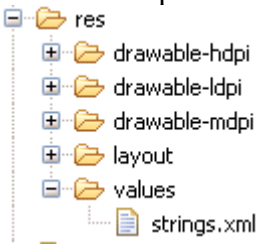
AndroiManifest.xml	Es un fichero que contiene la información de la aplicación que será mostrada al sistema Android. Es editar este fichero para permitir que la aplicación tenga acceso a internet y que habilita los dos modos de precisión a la hora de obtener la localización por GPS u otro medio.
	
	`<uses-permission android:name="android.permission.INTERNET" />`
	`<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>`
	`<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>`

main.xml	Fichero que contiene la definición del interfaz gráfico. Además de todos los elementos que forman parte del interfaz gráfico, cobra especial relevancia la declaración del mapa, MapView, donde se puede ver que además de algunos parámetros, es necesario introducir la clave para poder utilizar el API.
	
	`<com.google.android.maps.MapView`
	`android:id="@+id/mapView"`
	`android:layout_width="fill_parent"`
	`android:layout_height="fill_parent"`
	`android:enabled="true"`
	`android:clickable="true"`
	`android:apiKey="0SQkRRaxJHtMD753fkzJ8SWngBs08ynuhOsfBUQ"/>`

strings.xml	Fichero que contiene valores constantes que serán utilizados en la aplicación.
R.java	fichero que contiene los identificadores de los elementos contenidos en main.xml y string.xml. Es un fichero que no debe ser modificado por el usuario, se modifica automaticamente en función de los cambios en los ficheros de arriba.

Una vez creado el proyecto se puede observar la generación de una serie de directorios, que serán explicados a continuación:

Directorio	**Propósito**
/gen	Directorio que contiene el fichero R.java.  Como se puede observar, a su vez está contenido en el paquete com.fastprkviewer.app. Coincide con el paquete donde se encuentra la actividad principal.

/res/drawable	Es el directorio donde se incluyen todas las imágenes que son utilizadas por el interfaz gráfico. 
/res/layout	Es el directorio que contiene el fichero main.xml. 
/res/values	Es el directorio que contiene el fichero strings.xml 

8.3. Implementación del interfaz gráfico.

El interfaz gráfico de una aplicación en Android vendrá definido por entero en el fichero **main.xml** en el directorio /res/layout del proyecto. Por lo tanto, no se podrá mostrar nada de la aplicación que no se incluya en este fichero.

El marco principal del interfaz gráfico se realiza con un RelativeLayout. Con esto se van situando en posiciones relativas los demás del componentes del interfaz.

En primer lugar se incluye los componentes de la parte superior, esto son:

- Un botón general: es el botón ***Search*** de la clase **Button**. El método asociado a la acción de pulsar el botón se define en el fichero main.xml. Es el botón encargado de realizar búsquedas de localizaciones.

- Un botón de estados, denominado ***GPS***, que será de tipo **ToggleButton**. Es el botón encargado de activar y desactivar la detección de la localización del dispositivo móvil.

- Una etiqueta: Se trata de un **TextView** definido para que ocupe todo el ancho del marco principal (marco padre) y su altura se ajuste al tamaño de la letra.

- Un cuadro pra introducir texto: se trata de un elemento de tipo **EditText** definido para situarse a la izquierda del botón **Search**.

