UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TELEMÁTICA

PROYECTO FIN DE CARRERA

DESARROLLO DE UNA APLICACIÓN CON GEOLOCALIZACIÓN PARA ANDROID.

TUTOR: Mario Muñoz Organero

DEPARTAMENTO DE TELEMÁTICA

AUTOR: Elías Pardo de Donlebún Matilla

INGENIERÍA TÉCNICA DE TELECOMUNICACIONES SONIDO E IMAGEN

RESUMEN

El objetivo del siguiente proyecto es el de desarrollar una aplicación de realidad aumentada con geolocalización para el sistema operativo Android. La aplicación, haciendo uso del servicio GPS y brújula del terminal, geolocalizará al usuario y mostrará los puntos de interés cercanos obteniendo la información de bases de datos. El formato de estas bases de datos será XML y serán fácilmente editables por un usuario sin muchos conocimientos informáticos. La aplicación incluye una base de datos de restaurantes y de puntos turísticos de Madrid. El propósito de la aplicación es amplio y será definido dependiendo de las bases de datos que se utilicen.

La motivación del proyecto es, por un lado desarrollar una aplicación que sirva como plataforma para futuros desarrollos del departamento de Telemática y por otro lado que el estudiante se involucre en un proyecto completo de cierta envergadura.

Palabras clave: realidad aumentada, Android, geolocalización, GPS, brújula

ABSTRACT

The goal for the following project is to develop an augmented reality application for Android, using geolocation. Using the terminal´s GPS service and the compass, the application will geolocate the user and show the nearby points of interest using a database. The databases´ format will be XML and easily modified by an inexperienced user. The application offers a restaurant and touristic database of Madrid. The purpose of the application is broad and will depend on the databases used.

The motivation behind this project is to, on one hand, develop an application that can be used as a starting point for the Department´s future developments and, on the other hand, involve the student in a complete development project with considerable amount of workload.

Keywords: augmented reality, Android, geolocation, GPS, compass

ÍNDICE DE CONTENIDOS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS Y ECUACIONES

DEFINICIÓN DE OBJETIVOS E INTRODUCCIÓN

La primera sección de este proyecto tiene como objetivo dar una idea general al lector de lo que se va a encontrar en esta memoria. En esta sección se hará una breve introducción a este proyecto, incluyendo la motivación y los objetivos de éste.

1. Introducción

En este documento se presenta un proyecto fin de carrera que tiene el objetivo de desarrollar una guía geolocalizada con realidad aumentada para la plataforma Android. Se trata de una aplicación que, dependiendo de la localización en la que se encuentre el terminal, muestra al usuario los puntos de interés cercanos a él mediante una interfaz gráfica. De esta manera el usuario puede obtener información acerca de los lugares que visita o se encuentra sin necesidad de consultar una guía, mapa o preguntar a la gente local. La utilidad principal de la aplicación es servir de guía turística al usuario.

Otra de las características más destacables de la aplicación es que la interfaz es muy amigable, con la intención de que el público sea el máximo posible. La aplicación acepta bases de datos XML, fácilmente editables, en las que el usuario podrá popular las suyas propias. Esto abre un abanico de posibles usos que se le pueden dar a la aplicación, puesto que los usuarios que creen bases de datos definirán un nuevo uso para la aplicación. La intención es que la aplicación sea lo más social posible, de manera que se atraiga a nuevos usuarios, además de motivarles para crear nuevas bases de datos.

La aplicación desarrollada servirá de plataforma de lanzamiento para futuros desarrollos del departamento de Telemática de la universidad Carlos III. Además el código estará disponible para cualquier desarrollador que desee proseguir con el desarrollo o simplemente para usarlo como referencia.

Hasta el momento no se ha sabido de ninguna aplicación idéntica a ésta para Android. Similares son: *Compass* de Lonely Planet, *AR Travel Guide* de Wikitude. Las principales diferencias con éstas son la implementación de la realidad aumentada, que se hace en tiempo real mientras que con la aplicación desarrollada para este proyecto la hace cuando el usuario pulsa el obturador.

2. Motivación y objetivos

En la siguiente sección se exponen las distintas motivaciones que han llevado al desarrollo de este proyecto. La primera motivación es mantener el departamento de Telemática de la Univeridad Carlos III al día en cuanto a aplicaciones de realidad aumentada se refiere. La otra motivación principal es el aprendizaje del alumno, el cual adquirirá nuevas capacidades y conocimientos durante el desarrollo. Los objetivos se enumeran a continuación y son: desarrollar una aplicación que utilice la geolocalización y que el desarrollador tenga soltura en el desarrollo de aplicaciones para Android.

Gracias a los avances tecnológicos por parte de la industria del hardware, los dispositivos móviles han alcanzado una capacidad de cómputo antes impensable. Esto ha permitido que los terminales móviles no se limiten a realizar funciones de telefonía “clásica”, si no que han derivado en una suerte de ordenador portátil. Para explotar estas nuevas capacidades los fabricantes han decidido hacer públicas la mayoría de interfaces de programación para que desarrolladores ajenos a ellos puedan generar aplicaciones para sus terminales. De esta manera se crea una especie de sinergia, ya que los

desarrolladores tienen un nuevo ámbito en el que trabajar y los terminales se hacen más atractivos de cara al consumidor gracias a estas nuevas aplicaciones, lo cual obviamente repercute en el número de ventas.

Unos de los últimos avances introducidos a los dispositivos móviles han sido el GPS y los acelerómetros. Ambas abren un campo totalmente nuevo e inexplorado para los desarrolladores y uno de ellos es la Realidad Aumentada. La Realidad Aumentada es un paradigma de la programación en el que las aplicaciones dependen y utilizan información contextual.

Hasta ahora, las aplicaciones para terminales móviles no han sido más que una mera transcripción de los programas clásicos para los ordenadores de sobremesa, adaptados a las limitaciones de batería, capacidad de cómputo y tamaño de pantalla de los móviles. El desarrollo de aplicaciones que usasen GPS y acelerómetros ha sido escaso (si no nulo), ya que la utilidad de ellos en un ordenador de sobremesa es más bien poca. Por ello introducir estas capacidades ha provocado una nueva ola en el mundo de la programación, abriendo nuevas fronteras.

Las aplicaciones de Realidad Aumentada son hoy en día, valga la redundancia, una realidad. Hay aplicaciones comerciales para varias plataformas funcionando perfectamente y con multitud de propósitos (sociales, lúdicos, comerciales, turismo...), pero aún queda mucho por explorar. El futuro de estas aplicaciones es verdaderamente prometedor, y por ello hay que estar al día y conocer los recursos y capacidades para, en algún momento, poder desarrollar una aplicación de Realidad Aumentada verdaderamente innovadora, competitiva y con utilidad social.

Actualmente existen en el mercado multitud de aplicaciones de Realidad Aumentada y con naturalezas muy variadas, entre ellas, están las siguientes:

- Publicidad
- Navegación
- Militar
- Turismo
- Lúdico
- Comercio

El género escogido ha sido el turismo, puesto que es una de las más sencillas de implementar y de las más asequibles. Es mucho más sencillo y práctico que las de videojuegos o las militares. Las razones son las siguientes:

1. Las capacidades del desarrollador. Normalmente las aplicaciones las desarrolla un equipo de desarrolladores con experiencia, por lo que es lógico escoger un ámbito con una complejidad asociada razonable.
2. Limitaciones de tiempo. El plazo inicial es de 6 meses.
3. Requerimientos del terminal. Cuantos menores requisitos tenga la aplicación, mayor penetración de mercado tendrá.
4. Turismo. España es uno de los destinos turísticos mundiales por excelencia por lo que se parte con ventaja.
5. Ventajas respecto a los sustitutivos. El sustitutivo por excelencia de una guía

virtual como la que vamos a desarrollar sería la guía tradicional impresa, la cual se queda obsoleta con el paso el tiempo y resulta un lastre a la hora de cargar con ellas para hacer turismo. Otro sustitutivo es la guía digital, sobre la que la de Realidad Aumentada tiene la ventaja de su facilidad de uso.

6. Además, muchos de los demás temas, sobretodo la navegación o militar, exigen una cantidad de equipamiento mucho mayor y por ejemplo, la fase de pruebas sería mucho más difícil de realizar.

La motivación no es desarrollar una aplicación innovadora o rompedora, si no hacer una primera toma de contacto con el concepto, de manera que el departamento esté al día en cuanto a aplicaciones para terminales móviles respecta. De esta manera se sentarán las bases para que, en un futuro, quizás algún otro alumno de la Universidad Carlos III, siga estos pasos y pueda desarrollar la mencionada aplicación comercializable y realmente innovadora.

También hay una motivación adicional, el aprendizaje. Con este proyecto se explorarán las posibilidades de uno de los sistemas operativos para terminales móviles. La idea es que el estudiante asuma toda la responsabilidad del proyecto, siempre con el apoyo que brinda la experiencia y los consejos del tutor. Dado que se trata del trabajo de una sola persona y con un tiempo limitado, la carga de trabajo debía ser la suficiente como para ser un proyecto retador, sin excederse para que sea posible su finalización dentro de plazos razonables. Uno de los alicientes de este tipo de trabajos es que permite al estudiante enfrentarse con problemas reales con los que deberá lidiar en su futura carrera profesional, además de aplicar muchos de los conceptos teóricos aprendidos durante su formación en la universidad.

Además de todo esto, permite al estudiante obtener una visión global y nuevos conocimientos sobre un lenguaje de programación para dispositivos móviles, algo que está en plena expansión y ciertamente, le será muy útil en el futuro. También proporciona un escenario perfecto en el que el estudiante pueda enfrentarse con un lenguaje de programación desde cero, contando con la orientación proporcionada por el tutor del proyecto.

El objetivo de este proyecto es obtener una aplicación para terminales móviles Android que utilice las funcionalidades de geolocalización. El programa incluirá una base de datos y deberá mostrar puntos de interés (obtenidos de la base de datos) cercanos al usuario mediante su posición, obtenida mediante GPS, y su orientación, obtenida por la brújula integrada en el terminal. Además de esto, deberá proveer al usuario de información adicional sobre los puntos. Ésta podrá ser obtenida mediante conexión a internet o desde la misma base de datos.

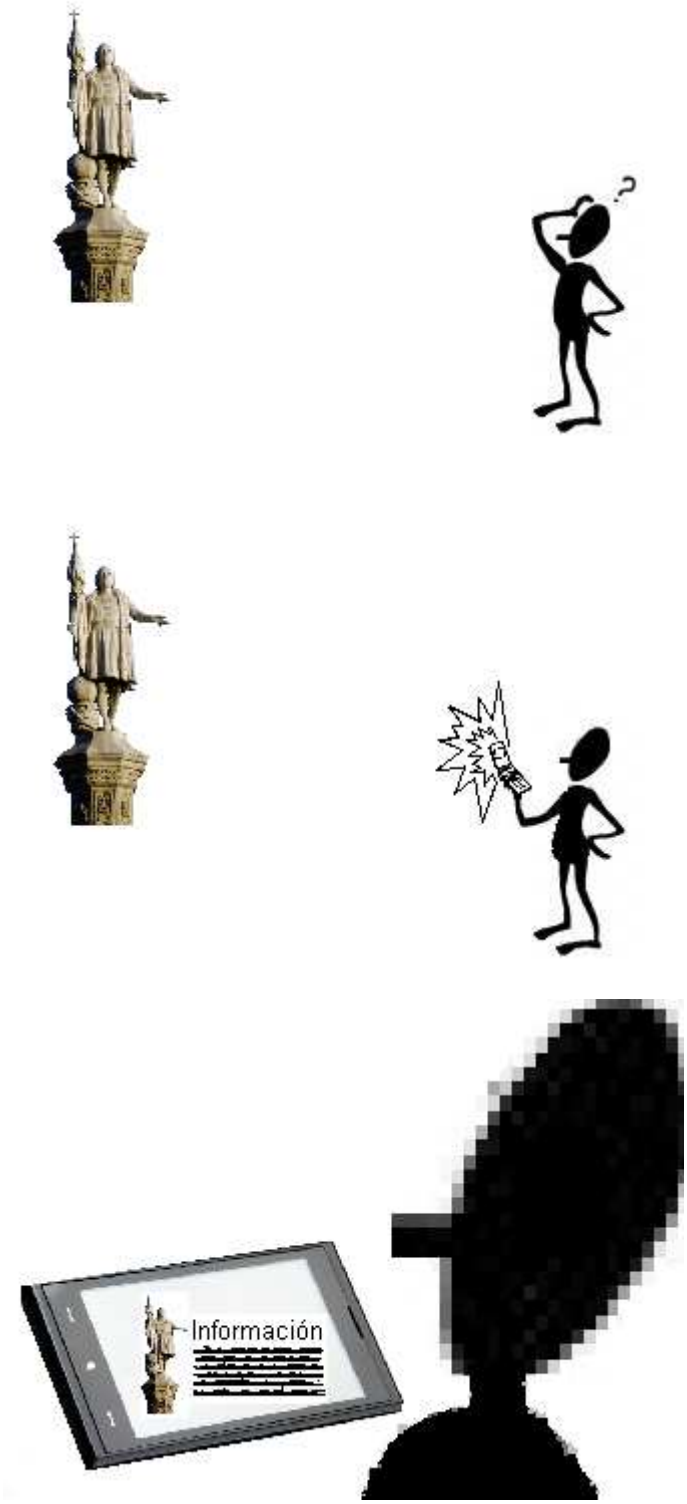


Ilustración 1: Funcionamiento básico de la aplicación

En la figura anterior se muestra la funcionalidad general de la aplicación. El usuario ve algo que le interesa, esto puede ser tanto un edificio como un monumento o algún sitio singular, y arranca la aplicación. Toma una foto y se le muestra la información del punto. El nombre del lugar lo obtiene gracias a la base de datos de la aplicación, pero toda la demás información la obtendrá a partir de búsquedas en internet.

Otro de los objetivos principales es que el estudiante adquiera una experiencia

global de cómo es un proyecto desarrollando la aplicación. Además de hacer una toma de contacto con un nuevo lenguaje de programación. Se le suma a todo esto el hecho de que el estudiante debe aprender a realizar todas las tareas de gestión de un proyecto, sobretodo la de planificación.

La intención es que la aplicación sea lo más social posible, de manera que los usuarios se animen a usarla e incluso a crear sus propias bases de datos. Para conseguir esto, se ha hecho hincapié en la sencillez y facilidad de uso, para incluir en el público potencial el máximo número de personas.

ESTADO DEL ARTE

En esta sección se analizarán todos los campos o ámbitos que tienen relación con este proyecto y con la aplicación a desarrollar. Al empezar el desarrollo de este proyecto, se analizaron los factores que tienen un impacto en el proyecto. Aquí se detallan éstos, así como la magnitud de su impacto y las perspectivas de que éstos cambien.

Se comienza esta sección con un análisis sobre la popularidad de las plataformas móviles en el mercado. A continuación, se llega a la sección que continúa con un análisis de la creciente capacidad de cómputo de los terminales móviles, seguido de otro estudio de la situación del mercado de las aplicaciones móviles. Después se introducirá un nuevo paradigma de programación, el context-aware programming.

En la penúltima sección se contrastará la posición del sistema operativo Android frente a sus principales rivales. Y por último se comentará la necesidad del Departamento de tener una aplicación base de realidad aumentada en Android.

1. Popularidad de los teléfonos inteligentes

Dado que la plataforma a la que va dirigida la aplicación desarrollada en esta sección se va a analizar en detalle el mercado de los teléfonos inteligentes. Se comenzará con una definición exhaustiva de lo que es un teléfono inteligente, continuando con un breve repaso a la historia de estos dispositivos, para pasar luego a comentar la situación actual del mercado. Para finalizar se expondrán las distintas predicciones que hacen los expertos sobre este sector.

El término teléfono inteligente (smartphone), tiene sentido si analizamos la historia de los terminales móviles. En un principio, los teléfonos móviles sólo tenían capacidad para hacer llamadas y enviar mensajes de texto (sms). Por otro lado, aparecieron las PDAs, que eran agendas electrónicas, donde el usuario podía apuntar citas en el calendario, información de contactos e incluso descargarse correos electrónicos de su ordenador.

A medida que el tiempo pasaba, los teléfonos ganaban más capacidades de las PDAs(como navegar por internet) de la misma manera que las PDAs agregaban capacidades de teléfonos móviles(podían descargarse correos directamente de un servidor), en ese momento nace el teléfono inteligente.

Lo que diferencia el teléfono inteligente de las PDAs (cada vez más en desuso) y de los teléfonos corrientes son las capacidades. Según about.com (Cassavoy, sin fecha) un teléfono inteligente debe tener: un sistema operativo (muchos de ellos basados en versiones ligeras de Linux), software (que permita realizar acciones tales como editar un fichero de texto), acceso a internet (tanto por WiFi como por la red telefónica), teclado QWERTY (ya sea físico o táctil) y mensajería (tanto sms, mms como o correos electrónicos).

El primer teléfono inteligente que se fabricó fue el IBM Simon en 1992, que tenía funcionalidades propias de una PDA, además de integrar todos los servicios y capacidades de un teléfono móvil. A pesar de esto, el término teléfono inteligente (smartphone), fue acuñado por Ericsson para uno de sus modelos, el GS88. Al principio, los teléfonos inteligentes se limitaban al ámbito empresarial y profesional, provocado

sobretodo por el alto precio de los terminales. Los teléfonos inteligentes se hicieron realmente populares cuando se hicieron accesibles y se bajaron los precios de los terminales y los de las tarifas de datos, extendiendo el uso de las redes 3G.

La verdadera revolución que provocó la actual popularidad de estos teléfonos fue la aparición del App Store en Apple (y todos sus homólogos como el Android Market o el BlackBerry App World). La posibilidad de descargar contenidos abría a desarrolladores de aplicaciones un mercado potencial enorme. La bajada de los precios de los terminales sumado a la de los planes de datos provocaron que ese mercado potencial pasara a ser una realidad.

En la actualidad el mercado de los teléfonos inteligentes está en alza y las estimaciones sobre el sector indican que esta tendencia continuará. Actualmente, según comScore hay 60 millones de teléfonos inteligentes sólo en Francia, España, Italia, Alemania y Reino Unido (comScore, 2010(1)). Con un crecimiento del 40% del año 2009 al 2010, donde más se ha notado este aumento ha sido en el Reino Unido, con un 70% a la alza (11 millones de terminales). Según Canalys(Clarke, 2011) en el primer cuarto del año 2011, se duplicó el número de terminales vendidos comparado con el mismo cuarto de 2010, de 55 millones a 101 millones de unidades.

Además, centrándonos en el objetivo de este proyecto, según el estudio antes referenciado de comScore, el 35,7% de los usuarios de teléfonos móviles han utilizado han utilizado alguna vez una aplicación de su terminal. Esto es una buena noticia, puesto que muchos de los móviles que hay en el mercado no tienen capacidad para ejecutar aplicaciones, por lo que a medida que los teléfonos inteligentes se extiendan lo usuarios usarán cada vez más las aplicaciones disponibles.

En cuanto a predicciones Analysys Mason (Analysys Mason, 2010) pronostica que habrá 1700 millones de teléfonos inteligentes para 2014, con un crecimiento anual del 32%. La mayor parte de este crecimiento provendrá de América Latina y de Asia. Últimamente esta tendencia se nota en que muchas compañías y operadoras han sacado continuamente al mercado nuevos modelos de teléfonos inteligentes.

Por otro lado Gartner hace una lectura menos conservadora, prevé 1820 millones de unidades para 2013 (Whitney, 2010). Nielsen por su lado, predice en 2010 que para finales del año 2011, las unidades vendidas de teléfonos inteligentes se igualará con la de los teléfonos corrientes (Small Business Lab, 2010).

2. Tecnologías disponibles

El objetivo de este proyecto es desarrollar una aplicación para una plataforma móvil. Esta plataforma debe tener unas tecnologías, sin las cuales la aplicación no podría ejecutarse en ella. Estas tecnologías son: plataforma de cómputo móvil, tecnologías de comunicación inalámbrica y tecnologías de posicionamiento. Se van a analizar las distintas alternativas existentes y justificar el porqué de la elección del teléfono móvil como plataforma destino para la aplicación.

Primero se estudiará la diferentes plataformas de cómputo móviles disponibles actualmente, así como los motivos que han llevado a escoger los teléfonos inteligentes para este proyecto. A continuación se tratarán las diferentes tecnologías de comunicación inalámbricas, así como las que ofrecen los teléfonos inteligentes. Se presentarán las tecnologías de posicionamiento existentes y las utilizadas para este proyecto. Por último se detallará la evolución y el estado actual de los teléfonos inteligentes.

Hoy en día hay gran variedad a la hora de elegir una plataforma de cómputo móvil. Las distintas opciones son: estaciones de trabajo móviles y wearables, tablets, UMPCs, PDAs y teléfonos inteligentes. De todas ellas, la más popular es sin duda los teléfonos inteligentes, es por ello que esta aplicación ha sido desarrollada para ésta.

Las estaciones de trabajo móviles hasta ahora se han basado en un ordenador portátil llevado en una mochila o algún prototipo similar (Cheok et Al., 2003). La capacidad de cómputo de éstas es mucho mayor que todas las demás plataformas. El tablet pc, también ha sido utilizado para varios proyectos de realidad aumentada (Klein et Al., 2004). Los UMPC, han aparecido recientemente y ya se están usando en varios ensayos de realidad aumentada, según Papagiannakis (Papagiannakis et Al, 2008), son la evolución de las estaciones de trabajo móviles. Las PDAs han servido de plataforma de computación para varios proyectos, pero ahora han caído en desuso después de la aparición de los UMPC y de los teléfonos inteligentes (Papagiannakis et Al, 2008). Una de las mayores limitaciones que tiene una PDA es la conectividad, ya que no posee conexión a la red telefónica. Los teléfonos inteligentes son teléfonos móviles con las capacidades de una PDA. De las mencionadas en este párrafo es la plataforma más utilizada, tanto por el público general como por desarrolladores de aplicaciones de realidad aumentada (como estación de cómputo o como display).

La principal razón para escoger los teléfonos inteligentes como plataforma frente a cualquiera de las demás es su popularidad, ya que cualquier otra limitaría la cantidad potencial de usuarios finales de la aplicación. Para una aplicación como la de este proyecto, las necesidades de capacidad de cómputo son relativamente pequeñas, de no ser así sólo se podría escoger entre las estaciones de trabajo móviles o los UMPC. Comparado con las PDAs y los tablet PC, las capacidades de los teléfonos inteligentes son similares.

Las tecnologías que permiten conectividad inalámbrica son muy numerosas. Aunque más adelante, en la sección “Evolución de las redes móviles”, se expondrán las virtudes y defectos de todas ellas con más detalle. Aquí se nombrarán por encima las que se han utilizado para proyectos similares a este (context-aware y realidad aumentada).

Una de las más comunes es WAN (telefónica) y 3G, la cobertura es amplia aunque el ancho de banda no es muy grande. Jonietza junto con Nokia desarrolló un prototipo que hallaba la distancia de cualquier objeto al que apuntase su cámara que usaba esta tecnología (Jonietza, 2007). Otra tecnología usó un prototipo de Pacman (Human Pacman), un juego multijugador en el que se seguía el movimiento de los jugadores mediante WLANs(Cheok et Al., 2003). Las redes de área personal (WPAN) se han usado en el proyecto Ubibus, para permitir a personas ciegas solicitar con antelación que el autobús de la linea que quiera pare en su parada, además de avisarle cuando el autobús llegue a la parada (Banâtre et Al., 2004).

Actualmente, los teléfonos móviles ofrecen conectividad tanto por redes de telefonía como redes de datos inalámbricas. En los teléfonos inteligentes, lo más común es encontrarse ya con conectividad 3G o 3,5G (UMTS, HSDPA, HSDPA+...) que permite conexiones de datos de 384Kbps hasta 21Mbps. Además de a este tipo de redes, también se pueden conectar a redes de datos inalámbricas (WiFi, WiMAX...), obteniendo por regla general mejores prestaciones en cuanto a ancho de banda (suele estar limitada por número de usuarios y estándar utilizado).

Para terminar, se muestra el siguiente gráfico de la evolución de las plataformas de computación, en el que relaciona la capacidad de cómputo con la conectividad de las

plataformas. La proyección que hace Hewitt (Hewitt, 1993), es que los desarrollos futuros tenderán hacia plataformas más móviles.

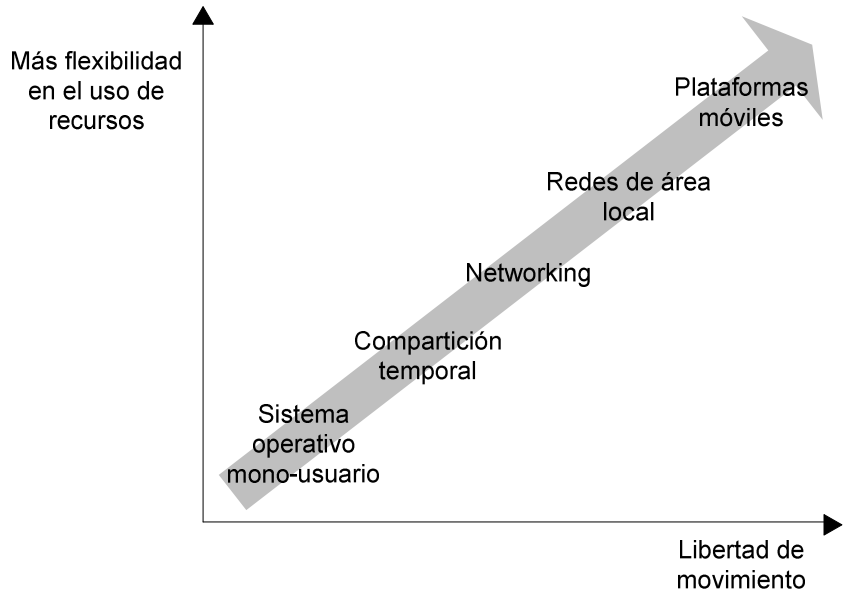


Ilustración 2: Evolución de la computación

Para las aplicaciones que utilizan realidad aumentada y context aware computing, obtener la posición del usuario es estrictamente necesario. Las tecnologías de posicionamiento son muy variadas, tanto las tecnologías en sí como el uso que se puede hacer de ellas.

Existen variadas tecnologías diferentes, en este apartado se mencionarán las que de más popularidad gozan y más relación tienen con este proyecto, lo que no quiere decir que no existan otras muchas.

Como tecnologías de posicionamiento predominan el GPS, el posicionamiento usando redes móviles y usando redes inalámbricas de datos. El GPS funciona en casi todos los lugares excepto en los que haya algún obstáculo, ya sea un techo o un árbol, que impida al navegador recibir la señal de al menos 4 satélites. Las redes móviles permiten localizar el terminal utilizando triangulaciones, resultando en un posicionamiento menos preciso que el del GPS. La localización por redes inalámbricas de datos se basa en la fuerza de la señal que llega al punto de acceso para calcular la posición del terminal. Un ejemplo de aplicación que utiliza este método es Google Latitude, que se basa en la triangulación con antenas de telefonía o por la fuerza de la señal del punto de acceso.

En cuanto a las tecnologías de seguimiento de movimiento destacan los acelerómetros y los magnéticos. Los acelerómetros funcionan bien si no se hacen movimientos bruscos. En cuanto a los magnéticos, son utilizados sobretodo como brújulas, muy útiles para cualquier aplicación que necesite saber la orientación del usuario.

En el caso de la aplicación que se ha desarrollado para este proyecto, la elección de la plataforma limita las opciones de tecnologías de posicionamiento. En la mayoría de los teléfonos inteligentes, el posicionamiento puede hacerse mediante GPS, las redes de datos inalámbricas o triangulando antenas de telefonía móvil.

3. Mercado de las aplicaciones móviles

Ya que la aplicación desarrollada tiene como plataforma objetivo los teléfonos inteligentes, en esta sección se analizará el mercado de las aplicaciones para este tipo de terminal. Primero se comentarán los inicios de las aplicaciones para estos terminales para pasar a analizar a continuación el estado actual del mercado. Para finalizar se expondrán las distintas predicciones que hacen los expertos en la materia.

A pesar de que siempre han existido aplicaciones para teléfonos móviles, no fue hasta la aparición de los teléfonos inteligentes cuando empezó a crecer enormemente el sector. El smartphone, además de tener unas mayores capacidades y funcionalidades que un teléfono móvil estándar, tiene un sistema operativo. Este sistema operativo sirve de base para que programadores puedan desarrollar sus propias aplicaciones. Los teléfonos anteriores a los smartphones, no disponían de esta posibilidad, por lo que no existía realmente un mercado de aplicaciones.

El primer mercado de aplicaciones nace con la platafoma Palm en 2002. Que es la primera en proveer un API para que programadores ajenos al fabricante del terminal o al sistema operativo, pudieran desarrollar nuevas aplicaciones. Pero hasta que realmente no se popularizaron los teléfonos inteligentes, no apareció el enorme mercado de las aplicaciones.

Actualmente, como se comentó en la sección “Popularidad de los teléfonos inteligentes”, según comScore (comScore, 2010(1)), el 35,7% de los usuarios de teléfonos móviles ha utilizado alguna vez una aplicación. Gracias a las continuas ventas de teléfonos inteligentes, el descenso en precios de los planes de datos y la aparición de las tiendas de aplicaciones la expansión del sector de las aplicaciones ha sido enorme. Según ABI Research, las descargas en el año 2010 fueron 7900 millones (Meyers, 2011), en comparación con las 6000 millones que predijeron en 2010.

En 2009 y 2010 comScore (comScore, 2010(2)) llevó a cabo, en un espacio de tiempo de 3 meses un estudio en el que se reflejaba las tendencias más populares entre el mercado de aplicaciones para teléfonos inteligentes. Después de los juegos nativos de los teléfonos, las aplicaciones de las redes sociales son las más populares, seguidos de las de noticias y las de deportes. La que sufrió un mayor incremento de popularidad fue las redes sociales, que creció un 240% en el mismo periodo del año 2009 al 2010.

Según un estudio llevado a cabo por Research2Guidance en 2009 los usuarios que utilizan aplicaciones en un teléfono inteligente fueron 100 millones, para el año 2013 la perspectiva es de casi 1000 millones. El mismo estudio prevée unos ingresos de 15.650 millones de dólares para 2013 comparado con los 1.940 del año 2009. El mercado de las aplicaciones para teléfonos inteligentes, según Gartner(Gartner, 2010), será uno de los segmentos con mayor crecimiento dentro del sector de las telecomunicaciones en los próximos años.

Uno de los factores a considerar sobre el número de descargas, es que muchas de éstas se realizan de manera gratuita (según Gartner el 82%) y la tendencia es que vaya a más. Aunque puede parecer que una descarga gratuita implica que el desarrollador no obtiene ingresos por ella, muchas veces se cobra, una vez en la aplicación, por contenidos extras. De esta manera la popularidad de las aplicaciones es mucho mayor, según Jake Saunders, vicepresidente del área de predicciones de ABI Research, "dado que la competencia es cada vez mayor, los propios responsables de las aplicaciones tienden a bajar los precios con la intención de permanecer en la cima de las listas de descarga". Esto provocará que las descargas tiendan a subir mientras que los ingresos

bajarán, dada la alta competitividad que hay en el sector.

Otro de los retos del sector es la diversidad de dispositivos móviles y sistemas operativos. Para los proveedores de contenidos, esto es una inconveniencia, puesto que todo les resultaría mucho más sencillos si sólo tuviesen que desarrollar para una plataforma. Los sistemas operativos iOS, Android, BlackBerry y muchos otros tienen características y capacidades muy diferentes.

4. Aparición del context-aware programming y la realidad aumentada

La aplicación desarrollada para este proyecto utiliza dos paradigmas de programación relativamente recientes. El primero de ellos, y de mayor peso en este proyecto, es el context-aware computing y el segundo, y no por ello menos importante, la realidad aumentada. En esta subsección se explicará los fundamentos de cada uno de ellos y la evolución a lo largo del tiempo de cada uno de ellos. Además se analizará la situación actual, las arquitecturas existentes y las previsiones que hacen los expertos sobre la proyección de ambos en un futuro relativamente cercano.

Se comenzará explicando el concepto y la historia del context-aware computing, continuando con la realidad aumentada. Al finalizar se proseguirá detallando la relación entre ambas. Para finalizar se analizará el estado del arte en cuanto a context-aware computing, las diferentes arquitecturas y su proyección futura estimada, análogamente se analizará la realidad aumentada.

El context aware computing es un paradigma de programación que encuentra y usa información del entorno. Hay dos visiones principales sobre el concepto de context-computing. La primera es de Weiser en “Ubiquitous Computing” (Weiser, 1991). Para él, hay dos factores primordiales que propician la creación del context-aware computing. El primero es que cuanta menos interacción del usuario necesita una aplicación, mejor es ésta. El segundo es la creciente capacidad de cómputo de los sistemas embebidos y su reducido coste. Según Weiser, estos dos factores están íntimamente relacionados y abrirían la posibilidad de que la computación formase parte del entorno (context-computing).

Aplicación
Almacenamiento/Gestión
Preprocesado
Recuperación de datos en bruto
Sensores

Tabla 1: Abstracción por capas del context-aware

La otra visión principal la tiene Ishii en “Tangible Bits” (Ishii et al, 1998), que tiene como origen el mencionado anteriormente “Ubiquitous Computing” (Weiser, 1991). En él Ishii plantea que el humano vive en dos mundos distintos, el mundo de la computación ("bits") y el mundo real ("átomos"). Aunque el mundo real sea con el que el humano está más familiarizado, su interacción con el mundo de la computación se ha

hecho sencilla, a través de la evolución. Actualmente la mayor parte de la interacción está basada en teclas y un ratón, mientras que Ishii se ha dedicado a buscar interacciones más amplias. Un ejemplo de ello es un kit físico de construcción que genera narrativa multimedia (Gorbet et al., 1998).

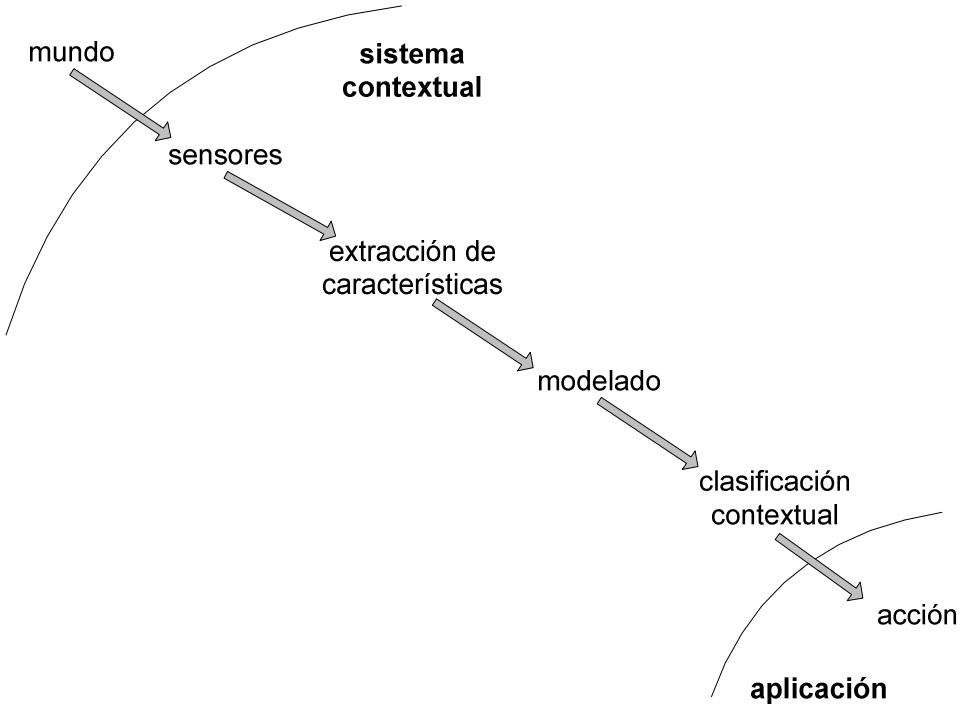


Ilustración 3: Arquitectura de un sistema de context-aware

En la Ilustración 3 se muestra lo que tienen en común las dos visiones expuestas, que es que toman el mundo como interfaz. Según Paul Dourish, en su artículo "Seeking a Foundation for Context-Aware Computing", hay muy poco consenso en lo que context-aware computing es exactamente. Lo que queda patente es que la aplicación desarrollada para este proyecto utiliza conceptos del context-aware computing puesto que dependiendo del entorno (lugar) donde se encuentre el usuario, los resultados que aparecerán en la pantalla variarán.

	Manual	**Automática**
Información	Selección por proximidad e información contextual	Reconfiguración contextual automática
Comando	Comandos contextuales	Acciones contextualmente ejecutadas

Tabla 2: Dimensiones de un programa context-aware

Según Schilit (Schilit et Al., 1995), existen en el context-aware computing 4 dimensiones, mostradas en la Tabla 2. Más adelante analizaremos la aplicación desarrollada en este proyecto desde el punto de vista de las dimensiones de Schilit. La aplicación que se ha desarrollado para este proyecto, visto desde las dimensiones de Schilit, se encontraría en Contextual Command. Se trata de acciones que realiza el usuario y que tienen diferentes resultados según el lugar en el que se ejecuten.

Las aplicaciones que se han utilizan el context-aware computing han sido de

bastante interés para universidades y centros de investigación durante los últimos años. Muchas de esas investigaciones han tenido por objetivo desarrollar una guía turística contextual. De todos ellos los que más impacto han causado han sido: el sistema GUIDE, una guía de la ciudad de Lancaster, desarrollada por la Universidad de Lancaster y sistemas a una escala menor han sido utilizados para museos (Bederson, 1995 y Oppermann, 1998).

La realidad aumentada es una variación de la realidad virtual. En la realidad virtual se sumerge completamente al usuario en un entorno puramente sintético, dejándole sin capacidad de interacción con su entorno real. Por otra parte en la realidad aumentada, el usuario ve en un dispositivo información sobreimpresa que le es relevante y además puede interactuar con su entrono. Como lo expresa Ronald T. Azuma, en A Survey of Augmented Reality (Ronald T. Azuma, 1997) "…se puede decir que la realidad aumentada complementa la realidad, en vez de reemplazarla". Milgram expresó en A Taxonomy of Mixed Reality Virtual Displays (Milgram et Al, 1994), que la realidad aumentada podía estar a medio camino entre la telepresencia y la realidad virtual.

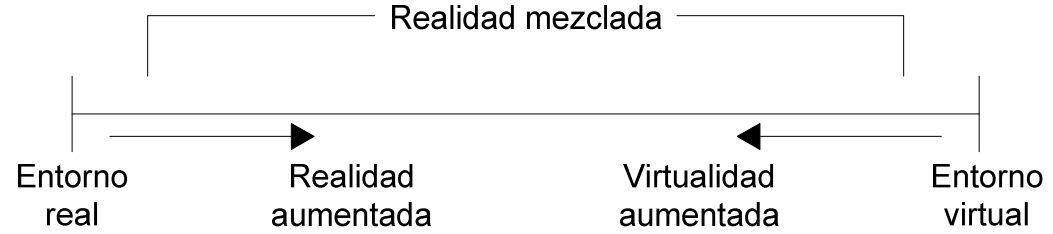


Ilustración 4: El espacio de Mildred

A pesar de que el término realidad aumentada fue acuñado en la década de los noventa, en 1968 Ivan Sutherland y algunos colegas suyos desarrollaron el primer sistema de realidad aumentada. Se trataban de unas gafas con un display que mostraba información sobreimpresa y se convirtió en la base de todos los estudios de hardware gráfico (Henry Fuchs, 2008). Entre los años setenta y ochenta, la realidad aumentada fue una de los ámbitos de investigación de varias instituciones estadounidenses, tales como el Ames Research Center de la NASA, el MIT y la Universidad de Carolina del Norte en Chapel Hill. Por último un sistema contextual y que además usaba realidad aumentada fue el Chameleon, desarrollado por la Univerisidad de Toronto (Fitzmaurice, 1993), que permitía explorar información (por ejemplo, la de un mapa) a través de un monitor de mano.

No fue hasta el año 1997 que la realidad aumentada pasó a ser móvil, gracias a los avances tecnológicos. Fue la Touring Machine de la Universidad de Columbia presentada en el International Symposyum on Wearable Computing (Feiner et Al, 1997), la que rompió esta barrera por primera vez. El proyecto trataba de hacer una guía turística del campus de la Universidad de Columbia. Para tener una funcionalidad similar a la del programa desarrollado para este proyecto, el usuario debía cargar con una mochila que llevaba: un ordenador, un módulo de GPS, una tarjeta para conectarse a Internet, una fuente de alimentación y un medidor de campo magnético que hacía de brújula. Además de todo esto, no bastaba con el módulo de GPS para saber la posición, si no que debía compararse con el resultado obtenido en otro módulo del cual se sabía la posición exacta. Con estos datos nos hacemos una idea de lo que ha avanzado el campo de la realidad aumentada gracias a las nuevas tecnologías y los avances en computación. Höllerer (Höllerer et Al, 2004) establece que para desarrollar una aplicación de realidad aumentada son necesarios los siguientes módulos: computadora, pantalla, sistema de

posicionamiento, interfaz con el usuario, conectividad inalámbrica, bases de datos y algún medio de acceso a ellas. Podemos comprobar que cuando Feiner y sus colegas desarrollaron su proyecto, era necesario un módulo físico casi para cada funcionalidad, mientras que hoy en día basta con un terminal móvil de prestaciones estándar.

La relación entre el context aware computing y la realidad aumentada en la aplicación desarrollada para este proyecto es la siguiente: la realidad aumentada es la interfaz con la cual mostramos los resultados de una programación dependiente del entorno (context-aware) (Kjeldskov, 2003). La mayoría de las aplicaciones que utilizan context-aware computing que tienen una interfaz gráfica, suelen contar con la realidad aumentada (o alguna de las variantes existentes en el Continuum de Milgram) para mostrar los resultados obtenidos por esa computación dependiente de la localización.

La mayoría de aplicaciones existentes que utilizan context-aware programming usan sólo ciertos valores contextuales, como son la localización o el tiempo. El primero de estos valores es el más utilizado en aplicaciones comerciales para teléfonos inteligentes. Fuera del mercado de aplicaciones para teléfonos inteligentes la programación context-aware programming se reducen a entornos de investigación en laboratorios y universidades. Esto se debe a la cantidad de recursos que utilizan y la cantidad de datos a procesar, además de lo costoso que es el desarrollo (Korkea, 2006).

El propósito más común de las aplicaciones comerciales que utilizan la programación dependiente del entorno es servir como guía, tanto turística como de tiendas. Una de las más populares es Layar, desarrollada para Android por Google (que además utiliza realidad aumentada para la interfaz de usuario). Existen también proyectos que pretender usar información contextual, como la posición, movimiento y sonido, para determinar la situación del usuario (si está en el trabajo, en casa, en una conversación...) y, en consecuencia, modificar el modo en el que se le avisan las llamadas o mensajes (con vibración o subiendo el volumen del timbre). Existen numerosos proyectos como este, como el de De Vaul y Dunn que usa el MIThril Inference Engine, que permite crear aplicaciones context-aware y que fue desarrollado por el MIT. El proyecto empezó en 2001 y la última actualización se hizo en Noviembre del mismo año.

Antes de nombrar y analizar algunas de las arquitecturas context-aware que han ido apareciendo, se va a definir el ámbito del estudio. Contexto, según Dey (Dey, 2000 2001), es: "*cualquier información que puede ser utilizada para caracterizar la situación de una entidad; una entidad es una persona, lugar o cosa considerada relevante para la interacción entre un usuario y una aplicación*". En el caso de este proyecto, nos centraremos en la interacción con lugares, puesto que las demás entidades darían lugar a un estudio mucho más amplio y con poca o nula relación con este proyecto.

El Context Toolkit, propuesto y desarrollado por Dey (Dey, Salber, and Abowd 2001), proporciona un nivel de abstracción que divide la aplicación en widgets, aggregators, interpreters, services and discoverers. Los widgets se encargan de recoger los datos de los sensores, los aggregators la recogen de varios sensores y los interpreters se encargan de traducirla en información de alto nivel. Está basado en Java, HTTP y XML.

Gaia es otra arquitectura propuesta por Roman (Roman et al 2002) que permite el desarrollo de aplicaciones para espacios activos. Los espacios activos son entornos programables, en los cuales el usuario puede interactuar con varios dispositivos y servicios. Ofrece cinco servicios: el *event manager* que distribuye los eventos en todo el

espacio activo, *presence service* que controla la informacion de componentes y dispositivos software, *space repositories* que guarda información de las entidades que se encuentra en el espacio activo, *context service* por el cual la aplicación busca y registra información contextual y *context file system*, un sistema de archivos dependiente del entorno.

La arquitectura MADAM, propuesta por Satyanarayanan (Satyanarayanan, 2001) y Mikalsen (Mikalsen et al, 2006) contempla dos tipos de entidades, los proveedores de contexto (*context-providers*) y los clientes de contexto (*context-clients*). Todo usuario de este tipo de arquitectura sería tanto cliente como proveedor. Además hay un repositorio central que almacena toda la información que se transmite y que actúa como nodo central. Esta arquitectura permite escalabilidad ya que se pueden interconectar nodos centrales que estén cerca (y por lo tanto, de cierta manera comparten el entorno y por lo tanto compartirán datos útiles entre ellos).

La proyección de la computación dependiente del entorno que hacen consultoras expertas en tecnología es muy alentadora. Para Gartner, las prioridades de un año a otro (2010-2011), han cambiado. Estos cambios se muestran en la Tabla 3 y como se puede apreciar, el context-aware computing (junto con el ubiquitous computing) es una de las novedades.

Posición	10 Áreas tecnológicas importantes para 2010	10 Áreas tecnológicas importantes para 2011
1	Cloud computing	Cloud computing
2	Análitica avanzada	Aplicaciones para móviles y tablets
3	Computación cliente	Analítica de próxima generación
4	IT verde	Analíticas sociales
5	Reestructuración de data center	Comunicación y colaboración social
6	Computación social	Video
7	Seguridad, monitoración de actividad	Computación context-aware
8	Memoria flash	Computación ubicua
9	Virtualización para disponibilidad	Memoria Storage Class
10	Aplicaciones móviles	Infraestructuras y computación textil

Tabla 3: Nuevas tecnologías importantes según Gartner

Esto demuestra, que el futuro de la computación dependiente del entorno tiene mucho recorrido por delante y que el momento actual es sólo el principio. Además, otra de las nuevas prioridades según Gartner, que también podemos apreciar en la Tabla 3, es que las aplicaciones móviles también serán un sector clave en el año 2011, lo cual no

hace más que validar las buenas perspectivas de este proyecto.

En la actualidad, hay tres variantes en el campo de la realidad aumentada. Encontramos la realidad aumentada basada en marcadores, la basada en reconocimiento de objetos y basada en GPS o posicionamiento.

La realidad aumentada basada en markers utiliza un patrón que se imprime sobre algún material (normalmente papel), el cual sirve para que el software lo reconozca y pueda representar el objeto en su posición. Además de la posición estos marcadores también determinan el objeto a representar, es decir que se puede tener variedad en los objetos a representar y dependiendo del patrón se representará uno u otro. Un ejemplo de este tipo de aplicación es el juego llamado Gizmondo, en el cual se utiliza una tarjeta que representa el tablero del juego y basándose en éste, el usuario posiciona torres en él.

Otra variante de la realidad aumentada está basada en el reconocimiento de objetos. Es probablemente la variante más costosa computacionalmente, ya que se tiene que realizar un reconocimiento y cotejarlo con una base de datos. Este campo es el más moderno y no es tan popular como los otros. Un ejemplo de aplicación es Polar Rose, adquirida por Apple, en la cual el usuario toma una foto y el software intenta reconocer las palabras que aparecen en la fotografía. También permite un procesado posterior de esas palabras, como la traducción por ejemplo.

La última variante son las aplicaciones basadas en el posicionamiento. Este posicionamiento puede ser realizado por GPS o por otros métodos como la fuerza con la que llega la señal de WiFi al dispositivo. Uno de los inconvenientes de estos sistemas es la poca precisión que se obtiene, llegando a ser inútil este sistema si se necesita una precisión de menos de varios metros (ya que ésta depende del alcance del router WiFi). Un ejemplo de aplicación comercial que utiliza esta tecnología es Layar.

Una de las arquitecturas más comunes actualmente a la hora de desarrollar una aplicación de realidad aumentada es la basada en plug-ins. Opuestamente a la filosofía utilizada al comienzo de la realidad aumentada, donde un sistema cumplía una sola función y tenían un software monolítico, ahora existen toolkits que permiten desarrollar aplicaciones con diversos propósitos para una misma plataforma. Una plataforma colaborativa llamada “Studierstube” (Schmalstieg et al, 2002), permite desarrollar distintas aplicaciones que son escritas y compiladas separadamente. Una vez desarrolladas las aplicaciones, se introducen en un entorno de ejecución que sólo acepta una instancia de cada aplicación, por lo que cumple múltiples funcionalidades a la vez, algo así como un sistema de realidad aumentada multitarea.

Otra arquitectura bastante novedosa es la utilizada por el proyecto MARS2002, en la que un núcleo central realiza una serie de tareas, por ejemplo actuando como backend manteniendo una base de datos y devolviendo resultados a los terminales. También basada en núcleos centrales también está el MR Software Suite (MRSS) (Hughes et al, 2005), donde cada núcleo se encarga de procesar distintos datos (tres para video y audio, uno para coordinar los demás).

Las perspectivas del mercado de las aplicaciones de realidad aumentada para teléfonos inteligentes es muy esperanzadora. Las crecientes funcionalidades y capacidades de los terminales de última generación permiten desarrollar aplicaciones de manera más sencilla (gracias a las APIs) y con funcionalidades más útiles para la sociedad en conjunto. Los grandes del sector, como Google y Apple, apuestan cada vez más por este tipo de aplicaciones. Según Mark Beccue de ABI Research, el mercado de aplicaciones de realidad aumentada llegará a los tres mil millones de dolares para 2016,

mientras que en 2010 sólo llegaba a los 21 millones (Clint Boulton, 2011).

5. Evolución de las redes móviles

Existen numerosas tecnologías de comunicación inalámbricas disponibles actualmente que podrían ser utilizadas para este proyecto. Sus características más relevantes, en el ámbito de la realidad aumentada, son el ancho de banda y la disponibilidad de la red. En el caso de una aplicación como la que se presenta en este proyecto, la latencia y jitter no son factores determinantes, pues los datos "críticos"(las bases de datos que contienen los puntos de interés) se almacenan en local, y la red se utiliza para obtener información adicional. A continuación presentamos todas aquellas tecnologías que han sido utilizadas para proyectos de context-aware programming.

La tecnología más usada en los teléfonos móviles es las redes las Wireless WAN y las redes 3G, que se sirven de la red telefónica para la transmisión de datos. Su principal ventaja respecto al resto es la disponibilidad, que en el caso de España es casi del 98%. Previa a la tecnología 3G, era común realizar conexiones de datos usando redes GSM (2G), pero la latencia y el ancho de banda eran muy pobres. Las comunicaciones 2,5G (GPSR) utilizan las redes 2G, y consiguen mejorar las características de éstas y además, como las comunicaciones 3G, soportan aplicaciones multimedia.

Otro tipo de red son las WLAN, que son redes inalámbricas implementadas en áreas acotadas como oficinas o domicilios particulares. La principal virtud de estas redes son el ancho de banda, el cual es muy superior al proporcionado por las redes de telefonía móvil. Como contrapartida, la movilidad dentro de estas redes está restringida a la cobertura de éstas (que suele rondar los entre 20 y 250 metros), añadiendo además un empobrecimiento del rendimiento cuanto más lejos del punto de acceso se esté. Recientemente ha aparecido una variante de estas redes llamada WiMax (WiFi Max), las cuales comparten el fundamento pero poseen un alcance mucho mayor a las redes WLAN convencionales (hasta 80 kilómetros).

Para finalizar, las WPAN son redes personales inalámbricas. Éstas están muy limitadas en cobertura (unos pocos metros), poseen un ancho de banda alto y se han utilizado sobretodo para tareas como impresión o control remoto. Un ejemplo de este tipo de conectividad es el Bluetooth.

Las distintas redes inalámbricas poseen características muy dispares. El ancho de banda y la disponibilidad de la red son los factores más determinantes del rendimiento de una aplicación que la usa. Las aplicaciones que utilizan realidad aumentada, generalmente hacen uso de comunicaciones inalámbricas, aunque no siempre es estrictamente necesario (si los datos están almacenados localmente).

El hecho de escoger una plataforma de cómputo limita el abanico de posibilidades de conectividad a las redes. Este caso, el de los teléfonos inteligentes, es de los que mayor conectividad posee, ya que en general existe la posibilidad de conectarlos a las redes de telefonía móvil y a las WLANs.

6. El sistema operativo Android

La aplicación desarrollada para este proyecto tiene por objetivo la platafoma Android, por ello en esta sección se tratará el sistema operativo Android, comenzando por su definición. Se continuará relatando brevemente su historia y su situación actual respecto a los demás sistemas operativos. Para finalizar se mencionarán las distintas

predicciones que se han hecho sobre el futuro del mercado de los sistemas operativos para teléfonos inteligentes.

Android es un sistema operativo para teléfonos inteligentes basado en Linux que se lanzó en 2007. También incluye middleware y aplicaciones clave (agenda, reproductor de música...). Google compró la empresa Android en 2005, y ahora ellos y la Open Handset Alliance son los encargados de desarrollar y publicar nuevas versiones.

Además Android es multimarca, varios fabricantes tienen modelos que corren el sistema Android, las más populares son HTC y Samsung. Además Google tiene su propia marca de teléfonos llamada Nexus, los cuales son fabricados también por la compañía Taiwanesa HTC.

Como se ha mencionado en la sección “Mercado de aplicaciones móviles”, existen varios sistemas operativos en el mercado y la convergencia, al menos a corto plazo, no parece probable. Por este motivo a la hora de desarrollar una aplicación ha de elegirse la plataforma objetivo, teniendo en cuenta que portarlo a otra para la que no fue desarrollada puede ser una tarea complicada, casi como desarrollar el software de nuevo.

En relación con sus competidores principales, Android está muy bien posicionado. Según IDC, el crecimiento de Android en el mercado europeo en el último trimestre del año 2009 al 2010 fue del 1580%, una cifra muy considerable. Esto supone una cuota de mercado del 31%, a pesar de que en el mercado europeo está muy arraigado el sistema operativo de Nokia, Symbian (Racoma, 2011).

La Ilustración 5, muestra la cuota de mercado mundial a finales de 2010 según el sistema operativo de los teléfonos inteligentes. Como se puede apreciar Android es el competidor dominante, seguido por Symbian. La anunciada fusión entre Nokia y Windows para desarrollar teléfonos inteligentes de alta gama podría arrebatar algo de cuota de mercado a Android.

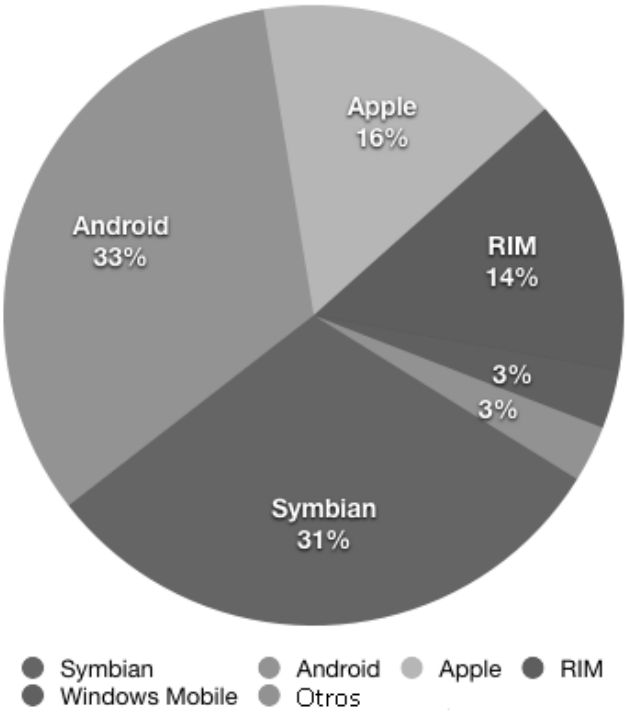


Ilustración 5: Cuota de mercado móvil según sistema operativo
(Fuente: Canalys, obtenido de Wikipedia)

Las principales consultoras del sector auguran un buen porvenir al sistema Android, tal como se muestra en la Tabla 5. Una de las razones de ello es que la mayor parte del crecimiento del sector provendrá de América Latina y de Asia. El principal competidor de Android es el iOS, que sólo se puede instalar en teléfonos Apple, que tienen un precio de venta al público alto, esto hace que sea poco probable que en los países en vías de desarrollo éste obtenga mucha popularidad (Analysys Mason,2010).

Sistema Operativo	Cuota de mercado 2016 (según ABI Research)	Cuota de mercado 2015 (según IDC)
Android	45%	45,4%
iOS	19%	15,3%
BlackBerry	14%	13,7%
Bada	10%	---
Windows Phone	7%	20,9%
Symbian	---	0,2%
Otros	5%	4,6%

Tabla 4: Comparativa de predicciones de ABI Research y IDC

Todo esto apunta a que la popularidad de Android en el mercado de los sistemas operativos, aunque ya es bastante grande, seguirá creciendo, asegurando así un gran público para la aplicación que se ha desarrollado para este proyecto.

7. Bases de datos XML

Para este proyecto se ha utilizado una base de datos XML nativa. Este tipo de base de datos utiliza XML como unidad fundamental y es muy parecida a las orientadas objetos, ya que cada uno de los nodos puede contener información muy heterogénea, como los objetos (Mertz, 2001).

Para crear una base de datos XML según XML:DB initiative(Kimbro Staken, 2001), se debe:

- Definir un modelo lógico para un documento XML que permita guardar y recuperar información. Como mínimo este modelo debe incluir elementos, atributos, PCDATA y un orden. Un ejemplo de estos modelos es XPath.

- La unidad fundamental de almacenamiento debe ser un documento XML.

- No necesita tener un modelo de almacenamiento físico en particular.

En el caso de la aplicación desarrollada para este proyecto se cumplen todos estos requisitos. Los elementos son las localizaciones, cada una de ellas tiene ciertos atributos, cuyos valores son PCDATA y además se sigue un orden al declarar una base

de datos (por ejemplo, se empieza con el nombre de la base de datos, seguido de todos los elementos que contiene, además de los atributos de cada uno de ellos).

Una de las razones para utilizar bases de datos XML, según O´Conell(O´Conell, 2005) es el incremento del uso de XML para transportar información a través de internet, por lo que se hacen conversiones de datos continuamente al formato XML y viceversa. Para mejorar el rendimiento de estos procesos de conversión, lo óptimo es guardar los datos en formato XML.

Además este tipo de base de datos, al ser independientes del esquema que se utilice, aporta mucha flexibilidad y hace el desarrollo mucho más fácil. Esto también puede ser un inconveniente, ya que esta flexibilidad puede traducirse en datos con poca homogeneidad. Dado que en este proyecto las bases de datos no son muy grandes y los esquemas relativamente sencillos, este obstáculo se podrá salvar con un mínimo trabajo de administración. Otra manera de evitar este inconveniente es usar un DTD que valide los datos, de manera que cumpla con el esquema (como por ejemplo el de W3C).

Una de las desventajas más importantes de este tipo de bases de datos es el modo de actualizar la información. No hay manera de hacerlo directamente, si no que se debe obtener el documento XML del servidor, modificarlo y devolverlo a su sitio. En el caso de que las consultas a la base de datos fuesen constantes, supondría un problema crítico, ya que se interrumpiría el servicio. Esto no supone un problema para este proyecto, ya que las bases de datos se guardarán localmente y no se verán modificadas. Como ampliación, se podrán crear nuevas bases de datos y subirlas al servidor, pero en ningún caso modificar las existentes(a menos que sea el administrador de la web, por lo que la interrupción del servicio será pequeña).

El uso óptimo de las bases de datos XML nativas es para organizar datos con una estructura poco rígida y definida, es decir, que es perfecta para este proyecto, en el caso de que se desarrollen más bases de datos que incluyan nuevos campos. Por poner un ejemplo, una nueva base de datos incluye en un campo una página web para que el usuario publique un comentario sobre el sitio donde se encuentra, con este tipo de base de datos seremos capaces de acceder a este campo y utilizarlo en el código. Permite, por tanto, definir el esquema de cada documento XML independientemente de las demás.

En un principio Android implementó para su sistema operativo dos tipos de parsers para XML. El primero DOM (Document Object Model) y el segundo SAX (Simple API for XML). Más adelante se introdujo el XML Pull Parser que, según Android, mejoraba el rendimiento para aplicaciones con poco uso de memoria, como las desarrolladas para plataformas móviles.

Un parser DOM transcribe todos los datos y su estructura a código nativo. Todo el procesado se hace de una vez y se analizan y se guardan todos los datos, por lo que es la opción que más memoria utiliza. El parser SAX permite al usuario implementar sus propias clases, dejándole así la opción de extraer sólo la información que le interesa. La última opción, el XML Pull Parser, se basa en un bucle que va leyendo las etiquetas y puede ser parado por el usuario en cualquier momento, por lo que se puede decir que se hace un procesado bajo demanda. Además tiene la ventaja respecto a SAX que elimina todo la sobrecarga que suponen las llamadas a métodos y clases. Android no incluye el StAX, Streaming API for XML, que es razonablemente parecido al XML Pull Parser. La ventaja de estos parsers frente a los demás es que con una variable booleana se puede terminar con el procesamiento, sin haber recorrido todo el documento entero. Esto tiene una utilidad mayor para los terminales móviles, que normalmente tienen un ancho de banda bastante limitado.

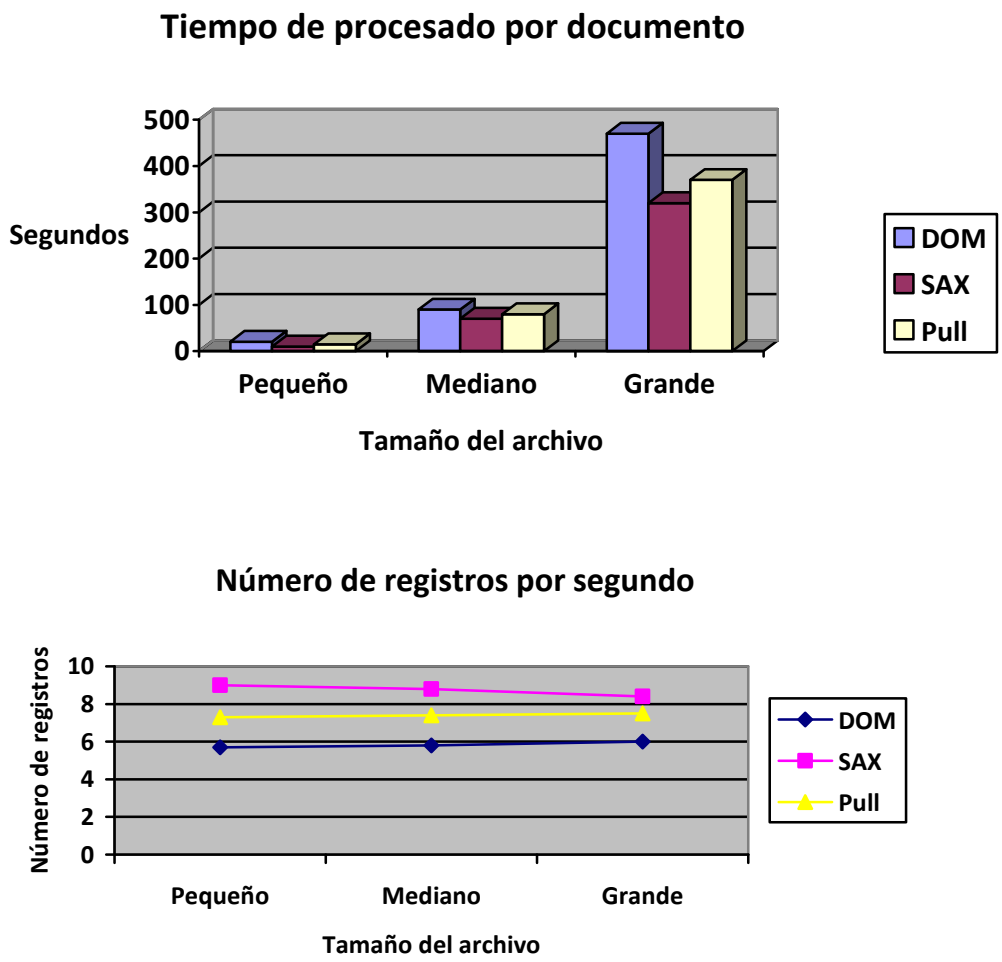


Ilustración 6: Comparativa de rendimiento de distintos Parsers XML

Un estudio hecho por Shane Conder(Conder, 2009), en el que compara los distintos parsers, obtiene los resultados mostrados en la Ilustración 6.