Debajo de la barra que forman los elementos arriba mencionados se introduce un nuevo marco RelativeLayout. Dentro de este marco se incluirá el mapa. El motivo por el que se realiza de esta manera es porque si se definiera, en el fichero **main.xml**, dentro del marco principal se superpondría al resto de elementos y estos no se verían, puesto que los parámetros de anchura y altura que se le asignan, tal y como se ve en la imagen:

```
<com.google.android.maps.MapView
    android:id="@+id/mapView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

Tienen que ser obligatoriamente "fill_parent".

La mayor parte de valores constantes, configuración, imágenes e interfaz gráfica de usuario vendrán predefinidas en los ficheros XML que acompañan a la aplicación.

Uno de estos ficheros es el R.java que es el fichero que contiene el identificador para todos los valores de constantes utilizados en la aplicación.

8.4. API's utilizadas.

Para el desarrollo de esta aplicación se han utilizado las siguientes API's de programación:

- **android.jar**: Es el API de Google para Android 2.3. Contiene las librerías básicas para el desarrollo de aplicaciones Android. Coincide practicamente en su totalidad con las librerías Java habituales.

- **maps.jar**: Es otro API de Google para Android 2.3. Contiene las librerías necesarias para implementar las funcionalidades de mapas. Algunas de las clases que pertenecen a esta librería son la MapView u Overlay.

- **JavaApiForKml.jar**: API liberada bajo licencia BSD de Micromata que se encarga de proveer una serie de interfaces para facilitar el acceso y el manejo de los ficheros kml.

8.5. Incidencias en el desarrollo.

En la fase final del desarrollo de esta aplicación se han hallado una serie de incidencias del API JavaApidForKml.

A día de hoy no existe un API estándar para KML como lo hay de XML para Java. La mayoría de aplicaciones que hacen uso de KML elaboran sus propias librerías basadas en XML.

El JavaApiForKML es uno de los pocos API's que se pueden hallar actualmente para poder manejar los ficheros KML.

El desarollo de la clase KmlReader, que utiliza el API citado, fué inicialmente sobre un proyecto estándar de Java. Esto fué debido a la necesidad de rapidez y facilidades de depuración que el Eclipse provee para este tipo de proyectos. La clase se creó y depuró sin más problemas obteniendo el resultado final deseado, obtener las coordenadas de los puntos incluidos en el fichero KML, así como los enlaces de los iconos que los representarán en el mapa.

Una vez obtenida la clase se procedió a importarla al proyecto de Android. Pese a incluir el API en el Java Build Path del proyecto, esta librería mostraba una incidencia en referencia a que no tenía la visibilidad de dos clases. Estas clases pertenecen a la librería javax.xml.bind.annotation las cuales se ha comprobado que de momento no están incluidas en la librería Java para Android.

9. Cronograma.

El proyecto se ha dividido en 4 fases fundamentales para su implementación. Para planificar estas fases se utilizará la metodología WBS (Work Breakdown Structure), con Diagramas de Gantt general para todo el proyecto y por cada una de las fases.

El modelo de desarrollo que se utilizará será incremental e iterativo, tratando de seguir en lo posible el modelo RUP (Rational Unified Process). Este modelo viene dado por la relación entre las fases, en las que se irá pasando de una otra a medida que se vayan alcanzando determinados hitos.

A continuación se describen las fases:

- ***Fase 1*:** Planificación del Trabajo Fin de Máster. Equivale a la elaboración de este documento.

- ***Fase 2*:** Adaptación al entorno y primer contacto. Familiarización con la tecnología a desarrollar,instalación del IDE Eclipse y su configuración, familiarización con el SDK de Android para Eclipse. Familizarización además con el API de Google Maps y los servicios web que esta proporciona.

- ***Fase 3*:** Diseño funcional de la aplicación.

    - ***Fase 3.1*:** Diseño arquitectónico. División y diferenciación de las distintas partes que compondrán la aplicación. Diagrama de Clases, Diagramas de Secuencia.

        - ***Fase 3.1.1***: Arquitectura de la aplicación: Diseño del funcionamiento general de la aplicación.

        - ***Fase 3.1.2***: Diagrama de clases (Lógica): Relación entre las clases que compondrán la parte lógica de la aplicación.

        - ***Fase 3.1.3***: Diagrama de clases (Interfaz gráfico): Relación entre las clases que compondrán la parte visible de la aplicación.

        - ***Fase 3.1.4***: Diagrama de secuencia: Diagrama de la evolución de la aplicación.

        - ***Fase 3.2*****:** Revisión y corrección. Comprobar la idoneidad de la solución adoptada con los diagramas.

- ***Fase 4*****:** Desarrollo. Trasladar el diseño de los diagramas al código.

    - ***Fase 4.1*****:** Estudio de viabilidad de la solución adoptada en los diagramas.

    - ***Fase 4.2*****:** Desarrollo. Escritura del código de la aplicación.

        - ***Fase 4.2.1***: Codificación de la lógica de la aplicación.

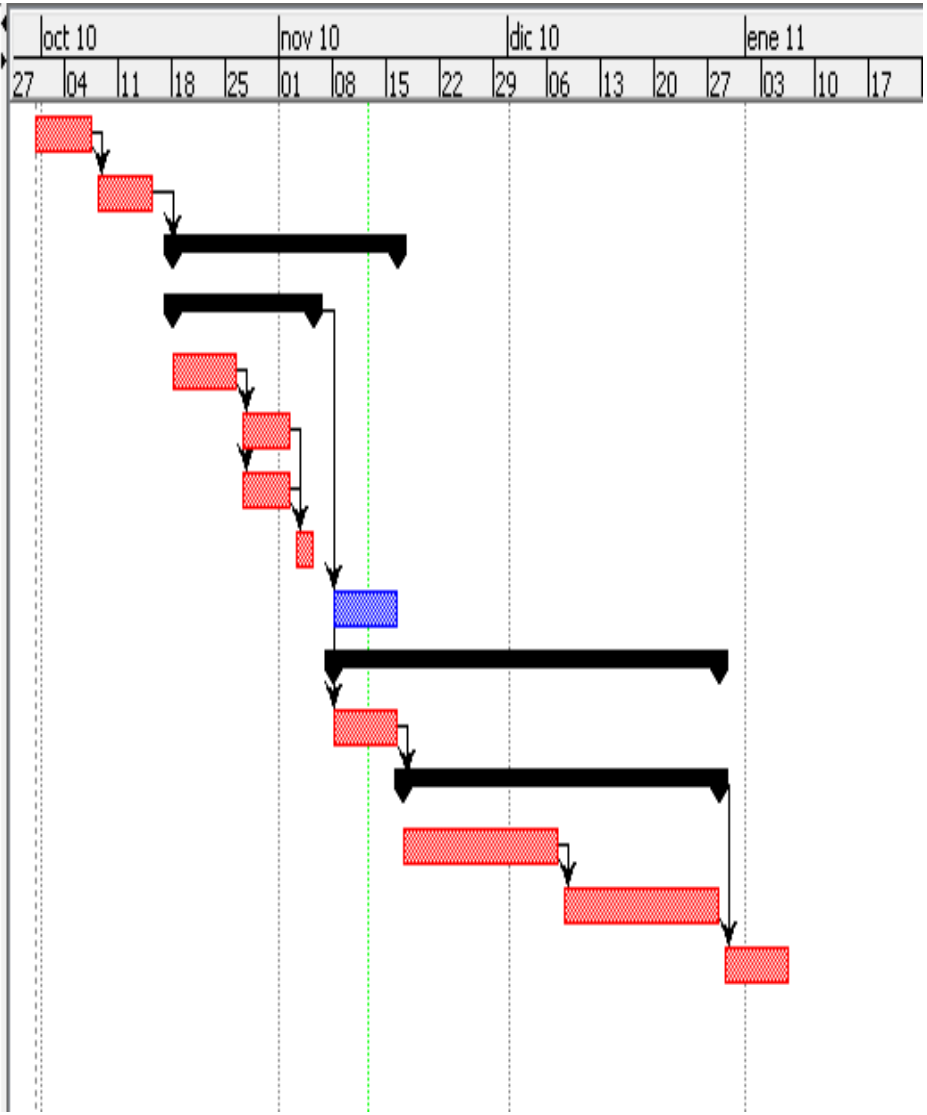
        - ***Fase 4.2.2***: Codificación de la parte de la aplicación que se ocupa del interfaz gráfico.

- ***Fase 5*****:** Pruebas y correcciones.

La evolución temporal de estas fases y su interrelación se verá representada en el siguiente Diagrama de Gantt.

También cabe mencionar que esta evolución temporal viene condicionada por la limitación de tiempo de este proyecto, teniendo su comienzo el 30-09-2010 y su finalización, de acuerdo con la entrega de la última PEC, el 06-01-2010.

		Nombre	Duración	Inicio	Terminado	Prede
1		Planificación	6 days	30/09/10 8:00	7/10/10 17:00	
2		Adaptación al entorno de tra	6 days	8/10/10 8:00	15/10/10 17:00	1
3		**Diseño Funcional**	22 days	**18/10/10 8:00**	**16/11/10 17:00**	**2**
4		**Diseño Arquitectónico**	15 days	**18/10/10 8:00**	**5/11/10 17:00**	
5		Arquitectura aplicación	7 days	18/10/10 8:00	26/10/10 17:00	
6		Diagrama de clases (Logi	5 days	27/10/10 8:00	2/11/10 17:00	5
7		Diagrama de clases (Inte	5 days	27/10/10 8:00	2/11/10 17:00	5
8		Diagrama de secuencia	3 days	3/11/10 8:00	5/11/10 17:00	7;6
9		Revisión y corrección	7 days	8/11/10 8:00	16/11/10 17:00	4
10		**Desarrollo**	37 days	**8/11/10 8:00**	**28/12/10 17:00**	
11		Estudio de viabilidad	7 days	8/11/10 8:00	16/11/10 17:00	4
12		**Codificación**	30 days	**17/11/10 8:00**	**28/12/10 17:00**	**11**
13		Codificación (Lógica)	15 days	17/11/10 8:00	7/12/10 17:00	
14		Codificación (Interfaz gra	15 days	8/12/10 8:00	28/12/10 17:00	13
15		Pruebas y correcciones	7 days	29/12/10 8:00	6/01/11 17:00	12



Las fechas serían:

Tarea	**Inicio**	**Fin**
1. Planificación	30/09/10	07/10/10
2. Adaptación al entorno de trabajo.	08/10/10	15/10/10
3. Diseño Funcional	18/10/10	16/11/10
3.1. Diseño arquitectónico.	18/10/10	05/11/10
3.1.1: Arquitectura de la aplicación	18/10/10	26/10/10