Según Jeffrey Sharkey (Sharkey, 2009) dijo en una conferencia de Google I/O en 2009, el sistema más económico (tanto de batería como de computación) para parsear un texto es el formato XML y utilizar un pull-parser. A pesar de que el formato JSON y Protbuf son más rápidos de parsear con un pull parser, éste método no está implementado en Android, solamente en modo Tree Parser en el que pierden la rapidez.

ANÁLISIS

En esta sección de análisis se empezará describiendo la metodología seguida para el desarrollo de la aplicación. En el segundo apartado se analizarán los sustitutivos, dando a conocer los puntos débiles y fuertes de la aplicación respecto a éstos. A continuación se detallarán los requisitos de la aplicación y en el último se propondrán los requisitos futuros que podría tener la aplicación, pero que no se han implementado en este proyecto.

1. Metodología

En esta sección se definirá la metodología a utilizar para el desarrollo de este proyecto. La elección de metodología es determinante durante el desarrollo puesto que, dependiendo de ella, habrá más cargas de trabajo durante ciertas fases y menos en otras. Hay también condicionantes a la hora de elegirla, por ejemplo la experiencia del desarrollador o la disponibilidad para realizar reuniones con el cliente o el supervisor del proyecto. A continuación se hace un análisis detallado de todos estos factores para llegar a la mejor elección.

Para definir la metodología usada para desarrollar este proyecto necesitamos una perspectiva de alto nivel. Nuestro objetivo es tanto desarrollar una aplicación informática en un tiempo muy limitado como desarrollarlo de la manera más práctica posible. Además, otro de los principales objetivos es didáctico, ya que el proyecto está orientado al estudio y aprendizaje del estudiante, no al desarrollo. Puesto que la experiencia del estudiante en proyectos de desarrollo de software era escasa, no más que las prácticas realizadas en la universidad, su conocimiento en la materia era bastante escaso. Esto obligaba a elegir una metodología con escaso trabajo previo, en favor de una fase de implementación temprana, puesto que es la mejor manera de obtener una primera toma de contacto.

Otro de los factores determinantes a la hora de elegir la metodología es la disponibilidad para concertar reuniones con el supervisor o el cliente. El papel del supervisor o cliente es el de validar los requisitos que se van implementando. En el caso de que no se pudieran realizar reuniones de manera rutinaria o fuera difícil concertarlas, la metodología a usar debería enfatizar la fase de análisis, para encontrar y definir de manera muy detallada y precisa los requisitos, puesto que no se podrían validar con facilidad. En el caso de este proyecto, la situación era la contraria, debido a que el estudiante y desarrollador seguía acudiendo a clases en la universidad, concertar citas era relativamente fácil. Además, gracias a las comunicaciones vía correo electrónico, muchas de las cuestiones podrían ser resueltas sin tener siquiera la necesidad de una reunión.

Debido a todo lo expuesto anteriormente, la metodología elegida debe dar poco peso a la planificación en favor de la obtención de prototipos en fases prematuras de desarrollo.

Estos prototipos eran necesarios ya que la especificación no estaba del todo clara, y se iba modificando a la vez que se iba avanzando en el proceso, debido a la inexperiencia del desarrollador. Por lo tanto se necesitaba una metodología que fuese de constante revisión y evolución, de manera que se descubriesen nuevos requisitos.

La metodología Rapid Application Development (James Martin, 1980) es la que mejor se ajusta a nuestras necesidades. Ésta metodología atrae muchas críticas dado el escaso nivel de planificación además de la descentralización de la responsabilidad, lo que puede derivar en multitud de prototipos inservibles que no llegan a la fase de producción (Butler, 1994). Sin embargo, al ser éste un proyecto de una sola persona, este problema de organización y responsabilidad no tiene lugar.

Dado que uno de los objetivos del proyecto es el aprendizaje del lenguaje de programación, esta metodología es idónea ya que enfatiza la fase de programación, minimizando el tiempo de planificación, permitiendo al desarrollador centrarse en la programación.

A continuación mostramos un diagrama que muestra el ciclo de desarrollo de la metodología RAD (Ilustración 7).

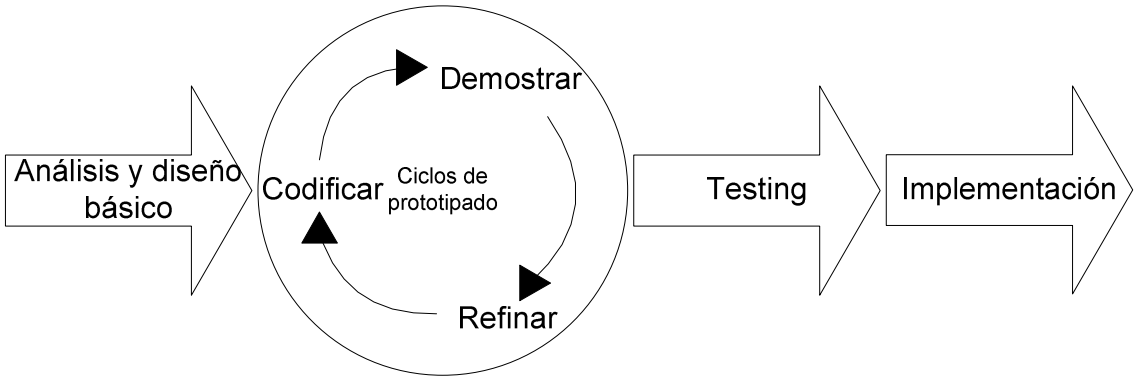


Ilustración 7: Metodología RAD

Aunque el RAD ha sido el enfoque principal que se le ha dado al proceso de desarrollo del software, también se ha utilizado otro paradigma auxiliar como método de resolución de problemas. Este paradigma ha sido el modelo iterativo basado en prototipos. Esta metodología es llamada también Ciclo de Vida en espiral.

El Ciclo de Vida en espiral o modelo iterativo basado en prototipos va a constar de las siguientes etapas:

- Identificación de requisitos: en esta fase se estudiarán los requisitos que debe cumplir el proyecto, basándose en las necesidades que se desean cubrir. Para ello también hay que tener en cuenta las limitaciones de la plataforma, puesto que condicionarán en gran medida las posibilidades del software.
- Programación: implementación de los prototipos necesarios para cumplir los requisitos. Muchos de estos prototipos serán desechables, puesto que se desarrollarán con la idea de comprobar el funcionamiento del sistema.
- Pruebas: de manera que se garantice que el software cumple los requisitos marcados. Los prototipos que finalmente se vayan a implementarse en el software final deben realizar correctamente las funciones para las que fueron desarrollados. Estas pruebas serán de validación y verificación es decir que lo que haga es lo que se quería y que lo haga bien.

Puesto que la metodología usada se basa en una continua búsqueda de requisitos, la revisión por parte de los usuarios debe ser constante, puesto que los nuevos requisitos introducidos deben ser validados, de manera que se compruebe que se implementa lo querido.

Este modelo permite, de manera sencilla y rápida, aclarar los requisitos, explorar puntos críticos e identificar características del programa que deben cambiarse. Usando este modelo se generan dos tipos de prototipos, los desechables y los evolutivos. Se muestra en la Figura 8 el gráfico que ilustra el modelo utilizado para desarrollar prototipos.

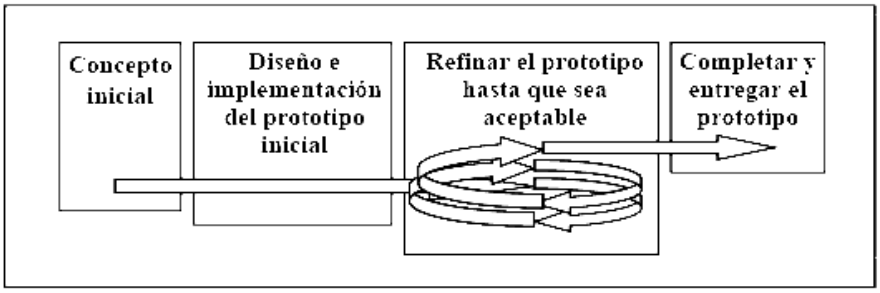


Ilustración 8: Modelo de prototipado evolutivo

El prototipo desechable es uno escrito de manera muy rudimentaria y se utiliza para programar las partes más complejas del código. Después se desecha.

El prototipo evolutivo suele incluir lo que se ha aprendido haciendo prototipos desechables. Es decir, un prototipo evolutivo es riguroso y se desarrolla usando bases sólidas (obtenidas mediante prototipos desechables) y el objetivo último es incluirlo en el sistema.

En el caso de este proyecto, numerosos prototipos desechables han sido programados. Por ejemplo, para desarrollar el módulo que muestra los puntos de interés cercanos al usuario. Se han creado clases y programas autónomos que implementaban esta función, con la idea de familiarizarse con el funcionamiento de la plataforma Android. Otro ejemplo de prototipo desechable ha sido usado también para los módulos de entrada y salida de información del disco duro. La proporción usada en el programa final de estos códigos ha sido poca o ninguna, conservando solamente los conceptos aprendidos desarrollándolos.

El programa desarrollado se puede considerar el prototipo evolutivo, ya que ha sido desarrollado a partir de los prototipos desechables.

Durante el desarrollo del proyecto se ha mantenido de manera constante e interrumpida un canal de comunicación vía correo electrónico, por el que se trataban todos los temas concernientes a este proyecto. Además, se han mantenido reuniones más o menos regulares para validar nuevas funcionalidades o requisitos con el tutor del proyecto. De esta manera, no se ha perdido mucho tiempo en desarrollos que luego no iban a ser validados. Para obtener una visión global de cómo ha sido el proceso de desarrollo se puede consultar la sección de planificación, donde queda todo detallado.

2. Análisis de sustitutivos

En esta sección empezaremos con una pequeña introducción, enumeraremos los sustitutivos que tiene nuestra aplicación, así como las ventajas de éstas frente a la aplicación desarrollada y viceversa. Al final se mostrará una tabla comparativa entre todos los sustitutivos.

Hay que tener en cuenta que esta aplicación no tiene una funcionalidad del todo

definida, es decir; dependiendo de la base de datos que se cargue se utilizará para una cosa o para otra. Se analizará en su faceta original, con las bases de datos iniciales, respecto a sus sustitutivos pero no se hará ese mismo análisis para los usos alternativos que pueda tener la aplicación, puesto que éstos pueden ser numerosos.

El propósito de la aplicación es servir de complemento a una guía turística tradicional o digital, pero en cualquier caso, apoyar con información a una guía más específica. Dado que el este mercado es algo novedoso y no existe ninguna aplicación similar, se va a proceder a compararla con las guías completas.

El sustitutivo más inmediato que tiene la aplicación es la guía de viajes impresa. Una de las pocas ventajas que tiene frente a la aplicación desarrollada es que los usuarios están más acostumbrados al uso de éstas, aunque esto es menos cierto para las nuevas generaciones. Las ventajas sobre la guía impresa son varias y significativas: se puede actualizar sin tener que comprar un nuevo ejemplar, la experiencia del usuario es mejor, da al usuario la posibilidad de elegir qué le interesa (selección de base de datos de puntos de interés) no pesa y no ocupa espacio y probablemente cueste bastante menos. La desventaja es que posiblemente contenga más información, así como planos y dibujos.

Otro sustitutivo es la guía digital, sin realidad aumentada. Éstas no conllevan ninguna mejora respecto a la desarrollada en este proyecto, excepto quizás, el precio. Las ventajas de una guía de realidad aumentada respecto a las digitales convencionales es, principalmente, la experiencia del usuario. Es posible que las guías desarrolladas por editoriales de guías tradicionales impresas contengan más información o más contrastada.

Una ventaja que las guías de realidad aumentada tienen con cualquier otra es que localiza al usuario, es decir, el usuario siempre sabrá dónde está y hacia donde quiere ir. Esto puede no parecer una ventaja muy importante, pero en el caso de que el usuario se encuentre en un país extranjero y no sepa hablar el idioma, esta ventaja puede suponer una diferencia notable para el usuario. Una de las mayores ventajas es que, dado que añadir un punto de interés adicional no supone apenas espacio en memoria (ni en disco duro), se pueden almacenar muchas referencias que quizás para una guía turística sean poco relevantes. También tiene un caso de uso añadido a la guía turística, este es el caso de los paseos: el usuario cuando está visitando una ciudad, normalmente en algún momento decide salir a pasear pero sin pensar en un lugar predefinido y sin guía. Si en ese paseo se encuentra con un edificio, plaza o monumento que le llame la atención si lleva su teléfono móvil consigo podrá obtener información de cualquiera de ellos.

En un ámbito institucional, en el que al Ayuntamiento o la Comunidad Autónoma les conviene promocionar los distintos lugares de la ciudad o comunidad, una aplicación como esta puede ser de inmensa utilidad. Sustituiría al guía turístico con las ventajas que esto conlleva en gasto tanto económico, burocrático y humano.

Producto	Escalabilidad	Actualización	Coste	Experiencia
GuiaInteractiva	Buena	Buena	Bueno	Regular
Guía impresa	Mala	Mala	Regular	Regular
Guía digital	Buena	Buena	Bueno	Mala
Guía realidad aumentada	Buena	Buena	Bueno	Regular/Buena
Guía turistico	Mala	Regular	Malo	Buena

Tabla 5: Comparativa de sustitutivos

En la Tabla 6 se muestra una comparativa entre todos los sustitutivos de la guía desarrollada. La columna producto contiene todos los sustitutivos analizados anteriormente. La columna escalabilidad representa la facilidad y el coste de escalar la solución dependiendo del número de usuarios que tenga, esto es un factor importante puesto que el turismo es muy dependiente de la temporada. En la columna coste se valorará tanto el coste para el usuario, como el de la administración (en caso de que aplique) así como el medioambiental. En la última columna se intenta cuantificar la experiencia que experimenta el usuario al usar la solución.

3. Análisis de requisitos

3.1 Prefacio

Para realizar el análisis de requisitos se va a utilizar como guía el estándar IEEE 830. Este estándar establece que para describir correctamente los requisitos de una aplicación, el documento debe seguir el índice mostrado en la Ilustración 9.

1. Introducción
    1. Propósito
    2. Alcance
    3. Definiciones, acrónimos y abreviaciones
    4. Referencias
    5. Resume
2. Descripción general
    1. Perpectiva del producto
    2. Funciones del producto
    3. Características del usuario
    4. Condicionantes
    5. Suposiciones y dependencias
3. Requisitos específicos

Apéndices

Índice

Ilustración 9: Estructura de un prototipo SRS

Hay secciones del índice que establece el estándar que no tienen cabida en este documento, dado que estarían duplicadas, por lo tanto se suprimirán las secciones de: Definiciones, acrónimos y abreviaciones, referencias, anexos e índice. Por ello el índice quedará de la siguiente manera:

1. Introducción
    a. Propósito
    b. Alcance
    c. Visión global
2. Descripción global
    a. Perspectiva del producto
    b. Funciones del producto
    c. Características del usuario
    d. Condicionantes
    e. Suposiciones y dependencias
    f. Distribución de requisitos
3. Requisitos específicos

Otra de las recomendaciones que hace el estándar es que el análisis sea trazable, es decir, que se pueda hacer un seguimiento en el tiempo de las modificaciones que sufre el documento y por tanto los requisitos que se han ido definiendo. Dada la metodología usada para el desarrollo de este proyecto, en concreto para la aplicación Android, la búsqueda de requisitos se realizaba en paralelo al desarrollo. Esta circunstancia, obligaría a presentar un documento de especificaciones por cada prototipo, tal como dicta el estándar. En aras de la brevedad sólo se seguirá el estándar (y su índice) para la versión final de las especificaciones, y, a modo de muestra a continuación se mencionarán brevemente la evolución de éstas. Para que la imagen global sea más clara, se recomienda leer los siguientes párrafos junto con la sección de Planificación de esta memoria.

La especificación inicial fue definida más con el propósito de familiarizar al desarrollador con el entorno de trabajo y el sistema operativo Android, desarrollando un prototipo, que realmente desarrollar parte de la aplicación. Los primeros requisitos fueron crear una aplicación que pudiese tomar una foto con la cámara del móvil y una vez tomada ésta, sobrescribir algo sobre ella.

Una vez cumplidos estos requisitos, un esbozo de lo que sería la aplicación final, se le añadió a la aplicación los de detectar la posición del usuario (mediante el sistema GPS, descartando completamente por triangulación de antenas de telefonía dada su inexactitud), su orientación y sobrescribir los puntos de interés en la interfaz gráfica(en función de estos últimos 2 datos).

Una vez implementado todo el algoritmo de context-aware, se propuso que por cada punto de interés que se mostrara en la pantalla se pudiera hacer click sobre ellos. Una vez hecho el click que apareciese un menú desplegable que mostrase las opciones disponibles para ese punto en concreto. Esto no llegó a la especificación final, puesto que Android no permite colocar botones en posiciones arbitrarias.

Una de las necesidades que apareció casi al término del proyecto fue crear una web informativa. Esta web contendría esta misma memoria, así como un manual de usuario para descargar. El código fuente también estaría disponible, que permitiría al usuario poder modificar el código junto con la posibilidad de poder descargarse la

aplicación para instalarla en un teléfono móvil. En esta web habrá un correo electrónico de contacto para poder enviar las bases de datos generadas por los usuarios, que también estarán disponibles para descargar.

3.2 Introducción

Propósito

Ateniéndose a la metodología que se ha seguido para el desarrollo de la aplicación, la definición de requisitos ha sido un proceso paralelo al desarrollo. La aparición de nuevos requisitos y la modificación de los existentes han sido constantes. En esta sección se detallará la última especificación de requisitos que se hizo.

Alcance

La aplicación que desarrollada para este proyecto se llama GuiaInteractiva. GuiaInteractiva es una aplicación de guía turística con realidad aumentada para plataformas Android X.X. Los usuarios de la aplicación serán turistas que tengan interés por encontrar información actualizada y con un mínimo esfuerzo del entorno en el que se encuentran. La aplicación no pretende sustituir la guía turística tradicional impresa, si no más bien complementarla, con información más reciente y de más sitios.

Visión global

En las siguientes subsecciones del Análisis de Requisitos se procederá a hacer una descripción global del producto, así como las funcionalidades que debe implementar, el prototipo de usuario que tendrá, además de los condicionantes que afectan al desarrollo de este producto. A continuación, se detallarán los requisitos que no se contemplarán para este proyecto y por último se detallaran los requisitos específicos.

3.3 Descripción global

Perspectiva del producto

El producto en cuestión, la aplicación GuiaInteractiva, es un producto autocontenido y no tiene dependencia de ningún otro, aparte de las funcionalidades que ofrece Android. La aplicación depende de las bases de datos que se tengan tanto en la carpeta privada del programa como en la carpeta descargas del terminal. La carpeta privada contendrá las bases de datos estándar, mientras que en la carpeta de descargas el usuario podrá introducir cualquier tipo de base de datos.

La aplicación será desarrollada para Android 2.1. Puesto que las nuevas versiones de Android que aparezcan en el mercado son retrocompatibles, se espera que la aplicación no tenga ningún problema para instalarse en terminales con esas versiones.

La interfaz gráfica debe ser sencilla y toda interacción debe ser por pantalla táctil (sin teclado alfanumérico).

Las ventanas deberán seguir una progresión lineal, es decir, si el usuario realiza una acción y quiere volver atrás, que sea posible.

La conexión que usará la aplicación para Internet será hará mediante redes de datos inalámbricas (WiFi) y redes de telefonía (2,5G y 3G). La localización se obtendrá mediante el sistema GPS y no se utilizará la triangulación de antenas de telefonía por su

poca precisión.

La aplicación debe aceptar nuevas bases de datos, que no estén incluidas por defecto en el paquete, introducidas por el usuario.

Funciones del producto

En las siguientes figuras (Ilustraciones 10 y 11), se muestran los casos de uso de la aplicación.

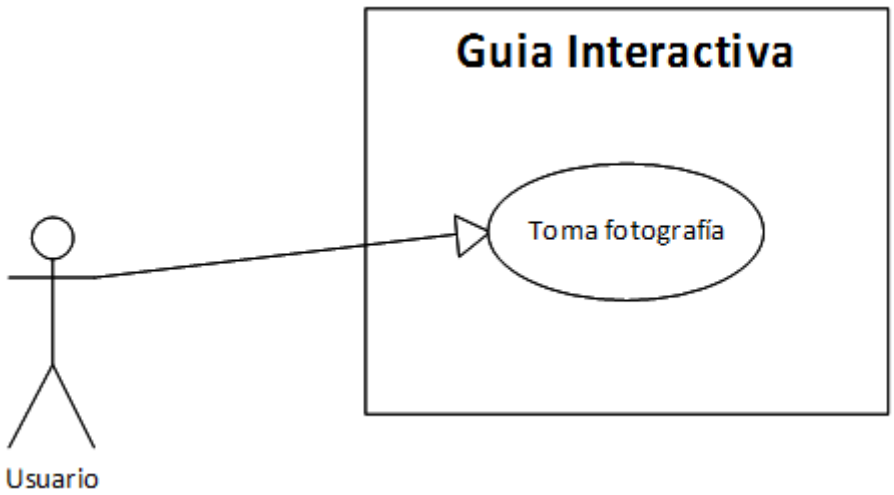


Ilustración 10: Caso de uso 1

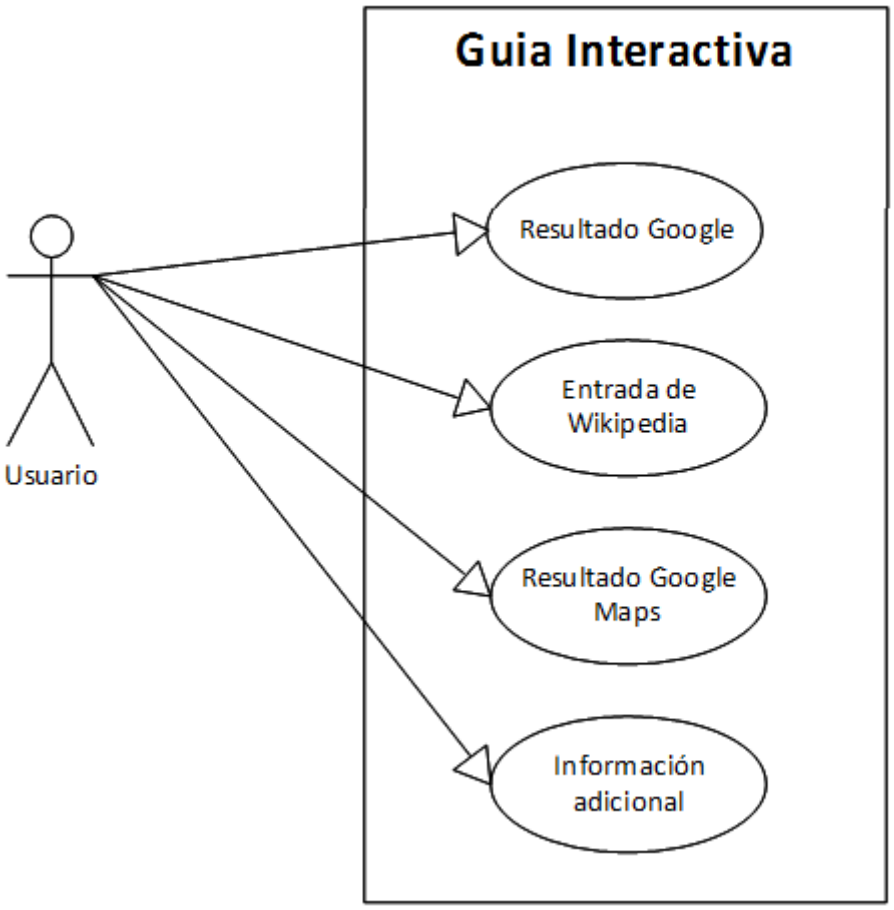


Ilustración 11: Caso de uso 2

El primer caso de uso muestra uno de los procesos más importantes de la aplicación. El usuario ve un punto del cual le interesa obtener información, arranca la aplicación (haciendo los ajustes necesarios) y toma una fotografía. Una vez hecho esto, el programa le mostrará los puntos de interés cercanos a su posición.

El segundo caso de uso es cuando la aplicación le muestra al usuario las diferentes opciones que tiene para obtener información sobre el punto. Estas opciones son, por defecto tres: buscar en Google, Wikipedia o en Google Maps. Hay una cuarta opción que no es obligatoria y que sólo aparecerá en caso de que la base de datos contenga los campos URL y/o INFO para ese punto de interés concreto.

Características del usuario

En cuanto al perfil del usuario habrá que distinguir entre el caso de que se utilice la aplicación solamente con las bases de datos por defecto y el caso en el que se introduzca alguna más, de naturaleza similar o distinta.

En el primer caso, el usuario de la aplicación tendrá una edad comprendida entre los 18 y los 65 años. Será orientado a usuarios con cualquier nivel de educación, pero sobretodo que tengan curiosidad e interés por aprender más sobre una ciudad. Se presupondrá cierto conocimiento e interés por las nuevas tecnologías.

En el segundo caso, gracias a la posibilidad de introducir nuevas bases de datos, el prototipo de usuario es todavía más ambiguo, por lo que no se concreta en ningún tipo de persona. Se espera que los usuarios con mayor conocimiento de estas tecnologías aporten su propia base de datos y la utilicen para cualquier uso.

Condicionantes

El mayor condicionante en este desarrollo es la plataforma objetivo, un teléfono móvil. Las limitaciones principales causadas por este motivo son la duración de la batería, conectividad a Internet limitada, escasa capacidad de cómputo, poca memoria RAM y el tamaño de la pantalla. La libertad de movimiento que ofrecen estos terminales también se convierte en un condicionante, aunque en este caso, es el que le da sentido a la aplicación.

Suposiciones y dependencias

En un futuro, con el avance de las capacidades de los terminales móviles las especificaciones se cambiaran acordemente. Además es de suponer que las nuevas actualizaciones del sistema operativo Android, cada vez más frecuentes, permitirán al programador hacer aplicaciones más avanzadas.

Además, dado que existen numerosos dispositivos que utilizan el sistema operativo Android, no se puede comprobar el correcto funcionamiento de la aplicación en todos ellos, a menos que se realicen las pruebas en todos ellos. Puesto que esta tarea sería muy costosa, tanto por tiempo como por recursos económicos, se probará en terminales de los dos mayores fabricantes de terminales con estas características. Éstos son HTC y Samsung.

La aplicación interactuará con varios módulos disponibles gracias a las librerías Android. Estos módulos son, Google Maps y el navegador. Se llamarán directamente desde el código de manera que no hay que implementarlo de nuevo. Además utilizará servicios disponibles tales como el GPS y la conexión a internet (ya sea por red

telefónica como por WiFi).

Distribución de requisitos

Dado que este proyecto se ha llevado a cabo en un tiempo limitado, se ha previsto requisitos futuros que no serán considerados para esta versión del programa. A continuación se muestran los más importantes. La enumeración y explicación completa de los posibles futuros pasos se encuentra en la sección de Conclusiones y Mantenimiento.

- Utilizar el API Geocoding de Google para obtener los puntos de interés cercanos, en vez de usar una base de datos creada anteriormente. Esto tendría una repercusión a tener en cuenta, la aplicación necesitaría de conexión a Internet en todo momento, en contraposición a como se realiza ahora.

- Traducción de todas las bases de datos y de los textos que se muestran al usuario, dando la opción de seleccionar el idioma al arrancar la aplicación.

- Permitir que un usuario agregue sus propios puntos de interés dentro de la aplicación a una base de datos ya creada.

- Crear un servicio de bases de datos en la nube. Al arrancar la aplicación, se descarga las bases de datos actualizadas de un servidor conectándose a Internet. Si un usuario crea un nuevo punto de interés, al cerrar la aplicación éste se añadirá a la base de datos del servidor. De esta manera toda la información está actualizada casi a tiempo real.

3.4 Requisitos específicos

Requisitos para interfaces externas

Interfaces de usuario

Las ventanas que se muestren al usuario deben ser autoexplicativas, indicando al usuario en todo momento lo que debe hacer para proceder a usar la aplicación. En ningún caso se mostrarán más de 2 botones, exceptuando los casos en los que se trate de una lista.

En caso de que haya un uso indebido o un error se le mostrará por pantalla al usuario, informándole de la situación.

La sobreimpresión de puntos de interés en la foto resultante debe hacerse de manera que esté alineado con el punto de interés real. Es decir, si el punto de interés esta a la derecha en la foto, el texto se sobreimprimirá a la derecha.

Los textos en la interfaz gráfica tratarán al usuario de tú, de esta manera se hace la aplicación más cercana. Puesto que es trata de motivar a los usuarios a construir sus bases de datos propias, se intentará hacer la aplicación lo más agradable y amigable posible.

La interfaz tanto gráfica de usuario deberá ser escueta, evitando, en la medida de lo posible las ventanas deslizantes. Sólo podrán ser deslizantes aquellas que contengan listas de longitud arbitraria y que, por lo tanto, no se puede ajustar el tamaño de fuente para asegurar que como máximo ocupen el ancho y largo de la pantalla.

Si dos puntos de interés se encuentren cerca uno del otro, puede darse el caso de que la representación gráfica de éstos se solape. Por ello se mostrarán los puntos de interés en distintas posiciones verticales (puesto que la horizontal está fijada por la localización). La posición vertical vendrá dada por el identificador del punto de interés, un entero de 1 a 10. La función que se utilizará será la siguiente:

$$alt_{PI} = \frac{5}{6} alt_{tel} - \frac{5id}{100}$$

Ecuación 1: Algoritmo que determina la posición vertical del punto de interés

Donde altpi es la altura a la que se mostrará el punto de interés, y alttel es la altura de la pantalla del teléfono. Id representa el identificador del punto de interés. De esta manera la posición vertical de la representación gráfica del punto de interés queda acotada entre aproximadamente el 80% y 33% de la altura de la pantalla. Siendo el 80% la posición más baja y 33% la más alta.

Se deberá dar al usuario la posibilidad de que él mismo seleccione el radio en el que aparecerán los puntos de interés. De esta manera, si el usuario ve, por poner un ejemplo, un monumento a un kilómetro de distancia, pero no se puede acercar a él, podrá ajustar la aplicación para que muestre puntos de interés que estén a más de un kilómetro de distancia. El rango en el que podrá ajustarse estará entre los 25 metros y 5 kilómetros, con una escala logarítmica. Esta escala permite tener mucha más resolución en distancias cortas (pasos de 2 metros) y menos en distancias mayores (pasos de alrededor de 100 metros). Se ha decidido hacer esto porque el humano es capaz de ser más preciso calculando distancias cortas que cuando calcula distancias muy grandes.

Al usuario se le mostrará un texto de advertencia en caso de que el radio de interés sea mayor a 3 kilómetros, puesto que puede haber muchos puntos de interés disponibles y es probable que no puedan aparecer todos.

Durante el tiempo de carga de las bases de datos a la memoria RAM, dado que será muy variable dependiendo del número de bases de datos y su tamaño, se le mostrará al usuario una barra de progreso. No debe ser una barra que muestre el progreso, puesto que para eso habría que hacer cálculos y leer las bases de datos de antemano, si no simplemente mostrar una imagen de carga, que es más sencillo y eficaz.

El ángulo válido para mostrar puntos de interés será de 60º. Esto es: 30 grados desde el punto al que se apunta con la cámara hacia la derecha y hacia la izquierda. Esto responde a que el ángulo de visión de la cámara es limitado y, tratándose de un gran angular en la mayoría de los terminales, solo se puede hacer una aproximación. 60º es algo menos que el ángulo de visión de un objetivo de 20 mm en una cámara de 35mm. Las ecuaciones del algoritmo usado se muestran en la Ecuación 2 de la sección de Implementación, bajo el apartado “Proceso de desarrollo”.

La interacción posible o exigida al usuario no debe exceder la cantidad de una acción por ventana. Es decir para pasar de una ventana a otra el usuario sólo deberá realizar una acción, ya sea pulsar un botón, clickar en algún punto de la pantalla o mantener pulsado un botón. Esto se cumplirá en todas las ventanas menos aquellas en

las que haya algún tipo de selección por parte del usuario donde, la interacción que habrá será la requerida para hacer la selección y para realizar la transición a la siguiente ventana.

La orientación del teléfono será siempre horizontal en las ventanas donde intervenga la previsualización de la cámara de fotos y la foto tomada tomada por ésta, y no habrá posibilidad de cambiarla. Esto se hace por motivos prácticos, puesto que los teléfonos Android la cámara sólo se puede mostrar de manera horizontal y mostrar la foto de manera vertical la deformaría. De manera se asegura el máximo de público posible para la aplicación. Además así la cámara abarcará más puntos de interés puesto que mostrará un ángulo de visión mayor.

A la hora de que el usuario quiera obtener más información sobre un punto de interés, se le mostrará una lista y deberá mantener pulsado el punto de interés del que deseé más información. Se ha decidido que mantenerlo pulsado es más práctico, puesto que en el caso de que haya muchos puntos de interés, el usuario, al navegar por ellos, es probable que haga click en alguno sin querer, lo cual puede resultar bastante molesto. Este problema se soluciona si la selección se hace manteniendo pulsado el botón.

Interfaces HW

El programa hará uso de la cámara de fotos del teléfono móvil. Puesto que la mayoría de teléfonos Android no disponen de botón para tomar fotografías, se utilizará el botón de menú para tomar la foto y pasar a mostrar los puntos de interés cercanos.

La aplicación hará uso del de las señales GPS para geolocalizar al usuario, por lo que el terminal debe poseer un adaptador GPS, además de estar el servicio disponible.

El terminal deberá tener conexión a Internet, ya sea mediante redes de telefonía como por adaptador WiFi. La aplicación seguirá funcionando a pesar de no tener conexión, pero no podrá acceder a ninguna información, aparte del nombre del punto de interés.

La aplicación hará uso de la brújula, por lo que el teléfono deberá poseer además de tener disponible el servicio de sensores.

Interfaces SW

La aplicación no debe usar ningún paquete adicional a los que vienen incluidas en la librería estándar de Android. Para el proceso de desarrollo se podrá usar cualquier librería, pero no se podrá usar para la versión final del producto.

Los puntos de interés se representarán mediante la clase PuntoInteres, así como las bases de datos lo serán por BaseDatos. Los objetos PuntoInteres serán un mero volcado de la información que contengan las bases de datos sobre un punto de interés, así como atributos y métodos adicionales que hagan la programación más fácil. En el caso de los objetos BaseDatos tendrán atributos que determinarán si son estándar o descargadas, entre otros.

Todo tipo de uso malintencionado o accidental será previsto y tratado debidamente, de manera que se proteja al usuario, su terminal y la aplicación en sí.

En ningún caso se permitirá al usuario modificar ninguna base de datos, así como el código. Como medida de seguridad, el usuario no podrá acceder directamente a la base de datos y editarla desde la aplicación, de manera que se evitará cualquier tipo de uso malintencionado. El código no deberá estar accesible para el usuario, indirecta o

directamente desde la aplicación, no permitiendo así su modificación.

La aplicación podrá consumir, como mucho 16MB de memoria RAM, es un requisito impuesto por el sistema operativo Android. Puesto que uno de los grandes consumidores de RAM de la aplicación serán las fotografías que tomará, éstas se comprimirán en formato JPEG a un 20% de la calidad original. Puesto que la codificación JPEG obtienen muy buenos resultados con tasas muy altas de compresión, la pérdida es apenas perceptible mientras que el uso de RAM (al cargar la imagen como fondo de pantalla) es notoriamente menor.

Interfaces de comunicación

En cuanto a las interfaces de comunicación, lo esencial es gestionarlas bien en aras de conservar la batería del terminal. Además un riesgo importante que se corre con estas interfaces es el de la seguridad, sobretodo con las redes WiFi.

Unas de las funcionalidades que más batería consumen en un teléfono son el adaptador GPS y WiFi. Lo óptimo para reducir el consumo de batería sería tenerlos deshabilitados durante todo la ejecución, exceptuando los momentos de adquisición y transmisión(en el caso de WiFi) de datos. En el caso del GPS, este comportamiento tendría un efecto secundario: cuanto más tiempo el GPS mejor será el fix que se obtenga, por lo que si sólo se activase el adaptador durante un instante el fix obtenido no sería muy preciso. El caso del adaptador WiFi es un poco más complejo, porque depende de las acciones que tome el usuario: usando la aplicación podría ser que el usuario sólo consultara Internet por un momento o que, por otro lado, estuviese constantemente entrando en Internet. Por lo tanto la variabilidad es muy grande y encender y apagar el adaptador continuamente tampoco es muy eficiente. Lo que se hará será pedirle al usuario que active ambos intefaces(si no los tiene activados ya) en el último momento, es decir, en la fase de adquisición del fix y en el caso de que se necesite conexión a Internet. Después de este momento el GPS intentará conseguir un fix más preciso y el WiFi estará inactivo (si vuelve para atrás) o seguirá activo si el usuario sigue navegando.

Se deberá comprobar que el fix obtenido por el GPS no está obsoleto. Puesto que el terminal almacenará la última posición obtenida, habrá que comprobar que ésta está medianamente actualizada. Para ello se determinará que pasado un minuto, el fix estará obsoleto, puesto que se supone que el usuario se moverá bastante y fijar un tiempo mayor puede resultar en un funcionamiento errático debido a una actualización de la posición insuficientemente frecuente.

Funciones

La funcionalidad básica de la aplicación será la de proveer al usuario de información sobre los distintos puntos que estén cercanos a él. Como interfaz para mostrar la información se utilizará una foto, que el usuario habrá tomado previamente. Para saber lo que está mirando la aplicación debe ser capaz de geolocalizar al usuario y obtener su orientación, puesto que sabiendo ambas, podrá determinar qué es lo que está viendo. Esto se hace una vez pulsado el obturador. Con estos datos, se lee una base de datos y se determinan qué puntos de interés de ellos está viendo el usuario, y se sobreimprimen sus nombres en la foto. De estos puntos podrá obtener información adicional, ya sea obteniéndola por internet o de la base de datos.

La aplicación deberá seguir la siguiente secuencia de ejecución: preguntar al usuario qué bases de datos quiere y el radio de interés, cargar las bases de datos correspondientes, mostrar una pantalla informativa de cómo funciona la aplicación,

mostrar la visualización de la cámara hasta que el usuario pulse el obturador, una vez realizada la foto se sobreimprimirá puntos de interés(en caso de que haya), si el usuario pulsa la pantalla se le mostrarán los puntos de interés los cuales podrá pulsar prolongadamente para que se muestren las opciones de cada uno de los puntos de interés. En cualquier momento el usuario podrá pulsar la tecla “back” y volver a la ventana anterior, o la tecla “home” y volver al escritorio, manteniendo la aplicación en segundo plano. El diagrama de flujos se muestra en la Ilustración 12.

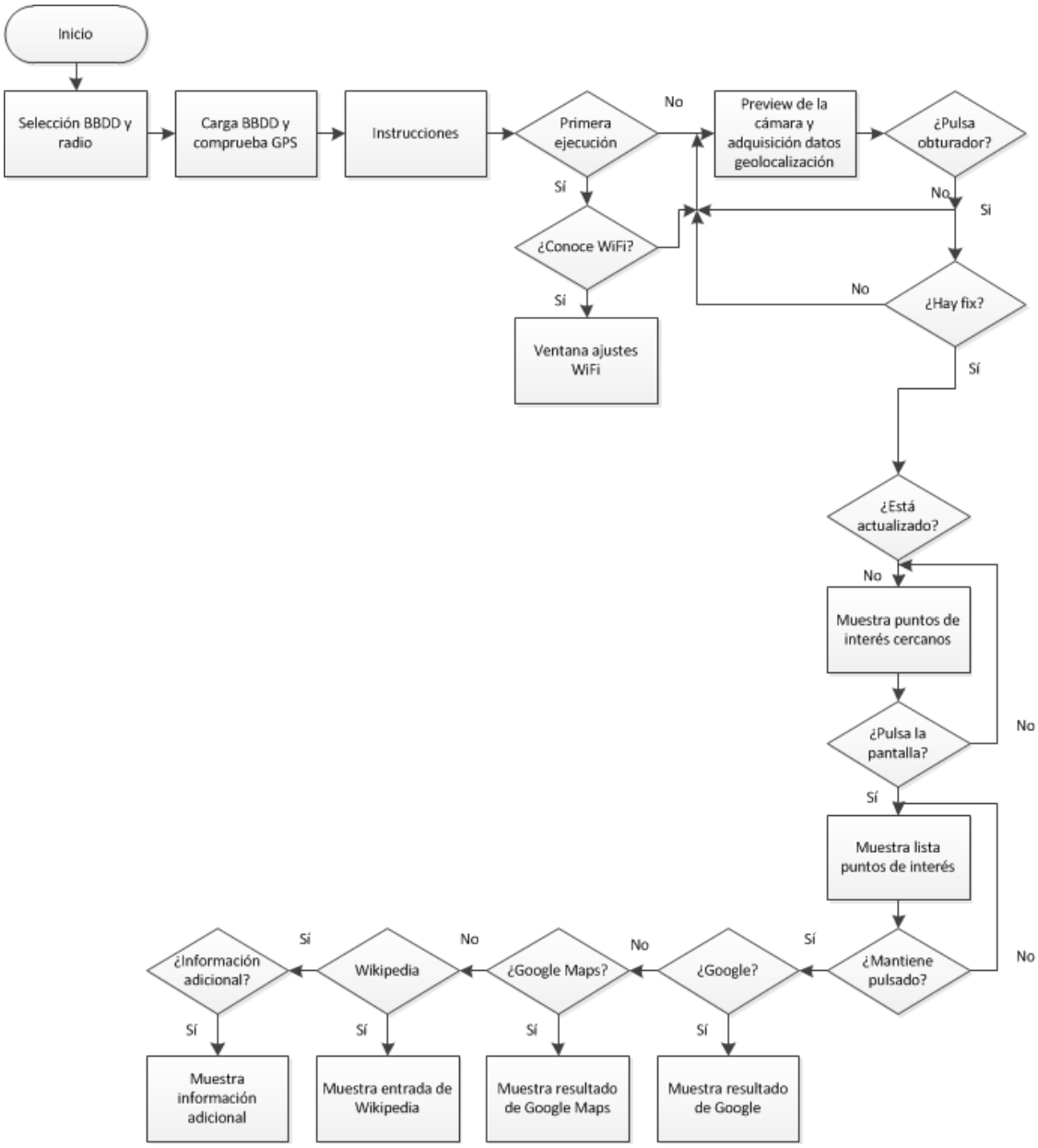


Ilustración 12: Diagrama de flujo de la aplicación GuiaInteractiva

En la figura 12, se expone el diagrama de flujos que deberá seguir la aplicación. La caja de decisión llamada “¿Pulsa la pantalla?” se refiere a si el usuario ha pulsado en cualquier lugar la pantalla. Como puede apreciarse, el diagrama no tiene un óvalo representando el término de la aplicación, esto es por que o bien se puede salir pulsando la tecla “Home” del terminal o bien se puede salir pulsando la tecla de retorno tantas veces como sea necesario hasta llegar a inicio. A inicio se llega desde el escritorio de Android, y si se termina la ejecución (mediante cualquiera de los dos métodos explicado anteriormente) también se volverá al escritorio. Hay una caja de decisión llamada

“¿Usuario sigue navegando?”, que representa si el usuario sigue navegando por Internet. La respuesta “No” representa que el usuario ha pulsado la tecla de retorno (Back) tantas veces como fuera necesario hasta volver a la selección de acciones sobre los puntos de interés. Esto se ha hecho en aras de la sencillez del diagrama, ya que el usuario ha podido navegar por un número arbitrario de páginas, por lo que no tiene sentido representar todos estos saltos (y posteriores retrocesos).

Por motivos de simplicidad, no se ha representado la posibilidad, que siempre deberá estar presente, de pulsar el botón “Back”. En los casos habituales, esta acción tendrá como consecuencia volver a la ventana anterior. En el caso de que el usuario pulse el botón “Back” cuando esté en la primera ventana (de selección de bases de datos y de radio de interés), el programa se pausará y mostrará el menú de aplicaciones o el escritorio (la ventana desde la que se arrancó la aplicación). Para conocer las teclas de un teléfono Android, se adjunta en el Anexo A, un ejemplo de uno de ellos.

La aplicación deberá comprobar que recibe correctamente la señal de GPS. Además deberá comprobar que los valores son correctos y están entre los posibles.

La aplicación deberá comprobar que tiene conectividad con internet. En caso de que no la tenga, se mostrará un mensaje que avise al usuario, además de desactivar todas las funciones que necesiten de Internet.

La aplicación deberá comprobar que el usuario ha seleccionado alguna base de datos. En caso contrario, se notificará al usuario y no se procederá con la ejecución del programa. Además deberá comprobar que se ha cargado correctamente la base de datos. En caso de que haya habido un error al cargar la base de datos, se tomará como no seleccionada. Por lo tanto, si fuera la única base de datos que fue seleccionada, no se continuaría con la ejecución del programa.

Requisitos de rendimiento

La aplicación sólo se instanciará una vez por terminal, limitando el número de usuarios posibles por teléfono a uno. La cantidad de información que manejará debe ser mínima, dadas las limitadas capacidades de los teléfonos. Debe ser capaz de ejecutarse en un terminal con las siguientes características:

- Android API 1
- 128 MB de RAM
- 32 MB de memoria flash
- Procesador de 200MHz. ARM9 o mejor

Puesto que esto son los requisitos mínimos en cuanto a rendimiento del terminal para que corra Android correctamente, puesto que se van a utilizar funcionalidades adicionales como el GPS o internet, se presupone que los requisitos serán mayores. Por ello se ha probado la aplicación en varios terminales y el de menos capacidad era un Samsung Galaxy 3 con Android 2.1. Por lo tanto, se debe garantizar el funcionamiento en un terminal con sus mismas características, éstas son:

- Android 2.1
- 256 MB de RAM
- 170 MB de memoria flash interna

- Procesador de 667MHz. ARMv6

Las bases de datos se cargarán dinámicamente en la memoria RAM, para tener un acceso mucho más rápido a ellas durante la ejecución. Las bases de datos son ligeras, además sólo se cargarán aquellas que el usuario elija.

Requisitos para bases de datos

Se usarán bases de datos para almacenar los puntos de interés. Éstas deben ser ficheros XML independientes, cada una conteniendo datos de cada canal de información (tiendas, turismo…). Las bases de datos deberán ser accedidas sólo al arrancar la aplicación, sin interactuar con ellas después de este momento. Las bases de datos por defecto deberán estar almacenadas en directorios privados de la aplicación, en el caso de que sean adicionales (descargadas de la página web), se almacenarán en el directorio de descargas del terminal. En el caso de que la aplicación se desinstale, se eliminarán las bases de datos por defecto, mientras que las descargadas se mantendrán en el sistema de ficheros del teléfono.

Las bases de datos contendrán etiquetas XML, texto y vínculos a imágenes de Internet.

Las bases de datos tendrán un campo opcional que contendrá dos subcampos: información adicional y una url. Estos dos subcampos serán, a su vez, opcionales, por lo que podrá estar uno y el otro no. La URL apuntará a una foto que esté en Internet.

Las bases de datos XML utilizarán los campos imprescindibles para contener la información necesaria, de manera que se minimice el tamaño de éstas.

En el caso de Android, la carpeta privada de la aplicación que se usará para guardar las bases de datos “estándar” será:

data/data/GuiaInteractiva/xml

Mientras que en el caso de almacenamiento externo se utilizará la siguiente carpeta:

/sdcard/download

Condicionantes de diseño

Puesto que la mayoría de los terminales Android carecen de teclado hardware, se diseñará la aplicación de manera que no sea necesario utilizar un teclado en ningún momento.

Las interfaces de los terminales Android con el usuario son las varias, entre las que se encuentran: la pantalla táctil y los botones. El uso de la pantalla táctil es mucho más intuitivo que los botones, por lo que todas las interacciones se realizarán mediante esta interfaz, excepto la de tomar la foto.

Cumplimiento de estándares

No se pretende cumplir ningún estándar con este producto. A pesar de ello, se han seguido como guía para el desarrollo algunos estándares. Para garantizar la calidad del software, se ha tenido como guía el estándar ISO 9126. Aún cuando el estándar no ha sido seguido al pie de la letra, puesto que no se quiere certificar la aplicación, usándolo como guía se cerciora de que el camino hacia la calidad del software es el correcto y está bien orientado.

Atributos del software

A continuación se expondrán alguno de los atributos que debe de tener el software y que pueden ser considerados como requisitos.

Fiabilidad y disponibilidad

El programa deberá poder ejecutar todas sus funcionalidades siempre que se den las siguientes condiciones:

- El teléfono tiene conexión a la red de datos (WiFi o telefónica)
- El teléfono dispone de servicio GPS y está activado
- El teléfono tiene cámara de fotos
- Disponga de batería disponible

Además de que el terminal cumpla con los condicionantes anteriores también se deben de cumplir unos requisitos adicionales para que el programa funcione correctamente y estén todas las funcionalidades disponibles. En el caso de la red de datos, no basta con tener conexión a una red de datos, también debe tener acceso a Internet (puesto que se puede estar conectado a una red inalámbrica WiFi pero no tener acceso a Internet, cosa bastante común sobretodo para los puntos de acceso de Internet pago públicos), en el caso de las redes telefónicas teniendo conexión 3G ya garantiza la conexión a Internet. Respecto al servicio GPS, además de tener habilitado y activado el adaptador GPS, también se deberá de asegurar que la señal se recibe correctamente en el terminal, para esto hay que evitar zonas arboladas, cubiertas y en general cualquier obstáculo que pueda hacer sombra a las señales de los satélites.

En caso de que alguna de las condiciones de ejecución del programa no se cumpla, se notificará al usuario y se procederá con las medidas definidas en los apartados anteriores.

Seguridad

La aplicación deberá solicitar el permiso del usuario para conectarse a internet, utilizar el GPS, controlar hardware (la cámara), leer el estado del telefono y modificar o eliminar contenidos de la tarjeta SD del terminal. No se requerirá ningún permiso más que los mencionados. En caso de que la aplicación se instale mediante el Android Market los permisos se solicitarán antes de instalar la aplicación.

En caso de uso de redes de datos inalámbricas (WiFi) la encriptación de los datos transmitidos será responsabilidad del punto de acceso.

La robustez del programa debe ser lo suficiente para resistir usos indebidos, accidentales o maliciosos.

Las bases de datos de puntos de interés no contienen ningún tipo de información personal o sensible a un ataque o publicación indeseada. Es por ello que no se debe tomar ninguna medida de seguridad adicional a las que se tomarían con un archivo al uso.

Mantenimiento

El mantenimiento de la aplicación debe ser mínimo. La aplicación debe ser robusta y no requerir cambios posteriores a la implantación, a parte de las mejoras que se pudieran hacer. Lo único que debe requerir mantenimiento es la página web, el

mantenimiento del dominio (o servidor) y la actualización de los datos (sobretodo las bases de datos) debe estar garantizado.

Portabilidad

En ningún caso el objetivo de este proyecto es desarrollar la aplicación con la intención de hacerla lo más portable posible. A pesar de que los futuros pasos de este proyecto probablemente sea portarlo a otra plataforma, no se hará ningún esfuerzo adicional durante este proyecto para facilitar la dicha.

La aplicación no es portable, pero mencionar que, al ser desarrollado sobre Android, se podrá ejecutar en cualquier dispositivo que soporte la versión 2.1 de Android o superior. Esto abre el abanico de dispositivos objetivo a teléfonos inteligentes y Internet-tablets.

DISEÑO

En esta sección se empezará detallando la arquitectura general de la aplicación. Se continuará con el análisis del modelo de datos y para finalizar se tratará la fase de adquisición de datos de la aplicación.

1. Arquitectura de la aplicación

Para entender mejor la manera en la que la aplicación fue desarrollada, el siguiente diagrama, en la Ilustración 13, muestra la arquitectura general del programa.

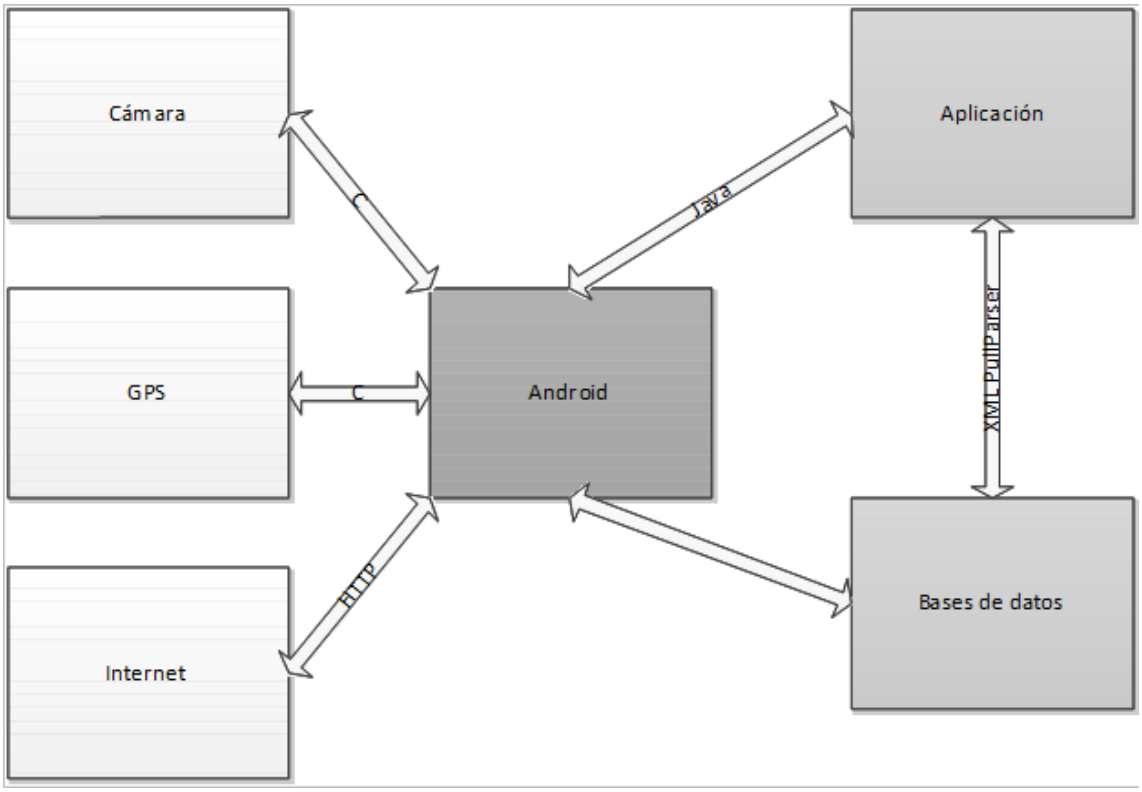


Ilustración 13: Arquitectura general de la aplicación

Como se puede apreciar en el diagrama, el almacenamiento del teléfono contiene las bases de datos, estas bases de datos serán gestionadas por una clase del programa, que será detallada a continuación. Además se detallará las clases que se encargan de hacer peticiones HTTP para obtener datos de Internet.

Diagrama de clases general

En el siguiente diagrama de clases simplificado de la Ilustración 14 nos da una idea general de la distribución de clases de la aplicación. Sólo se muestran los atributos y métodos más importantes de cada clase.

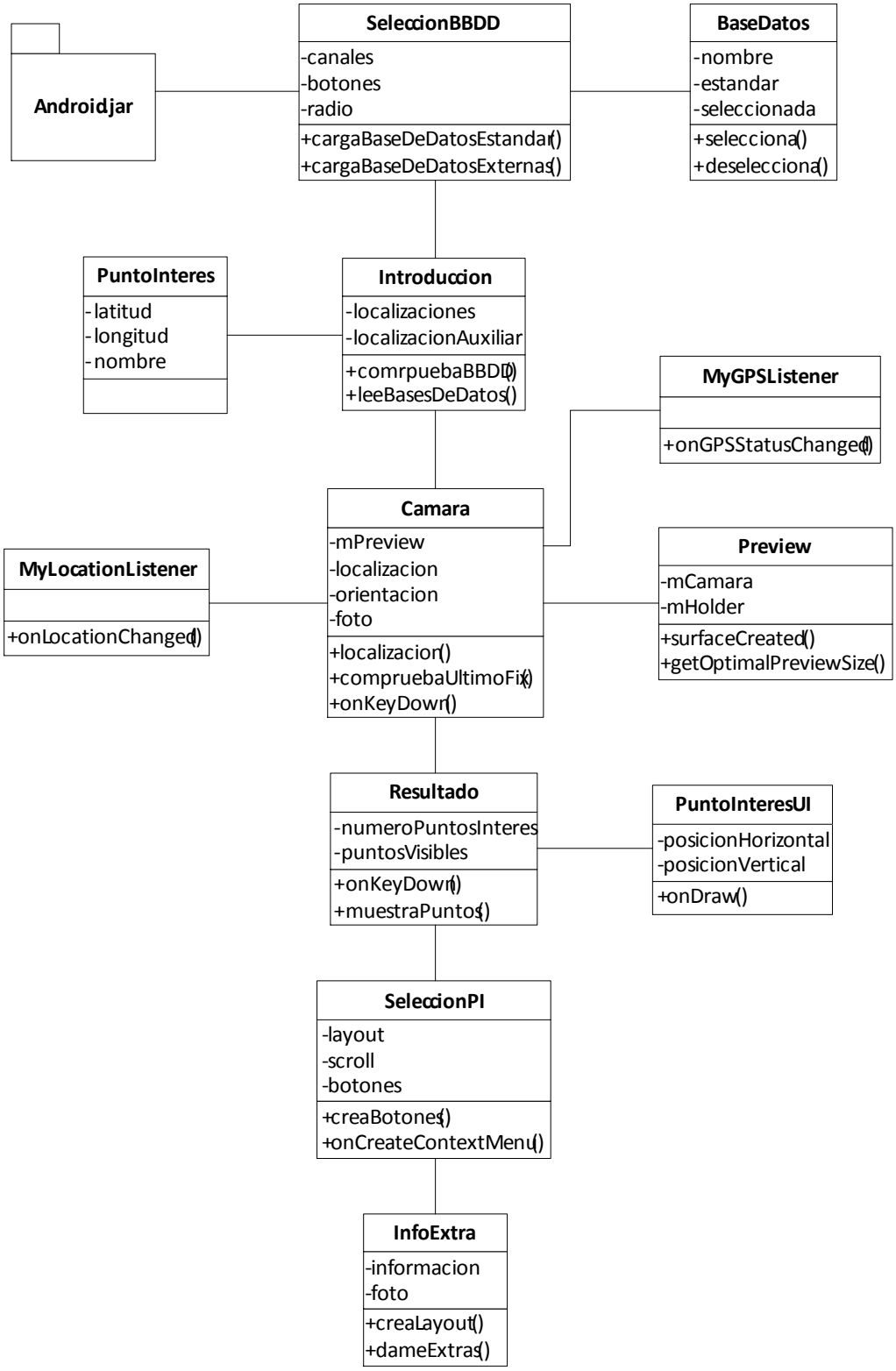


Ilustración 14: Diagrama de clases de la aplicación

Las clases que son Activitys se muestran en el eje principal del diagrama, es decir en el medio. Como se puede apreciar, aparte de algunas clases auxiliares, la mayoría de clases son Activitys que se corresponde a la interfaz gráfica de usuario. Esto es debido a la arquitectura de Android, que está basada en crear Activitys por cada ventana en la que el usuario puede interactuar. Si se sigue el hilo principal, se ve que cada Activity crea otra, hasta llegar al final de la aplicación, por lo que se parece

bastante a un diagrama de flujos.

Además de las clases Activity, existen algunas auxiliares que se encargan principalmente de adquisición de datos, representación de éstos y llamadas al sistema operativo para obtener alguna funcionalidad (como la cámara o los datos GPS).

Aunque sólo la primera Activity está conectada a la librería Android.jar, todas las clases de este proyecto están conectadas a ella. Se ha suprimido esta conexión por motivos de simplicidad y de facilidad de lectura.

2. Análisis detallado de las clases

A continuación se hace un análisis de las distintas clases que conforman este software. Se ajuntan los diagramas de clases más importantes, así como las clases auxiliares que están asociadas a éstos. Se hará un comentario pormenorizado de los métodos y variables más importantes del código.

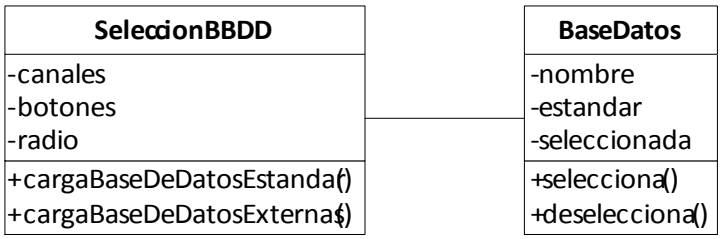


Ilustración 15: Diagrama de las clases SeleccionBBDD y BaseDatos

La clase SeleccionBBDD es la encargada de crear los objetos Base de Datos, es por ello que están fuertemente ligadas.

La Activity SeleccionBBDD se encarga de ofrecer visualmente al usuario todas las bases de datos almacenadas en el teléfono móvil además de permitirle ajustar el radio de acción de la aplicación. Aparte de los métodos para crear la interfaz gráfica de usuario, los más importantes son los que leen las bases de datos. Éstos son:

- cargaBasesDeDatosEstandar()
- cargaBasesDeDatosExternas()

Ninguno de los dos métodos leen el contenido de las bases de datos, si no que crean objetos que representan las bases de datos presentes en el terminal. Para ello, pro cada una de ellas crean un objeto BaseDatos y lo añaden al vector *canales*.

El primer método se encarga de buscar la base de datos llamada "historicodemadrid.xml", "leganes.xml" y "restaurantesdemadrid.xml", que se encuentran en la carpeta privada de la aplicación y siempre se cargarán. Estas tres bases de datos, puesto que son predefinidas y siempre estarán presentes (salvo por modificación del paquete apk) se cargan de manera programática. Por otro lado el método *cargaBasesDeDatosExternas* se encarga de recorrer la carpeta de descargas del teléfono en busca de archivos XML y agregar el objeto BaseDatos por cada uno de ellos al vector *canales*. El siguiente diagrama, mostrado en la Figura 16 representa la exploración que realizan los dos métodos.

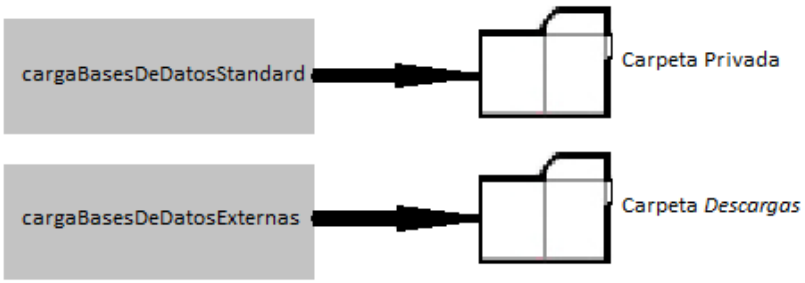
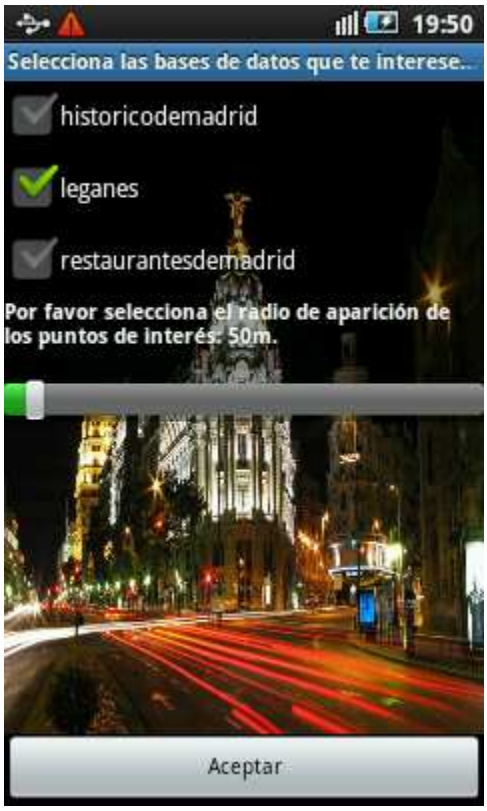


Ilustración 16: Carpetas tratadas por los métodos cargaBasesDeDatos

Además de estos métodos, en la clase SeleccionBBDD, el método compruebaBasesDeDatos permite controlar el caso de que no existan bases de datos (o no se cargue ninguna por algún motivo). Aunque es poco probable que pase, puesto que hay que modificar el código y la estructura interna del paquete, es posible que algún desarrollador haga alguna modificación y pueda darse este caso. También como medida de control se crea el array de booleanos *seleccionados*, en el cual se guardan si las bases de datos que se han seleccionado o no. De esta manera, cuando se vuelve a pintar la ventana (por que se cambia la orientación del teléfono, por ejemplo), se guarda la selección de bases de datos que estaba hecha por el usuario. La siguiente captura de pantalla muestra: primero la ventana original, segundo la ventana sin gestión de orientación y por último con gestión de orientación. Como puede observarse en los distintos casos, de esta manera se mantienen los cambios hechos por el usuario.