3.1.2: Diagrama de clases (Lógica)	27/10/10	02/11/10
3.1.3: Diagrama de clases (Interfaz gráfico)	27/10/10	02/11/10
3.1.4: Diagrama de secuencia:	03/10/10	05/11/10
3.2. Revisión y corrección	08/11/10	16/11/10
4. Desarrollo	08/11/10	28/12/10
4.1. Estudio de viabilidad.	08/11/10	16/11/10
4.2. Escritura del código	17/11/10	28/12/10
4.2.1: Codificación (lógica)	17/11/10	07/12/10
4.2.2: Codificación (interfaz gráfico)	08/12/10	28/12/10
5. Pruebas y correcciones.	29/12/10	06/01/11

10. Presupuesto.

Este es un proyecto de desarrollo software en el que se utilizarán herramientas de software libr y se utilice código licenciado bajo alguna licencia libre por lo que no tendrá ningún coste económico ligado a licencias de uso.

Pese a esto es necesario hacer un cálculo del coste del desarrollo, el cual incluye:

- Coste total de hardware y software.
- Mantenimiento.
- Soporte Técnico.

El hardware a utilizar será un PC estándar, para el desarrollo de la aplicación. Las primeras pruebas serán realizadas sobre el simulador que proporciona el Kit de Desarrollo Software de Android. Posteriormente sería necesario probar esta aplicación con un dispositivo móvil cuyo Sistema Operativo sea Android para probar las funcionalidades de actualización en tiempo real de las localizaciones de los sensores a medida que el usuario se vaya desplazando. El precio del PC se calcula alrededor de 800€ y el móvil 250€.

Tomando en consideración que el proyecto lo lleve a cabo un ingeniero especialista en desarrollo software para aplicaciones móviles con un sueldo mensual de 2000€, trabajando 20 días al mes, 40 horas por semana, el precio de las fases serían:

- La fase de Planificación son de 5 días por lo que el coste sería de 500€
- La fase de Adaptación y configuración del entorno de desarrollo son de 5 días por lo que el coste sería de 500€.
- La fase de Diseño Funcional completa son 22 días por lo que el coste sería de 2200€.

- La fase Desarrollo son 42 días pero la subtarea de la fase anterior Diseño Funcional, Revisión y corrección, y el Estudio de Viabilidad, de la fase actual, coinciden en el calendario. Como son complementarias se realizarán simultáneamente por lo que la fase de Desarrollo sólo se le consideraría los 35 días para la escritura del código. Dicho coste sería de 2500€.

- Por último la fase de Pruebas son 7 días laborales por lo que el coste sería de 700€.

Resumiendo el presupuesto sería de:

Descripción.	**Tiempo (días)**	**Precio (€)**
PC estándar	--	800
Teléfono móvil con Android	--	250
Planificación	5	500
Adaptación y configuración del SDK en el PC	5	500
Diseño funcional	22	2200
Desarrollo	35	3500
Pruebas	7	700
	TOTAL	**8450**

11. Conclusiones.

11.1. Sistema.

Con la elaboración e implantación de sensores en las ciudades, WorldSensing ha dado un gran paso en el avance del bienestar de los ciudadanos. Con la red de sensores se puede observar lo siguiente:

- Introducir todavía más a las personas en la sociedad de la información. Hoy en día, con los avances tecnológicos, las personas pueden obtener información sobre casi cualquier cosa de manera rápida y sencilla. Ofreciendo información sobre la localización y el estado de plazas de aparcamiento en la vía pública, se está consiguiendo ir un paso más allá.

- Se facilita la vida de los conductores. Sabiendo dónde puede hallar plazas libres los conductores dejarán dar innumerables vueltas por zonas a la busca de una plaza. Esto evitará aglomeraciones en el tráfico.