Caso 1: Posición y selección original

Caso 2: Rotación sin gestión de orientación

Caso 3: Rotación con gestión de orientación

Ilustración 17: Comparación de ventanas con y sin gestión de orientación

En la Activity Introducción, el método principal es *leeBasesDeDatos* que, por cada objeto BaseDatos en *canales* que haya sido seleccionado, lee el archivo XML correspondiente. El método posee mecanismos que permiten detectar si la base de datos está mal formada, o simplemente no se trata de una. Además, por cada punto de interés que lea creará un objeto de la clase PuntoInteres.

Introducción lanza un método llamado *compruebaBBDD* que controla que el vector *canales* no sea nulo, a pesar de que se comprueba anteriormente, por si acaso el usuario no elige ninguna base de datos. También tiene otro mecanismo de control compuesto por dos métodos que comprueban que el servicio GPS está activado. El primero llamado *compruebaGPS* se encarga de comprobar si el GPS está activado, si la respuesta es negativa, se lanza el método *alertaNoGPS* que muestra al usuario una ventana emergente avisándole de que lo active, además se encarga de abrir la ventana de configuración del servicio de posicionamiento del terminal.

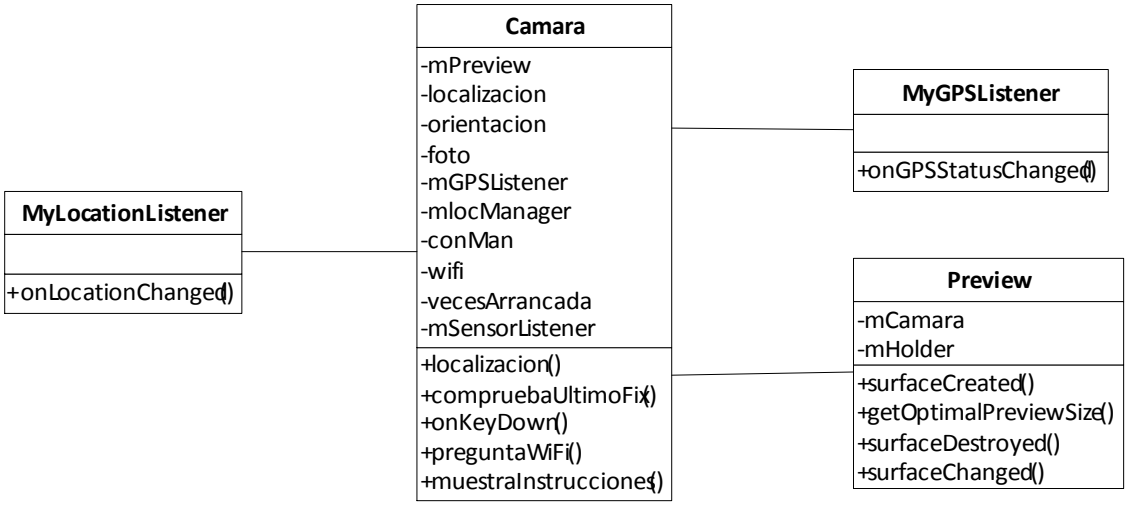


Ilustración 18: Diagrama de las clases Camara y auxiliares

La siguiente Activity, *Camara*, es una de las más importantes y complejas de todo el software. Cuando se lanza ésta Activity, todos los puntos de interés de las bases de datos seleccionadas estarán cargados en memoria. Lo primero que se hace es crear un objeto de la clase auxiliar Preview, que permite mostrar por pantalla al usuario lo que captura el objetivo de la cámara. Una vez hecho esto, se llama al método *muestraInstrucciones* que muestra al usuario qué botón debe pulsar para tomar una fotografía.

En cuanto al servicio de posicionamiento, existen dos métodos principales (*localizacion* y *preguntaWiFi*) y uno secundario (*compruebaUltimoFix*).El primer método, *localizacion,* se encarga de activar el servicio de localización del GPS añadiendo escuchadores a los adaptadores GPS (puesto que el posicionamiento por GPS esté activo ya se comprobó en la anterior Activity). Además el método *preguntaWiFi* detecta si el usuario está conectado a alguna red WiFi y, en caso de que lo esté, ajusta el servicio de posicionamiento para que se sirva de esos datos para localizar la posición, en el caso de que no lo esté preguntará al usuario si lo quiere activar. Esta pregunta sólo se realizará en caso de que sea la primera vez que se pasa por esta Activity (atributo *vecesArrancada*). Si el usuario lo quiere activar, se mostrará la ventana de ajustes de redes inalámbricas del terminal. El último método relacionado con el posicionamiento es *compruebaUltimoFix*, un método encargado de comprobar que los fix que usa el programa no están obsoletos (tienen mucho tiempo), de esta manera la posición del usuario se actualizará en el caso de que haya pasado 1 minuto.

El método *onKeyDown* (un método callback que se llama cuando se pulsa un botón) se encarga de comprobar que se han obtenido datos de posicionamiento y que éstos están actualizados (llamando a *compruebaUltimoFix*), adquirir la orientación, tomar una fotografía y a la vez pasar a la siguiente Activity. En caso de que no haya datos de localización, se mostrará un mensaje de advertencia pidiéndole que espere hasta que se consiga un fix.

El escuchador *mSensorListener* es el encargado de actualizar los valores de la orientación del terminal. La clase *MyLocationListener*, que implementa la interfaz *LocationListener*, se encarga de actualizar los datos de posicionamiento del terminal. El método *onLocationChanged* es llamado cada vez que el GPS consigue un fix. Este método se utiliza para actualizar el valor de *localizacion*. Además, también muestra al usuario una alerta, para que sepa que se ha obtenido un fix y que se puede pulsar el obturador (para pasar a la siguiente Activity).

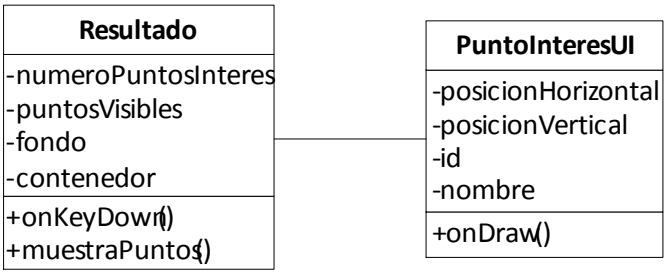


Ilustración 19: Diagrama de las clases Resultado y auxiliar

La clase *Resultado* recibe los datos de orientación y posición de *Camara* y los datos del radio de interés de *SeleccionBBDD*. Con estos datos, el método *muestraPuntos* se encarga de calcular los puntos de interés que se ven y crea un objeto *PuntoInteresUI* por cada uno de ellos. El vector *puntosVisibles* contiene todos estos objetos. Además, la clase *Resultado* se encarga de establecer como fondo de pantalla la foto tomada por *Camara* anteriormente, esto lo hace con el TextView *contenedor* y el Drawable *fondo*.

PuntoInteresUI representa gráficamente los puntos de interés que deben ser mostrados al usuario. Se crean a base de PuntoInteres que tienen el atributo *seve* a *true*. Éstos serán mostrados dependiendo del ángulo en el que se encuentran sus PuntoInteres correspondiente (representado por el atributo *anguloseve*), además se les aplicará una transformación para ajustarse a la pantalla del telefono móvil en el que se ejecute (puesto que pueden tener distintas resoluciones y tamaños). También tienen un identificador (un entero del 1 al 10). El atributo *nombre* es el texto que representará gráficamente al punto de interés para el usuario. El identificador determina el orden de los puntos de interés y también determinará la posición vertical en la que aparecerán, puesto que dos puntos de interés cercanos pueden solaparse si están en la misma posición vertical.

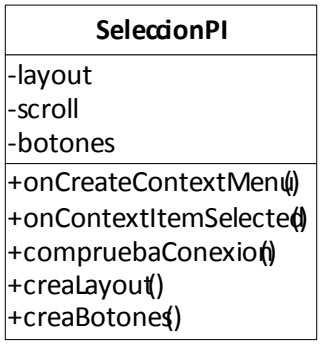


Ilustración 20: Clase seleccionPI

La Activity *SeleccionPI* crea un botón por cada uno de los objetos contenidos en *puntosVisibles*, esto es el array *botones*. También crea el menú emergente que aparece cuando se mantiene pulsado algunos de los botones. Además el método *compruebaConexion* comprueba si el teléfono tiene conexión a Internet, y en caso de que no la tenga mostrará la ventana de configuración de redes inalámbricas. También crea un *ContextMenu*, un menú desplegable que aparece cuando se mantiene pulsado el botón de alguno de los puntos de interés cercano. Este menú es creado por los métodos *onCreateContextMenu* y *onContextItemSelected*, donde se define cuántos botones se mostrarán y qué harán éstos. Como mínimo todos los puntos de interés tendrán 3 botones (buscar en Google, ver en Google Maps y ver entrada de Wikipedia) pero también se controla si los puntos de interés que se muestran contienen información extra(campos <INFO> y/o <URL> en la base de datos); en caso afirmativo, se mostrará un botón adicional para acceder a esa información. Una vez seleccionada la acción

pulsando un botón se obtiene la información del punto de interés para pasársela a la siguiente Activity, sea la que sea.

La Activity InfoExtra solo se podrá llamar si el punto de interés contiene alguno de los dos siguientes campos o ambos: <INFO> y/o <URL>. Esta Activity mostrará tanto el texto contenido en el campo <INFO> como la foto a la que apunta la URL del campo <URL>, en caso de que existan. Además tiene mecanismos de detección y aviso de URLs incorrectas.

Además de esto, encontramos la clase PuntoInteres, que representa cada uno de los puntos de interés existentes en las bases de datos. Contiene la siguiente información: geolocalización del punto, nombre y, opcionalmente: un campo *url* e *info*. Los atributos *cerca*, *seve* y *anguloseve*, son rellenados por Resultado, y determina si el punto está cerca de la posición actual del terminal (dentro del radio seleccionado por el usuario), si el punto se ve (está dentro del ángulo que cubre la lente del terminal) y el ángulo en el que se ve (respecto a la dirección en la que apunta la cámara), respectivamente. Si el atributo booleano *seve* está a *true* entonces ese objeto PuntoInteres estará representado en la interfaz gráfica del usuario mediante un objeto PuntoInteresUI. Los campos opcionales *url* e *info* estarán presentes en el caso de que el punto de interés tenga estos campos en la base de datos (y contendrán ésta información).

La clase BaseDatos representa las bases de datos y permite controlar si éstas son seleccionadas (y por tanto cargadas en memoria) y si son bases de datos estándar o descargadas. Los atributos de esta clase son: una cadena *nombre* (nombre del archivo XML), un booleano *estandar* que si está a *true* significa que es descargada y un booleano *seleccionada.* Estos objetos BaseDatos son instanciados en SeleccionBBDD y son tratados en Introducción donde, si el atributo *seleccionada* está a *true*, se cargarán en memoria la información contenida en las bases de datos XML.

3. Modelo de datos

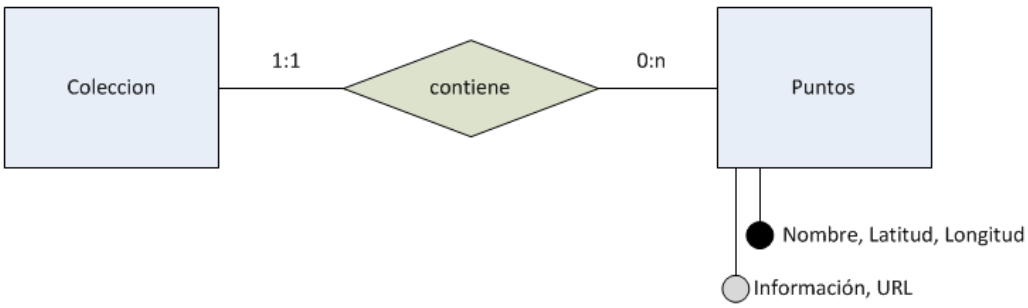


Ilustración 21: Diagrama Entidad-Relación

El diagrama entidad relación mostrado en la Ilustración 21 muestra el modelo de datos utilizado para las bases de datos de puntos de interés. La colección contiene todos los puntos de interés de una base de datos. Por un lado, un punto de interés tiene los siguientes atributos obligatorios: nombre, latitud y longitud (ambas medidas en grados decimales). Por otro, un punto de interés puede tener información adicional que añada el creador, y ésta puede ser texto y/o una url apuntando a una fotografía.

Las claves primarias están representadas en el diagrama con el punto negro, mientras que las opcionales están representadas por un punto gris. Las claves son

elegidas así por que un nombre puede repetirse, a lo mejor en distintas ciudades (por ejemplo, la Plaza de España). Por otro lado ni la longitud ni la latitud son claves por sí solas, puesto que en un mismo punto puede haber dos puntos de interés (en Madrid por ejemplo, la Plaza de Callao y el edificio Capitol). Por lo tanto, ni el nombre por sí solo puede ser clave, ni la latitud ni la longitud, si no el conjunto de todas. La información adicional y la URL no son claves, puesto que no identifican inequívocamente un punto de interés, además de el hecho de que son campos opcionales.

4. Diagrama de secuencia de los procesos importantes

A continuación se muestran los diagramas de secuencia de los procesos más importantes que se llevan a cabo durante la ejecución de la aplicación. Después de cada diagrama habrá una breve explicación de cada uno de ellos.

Los dos procesos principales que tienen lugar en el programa son la secuencia de lanzamiento de la aplicación y la obtención de datos de geolocalización, orientación y cotejo con las bases de datos cargadas en la memoria.

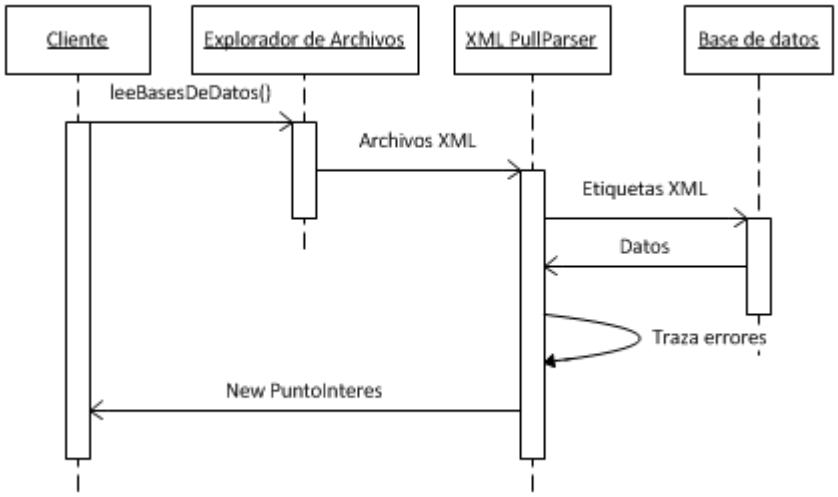


Ilustración 22: Diagrama de secuencia de lanzamiento de la aplicación

En la Ilustración 22 se muestra la secuencia de lanzamiento de la aplicación. Todo el proceso se realiza en la Activity Introducción, que además de mostrar las instrucciones de la aplicación, carga los puntos de interés en la memoria RAM. El primer paso que se toma es llamar a método leeBasesDeDatos, que se encarga de ir pasándole al explorador todas las bases de datos que el usuario haya seleccionado. Las bases de datos seleccionadas por el usuario están contenidas en el vector *canales*. El explorador, por su parte le pasa los archivos XML a un objeto parser(una instancia de XML PullParser). Este parser se encarga de ir leyendo la base de datos XML mediante etiquetas XML y leyendo los campos que éstas contienen. A medida que el parser va recibiendo datos de cada uno de los puntos de interés contenidos en la base de datos, éste va creando objetos PuntoInteres y los va introduciendo al vector *localizaciones*. Estos objetos PuntoInteres representan los puntos de interés contenidos en las bases de datos y están cargados en la memoria RAM, para tener un acceso más rápido a los datos. Mientras que se van parseando los datos de las bases de datos XML, se van generando trazas por si hay algún tipo de error.

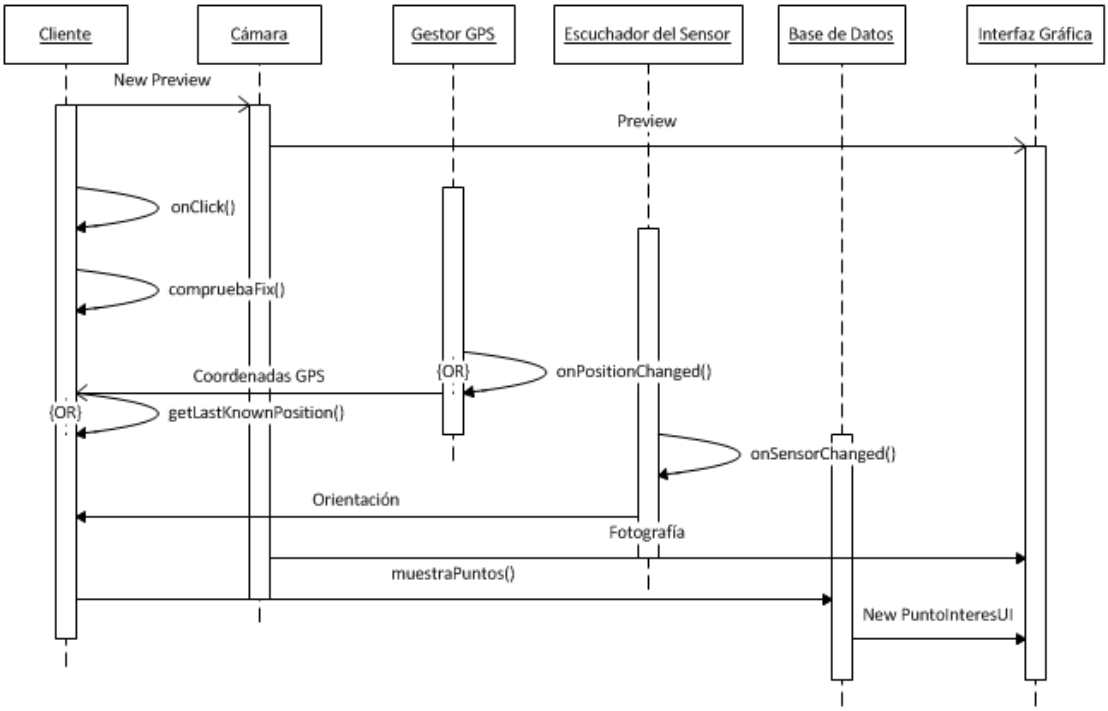


Ilustración 23: Diagrama de secuencia de la fase de adquisición de datos

Lo que se muestra en la figura anterior (Ilustración 23) sucede entre las Activitys Camara, Resultado y La base de datos es *localizaciones*, la que se creó en la secuencia de arranque. Al iniciar la Activity Camara, se crea un objeto Preview que muestra por pantalla lo que está capturando el objetivo de la cámara del teléfono. Cuando el usuario pulsa el botón de menú, representado por el método *onClick()*, se comprueba si hay datos de localización GPS. Ésto se hace con el método *compruebaFix()*, que también comprueba si el fix está actualizado u obsoleto. En caso de que esté obsoleto, se esperará a que el gestor de GPS actualice la posición cuando llame al método *onPositionChanged()*. Si el fix es correcto, es decir: existe y está actualizado se llama al método *getLastKnownLocation()*. En cuanto se tiene la posición, se lee el valor que devuelve el método *onSensorChanged()*, para obtener la orientación del terminal. se tomará una foto y ésta se establece de fondo de pantalla. El método onPositionChanged() es un método callback que es llamado por MyLocationListener, clase que implementa la interfaz LocationListener. Este método devuelve un objeto Location llamado *localizacion*, al cual más adelante se le extrae la latitud y la longitud El método getLastKnownLocation devuelve la última posición conocida en forma de objeto Location al igual que *onPositionChanged()*. onSensorChanged es un método callback que llama SensorListener cuando la orientación del dispositivo cambia. Este método devuelve un float el cual se guarda en la variable *orientacion*.

En el momento en que se obtienen los datos de localización y orientación, se toma la fotografía, que se establece como fondo de pantalla de la siguiente Activity. Además con los dos datos obtenidos anteriormente, localización y orientación (en las variables *localizacion* y *orientacion*, respectivamente), se consulta la base de datos mediante el método *muestraPuntos()*. El método muestraPuntos se encarga de crear objetos PuntoInteresUI que son las representaciones gráficas de los puntos de interés cercanos. Todos estos objetos PuntoInteresUI se introducen en un vector llamado *puntosVisibles*.

5. Aspectos Generales del Diseño

Dado que los requisitos definidos para la aplicación eran bastante claros en cuanto a que el uso de ésta debía ser fácil y claro, se tuvo que pensar en una manera de disponer la información sobre la pantalla de la manera más clara posible. Hay que mantener cierta coherencia entre ventana y ventana, puesto que así, el usuario se siente más cómodo y le resulta más fácil moverse por la aplicación.

A continuación se muestra la plantilla básica de cualquier aplicación Android. Es la disposición que tienen los objetos de la interfaz gráfica por defecto. En esta sección solo se tratarán las partes de la interfaz que tienen algo que ver con el software desarrollado. Por supuesto, todo lo descrito aquí, a pesar de que se muestre en modo apaisado, vale también para el modo vertical.

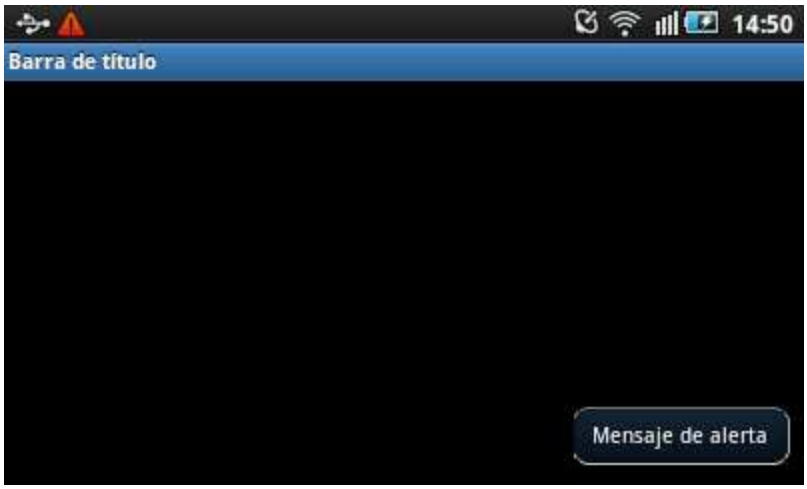


Ilustración 24: Plantilla básica de una aplicación Android

Existen varios iconos que indican qué tipo de servicios están corriendo en el teléfono. Ninguna de estas funcionalidades ha sido cambiada. En el caso que vemos en la pantalla indica, de izquierda a derecha: el terminal está conectado por USB, está en modo debbugging, está adquiriendo datos del GPS, tiene WiFi conectado, tiene cobertura telefónica, la carga de de batería y la hora.

Además de esos iconos, existe una barra de título para la ventana. Aquí se muestra, en el caso que corresponda, algún tipo de explicación sobre lo que se muestra en la ventana. Puede haber ventanas sin esta barra.

Abajo a la derecha se muestra un mensaje de alerta. Éstos pueden o pueden no estar presentes en la aplicación. Son mensajes cortos y que desaparecen en poco tiempo, como máximo dos segundos y medio. Son muy útiles para avisar al usuario de que haga algo, o de que algo va mal.

La interacción con el usuario puede hacerse mediante los botones físicos del terminal y los táctiles que aparecen en la pantalla. Normalmente los táctiles son autoexplicativos, ya que contienen textos que normalmente explican la acción que realizan. En el caso de los botones físicos suelen ser estándar (back para volver a la pantalla anterior, home para volver al escritorio, etc...) y en caso de que no lo sean se suele aclarar.

6. Plantilla básica de la aplicación GuiaInteractiva

A continuación se detalla la plantilla básica de las ventanas de la aplicación. La

plantilla básica utilizada está fuertemente ligada a la plantilla de Android. Esta plantilla básica, es una mera guía utilizada para implementar las diferentes ventanas de este software.

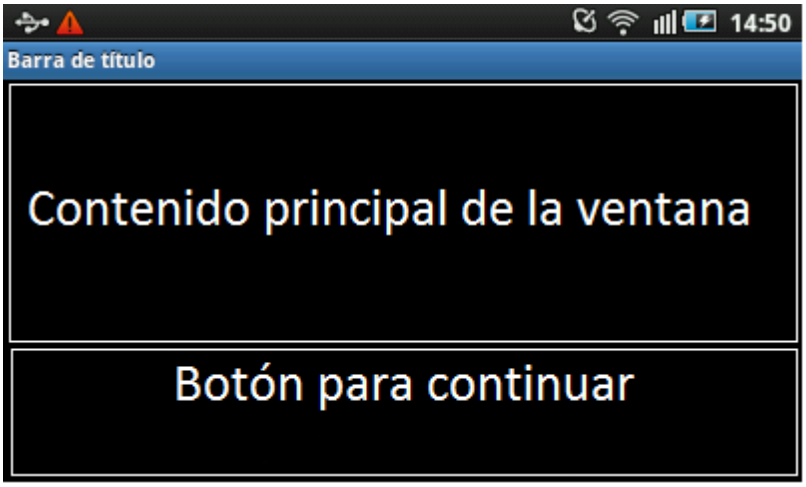


Ilustración 25: Disposición general de la aplicación

En la Ilustración 25, se muestra la disposición que se utilizará en la aplicación GuiaInteractiva, en los casos que corresponda. A continuación se detalla con mayor precisión la motivación para utilizarla.

La tipografía utilizada es la estándar de Android. Al usuario se le trata siempre de “tú”, de manera que sienta que es un programa cercano y amigable.

En las ventanas que no sean explícitamente de instrucciones tendrán un título que explicará por encima lo que se le está mostrando al usuario, así como lo que debe hacer. De esta manera, se está guiando al usuario todo el tiempo que ejecute la aplicación, y no se limita a explicarle todo el funcionamiento solamente en una ventana de instrucciones.

El contenido principal de las ventanas se intentará mantener en la parte de arriba, dado que se considera que es la parte más vistosa y a la primera que mira el usuario. De esta manera se consigue un cierto equilibrio, puesto que los botones adicionales, como los de continuar, estarán en la parte baja.

Una excepción a la regla mencionada en el párrafo anterior es la aplicada en el caso de mostrar gráficamente los puntos de interés cercanos sobreimpresos en la fotografía. Puesto que normalmente, cuando se toma una foto, se suele encuadrar bastante más cielo que tierra, se ha decidido que la zona donde se mostrarán estos puntos será a partir de la mitad hacia abajo de la pantalla.

Las ventanas que necesiten solamente de un botón táctil para continuar, lo tendrán ubicado en la parte baja de la pantalla. De esta manera el usuario asociará el concepto de continuar hacia la siguiente ventana con pulsar un botón en la parte de abajo de la pantalla. Esta disposición no se utilizará en el caso de que haya más de una posibilidad, puesto que todas deberían estar abajo, y quedaría descompensado el contenido de la ventana.

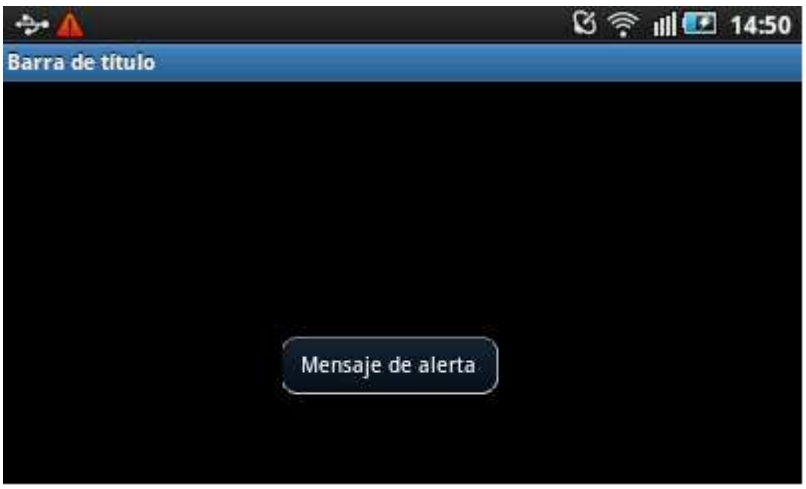


Ilustración 26: Alertas generales de la aplicación

En la figura anterior (Ilustración 26) se muestra la disposición de las alertas. Las alertas generales (el usuario no ha seleccionado ninguna base de datos, no ha activado el GPS), se mostrarán en el centro de la pantalla y un poco abajo verticalmente hablando. Estas alertas se mostrarán durante 2,5 segundos, por lo que el texto debe ser los suficientemente corto como para que el usuario lo pueda leer.

En caso de que se pida al usuario pulsar un botón físico, se mostrará un mensaje señalando a éste, de manera que si el usuario no sabe qué botón es, este mensaje se lo aclarará.

7. Interfaz gráfica de la aplicación GuiaInteractiva

A continuación se muestran capturas de pantalla de la interfaz de usuario. Ésta cumple las especificaciones definidas y explicadas en la sección de Análisis, bajo el apartado “Análisis de requisitos”. Además se sigue rigurosamente el diagrama de flujos definido.

Las impresiones, opiniones y sugerencias hechas por los usuarios a lo largo del desarrollo han sido tenidas en cuenta a la hora de darle forma a la interfaz de usuario final.

Selección de la base de datos

Una vez arrancada la aplicación, lo primero que hará ésta es preguntar al usuario por la base de datos que quiere utilizar. Haciendo click en cualquiera de las que aparezca y pulsando en el botón “Aceptar” llevará al usuario a la siguiente pantalla, además de cargar las bases de datos correspondientes.

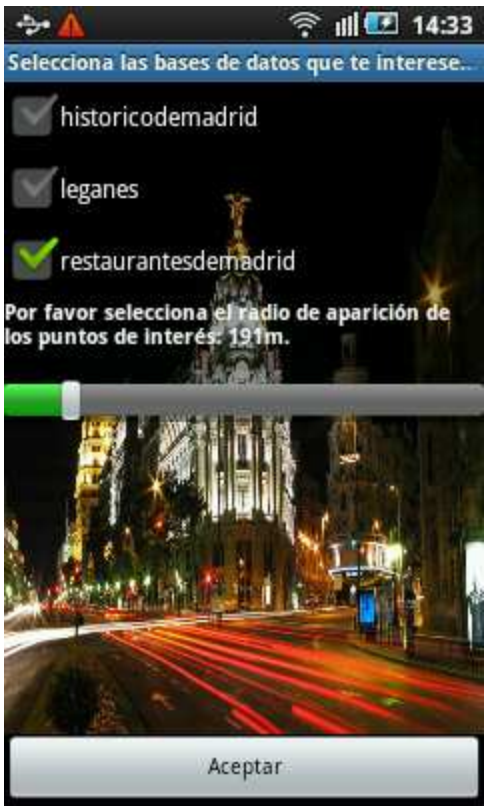


Ilustración 27: Selección de bases de datos

En el caso de que haya algún tipo de error cargando la base de datos, se mostrará por pantalla un mensaje como el siguiente, especificando cuál de las bases de datos ha sido la que sufrió el problema.

Ventana introductoria

Ilustración 28: Introducción

La siguiente ventana no tiene más utilidad que la simple información del usuario, el cual tendrá que pulsar en el botón “Pulse aquí para continuar” para pasar a la siguiente ventana. En esta captura de pantalla no se muestra ese botón, puesto que está más abajo, por ello la ventana se ha hecho deslizante (se solventan así los problemas de tamaño en teléfonos con pantalla pequeña). El usuario también podrá pulsar el botón “Back” para volver atrás y seleccionar otra base de datos.

Si, por el motivo que sea, el servicio GPS no está disponible, la aplicación dirigirá al usuario hacia la ventana de configuración, donde podrá habilitar el adaptador de GPS. Si no se activa, cuando el usuario vuelva intentar pasar a la siguiente ventana, se le volverá a redirigir hacia la pantalla de configuración de GPS.

Preview de la cámara

Antes de arrancar el preview de la cámara, si es la primera vez que se arranca la ventana, se preguntará al usuario, en caso de que no lo tenga activado, si quiere conectarse a una red WiFi, puesto que de esa manera se conseguirá un fix de manera más rápida. Esta pregunta se realizará mostrando una ventana emergente, mostrada en la Figura 29.

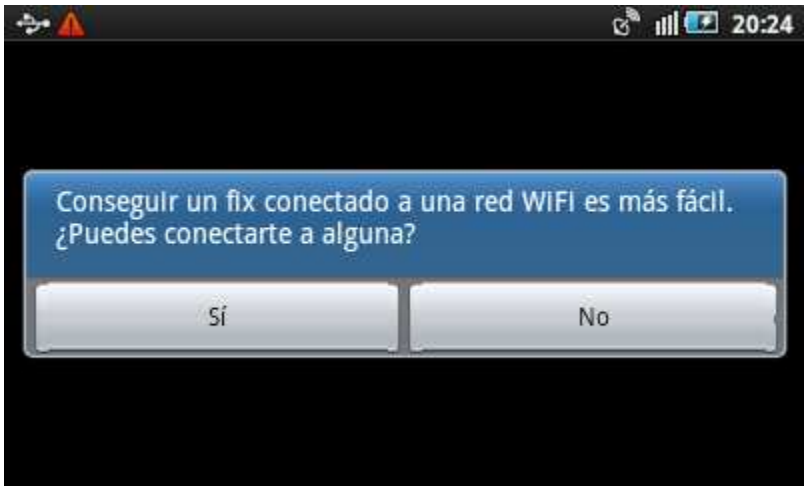


Ilustración 29: Consulta sobre conexión a una red WiFi

Esta ventana mostrará las imágenes que capture el objetivo de la cámara de fotos, de manera que el usuario pueda ver hacia dónde dirige su cámara, y por tanto, sobre qué lugares va a tener información. Para tomar una fotografía y pasar a la siguiente ventana deberá pulsar el botón de menú. También podrá volver atrás, a la ventana introductoria, pulsando el botón “Back”.

Ilustración 30: Preview de la cámara

En el momento en el que el usuario pulsa el botón menú, la aplicación solicita datos de orientación y de localización al sistema GPS y a las redes inalámbricas (en caso de que estén habilitadas). En caso de que los datos de localización no se hayan conseguido o estén obsoletos (tengan más de un minuto), se mostrará la siguiente advertencia (Figura 31).

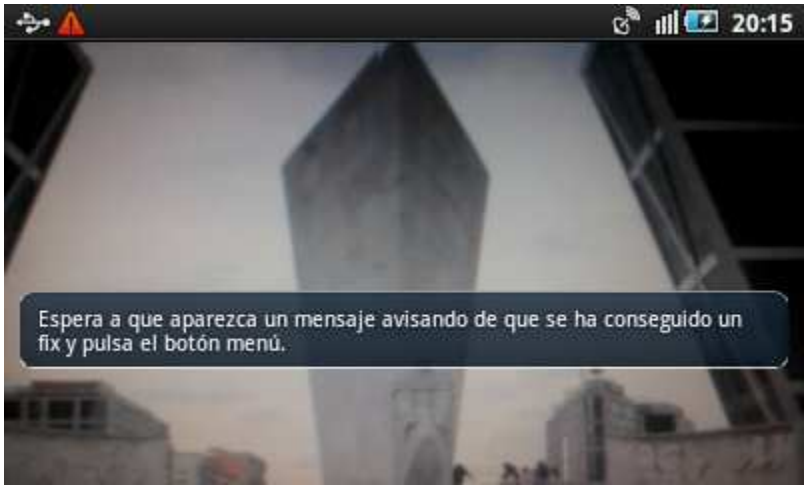


Ilustración 31: Alerta por falta de fix

En el caso de que se adquieran los datos correctamente, se tomará una fotografía y se pasará a la siguiente Activity. Además se oirá un sonido de obturador cuando se tome la fotografía.

Puntos de interés cercanos

Una vez que el usuario ha realizado la fotografía, ésta pasará al fondo de la pantalla de la ventana actual. En caso de que haya algún punto de interés cercano, éste se mostrará mediante un texto sobreescrito, en el que se indica su nombre. En caso de que el usuario quiera obtener más información de cualquiera de los puntos de interés que se muestran deberá pulsar cualquier lugar de la pantalla. En caso de que quiera volver atrás para tomar otra foto, o por cualquier motivo, deberá pulsar el botón “Back”.

Ilustración 32: Muestra de puntos de interés

En caso de que el número de puntos de interés cercanos a mostrar sea mayor que nueve, sólo se mostrarán los nueve primeros, además de lanzar un mensaje de aviso al usuario.

Selección de punto de interés

En esta ventana, se le muestran al usuario todos los puntos de interés cercanos a modo de lista. En caso de que quiera obtener más información de alguno de ellos, bastará con que mantenga pulsado el botón con su nombre. Se ha decidido que mantener pulsado el botón es más práctico, puesto que si hay muchos puntos de interés, el usuario al desplazarse por ellos puede hacer un simple click inintencionadamente sobre ellos y entrar a ver la información, lo cual puede resultar molesto. Haciendo esto le aparecerá un menú emergente en el cual podrá elegir la opción que más le interese. En esta ventana también podrá utilizar el botón “Back” para volver atrás en cualquiera de las dos ventanas.

Ilustración 33: Selección de punto de interés

Ilustración 34: Opciones por punto de interés

Dependiendo de la acción que elija el usuario, aparecerá la siguiente ventana.

Debido a la libertad que tiene el usuario al introducirse en la red, puesto que puede navegar por cualquier página, solo mencionar que para volver atrás, deberá pulsar el botón “Back” tantas veces como sea necesario, hasta que vuelva al menú mostrado en la Ilustración 33. Otra opción, para volver al escritorio, el usuario podrá pulsar el botón “Home”, lo cual dejará a la aplicación corriendo en segundo plano. En el caso de que el usuario volviese a la aplicación, pulsando el icono del escritorio, volvería a la pantalla donde dejó la aplicación por última vez.

IMPLEMENTACIÓN

Una vez hecho el diseño hay que seleccionar las tecnologías e infraestructuras necesarias para implementarlo. Una vez hecho esto empieza el proceso de desarrollo. A continuación se explican los contenidos de cada subsección.

En primer lugar, en la subsección de tecnologías utilizadas, se detalla el software utilizado en el desarrollo de la aplicación. En el apartado de infraestructuras se indica las máquinas físicas que se han utilizado en el proceso. En la última subsección se detalla todo el proceso de desarrollo de la aplicación, teniendo en cuenta la metodología que se ha seguido.

1. Tecnologías utilizadas

Se han usado varias y diversas tecnologías en el proceso de desarrollo e implementación de esta aplicación. El software que más se ha utilizado durante el desarrollo ha sido el entorno de desarrollo Eclipse Galileo 3.6.1, con Java 1.6.0_22 para sistemas de 64 bits. Este entorno es en el que se ha editado todo el código fuente.

A este entorno de desarrollo se le ha instalado el ADT un plug-in para poder desarrollar aplicaciones en Android usando el Eclipse. Este plug-in nos permite crear proyectos Andorid, crear interfaces de usuario para aplicaciones, añadir componentes basados en el Android Framework API, debuggear aplicaciones Android y exportar APKs, lo que permite distribuir aplicaciones Android. Además nos permite gestionar y crear terminales emulados Android. Para descargarse este plug-in basta con dirigirse a: http://developer.android.com/sdk/eclipse-adt.html.

Para emular el entorno de ejecución de la aplicación (un terminal móvil), se ha utilizado el Android Virtual Device Manager. Este gestor de terminales móviles virtuales permite crear emuladores de terminales con distintas características. Cada uno de estos terminales emulados es una aplicación basada en QEMU que provee al usuario de un terminal móvil ARM virtual en el que poder ejecutar sus aplicaciones. Este emulador es una pila del sistema Android al completo, y la versión del sistema operativo es seleccionable, dando como opción varias versiones del sistema operativo Android. El SDK que se distribuye contiene código para el kernel Linux de Android, las librerías nativas, la máquina virtual Dalvik y varios paquetes Android. Este emulador se puede lanzar y recibir datos desde un terminal con telnet o desde el propio Eclipse, al cual se integra al instalar el ADT antes mencionado.

A continuación, en la Figura 35, se muestra un diagrama en el que se representa el entorno de desarrollo que se ha utilizado y en el que aparecen todos los módulos que interactúan entre sí. De esta manera el desarrollador puede hacerse una idea de cómo funciona el entorno de desarrollo.

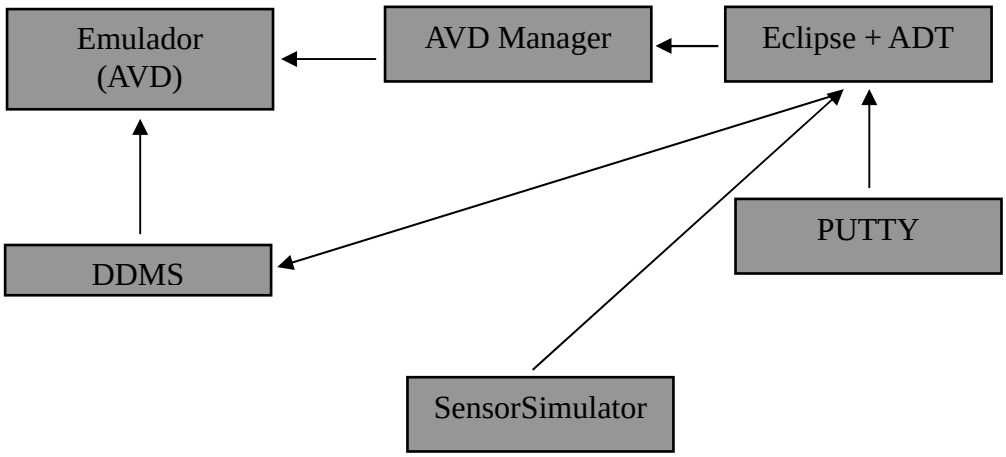


Ilustración 35: Entorno de desarrollo completo

A continuación se explican tanto las funciones de cada uno de éstos módulos, así como las instrucciones de cómo crearlos (en el caso de que corresponda). Como se ha mencionado antes Eclipse y el AVD Manager permiten crear terminales Android emulados (AVD). Para obtener instrucciones de cómo crearlos, referirse al Anexo E.

Uno de los mejores aliados durante el desarrollo ha sido la perspectiva DDMS de Eclipse. Esta perspectiva permite ver las trazas tanto de usuario como los mensajes que lanza el código Android. También permite obtener capturas de pantalla, así como explorar el contenido de las carpetas del terminal.

Durante el desarrollo era necesario pasarle datos ficticios al GPS del emulador, para ello se ha utilizado la consola de Microsoft Windows, y se tuvo que habilitar el servicio Telnet desde el panel de control de Windows. Se puede utilizar como guía el siguiente tutorial: http://www.fettesps.com/windows-7-enable-telnet/.

Otra opción que también se utilizó fue el programa gratuito PUTTY, que permite conectarse mediante Telnet, SSH, rlogin, raw o serial a cualquier máquina, ya sea real o virtual. Para conectarnos al emulador deberemos usar Telnet, usar localhost (o 127.0.0.1) como Host Name y cambiar el puerto a el que se muestra en el marco de la ventana del emulador. La configuración por defecto para conectarse al terminal virtual de Android se muestra en la Figura 36. Para descargarse la aplicación basta con visitar la siguiente página: http://www.putty.org/.

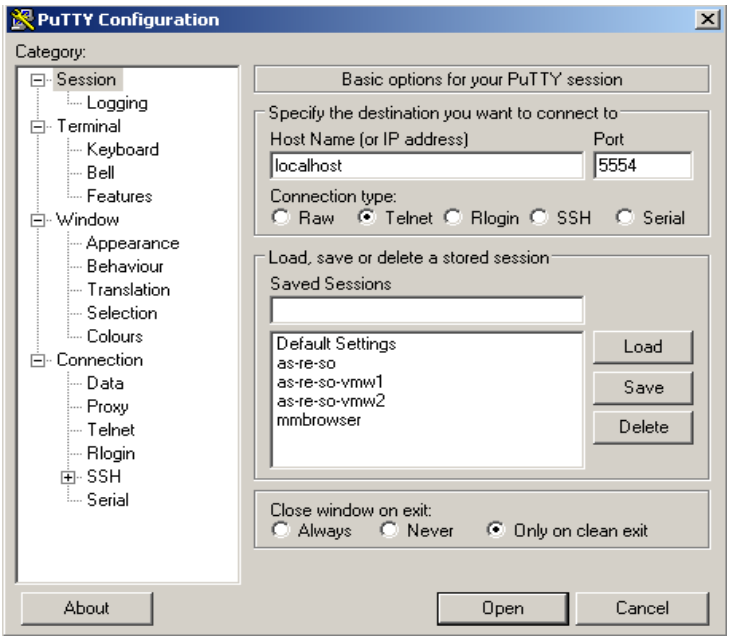


Ilustración 36: Configuración de PUTTY para conexión con emulador

Para enviar al terminal virtual datos simulados de GPS se introducía la siguiente línea en la consola que abre PUTTY:

geo fix <longitud> <latitud>

Cierto es que en las instrucciones oficiales la línea de comando especificada requiere primero la latitud y después la longitud, pero por algún motivo la implementación se ha hecho al revés. Esto ha sido probado en terminales físicos, por lo que no hay otra explicación que esa.

A modo de ejemplo se muestra la Ilustración 37, con unos datos de prueba. Es probable que la primera vez que se introduzcan la instrucción, la consola no la reconozca (como en el ejemplo de la figura). Basta con intentarlo de nuevo y funcionará.

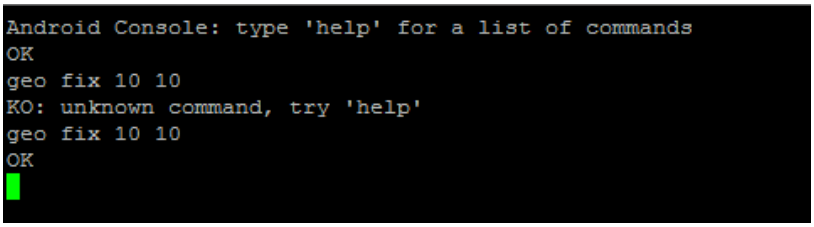


Ilustración 37: Cómo introducir datos GPS al terminal emulado

Además de los datos ficticios del GPS también se tuvieron que generar datos de orientación ficticios al emulador. Para ello se utilizó Sensor Simulator desarrollado por Open Intents, que provee al usuario de una brújula. Este programa consta de dos módulos: una aplicación que se instala en el terminal emulado (cliente) y otra que se instala en la máquina en la que se desarrolla la aplicación (servidor), en este caso un PC. Desde ésta última se envían señales que simulan a la brújula por un puerto y dirección IP determinada, este puerto y dirección se deberán configurar en la aplicación instalada en el emulador (cliente). De esta manera el simulador instalado en el PC (servidor) transmite al teléfono virtual datos de orientación. Para lograr que estos dos módulos se comuniquen el servidor (PC) debe tener una dirección IP, aunque no necesita de Internet. Las instrucciones de cómo instalar el cliente en el emulador se encuentran

detalladas en el Anexo G. En dicha sección se detalla como instalar la aplicación desarrollada en el emulador, por lo que habrá que aplicar los mismos pasos pero cambiando todas las referencias a la aplicación a la de SensorSimulator. Para descargarse la aplicación basta con seguir el siguiente hipervínculo: http://code.google.com/p/openintents/wiki/SensorSimulator.

Para introducir archivos en el terminal emulado, simulando que han sido descargados de internet, debían introducirse en la tarjeta SD. Para ello, en el emulador habría que montar el sistema de archivos para simularla. Si el terminal se ha creado siguiendo las instrucciones dadas en el Anexo E, no habrá que hacer nada para montar el sistema de archivos, más que continuar siguiendo las instrucciones que muestran cómo crear una carpeta en la tarjeta SD.

Puesto que para simular la carpeta de descargas hay que montar una tarjeta SD virtual, en el caso de que no se haya montado el sistema de archivos durante la creación del terminal emulado, habrá que montarlo expresamente cada vez que se arranque de nuevo el emulador. Para ello se introducía en el terminal, situándolo en la carpeta "tools" del sdk de Android (por defecto: C:\Program Files\Android\android-sdk-windows\tools), el siguiente comando:

mksdcard 512M sdcard.iso

Con este comando se monta una tarjeta SD de 512MBytes. Una vez montado el sistema de archivos se arrancará el terminal desde la consola, para que cargue los cambios, con el siguiente comando:

emulator -avd <nombre_del_terminal> -sdcard sdcard.iso

Para crear una carpeta en la tarjeta SD del terminal emulado por línea de comandos (terminal o consola) se utilizó la siguiente instrucción estando en la carpeta de instalación de Android SDK:

adb shell "mkdir /sdcard/download"

De esta manera ya se tenía la tarjeta montada y una carpeta llamada *download*, que simula la carpeta donde se descargarían los archivos descargados en un terminal real.

Para introducir archivos en el teléfono inteligente virtual, se ha utilizado la Dalvik Debug Monitor Server (DDMS) de Eclipse. DDMS es una herramienta de depuración que proporciona varios servicios muy útiles a la hora de desarrollar aplicaciones para Android. En este proyecto se ha utilizado para guardar las bases de datos en el teléfono virtual. Para hacerlo debemos abrir la perspectiva DDMS en Eclipse, seleccionar el botón "Push a file onto the device" (mostrado en la Ilustración 38) que abrirá una ventana en la que se elige el archivo que se quiere subir al terminal.

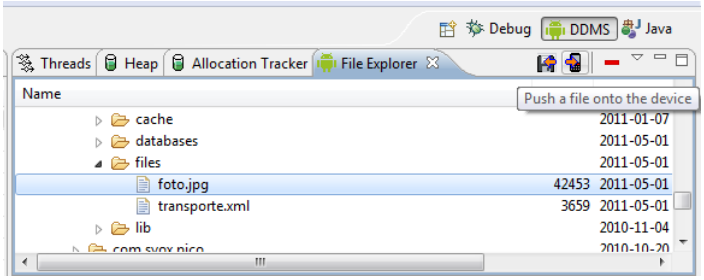


Ilustración 38: Perspectiva DDMS en carpeta privada de la aplicación

Como rutina para empezar a desarrollar, se arrancaba el servidor de SensorSimulator que por defecto enviará los datos por la IP 192.168.1.100:8010. Una vez arrancado, se abría el Eclipse, con el que se arrancaba el emulador mediante el AVD Manager. Se arrancaba también PUTTY para enviar datos de GPS al terminal. Una vez cargado el emulador, en caso de querer introducirle un fichero se usará la perspectiva DDMS tal y como se explicó anteriormente. Una vez arrancados todos los servicios se ejecutaba la aplicación a desarrollar y, en caso de ser necesario, mandar datos GPS (mediante PUTTY) o de la brújula (mediante SensorSimulator).

Para editar la base de datos en XML, se ha utilizado el MS Wordpad además del Notepad++. Además se han utilizado Mozilla Firefox 5.0 y Google Maps, para obtener las coordenadas GPS de los puntos de interés de prueba. Además Mozilla Firefox ha sido usado para recabar la mayoría de información necesaria para desarrollar la aplicación: documentación acerca de Android, datos de geoposición para los puntos de interés, emulador de brújula...

Para instalar la aplicación en el terminal móvil se han seguido los pasos descritos en el Anexo F. Puesto que hay tres maneras de hacerlo, se han llevado a cabo instalaciones de todas las maneras posibles para probar que no existe algún tipo de error. La primera de ellas fue con el Eclipse, puesto que da la posibilidad de monitorizar las trazas del programa y del propio teléfono, lo cual es muy útil durante el desarrollo para detectar errores u obtener más información que la que se obtendría como un simple usuario.

2. Infraestructura utilizada

Todo el desarrollo de la aplicación ha sido llevado a cabo en un ordenador portátil personal y un ordenador personal de sobremesa. El proceso de desarrollo se realiza de igual manera en arquitecturas Linux y Microsoft Windows.

Las características de los ordenadores usados en el desarrollo son:

Ordenador portátil:

- Procesador: Intel I5 M430. 2 Nucleos a 2,27GHz.
- Memoria: 4GB DDR3.
- Sistema operativo: Windows 7 Home Premium 64 bits.

Ordenador sobremesa:

- Procesador: AMD Athlon 64 X2 4200 a 2.20 GHz.
- Memoria: 1,5GB DDR2.
- Sistema operativo: Linux Ubuntu 10.04 64 bits.

Se ha utilizado una red de datos inalámbrica (WiFi) sin ningún tipo de restricción. El firewall debe de permitir la comunicación de Java ya que es necesario a la hora de desarrollar que el programa Sensor Simulator se comunique con el Android Emulator (ADV).

Además se ha utilizado un terminal móvil Samsung GT-I5800 (o Galaxy 3), para realizar las pruebas correspondientes. El procesador es de 667MHz y posee una RAM de 256MB. Los servicios WiFi y GPS deben de estar tanto activados como disponibles.

Esto es, tener conexión a una red WiFi además de tener acceso a Internet. En el caso del GPS significa tener activado el adaptador GPS del terminal, además de recibir correctamente las señales para obtener un fix.

3. Proceso de desarrollo

Esta sección está orientada a detallar todo el proceso de implementación, comentando todos los pasos seguidos para desarrollar cada uno de los módulos que componen el software final.

3.1 Visión general

Debido a la metodología escogida, que favorece el desarrollo de prototipos en etapas tempranas del proyecto a costa de sacrificar la planificación, el proceso de desarrollo fue un constante ciclo. Este ciclo constaba de las siguientes fases: definición de requisitos, implementación y validación. Una vez completado, en caso de ser necesario, se pasaba otra vez a redefinir los requisitos. Las validaciones y la definición de requisitos las realizaba el tutor del proyecto. La mayor parte del proceso de validación se realizaba concertando reuniones en la universidad Carlos III aunque, excepcionalmente y para funcionalidades sencillas, se hacía mediante correo electrónico. Para obtener una visión global y obtener una mejor imagen de todo el proceso, se puede consultar la sección Planificación de este documento.

En el Anexo H se encuentra la nomenclatura usada durante el desarrollo de la aplicación. Puede consultarse para obtener una mejor visión de las reglas que se han seguido durante el desarrollo. Otro documento contenido en esta memoria que puede ser de gran utilidad para obtener una idea general del proceso de desarrollo es el roadmap, que se encuentra en el Anexo I.

El primer prototipo desarrollado tenía unos requisitos bastante básicos, éstos eran: leer un archivo de fotografía y sobreimprimir un texto en ella. Se trataba de un prototipo desechable, puesto que la estructura nada tenía que ver con la del software final, pero se adquirieron conocimientos que luego se aplicarían en el programa final.

3.2 Proceso de desarrollo

Dada la metodología escogida para el proyecto, el proceso de desarrollo comenzaría con la especificación de requisitos básicos y generales, que a medida que se fueran desarrollando, se iría especificando de manera más precisa.

El primer paso para comenzar el desarrollo fue establecer y configurar el entorno de trabajo. Puesto que en éste intervienen numerosos módulos con distintas funciones que deben interactuar, no fue tarea sencilla. Para una enumeración detallada y explicación de estos módulos se puede consultar la primera subsección de este apartado.

Una vez establecido el entorno de desarrollo se comenzó a escribir el código. El primer prototipo sirvió de toma de contacto con el sistema Android, además de para cerciorar que el entorno de desarrollo estaba correctamente configurado (por ejemplo, que se recibían correctamente los datos del GPS).

Basado en el prototipo mencionado en el párrafo anterior, se creó una ampliación de éste. Los requisitos era que se tomara una fotografía, y sobrescribir sobre ésta un texto. Esto ya entrañaba muchísima más complejidad, puesto que al estar desarrollando

todo el código sobre el emulador (el cual carece de cámara) era difícil saber si el programa funcionaría correctamente en un terminal físico.

Una vez que se tomó contacto con el entorno de desarrollo parcial (dado que no se había introducido todavía los datos de GPS y brújula) y con la programación en Android, se pasó a la evolución del prototipo para que manejara datos de orientación y posicionamiento. En un primer acercamiento sólo se pretendía mostrar ambos datos, capturados al pulsar el obturador, impresos por pantalla. Para esta parte en particular se usaron dos prototipos desechables que hacían las funciones de recoger e imprimir los datos de orientación y posición. Además del reto en cuanto a programación, había otro inherente: cómo introducir datos de GPS y brújula de prueba.

El siguiente paso fue desarrollar un prototipo que cargara y leyera bases de datos XML. Estas bases de datos contenían información sobre puntos de interés, y se debían cargar en memoria para que la aplicación pudiera hacer uso de ella. Puesto que la información debía *parsearse*, entre las muchas opciones que había para hacerlo se utilizó un XML PullParser, dada su sencillez y su modularidad.

Una vez se tenía un prototipo que podía realizar fotos, sobreimprimir texto en ellas, y adquirir datos de posición y orientación se pasó a la integración con la base de datos. Una vez validado el prototipo de lectura de bases de datos, se introdujeron las funciones de adquisición de localización y orientación, de manera que si el "usuario" estaba cerca de algún punto de interés al pulsar un botón, se mostraba ésos puntos de interés cercanos. La integración de la interfaz entre la base de datos, la posición y la orientación del usuario fue una de las labores más complicadas.

Una vez se tuvo un prototipo que integraba todas las funciones desarrolladas (módulos de adquisición de datos, carga de base de datos y cámara), sólo faltaba sobreimprimir los resultados en la foto resultante.

En un primer momento, la aplicación se pensó para usar en un radio muy pequeño, por lo que sólo se debía mostrar un punto de interés por cada foto tomada por el usuario (en caso de que hubiese alguno). No se desarrollo ningún prototipo ya que, a medida que la idea se iba desarrollando, se llegó a la conclusión de que limitar los puntos de interés que se muestran a uno era una solución pobre además de introducir problemas de decisión (cual mostrar y cual no en caso de duda).

Finalmente, se optó por poder mostrar varios puntos de interés al usuario. Para ello hacía falta diseñar e implementar un algoritmo que calculase la orientación de los puntos de interés respecto a la orientación del terminal. El algoritmo usado para hallar el ángulo en el que se encuentran los puntos de interés se muestran a continuación en la Ecuación 2.

Donde γ es el ángulo en el que se encuentra el punto de interés respecto del ángulo 0º. Haciendo el siguiente cálculo se halla tal ángulo:

$$\gamma = arctan\left(\frac{\Delta lat}{\Delta lon}\right) \qquad \Delta lat = lat_{pi} - lat_{act}$$
$$\Delta lon = lon_{pi} - lon_{act}$$

Ecuación 2: Algoritmo de obtención de ángulo de muestra

Una vez obtenido el ángulo γ, se transforma para que esté referido respecto al norte; esto se hace restándole a 90º el ángulo γ. Cuando ya se tiene el ángulo respecto al norte, se compara con la orientación actual (del terminal) y si está dentro de los límites de 80º (40 a la derecha y 40 a la izquierda), el punto deberá mostrarse, en caso contrario se considera que el punto está cerca pero no se encuentra en un ángulo visible.

Anterior a este algoritmo, se había desarrollado uno mucho más complejo y menos eficiente e intuitivo, a continuación se detalla su fucnionamiento. Los 360º en los que el usuario puede orientar su teléfono, están separados en porciones de 50º, cada una de ellas tiene asignada una plantilla rectangular. Si el usuario toma una foto orientado a un ángulo, este ángulo tendrá su plantilla rectangular asignada y los puntos de interés que estén dentro de esta plantilla rectangular serán los que aparezcan en la pantalla. En la Figura 39 se muestran las porciones (ángulos) junto a sus plantillas rectangulares asignadas. Obviamente el método es bastante poco fiel a la realidad, además de dar muchos problemas en los ángulos fronteras (cerca de los límites entre las porciones). Una vez desechado el algoritmo, se utilizó el implementado en la aplicación, descrito en el párrafo anterior, en el que se muestran los puntos de interés en 80º de ángulo de visión, de manera continua y sin plantillas.

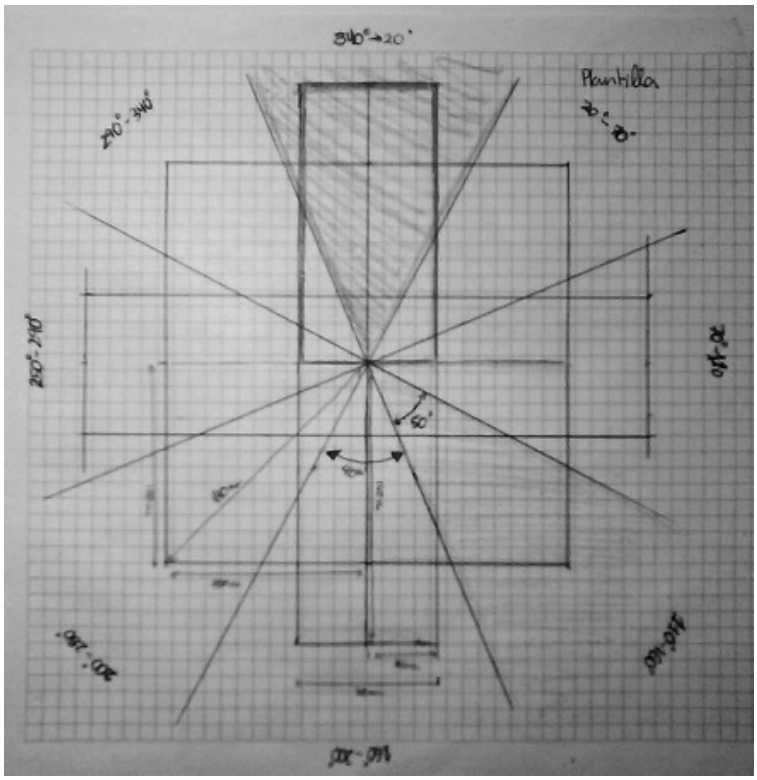


Ilustración 39: Plantilla utilizada en un prototipo desechado

Parte de este último prototipo integraba una nueva funcionalidad: obtener la localización además de por GPS por información de redes inalámbricas, ya sean redes WiFi o telefónicas. De esta manera, aunque sin tener tanta precisión, se conseguía un fix más rápidamente.

Lo único que faltaba por hacer después era: una vez que el usuario ya tenía sobreimpresos todos los puntos de interés en la foto, mostrarle las distintas opciones que tenía por cada punto. Se decidió que la manera más amigable para el usuario era que pulsase en la pantalla y le apareciesen todos los puntos disponibles en una lista. Manteniendo pulsado un punto de interés de la lista aparecería un popup que mostrase todas las opciones de ese punto.

Para obtener información sobre los distintos puntos de interés se tuvo que integrar toda la gestión de redes de datos, ya fueran telefónicas o WiFi. Además también se incluyó un campo opcional en los puntos de interés contenidos en las bases de datos. Estos campos opcionales eran dos <URL> y/o <INFO>, el primer campo contiene una URL que apunta a una fotografía (que se muestra al usuario) y el segundo contiene información adicional en modo texto. Para acceder a esta información adicional se decidió que lo mejor era crear un botón adicional para los puntos de interés que contuvieran esta información. Este botón adicional aparecería en el menú emergente mencionado en el párrafo anterior, y tendría como nombre “Información adicional”.

VERIFICACIÓN Y VALIDACIÓN

Una vez que todo estaba implementado, se puso en marcha un proceso de verificación y de validación final, para comprobar que el programa cumplía tal y como se especificaba en los requisitos.

Debido a la metodología utilizada, de constante búsqueda de nuevos requisitos, la validación fue un proceso continuo durante el desarrollo hasta llegar a la validación final. El proceso de verificación no se puso en marcha hasta que la aplicación estuvo desarrollada y validada.

Por un lado, un conjunto de pruebas se desarrollaron para comprobar que las nuevas funcionalidades fueran implementadas de manera correcta. Por otro lado, las nuevas funcionalidades eran validadas por el departamento para cerciorarse de que se cubrían las necesidades para las que estaban diseñadas.

1. Verificación

Un conjunto de pruebas son procesos que permiten verificar y comprobar la calidad de un programa.

En este proyecto en particular, casi todo el testing fue realizado en un ordenador portátil y no en un terminal Android, puesto que no estaba disponible, además de ser bastante más dificultoso hacerlo. Aún así, algunos test fueron realizados en terminales una vez la aplicación estaba desarrollada casi completamente.

El método de testing utilizado en este proyecto ha sido el de las cajas blancas, este método nos permite cerciorarse de que el software hace bien lo que hace. Se trata de hacer pruebas individuales a los métodos o funcionalidades, de manera que el procesado de datos es estudiado y verificado de manera rigurosa. Estos tests son adecuados puesto que se conoce el código minuciosamente y se puede inspeccionar.