- Se ayudará a la reducción de la emisión de gases contaminantes a la atmósfera. Cuando un conductor se encuentra en la búsqueda de una plaza de aparcamiento suele cambiar la forma de conducción, siendo esta más lenta para tener un mayor margen de maniobra. Esta conducción se realiza durante tiempo prolongado sobre marchas cortas, que son las más contaminantes. El hecho de que un usuario disponga de la localización de una plaza libre hará que su conducción sea más ligera para llegar antes a la plaza. Cuanto mayor sea el número de personas utilizando el sistema, menor será la tasa de emisión de gases total.

A primera vista se puede observar que es un sistema que presenta, invevitablemente, algunas limitaciones. Aunque la actualización del estado de una plaza de aparcamiento se produzca en tiempo real en la base de datos de WorldSensing, el refresco en la aplicación se produce con una cadencia de unos 30 segundos, por lo tanto cabe la posibilidad que otro conductor se encontrara cerca y la ocupe cuando el usuario de la aplicación se esté acercando. Este tipo de limitación ya no depende del sistema de sensores ni de la aplicación si no al albedrío de las circunstancias.

11.2. Software.

Con el desarrollo de este software se han obtenido las siguientes observaciones:

- La arquitectura de una aplicación para Android difiere de la arquitectura de una aplicación para PC. Pese a que el lenguaje de programación sea un lenguaje tan extendido como Java, el framework de la aplicación es diferente. El framework más utilizado para una aplicación para PC se componen de los siguientes elementos:

  - **Controlador**: es el elemento que controla el acceso a la aplicación.

  - **Modelo**: es un miembro del controlador que maneja las operaciones lógicas y de uso de la información.

  - **Vista**: es el elemento que se encarga de dibujar el interfaz gráfico de usuario.

  Una aplicación para Android se compone de los siguientes elementos:

  - **Activity**: las actividades representan el componente principal del interfaz gráfico. Se puede ver a una actividad como el componente análogo a una ventana en cualquier otro lenguaje visual.

  - **Services**: los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. Conceptualmente son similares a los servicios en cualquier otra arquitectura.

  - **Content Provider**: es el mecanismo definido en Android para compartir datos entre aplicaciones.

  - **Broadcast Receiver**: es el componente destinado a detectar y reaccionar frente a determinados mensaje o eventos globales generados por el sistema.

Como se puede observar algunos elementos de la arquitectura para Android, conceptualmente son muy parecidos a los que se utilizarían para una aplicación para PC aunque se ve claramente que la arquitectura de una aplicación para Android está adaptada a las limitaciones hardware que tiene un teléfono móvil:

- Los elementos, citados en el punto anterior, que componen la aplicación deben estar declarados en el AndroidManifest. El AndroidManifest es un fichero de tipo xml que contiene la información de la aplicación que será mostrada al sistema Android. Si no se encuentra declarado en este fichero no se puede crear de forma dinámica en la aplicación.