Para realizar estas pruebas se utiliza el framework de tests de Android, el cual se basa en JUnit. Se trata de un proceso que corre en el terminal emulado y que consta de dos hilos, uno el de la aplicación a probar y en el otro los casos de test (test cases). Para ello hay que configurar el entorno de la misma manera que en el desarrollo de la aplicación (adquisición de datos GPS, brújula, Internet) todo ello dependiendo del caso de test que se esté probando.

Algunas pruebas más se realizaron, para comprobar los requisitos no funcionales y algún otro aspecto. Por supuesto, al realizar estos tests, surgieron algunos errores en el código. Algunos ejemplos se pueden encontrar más adelante, en el apartado “Ejemplo de errores encontrados”.

Test de Junit

Las pruebas han sido desarrolladas con la ayuda de Junit. Las pruebas implementadas han sido realizadas usando la siguiente metodología:

- Identificar el método o funcionalidad a probar.
- Determinar la entrada (válida o no válida) y la salida deseada.
- Si una nueva funcionalidad o método es desarrollado, se creará un nuevo

test para él.

- Todos los test fueron realizados en modo desarrollo y producción.

Los test de Junit que se han creado están especificados en el Anexo D, junto con su descripción y sus resultados.

Test no funcionales

La siguiente tabla detalla todo el conjunto de pruebas realizadas para verificar los requisitos no funcionales, junto con su resultado:

Nombre y descripción	**Resultado**
Compatibilidad con otros sistemas Android	
Android 2.1	OK
Android 2.2 y mayores	OK
Compatibilidad con distintos terminales	
Samsung Galaxy (GT-i5800)	OK
Rendimiento	
Requisitos mínimos	OK
Seguridad	
Solicitud de permisos	OK
Robustez a usos indebidos	OK
Usabilidad	
Sencillez de la interfaz de usuario	OK
Ventanas autoexplicativas	OK
Documentación y mantenimiento	
Documentación del proceso de desarrollo	OK
Manual de usuario	OK
Manual de referencia para desarrollador	OK
Documentación del código	OK
Documentación sobre mantenimiento y desarrollo futuro	OK

Otros tests

Alguna prueba más se realizó mientras que la aplicación se estaba desarrollando. Estas pruebas no han sido documentadas por lo que han sido dejadas de lado.

Se realizaron pruebas para cargar bases de datos al programa. En la clase Introduccion se comprueba que todas las bases de datos se cargan correctamente, y si hay algún tipo de error, se utiliza la información cargada correctamente. Cada vez que se carga una base de datos se comprueba su integridad.

Otra práctica seguida durante el desarrollo era la generación de mensajes emergentes que avisan de que algo ha ocurrido, ayudando a detectar dónde estaba el error.

Ejemplos de errores encontrados

A medida que se iban realizando todas las pruebas mencionadas, errores fueron apareciendo. Para obtener una idea clara de en qué medida ayuda el testing a encontrar problemas y solucionarlos, en esta sección se detallan un error de cada tipo.

Dentro de la categoría de errores de Junit, un error importante se detectó en el módulo de adquisición de datos de GPS. Durante el proceso de desarrollo, se utiliza un generador de coordenadas GPS, en el que se puede introducir cualquier valor y éste se encarga de mandárselo al terminal Android simulado. Si se introducía un valor incorrecto, por ejemplo unas coordenadas que no pueden existir (latitudes fuera del rango [-90,90]), la aplicación aceptaba los datos y seguía el proceso, lo que podría causar problemas mayores en el futuro, ya que no se puede confiar en la integridad de los datos que mandan los satélites GPS.

En cuanto a los requisitos no funcionales, en el ámbito del rendimiento se quería llegar a, dentro de lo que cabe, unos requisitos recomendados para el terminal mínimos. Para conseguirlo, la carga en memoria debe ser el mínimo indispensable. Al comenzar el desarrollo, se cargaban todas las bases de datos al arrancar en memoria. Para aliviar, de alguna manera la carga en memoria, se introdujo la nueva especificación, que permitía al usuario elegir qué base de datos quiere usar.

Por último, era indispensable cerciorarse de la integridad de la información almacenada en las bases de datos. Para ello se desarrolló un método que no sólo cargaba las bases de datos en memoria, si no que comprobaba que estaban bien formadas, un requisito indispensable para el correcto funcionamiento del software.

2. Validación

El proceso de validación fue realizado en conjunto con el profesor del Departamento de Ingeniería Telemática Mario Muñoz Organero, principalmente mediante reuniones en la Universidad Carlos III. Algunos cambios o comentarios fueron realizados vía correo electrónico. También otros actores entraron a participar en el proceso de validación, sobretodo en el entorno del desarrollador, estas participaciones venían en forma de sugerencias, sobretodo en cuanto a la interfaz gráfica.

El medio de demostración principal ha sido la presentación de prototipos, de manera que podía ser validado directamente por el usuario. Estos prototipos, en una primera fase, fueron presentados mediante terminales emulados de Android. Una vez la aplicación era estable y había seguridad de que el software no era malicioso, se empezó a probar en terminales físicos. Este proceso de validación ha sido constante y realizado tanto por el Departamento como por personas ajenas al proyecto, que han dado una visión neutra sobre la aplicación.

La validación final de la aplicación se realizó algo después de la puesta en marcha y la instalación en un terminal físico Android, puesto que muchos de los requisitos se validaron directamente probando la aplicación en el terminal. En el Anexo L se adjunta el documento de Pruebas de Aceptación de Fábrica (Factory Acceptance Test), donde el tutor del proyecto y el desarrollador (el estudiante) validan que el producto desarrollado se ha probado de manera correcta mientras se desarrollaba y que cubre los requisitos que demandaban los usuarios. Firmándolo, el usuario tiene la garantía de que el software funciona correctamente, de la misma manera que el desarrollador tiene un documento que acredita que el usuario acepta el software.

Además de la validación hecha por el tutor del proyecto, se han hecho validaciones externas de manera rutinaria, realizadas por varios interesados en el proyecto. Su trabajo ha sido el de probar la aplicación a nivel de usuario y aportar retroalimentación en forma de sugerencias e ideas de nuevas funcionalidades.

Debido a que el sistema operativo Android puede ejecutarse en numerosos terminales de diferentes fabricantes, probar la aplicación para todos ellos hubiera sido imposible. Se han hecho pruebas en terminales Samsung y HTC, puesto que son unos de los dos fabricantes que más cuota de mercado tienen en cuanto a terminales Android. También ha sido una cuestión práctica probarlo en terminales Samsung, puesto que de éstos es sabido que el sistema GPS no funciona del todo bien (para conseguir un fix puede llegar a necesitar la asistencia de una red WiFi), por lo que si la aplicación funcionaba correctamente en uno de estos terminales, no habría más problema en cualquiera de los demás fabricantes.

PUESTA EN MARCHA

Después de que la implementación del software se completase y fuera verificado y validado de manera íntegra, era el momento de la puesta en marcha. Hasta entonces se habían instalado prototipos en terminales físicos Android, pero siempre en modo debug, es decir, en pruebas. Una vez completada esa fase era el momento de la puesta en marcha definitiva, de igual manera que lo haría un usuario. Este proceso incluye la definición de requisitos de los terminales que vayan a usar esta aplicación, cómo instalarla además de un manual para el usuario así como uno para los posibles futuros desarrolladores.

1. Requisitos

En esta sección se van a detallar los requisitos necesarios para instalar y ejecutar correctamente la aplicación desarrollada. Dado que se trata de un programa para dispositivos móviles Android y debido a la variedad de modelos y características que éstos tienen, definir los requisitos, sobretodo de memoria y capacidad del procesador, no es tarea fácil. Para ello habría que probar en multitud de terminales y comprobar el correcto funcionamiento del software. Puesto que esto es inviable, a continuación mencionan aquí las capacidades mínimas que debe tener un terminal para poder arrancar Android, lo que debería dar al menos, la posibilidad de instalar esta aplicación sin garantizar un funcionamiento fluido. Más adelante se determinarán las funcionalidades que deberá proporcionar el terminal para ejecutar la aplicación y por último las necesidades de almacenamiento.

Como se ha mencionado en el párrafo anterior, determinar los requisitos en cuanto a procesador y memoria para la aplicación es una tarea ardua. Se sabe que para ejecutar Android, los requisitos mínimos para los móviles son memorias RAM de 32 MB, 32 MB de memoria flash y un procesador de 200 Mhz, según Andy Rubin, director de plataformas móviles de Google. Pero, previsiblemente se espera que los requisitos para una aplicación sean más altos, ya que un móvil basado en Linux requiere una memoria de 128x128 y un procesador más potente que 200MHz. Además esta aplicación necesita hacer uso de hardware adicional, como el GPS, la cámara o los datos. Si nos atenemos a estas especificaciones, un procesador *ARM9* sería suficiente para correr *Android* en un dispositivo móvil real (Pérez, 2009). Además el terminal móvil debe de disponer del sistema operativo Android 2.1 (también conocida como Eclair) o superior.

La instalación de la aplicación ocupará 833KB que corresponde a todo el código y lógica del programa, a lo que hay que añadirle lo que ocupe la base de datos con los puntos de interés.

En cuanto a las capacidades funcionales que deberá tener el teléfono, éste debe disponer de cámara así como sistema de localización global (GPS) y brújula. En caso de no tener GPS, se podrá usar el posicionamiento dado por la red de telefonía móvil, aunque ésta será mucho menos precisa. Si el usuario quiere acceder a la información adicional de los puntos de interés, deberá tener conexión a internet. Esta conexión puede realizarse mediante red de datos inalámbrica (WiFi) o por red de datos telefónica. La necesidad de un adaptador WiFi no es imprescindible, pero si muy recomendable puesto que muchos teléfonos Android de baja gama tienen problemas con el GPS y no consiguen un fix sin estar conectados a algún punto de acceso inalámbrico.

En cuanto a las necesidades de almacenamiento la aplicación, con sus bases de datos estándar, ocupa 412KB, de éstos 96 son de la aplicación 108 de datos y otros 208 de caché. Hay un variable en cuanto a la necesidad de almacenamiento y esto viene dado por la posibilidad de incluir bases de datos externas.

El tamaño de la base de datos dependerá de la cantidad de puntos de interés que contenga y, sobretodo de si contienen o no los campos opcionales *info* y *URL*, además del tamaño de éstos. Como aproximación para calcular cuánto ocupará la base de datos se puede decir que la base de datos vacía ocupa 23 Bytes y cada entrada (punto de interés) ocupará 115 Bytes. Por lo tanto, el paquete básico, con 29 puntos de interés, aproximadamente ocupará 3358 Bytes ((29x115)+23), lo que no dista mucho de los 3.659 Bytes que realmente ocupa. Hay que tener en cuenta que el tamaño que realmente ocupará en el disco depende del sistema de ficheros que utilice la tarjeta SD, que determinará el tamaño mínimo del archivo (así como los pasos en el que éste crecerá).

2. Instalación

Antes de poder instalar la aplicación hay que crear un paquete APK (Android package), aunque que se podría instalar con el Eclipse sin necesidad de hacer el paquete pero para ello habría que disponer del código fuente. Para obtener el paquete apk del proyecto en desarrollo bastó con explorar la carpeta bin del *workspace* del proyecto en Eclipse. La ruta es la siguiente:

<ruta de workspace>\GuiaInteractiva\bin\GuiaInteractiva.apk

El proceso de instalación de la aplicación a un terminal es relativamente sencillo. Las instrucciones vienen detalladas en el Anexo F. Puesto que lo que se distribuye es un paquete APK, sólo se podrá utilizar alguno de los dos primeros métodos explicados en tal anexo, puesto que el último (con Eclipse) es para aplicaciones en desarrollo y con el código fuente disponible.

3. Documentación

A documentación que se adjunta en las siguientes secciones está dirigida a distintos públicos. El primer documento está dirigido a los usuarios de la aplicación y les guiará a lo largo de la aplicación, así como resolverá los problemas o dudas más comunes. El segundo documento está dirigido a futuros desarrolladores de la aplicación, que quieran modificar el código para añadir nuevas funcionalidades, o con cualquier otro fin. Se ha intentado incluir la información que se considera más relevante, en función de los conocimientos adquiridos en este proyecto. En cada una de las secciones se especifica el anexo en el que se encuentran los manuales.

3.1 Manual de usuario

Como puede comprobarse en la definición de requisitos, la aplicación fue desarrollada para ser lo más sencilla y auto explicativa posible. A pesar de eso, se ha editado un documento para solventar cualquier tipo de duda que pudiese tener un usuario. El manual de usuario se encuentra en el Anexo J.

Este manual comprende las dudas o instrucciones más usuales y generales. Para un análisis en detalle, lo mejor es referirse a esta memoria, puesto que todo está mucho más precisado.

Se entiende que un usuario corriente, tendrá un paquete APK, conseguido por cualquier tipo de vía, que deberá instalar. Una vez instalada la aplicación, procederá a ejecutarla. Además también se explicará como desinstalar la aplicación, así como introducir bases de datos externas nuevas.

3.2 Manual de referencia

El manual de referencia se encuentra en el Anexo K. Describe de manera general el entorno de desarrollo, así como la manera de establecerlo para que cualquier programador pueda seguir con el desarrollo. Además, se explica por encima el código desarrollado, para que el desarrollador obtenga una visión global del proyecto.

Adicionalmente, se explica cómo realizar ciertas actividades relativamente frecuentes durante el desarrollo, como son: montar una tarjeta SD o crear un terminal Android emulado.

Para aclarar cualquier tipo de duda más específica lo más recomendable es consultar este documento.

MANTENIMIENTO

En esta sección se detallará el mantenimiento que debe de hacerse a la aplicación así como futuras mejoras que se harán del software.

1. Mantenimiento

El mantenimiento de la aplicación es más bien sencillo. Esto se debe, sobretodo al hecho de que las nuevas versiones de Android que van apareciendo son retro compatibles, por lo que, aparentemente no habrá necesidad de modificar o adaptar la aplicación en un futuro. Aún así, se deberá hacer un seguimiento para cerciorar que esto es cierto.

Por otro lado queda la actualización de las bases de datos. Aunque es poco probable, es posible que se de algún cambio en las localizaciones de los puntos de interés. Puede suceder que trasladen un museo o que reubiquen un monumento.

También se deberá mantener un canal de comunicación con los usuarios para resolver todo tipo de dudas o problemas que surjan. Debido a que los terminales Android son fabricados por numerosas compañías, es posible que se dé algún tipo de conflicto o problema en algún terminal en concreto. Este canal será el correo electrónico.

2. Ampliaciones futuras

En esta sección se detallarán las ampliaciones que se podrían hacer para este proyecto. Se trata de una aproximación, que puede servir de guía para los posibles desarrolladores. No se puede estimar el tiempo que llevaría implementar todas estas mejoras, puesto que dependería de la experiencia del programador, además de que hay muchas cuestiones que deben estudiarse y analizarse previamente para ver si son factibles.

Las prioridades en cuanto a mejoras y ampliaciones en este proyecto están centradas en la optimización del uso de la batería, popularizar la aplicación y mejorar la robustez de la aplicación.

Hay múltiples posibles futuros pasos para este proyecto. Al haber sentado unas bases, las opciones de futuros desarrollos son varias. Se recomienda que esta sección se lea conjuntamente con el roadmap, que se encuentra en el Anexo I.

La primera ampliación que se podría hacer es portar el código a otras máquinas. Actualmente, aparte del sistema operativo Android, también son muy populares las BlackBerrys, con BlackBerry OS (desarrollado por RIM) y los iPhones con el sistema operativo iOS. Extender la aplicación a otras plataformas sería un paso muy positivo, y abriría las puertas a un mercado mucho mayor, con más usuarios finales. Lo óptimo sería que todas usasen las mismas bases de datos, de manera que no hubiese diferencias entre el uso de la aplicación en una plataforma o en otra. Esta ampliación no se encuentra en el roadmap recomendado puesto que es una modificación tan sumamente grande que equivaldría a desarrollar otro proyecto de la misma magnitud que éste, por lo que no se consideraría una ampliación, si no un proyecto paralelo.

Tal cual está implementada la aplicación, actualmente, el usuario debe pasar por

la ventana de instrucciones cada vez que utiliza la aplicación. Esto es debido a que en ésta Activity, se encuentra y ejecuta toda la lógica de lectura de bases de datos. Dado lo ajustada que está la planificación, se optará por dejarlo como posible futuro desarrollo.

Mejorar la gestión de batería. Un punto crítico de la aplicación es el consumo de batería que provoca. Esto se da sobretodo por las comunicaciones que se mantienen en ejecución: el GPS y las redes de datos. La utilización del GPS consume mucha batería, pero por ahora no existe ninguna solución de compromiso ya que la geolocalización por redes telefónicas no tiene la precisión deseada(al igual que las realizadas por redes WiFi).

Ampliar y crear nuevas bases de datos. Una manera podría ser creando un traductor de bases de datos. Con esta herramienta se podrían portar los datos de otras bases de datos al formato de esta aplicación. De esta manera se conseguirían la cantidad de datos necesarios para atraer más usuarios, aumentando así las probabilidades de que éstos se animen a crear sus propias bases de datos o actualizar alguna existente. Otra manera podría ser integrándolo con Google Geocoding API. También se podría modificar la manera de obtener datos: sin utilizar bases de datos, accediendo directamente a los datos de Google Maps por internet mediante el mencionado API.

Una mejora en cuanto a la implementación del algoritmo usado para mostrar gráficamente los puntos de interés cercanos al usuario sería que el ángulo frontera fuera variable. Ésto viene motivado por que, al estar cerca de un punto de interés, el ángulo de podría ser más pequeño, puesto que el usuario puede determinar la orientación con mayor precisión que uno que está lejos.

Continuar con el desarrollo de la aplicación, añadiendo nuevas funciones, sería muy positivo. Actualmente hay muchas aplicaciones similares a la de este proyecto, con lo que añadir nuevas funcionalidades aportaría un valor añadido, que el usuario sin duda valoraría. Una de las mejoras más factibles sería que los propios usuarios pudieran añadir sus propios puntos de interés. Esto haría el proyecto mucho más social, y con un coste de mantenimiento y generación de bases de datos mucho menor. La promoción del proyecto sería un factor esencial para que esto sea factible.

Presentar el proyecto a algún organismo público que tenga interés en impulsar el turismo de su región. Se podría crear una base de datos para un lugar concreto, y en varios idiomas. Conseguir que la administración pública se involucre, simplemente habilitando la opción de descarga de la aplicación en su página web, significaría un impulso enorme al proyecto.

Instalar un servidor de provisión de puntos de interés en varios idiomas y de varias ciudades. De esta manera, el usuario potencial de la aplicación, entraría en la página y se descargaría el paquete con los puntos de interés de la ciudad que va a visitar y en su idioma. La promoción del proyecto también sería primordial para que esta situación se dé.

Introducir publicidad en la aplicación. En los tiempos de espera que el usuario sufre, por ejemplo: al esperar la señal del GPS, se puede introducir publicidad. Lo interesante es que ésta publicidad podría ser dependiente del contexto, es decir, que tuviese en cuenta donde se encuentra el usuario. Se podrían desarrollar dos versiones, una de pago sin publicidad y otra gratuita con anuncios. Los beneficios generados se reinvertirían para desarrollar más mejoras y se destinarían al mantenimiento de la aplicación. El objetivo no debería ser obtener beneficios con la aplicación, si no paliar los costes producidos por el desarrollo de las mejoras o para reinvertir con miras a

mejorar la proyección de la aplicación.

Por último y no menos importante, una mejora que podría abrir mucho el mercado de la aplicación sería la posibilidad de permitir elegir el idioma de la aplicación, durante la instalación o en ejecución. Se podrían incluir bases de datos multi-idioma y según el usuario seleccionase un idioma u otro, se cargaría la información correspondiente. Esto también aumentaría más las posibilidades de que los usuarios creasen sus propias bases de datos.

PLANIFICACIÓN

En esta sección se muestran dos planificaciones. La primera es la teórica, que se planteó al principio del proceso de desarrollo de la aplicación. La segunda es la planificación real que se ha llevado a cabo. En el punto cuatro de esta sección se hace una comparación entre ambas planificaciones.

1. Aspectos generales

Antes de comenzar especificando la planificación original y la real, unos conceptos generales serán explicados. De esta manera aseguramos que la información que le sigue sean comprendidas correctamente.

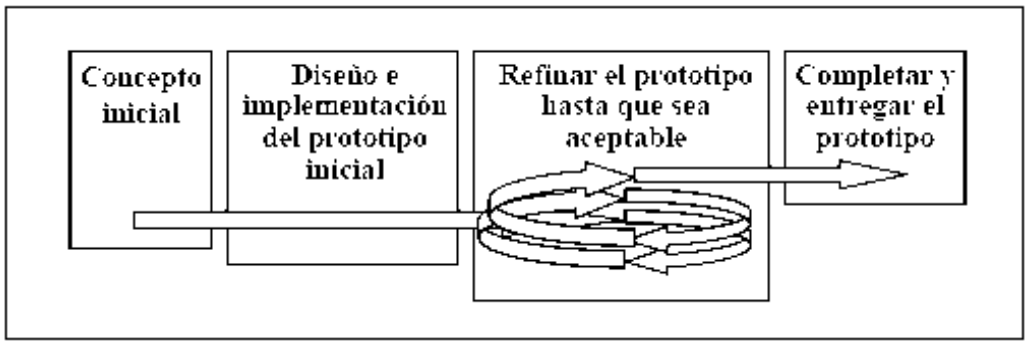


Ilustración 40: Metodología utilizada en el proyecto

Como ha sido explicado en la primera sección de esta memoria, la metodología determina en gran medida la planificación que se va a llevar a cabo. En este caso, dado que el desarrollo de prototipos era constante, se han dividido en cuatro módulos que se explicarán más adelante.

Basada en esta metodología se hizo una planificación inicial.

Todos los diagramas de Gantt están basados en días de 4 horas laborables, puesto que era el tiempo real del que se disponía durante el desarrollo del proyecto.

2. Planificación inicial

La planificación inicial era una aproximación a lo que debería ser el proceso de desarrollo, incluyendo hitos y fechas de entrega. Puesto que se trata de una aproximación, no se consideraron tiempos de vacaciones o que surgiera cualquier tipo de imprevisto, de manera que se esperaba que hubiese retrasos respecto a la planificación original.

La siguiente tabla da una idea de la planificación mencionada. El diagrama de Gantt asociado a esta tabla se encuentra en el Anexo B.

Los ocho prototipos que se planteaban en un principio para el desarrollo de la aplicación son los siguientes y están en el diagrama de Gantt nombrados de la siguiente manera:

- Prototipo 1: Se trata de una toma de contacto con la interfaz gráfica. Se crea un prototipo que sobreimprime en un archivo de fotografía un texto.

- Prototipo 2: Módulo de cámara. El prototipo debe ser capaz de tomar una fotografía y sobreimprimir un texto sobre ella.
- Prototipo 3: Un prototipo que implementa la adquisición de datos de localización GPS y de orientación.
- Prototipo 4: Un prototipo que gestiona la lectura y acceso a la información de las bases de datos XML. Se cargan la información de los puntos de interés en memoria.
- Prototipo 1 mejorado: Una evolución del prototipo 1. En este caso, se sobreimprimirán los puntos de interés cercanos en función de la orientación y localización.
- Prototipo 2 mejorado: Desarrollado a partir del prototipo 3, éste implementa localización por redes inalámbricas (tanto telefónicas como WiFi). Además gestiona de mejor manera la adquisición de datos GPS.
- Prototipo 3 mejorado: Basado en el prototipo 4, integra la gestión de la conexión a internet así como el acceso a páginas y servicios web con las bases de datos XML.
- Prototipo 4 mejorado: Integración completa de todos los módulos, para conformar la aplicación final.

Cuando se realizó esta planificación había una variable a tener en cuenta, la fecha de presentación del proyecto. Esta fecha iba a ser, previsiblemente a principios del año 2011, pero no se sabría a ciencia cierta hasta unas semanas antes.

Los prototipos nombrados como mejorados son prototipos que serían validados por el cliente. Puesto que la metodología usada era de constante validación y búsqueda de requisitos, se desarrollaron prototipos para que el cliente obtuviera una visión general de la funcionalidad y aportara su opinión y los cambios que debían realizarse.

A continuación se adjunta una tabla que resume la planificación planeada originalmente. Para una mejor compresión, referirse al diagrama de Gantt que se encuentra en el Anexo B.

Tarea	**Días**	**Fecha de entrega**
Introducción		
Primera reunión	1	02/09/10
Planificación, índice de memoria	9	15/09/10
Primera aproximación, objetivos e iniciación	10	29/09/10
Análisis completo		
Primer análisis de requisitos	20	27/10/10
Diseño básico		
Definición de clases	13	13/11/10
Modelo de datos	10	26/11/10
Borrador de memoria	2	30/11/10

Implementación y testing		
Primer prototipo	5	08/12/10
Segundo prototipo	5	15/12/10
Tercer prototipo	5	22/12/10
Cuarto prototipo	5	29/12/10
Primer prototipo mejorado	5	05/01/11
Segundo prototipo mejorado	5	12/01/11
Tercer prototipo mejorado	5	19/01/11
Cuarto prototipo mejorado	5	25/01/11
Puesta en marcha y validación final	5	02/02/11
Memoria final	10	15/02/11
Presentación del proyecto	5	21/02/11

3. Planificación real

En esta sección se tratará la planificación que realmente se siguió. A pesar de que durante el proyecto se intentó seguir la planificación inicial, ésta tuvo que ser adaptada a las circunstancias. Los cambios sucedidos pueden apreciarse en el diagrama de Gantt del Anexo C.

Los prototipos que se mencionan en el diagrama de Gantt se detallan aquí:

- Prototipo 1: Se trata de una toma de contacto con la interfaz gráfica. Se crea un prototipo que sobreimprime en un archivo de fotografía un texto.
- Prototipo 2: Es una ampliación del primer prototipo, en el que, en lugar de crear un fondo de pantalla a partir de un archivo fotográfico, lo hace con una fotografía recién tomada por la cámara del teléfono. También incluye crear un preview de la cámara.
- Prototipo 3: Un nuevo prototipo que implementa la adquisición de datos de localización GPS y de orientación. Se trata de actualizar variables para que después puedan ser leídas por el software y actuar en consecuencia.
- Prototipo 4: Un prototipo que gestiona la lectura y acceso a la información de las bases de datos XML. Se cargan la información de los puntos de interés en memoria. Se integran todas las demás funcionalidades (cámara, orientación, posición, bases de datos y sobreimpresión de textos) en un sólo prototipo evolutivo.
- Prototipo 1 mejorado: Una evolución del prototipo 4, que integra todas las funcionalidades, pero que sólo mejora en el aspecto de la sobreimpresión del texto. En este caso, se sobreimprimirán los puntos de interés cercanos en función de la orientación y localización.
- Prototipo 2 mejorado: Desarrollado a partir del 1 mejorado, éste implementa localización por redes inalámbricas (tanto telefónicas como WiFi). Además gestiona de mejor manera la adquisición de datos GPS. Se mejora el algoritmo

que determina dónde se sobreimprimirán los puntos de interés cercanos.

- Prototipo 3 mejorado: Basado en el 2 mejorado, incluye gestión de la conexión a internet así como el acceso a páginas y servicios web. Además se incluye un nuevo campo en las bases de datos, el de <INFO> y/o <URL>, que contienen información adicional en modo texto y/o una url que apunta a una fotografía.

- Prototipo 4 mejorado: Se trata de una versión mejorada de la anterior. Se optimiza el código, así como se mejora el aspecto gráfico de la aplicación.

Aunque no se menciona explícitamente, después del desarrollo de cada uno de los prototipos, éstos eran validados por el tutor del proyecto. De esta manera se redefinían los requisitos y se implementaban las funcionalidades de la manera en la que el cliente las quería.

Los prototipos 1 y 2 en realidad eran uno, solo que en distintas fases de desarrollo, es decir el prototipo 2 era una ampliación del primero. En cuanto al prototipo 3, éste si que era autónomo, mientras que el 4, al integrar todos los módulos, se convirtió en el prototipo evolutivo que pasaría a ser la aplicación final. Los prototipos 1, 2, 3 y 4 mejorados son también el prototipo 4 evolucionado. Es decir el prototipo 4 mejorado es la aplicación final.

En cuanto a la presentación del proyecto, a pesar de que en el diagrama de Gantt se establece en el día 5 de Agosto, no es la fecha real en la que se realizó. Puesto que durante la redacción de esta memoria y el desarrollo del proyecto no se sabía la fecha que sería adjudicada para la presentación del proyecto, se propuso representarla en el diagrama justo después del término de todas las tareas, aunque no fuese la fechar real.

A continuación se relatan con detalle los distintos momentos del desarrollo de esta aplicación, sobretodo los motivos de los altos en el proceso que se pueden observar en el diagrama de Gantt.

El tiempo que llevó configurar todo el entorno de desarrollo, buscar las herramientas que se iban a usar, además de aprender a usarlas llevó algo más de tiempo del esperado, puesto que el desarrollador nunca había hecho un programa en Android y la incertidumbre era total.

Dado que el desarrollador de la aplicación se encontraba cursando asignaturas en la Universidad Carlos III, el proyecto quedó parado a principios del año lectivo. El alto en el proyecto también vino motivado por que el desarrollador se encontraba en varios procesos de selección. Esta parada duró aproximadamente mes y medio, hasta principios de Noviembre.

A mediados de Noviembre de 2010, el desarrollador comenzó a trabajar en una beca, para Alcatel-Lucent. El trabajo era de media jornada y duró hasta el final del proyecto, por lo que el desarrollo se vio bastante perjudicado en cuanto a planificación. A todo esto hay que aclarar que el desarrollador también estaba enrolado en la universidad recibiendo clases.

A finales del año 2010 el desarrollador se encontraba estudiando para los exámenes finales de las asignaturas cuatrimestrales que cursaba. Además, este hecho se solapó con las vacaciones de navidades.

Uno de los mayores retrasos en la parte inicial del proyecto fue que los requisitos del prototipo no estaban claramente definidos, por lo que el desarrollador un prototipo que no los cumplía. En un primer momento se desarrolló un prototipo que

tomaba una foto, la mostraba y, cuando el usuario pulsaba la pantalla, aparecían todos los puntos de interés cercanos en forma de lista. Esto hacía muy poco intuitiva la aplicación, hasta llegar al punto de degradar la experiencia del usuario tanto que podía dejar de usarla. Se rechazó el prototipo y se establecieron nuevos requisitos más definidos y claros. Estos eran: los puntos de interés cercanos deben mostrarse sobreimpresos en la fotografía tomada y alineados con su posición (en el mismo ángulo).

Durante el principio de año, el proyecto se paralizó debido a las vacaciones de semana santa. Éstas fueron aprovechadas por el desarrollador para adelantar trabajo de la universidad, puesto que se acercaban los exámenes finales del curso.

Otro cambio importante que hubo en la especificación fue la forma en la que se calculaban los puntos de interés visibles. Por la inexperiencia del desarrollador, se empezó utilizando un algoritmo que era, además de excesivamente complejo, poco eficiente y realista. Volver a diseñar y desarrollar otro algoritmo nuevo llevó un tiempo considerable. Para conocer los dos algoritmos utilizados se puede consultar el apartado “Proceso de desarrollo” de la sección de Implementación.

A partir de la aceptación del prototipo con el nuevo algoritmo, se empezó a trabajar en vistas a conseguir un prototipo de aplicación que implementase la realidad aumentada de manera completa. Esta aplicación debiera implementar el mismo algoritmo que el que utiliza la aplicación final, pero utilizando video (en lugar de fotografías) para mostrar los puntos de interés cercano. Es decir, sería hacer los mismos cálculos que se realizan cuando el usuario aprieta el obturador pero, al menos 16 veces por segundo (para dar sensación de fluidez). Esta tarea consumió bastante tiempo y además se llegó a la conclusión de que era inviable que una sola persona y con el poco tiempo disponible desarrollase una aplicación con tales características.

Durante el mes de Mayo el desarrollador se encontraba estudiando y realizando los exámenes finales de la universidad. Además coincidió con un pico de trabajo en su beca en la empresa. Esto repercutió en un paro total en el desarrollo del proyecto.

En la última reunión para definir la especificación final de la aplicación, que tuvo lugar en Junio, se introdujeron dos funcionalidades adicionales, las cuales tuvieron su repercusión en cuanto a la fecha del final del proyecto. Una de las dos funcionales era permitir que el usuario pudiera seleccionar el radio en el que quisiera que aparecieran los puntos de interés. La segunda funcionalidad fue un poco más compleja de implementar, se trataba de introducir dos campos opcionales a los puntos de interés, uno información en forma de texto y el otro una URL apuntando a una imagen. Además de modificar el método de lectura de bases de datos había que añadir un botón para los puntos de interés que tuvieran este campo, para que el usuario pudiese acceder a esta información.