- Al igual que los componentes deben estar en el AndroidManifest, los elementos que componen el interfaz gráfico deben estar declarados en el fichero /layout/main.xml del proyecto de la aplicación. Las instancias realizadas, de elementos visuales de la aplicación, se deben hacer en referencia a lo declarado en este fichero.

- El emulador de Android para PC incluído en el SDK de Android para Eclipse es una buena herramienta de trabajo, útil para testear y depurar la aplicación pero existen determinados elementos que forman parte de la aplicación que no pueden ser probados en el emulador. Un ejemplo es el NETWORK_PROVIDER, es decir, el proveedor de localización a través de la red. Para probar las obtención de localización del dispositivo móvil a través de la red es necesario realizar las pruebas sobre un móvil real con Android.

En el aspecto de las limitaciones hardware, desarrollar una aplicación para Android puede recordar al desarrollo de una aplicación para microprocesadores embebidos o microcontroladores.

La diferencia está en que los microcontroladores tienen aplicaciones que generalmente son procesos y servicios que obedecen a paradigmas de programación más habituales de los lenguajes estructurados.

Una aplicación para Android pese a que también puede utilizar servicios está basado en programación orientada a objetos yendo más allá y adaptándose a la arquitectura de un teléfono móvil, que aunque avance a pasos agigantados su tecnología, poco tiene en común con la arquitectura de un PC.

11.3 Aspectos generales.

Pese a la dificultad inicial que un desarrollador puede encontrarse al realizar por primera vez una aplicación para Android, sí es cierto que se encontrará con un sistema diseñado para facilitar el trabajo de los desarrolladores.

También se puede observar que ofrece un alto nivel de abstracción frente al hardware lo cual facilita enormemente el desarrollo e integración entre aplicaciones.

Son muchas las facilidades que un desarrollador se puede encontrar a la hora de elaborar una aplicación puesto que hay mucho código fuente, bibliotecas, referencias, tutoriales y software de simulación.

A través de la web oficial de Android, además de encontrarse con las últimas novedades del sistema operativo, versiones, etc, también se encontrará con una guía de presentación y desarrollo de aplicaciones que desmenuza la mayor parte de los entresijos del desarrollo de aplicaciones. Por otro lado también se puede tener acceso a prácticamente toda la biblioteca básica de clases de Java para Android.

Pese a tanta facilidad es altamente recomendable que el desarrollador tenga a su disposición un teléfono móvil con Android para que las pruebas sean lo más fiables posibles y así poder probar todos los periféricos de qué dispone el móvil, observar la eficiencia de la aplicación en lo que a consumo de recursos se refiere (cpu, batería, etc) y como no hacer las pruebas finales de cualquier aplicación sobre el medio al que está destinado.

12. Referencias bibliográficas.

Bibliografía utilizada:

- http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software
- http://www.google.com
- http://code.google.com/intl/es-ES/apis/maps/
- http://developer.android.com/index.html
- http://www.worldsensing.com/
- http://code.google.com/p/javaapiforkml
- "Android Essentials" de Chris Haseman, Agosto 2008.
- "Metodología para el Análisis de Requisitos de Sistemas Software" de Amador Durán Toro y Beatriz Bernárdez Jiménez. Universidad de Sevilla, Diciembre de 2001.
- "Proffesional Android Application Development" de Reto Meier, Noviembre de 2008, de Wrox.