Durante la segunda quincena de Julio el desarrollador disfrutó de vacaciones, puesto que en la beca que estaba realizando no podía permitírselas en cualquier otra fecha. Esto supuso un retraso de 15 días.

Uno de los mayores retrasos que ha sufrido el proyecto fue en las últimas fases. En la etapa de validación y testing, casi todo el código estaba probado, depurado y puesto en marcha en terminales con resultados satisfactorios. El problema surgió cuando se probó en un terminal Samsung Galaxy S, para el cual el sistema de adquisición de datos GPS implementado no funcionaba correctamente. Es conocido por todos los iniciados que los terminales Samsung (sobretodo los Galaxy S y Galaxy 3) tienen un

servicio GPS que deja bastante que desear, pero dada la gran popularidad de estos móviles en el mercado Android, no se podía dejar sin arreglar. La corrección del código y las subsiguientes pruebas llevaron 1(2) semanas.

A continuación se adjunta una tabla que resume la planificación que realmente se utilizó. Para una mejor compresión, referirse al diagrama de Gantt que se encuentra en el Anexo C.

Tarea	**Días**	**Fecha de entrega**
Introducción		
Primera reunión	1	02/09/10
Planificación, índice de memoria	9	15/09/10
Comienzo de las clases	47	31/10/10
Primera aproximación, objetivos e iniciación	10	13/11/10
Análisis completo		
Primer análisis de requisitos	20	10/12/10
Exámenes finales	31	09/01/11
Diseño básico		
Definición de clases	13	27/01/11
Modelo de datos	10	10/02/11
Borrador de memoria	2	12/02/11
Implementación y testing		
Primer prototipo	10	25/02/11
Vacaciones de semana santa	10	06/03/11
Segundo prototipo	5	14/03/11
Tercer prototipo	5	21/03/11
Cuarto prototipo	5	28/03/11
Primer prototipo mejorado	10	11/04/11
Segundo prototipo mejorado	10	25/04/11
Tercer prototipo mejorado	8	05/05/11
Exámenes fin de curso	42	15/06/11
Cuarto prototipo mejorado	7	23/06/11
Puesta en marcha y validación final	5	01/07/11
Memoria final	10	15/07/11
Vacaciones	15	31/07/11
Presentación del proyecto	5	05/08/11

4. Comparación

Comparando ambas planificaciones se pueden observar varias diferencias.

En primer lugar, dada la metodología utilizada, que contaba con poca planificación y muchos desarrollos de prototipos, de cambios casi completos en la especificación, la planificación inicial fue hecha de manera que se presuponía que, a pesar de añadir más requisitos durante el desarrollo, no iba a haber cambios tan drásticos en éstos como para tener que rehacer funcionalidades sin apenas reutilizar código. A pesar de esto mucho del tiempo se empleó en desarrollar prototipos o conceptos que luego no se validaban y, por lo tanto, era tiempo poco aprovechado. Poco aprovechado y no inútil, puesto que muchos de los conceptos implementados servían de alguna manera para desarrollos posteriores en funcionalidades aparentemente sin vínculo ninguno.

Se observan diferencias, por ejemplo, en los prototipos desarrollados. Mientras que en la planificación original los prototipos estaban perfectamente acotados y diferenciados, en los prototipos reales desarrollados esa diferenciación entre módulos no es tanta. A partir del cuarto prototipo, en el que se integra todo, en la planificación real se observa que las mejoras o nuevas funcionalidades que se implementan no tienen tanto que ver como en la real.

Uno de las principales fuentes de retraso fue la introducción del desarrollador en el mercado laboral, lo que limitaba mucho las horas que se podían dedicar a este proyecto. En concreto, los retrasos más importantes quedan detallados en la subsección anterior, producidos por una carga de trabajo muy alta y por motivos diversos, por lo que el proyecto quedó parado todo ese tiempo.

La consecuencia de estos factores y condiciones ha provocado que la fecha de presentación del proyecto fuera retrasada siete meses.

PRESUPUESTO

El propósito de esta sección es detallar y desglosar los distintos costes asociados a este proyecto. Con este documento se podrá saber de manera específica y detallada los costes del desarrollo de la aplicación.

Para una mejor comprensión y trazabilidad, todos los diferentes costes están agrupados dependiendo de su naturaleza. La última sección está dedicada al mantenimiento. Todos los costes son mensuales, excepto que se especifique lo contrario.

1. Recursos humanos

En esta sección, se detallan todos los costes asociados a salarios. La cantidad de horas se especifican según el rol asumido por el desarrollador dependiendo de las fase en la que se encontraba el proyecto, de esta manera la carga de trabajo queda más detallada.

En la primera tabla se detallan los costes del Ingeniero de Programación. Las tareas asumidas por éste son de alto nivel, además de ser el responsable del proyecto en su conjunto. Entre sus tareas se incluye la planificación, organización y obtener una visión general del software, analizando los requisitos del cliente y diseñando íntegramente el software. En este proyecto, puesto que la metodología utilizada es de continua búsqueda de requisitos, la tarea de diseño y análisis no se realizó sólo al principio, como pasa en muchos proyectos, si no que había cambios constantes.

Concepto	**Coste**
Introducción y planificación	40 horas (5 días)
Análisis integral	80 horas (10 días)
Diseño	80 horas (10 días)
Total de horas	200 horas
Coste por hora	50 € / hora
Total	**10.000,00 €**

En la siguiente tabla se especifican los costes correspondientes al Desarrollador y Tester. Puesto que el diseño ya ha sido llevado a cabo por el Ingeniero de Programación, el Desarrollador debe implementarlo. En este proyecto ha habido sólo un Desarrollador (y Tester a su vez) y varios Testers, pero puesto que se desglosa el coste por horas, no influye en el cálculo.

Concepto	**Coste**
Implementación	160 horas (20 días)
Testing	80 horas (10 días)
Total de horas	200 horas
Coste por horas	30 € / hora
Total	**6.000,00 €**

Poner estas dos tablas juntas y que testing y documentación sumen 10 dias
(documentacion es juntar todo, completar, etc...)

En la siguiente tabla se detallan los costes del Técnico. El Técnico es el encargado de las últimas fases del proyecto. Una vez que el proyecto está implementado y probado, el técnico genera toda la documentación y escribe los distintos manuales (tanto de usuario como de referencia).

Concepto	**Coste**
Documentación	40 horas (5 días)
Manuales	8 horas (1 día)
Total de horas	48 horas
Coste por horas	30 € / hora
Total	**1.440,00 €**

En la siguiente tabla se muestran todos los salarios antes desglosados. A todos los salarios hay que sumarles el I.V.A., puesto que todas las personas involucradas en el proyecto son trabajadores autónomos, este cálculo se hará al final de la sección. El total se muestra en la tabla:

Concepto	**Coste**
Ingeniero de Programación	10.000,00 €
Desarrollador y Tester	6.000,00 €
Técnico	1.440,00 €
Total	**17.440,00 €**

2. Equipamiento

Todos los dispositivos y tecnologías usadas se incluyen en la siguiente tabla. Entre ellos las licencias para el sistema operativo sobre el que se desarrollaba la aplicación (Windows 7 Home Premium), el editor de textos y programas de ofimática para generar la documentación (Microsoft Office Professional 2007), aunque muchos de los programas utilizados era gratuitos (Eclipse, PUTTY y notepad++).

Concepto	**Coste**
Ordenador de desarrollo	1.500,00 €
Terminales Android de pruebas	400,00 €
Impresora	100,00 €
Licencias de software	500,00 €
Total	**2.500,00 €**

3. Consumibles

En la siguiente tabla se especifican los costes de los consumibles utilizados durante el desarrollo del proyecto.

Concepto	Coste Unitario	Cantidad	Total
Folios	0,05 €	200	10,00 €
Cartuchos de tinta	30,00 €	3	90,00 €
Impresión memoria	30,00 €	5	150,00 €
		Total	**250,00 €**

4. Desplazamientos

Durante el desarrollo de la aplicación, varias reuniones se tuvieron que realizar con el cliente. Un total de 5 reuniones tuvieron lugar en Leganés, además del desplazamiento para realizar la presentación. Además, hay que añadir otras 4 reuniones hechas con los distintos Testers, puesto que éstos eran autónomos y no compartían el lugar de trabajo.

El coste del viaje está especificado por trayecto, por lo que para las 5 reuniones hubo que hacer 10 viajes (uno de ida y uno de vuelta), con un coste unitario de 20 €.

Concepto	Coste Unitario	Cantidad	Total
Reunión Cliente	20,00 €	10	200,00 €
Presentación	20,00 €	2	40,00 €
Reunión Testers	10,00 €	8	80,00 €
		Total	**320,00 €**

5. Costes adicionales

Aquí se detallan los costes que no entran en ninguna de las anteriores secciones, pero que están asociados al proyecto y deben quedar registrados.

Concepto	Coste Mensual	Meses	Total
Electricidad	40,00 €	9	360,00 €
Conexión a Internet	20,00 €	9	180,00 €
		Total	**540,00 €**

6. Total

La siguiente tabla detalla todos los costes desglosados anteriormente, a lo que le añade el I.V.A..

Concepto	Coste
Recursos humanos	17.440,00 €
Equipamiento	2.500,00 €
Consumibles	250,00 €
Desplazamientos	320,00 €
Costes adicionales	540,00 €
Subtotal	**21.050,00 €**
I.V.A. (18%)	3.789,00 €
Total	**24.839,00 €**

7. Costes de mantenimiento

El mantenimiento es más bien sencillo, por lo que puede llevarse a cabo por una persona de manera puntual y en periodos semanales. Su tarea principal será la de atender el canal de comunicación establecido con los usuarios de la aplicación vía correo electrónico. Su labor consistirá en resolver dudas y problemas de los usuarios así como recoger sugerencias para una posible ampliación de la aplicación.

Además de esto el encargado del mantenimiento se encargará de comprobar el correcto funcionamiento de la aplicación con las nuevas versiones que aparezcan del sistema operativo Android. La retrocompatibilidad es algo que Android da por sentado, pero habrá que comprobar que es así.

Otra de las tareas será la actualización de las bases de datos. Aunque poco probable, es posible que se de algún cambio en la disposición o en los nombres de los puntos de interés contenidos en las bases de datos.

Puesto que no son tareas con mucha carga de trabajo ni complicadas, se estima que podrá ser realizada por una persona un día por semana a media jornada.

Concepto	Coste Diario	Coste mensual
Técnico	100,00 €	400,00 €
Costes adicionales(consumibles, electricidad)	25,00 €	100,00 €
	Subtotal	500,00 € / mes
	I.V.A. (18%)	90,00 €
	Total	**590,00 €**

CONCLUSIONES

En la siguiente sección se analizará los resultados obtenidos con este proyecto, así como estimaciones del posible futuro de la aplicación. Se empezará discutiendo el impacto que ha podido tener el proyecto y se continuará comentando los objetivos cumplidos. Se continuará haciendo una previsión o sentando las bases para los futuros desarrollos. Para terminar el autor hará balance de la experiencia durante el proyecto, así como sus agradecimientos.

1. Impacto del proyecto

Se empezará haciendo una previsión del impacto a corto plazo del proyecto para, más adelante, continuar con una estimación del impacto a largo plazo. Hay que tener en cuenta que se trata de estimaciones, y por lo tanto sólo se trata de una opinión, bastante meditada, pero imprevisible en cualquier caso.

Impacto a corto plazo.

El resultado principal producido por este proyecto es la aplicación en sí. A pesar de no tratarse de un programa de última tecnología, estará muy al día con las tecnologías existentes en el mercado, por lo que no se descarta cierto éxito en cuanto a número de usuarios.

Además el impacto a corto plazo de este proyecto se dará sobretodo en el departamento de Telemática de la Universidad Carlos III, ya que el código escrito y la aplicación desarrollada puede ser utilizada como plataforma para desarrollar aplicaciones más complejas y alcanzar objetivos más ambiciosos.

Por otro lado, la aplicación se distribuirá por varios canales, con la intención de popularizarla y aumentar su público. También se espera motivación por parte de los usuarios para crear sus propias bases de datos y abrir así el abanico de usos de la aplicación.

Impacto a largo plazo.

El impacto a largo plazo depende en gran medida de la popularidad adquirida en las primeras fases posteriores al término del proyecto. Es decir, cuanto más impacto tenga el proyecto a corto plazo, mayor será también el impacto a largo plazo.

Como simple aplicación, es decir, sin esperar desarrollos futuros o ampliaciones, el éxito y el impacto de este proyecto dependerán del empeño que se invierta en su promoción, ya que se trata de un proyecto ante todo social. De los usuarios que añadan sus puntos de interés y el boca a boca entre ellos depende que esta aplicación por sí sola siga adelante, animando a otros programadores a proseguir el desarrollo de la aplicación.

Dado que el objetivo de este proyecto ha sido crear una aplicación que sirva de plataforma para futuros desarrollo, es lógico pensar que gran parte del impacto será provocado por los mencionados futuros desarrollos. A partir de este proyecto se espera que se creen muchos otros y, dado que el futuro del sector es prometedor, se puede suponer que el impacto será importante.

También se esperan que los usos de la aplicación hayan crecido, debido a la

creación de nuevas bases de datos, para nuevas ciudades o lugares así como distintas naturalezas. Este es uno de los bastiones del proyecto, y puesto que se trata de que los propios usuarios utilicen sus propias ideas e iniciativas, el impacto que pueda tener es imprevisible, pero en cualquier caso resulta bastante alentador.

A los desarrollos que se sigan en el Departamento, habrá que sumarles los que hagan los individuos que se utilicen el código fuente obteniéndolo por distintas vías. Estos individuos, que probablemente serán estudiantes, podrán modificar el código e introducir mejoras. Éstas últimas pueden ser las citadas en este proyecto u otras que al desarrollador le parezcan más interesantes.

Estas dos posibles futuras corrientes ayudarán a crear más y mejores aplicaciones de realidad aumentada en un futuro, lo que repercutirá en una mayor y mejor oferta para los usuarios de aplicaciones móviles y para la sociedad en general.

2. Objetivos alcanzados

El principal objetivo propuesto era el de crear una aplicación que sirviera de punto de partida para los futuros proyectos del departamento. Este objetivo se ha cumplido satisfactoriamente, la aplicación resultante es fiable, robusta y cumple todas las especificaciones, como se menciona en la Sección “Verificación y validación” de esta memoria.

Uno de los objetivos secundarios era mejorar los conocimientos del desarrollador de la programación para plataformas móviles, el cual ha sido cumplido. Al llevar a cabo el proyecto el desarrollador ha adquirido nociones y conceptos aplicables sólo a la programación para terminales móviles, y nada tienen que ver con el desarrollo de software para ordenadores personales. Uno de los ejemplos más claros es la Realidad Aumentada, la cual es impensable para un ordenador de sobremesa.

Uno de los objetivos que se tenía al principio del proyecto era implementar la realidad aumentada en la aplicación, de manera que los puntos de interés aparecieran en tiempo real y a medida que el teléfono iba mostrando lo que cámara capturaba. Este requisito se descartó por dos motivos, principalmente: por un lado, la complejidad que supone realizar los cálculos necesarios para obtener los puntos de interés y además hacerlo 16 veces por segundo (para conseguir una sensación de fluidez de la imagen) y por otro lado se pensó que ir andando manteniendo el teléfono móvil a la altura de los ojos podía no ser lo más práctico, siendo más cómodo tomar una foto y obtener la información.

El desarrollador también ha adquirido conocimientos de la arquitectura Android. Con unas nociones básicas del lenguaje de programación Java, desarrollar en Android no supone un gran reto. Donde más conceptos nuevos han aparecido es cuando se realiza el estudio del sistema operativo Android y su funcionamiento.

3. Posibles futuros pasos

En esta sección se mencionarán, sin entrar mucho en detalle los posibles futuros desarrollos que se pueden hacer a partir del código escrito para este proyecto. El análisis detallado de éstos se hace en los apartados de mantenimiento y ampliaciones futuras. Además también se puede consultar el roadmap, que se encuentra en el Anexo I.

Los futuros desarrollos basados en este proyecto pueden ser muchos y variados. Los que aquí se exponen son simplemente una recomendación de hacia dónde se

deberían enfocar los esfuerzos, pero no significa que cualquier otra funcionalidad o mejora sea menos recomendable.

Los dos asuntos que deben de resolverse lo antes posible son la gestión de la batería y aumentar la robustez de la aplicación. La aplicación consume bastante batería, sobretodo por el uso que hace de las conexiones de datos y del servicio de GPS. La gestión de éste último deja bastante margen para mejorarla. Muchos esfuerzos se han dedicado a aumentar la robustez de la aplicación, aún así, algunos asuntos han quedado sin resolver. Por ejemplo, chequear que los archivos XML son bases de datos, puesto que lo implementado hasta ahora muestra como base de datos cualquier archivo XML.

Además de eso, otra de las prioridades es popularizar la aplicación. Para atraer nuevos usuarios, las bases de datos deben ser grandes y estar actualizadas. Para ello se podría utilizar Google Geocoding API para obtener información de puntos de interés cercanos (aunque se necesitaría conexión a internet). Otra manera sería diseñar un traductor de bases de datos y utilizar como fuente los datos de Google Maps.

Estas son las ampliaciones más realizables a corto plazo, aunque no se puede estimar el tiempo de implementación, puesto que depende de la experiencia del programador, además de ser necesario hacer un estudio previo dependiendo de la ampliación.

A más largo plazo, una de las mejoras más importantes podría ser implementar completamente la realidad aumentada, utilizando el mismo algoritmo que el utilizado en este proyecto. Esto haría mucho más intuitiva la aplicación, además ganaría vistosidad lo cual repercutiría en su popularidad.

4. Experiencia personal y agradecimientos

Me gustaría dar las gracias a todas las personas que han estado conmigo durante el desarrollo de este proyecto, así como las que me han acompañado en algún momento de mi vida. Agradecer, primero y ante todos a mis padres por el apoyo, cariño y paciencia con la que me han criado y guiado durante toda mi vida. A mi hermano, sin el cual creo que no hubiera terminado el proyecto, además también por servirme de guía tanto en lo sentimental como en lo profesional. A mis hermanos de otra madre, por haber estado ahí siempre siempre, éstos son: Elisa, Isa, Ale y Julio. A otro compañero y amigo de casi toda la vida: Alayi, aunque ha habido un parón, ahora parece que estamos como antes y que el tiempo no ha pasado. Andreas, Marine y Javi, sin vosotros Bélgica hubiera sido insufrible y Madrid mucho menos agradable. A Murdok, Pollo, Alexei, Tamara y Turre, por hacer de Madrid mi casa y aceptarme como uno más. Agradecer también a los que me acompañaron en la universidad, compañeros de penas y alegrías: Diego, Julio, Miguelín, Charli, Muñoz y Juanki. A Ana, por haberme soportado durante momentos difíciles y estresantes.

En el plano técnico, agradecer a mi hermano, Miguel (Viri) y Julián (Walter) por hacerme un testing tan exhaustivo y ayudarme a hacer un mejor proyecto (ya repartiré dividendos).

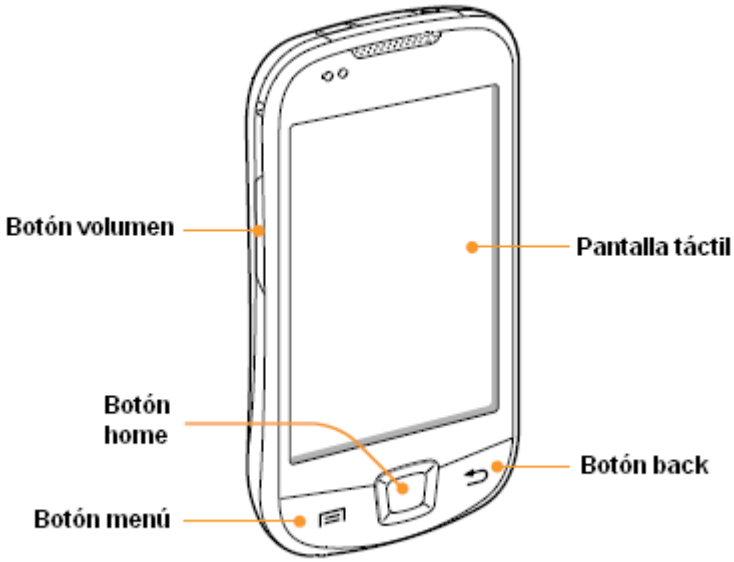
Por último, una cita:

The Dude abides

“El Nota aguanta”

ANEXOS

Anexo A: Ejemplo de disposición de teclas en un teléfono Android

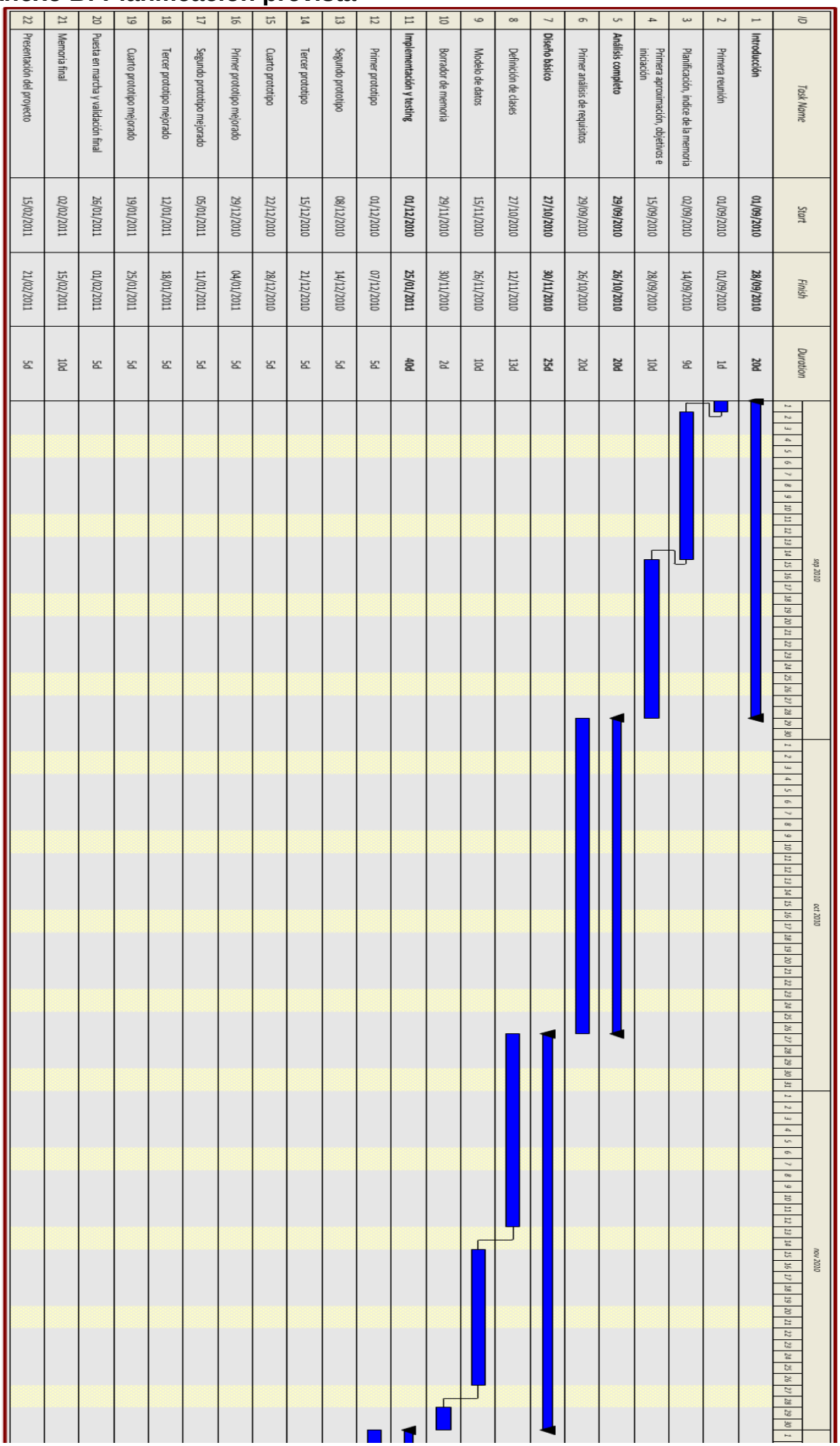


En general, los teléfonos Android tienen las teclas que se muestran en la figura anterior. La disposición y forma de las teclas varía dependiendo del fabricante y el modelo, pero todas suelen tener una disposición similar y un funcionamiento idéntico.

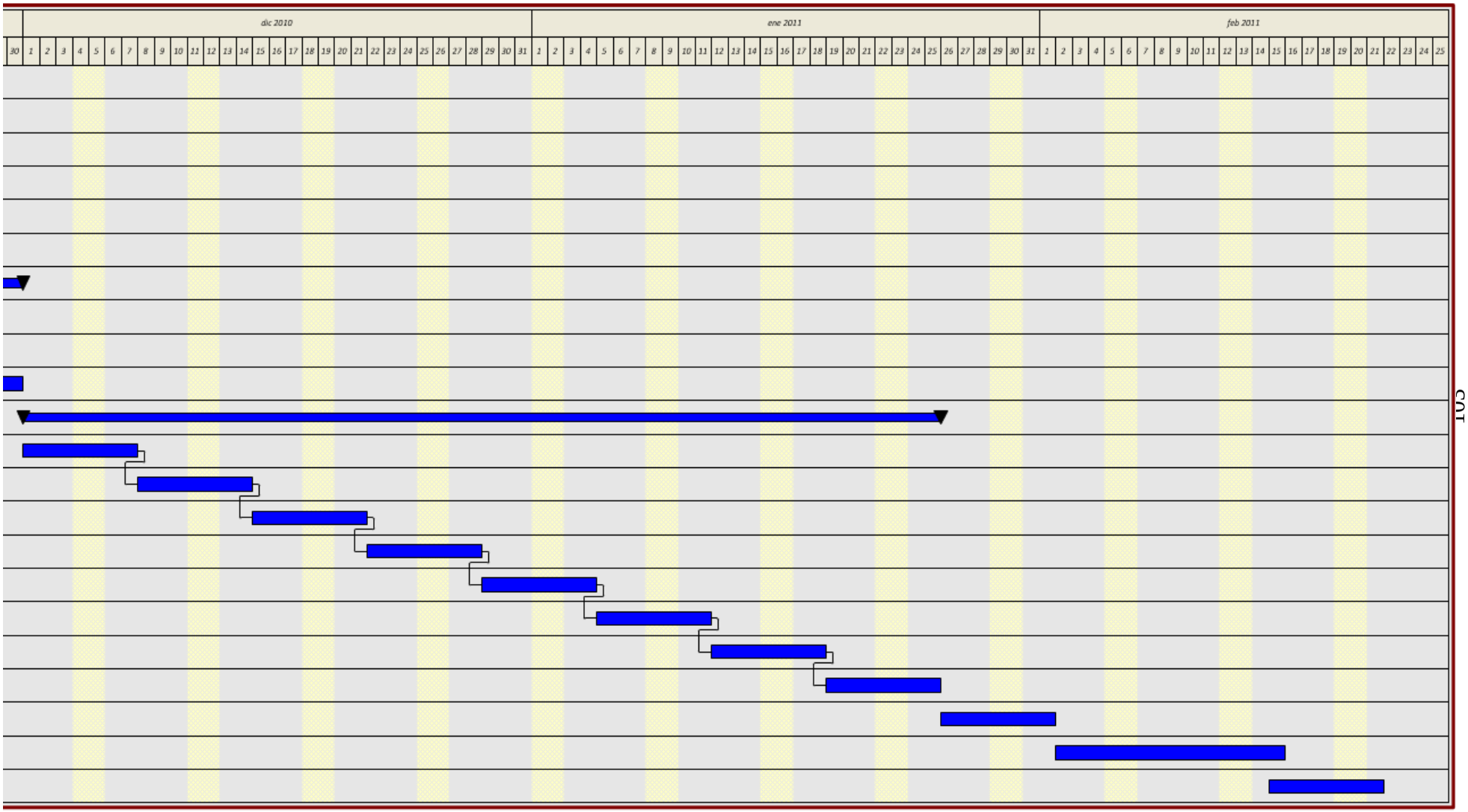
Se detallan a continuación las funcionalidades estandar de cada una de las teclas del teléfono:

- Botón volumen: subir y bajar el volumen, tanto del multimedia como del timbre y avisos.
- Botón home: vuelve al escritorio o al menú de aplicaciones, en general sale de cualquier aplicación para mostrar desde dónde se lanzó. Es algo así como minimizar todo en un entorno Windows.
- Botón menú: muestra las opciones disponibles.
- Botón back: por lo general vuelve a la pantalla anterior. En caso de que salga el teclado táctil, lo esconde para que el usuario pueda ver bien la ventana.
- Pantalla táctil: tiene multitud de usos, sobretodo como interfaz con el usuario. Permite usar botones táctiles programados por software.

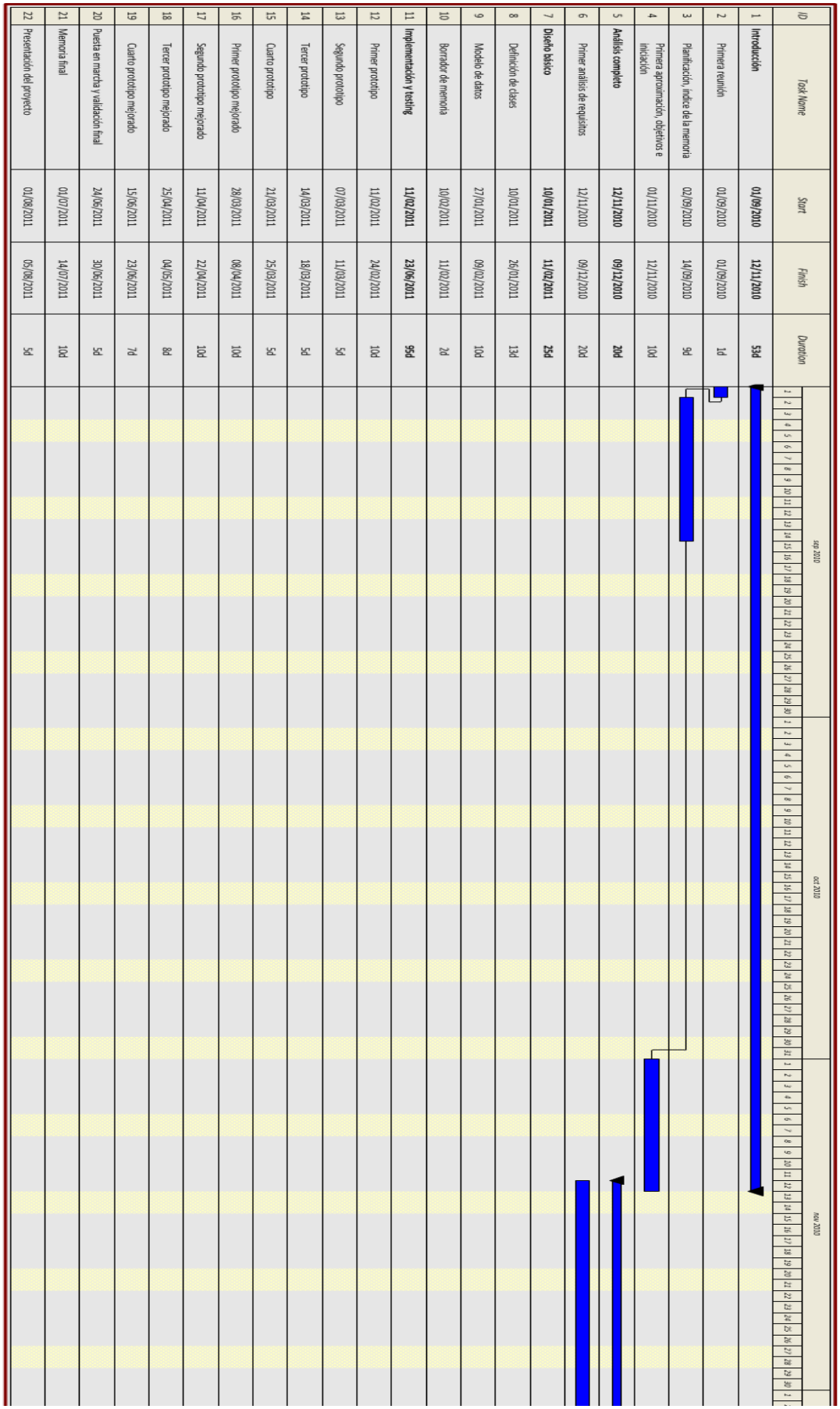
Anexo B: Planificación prevista



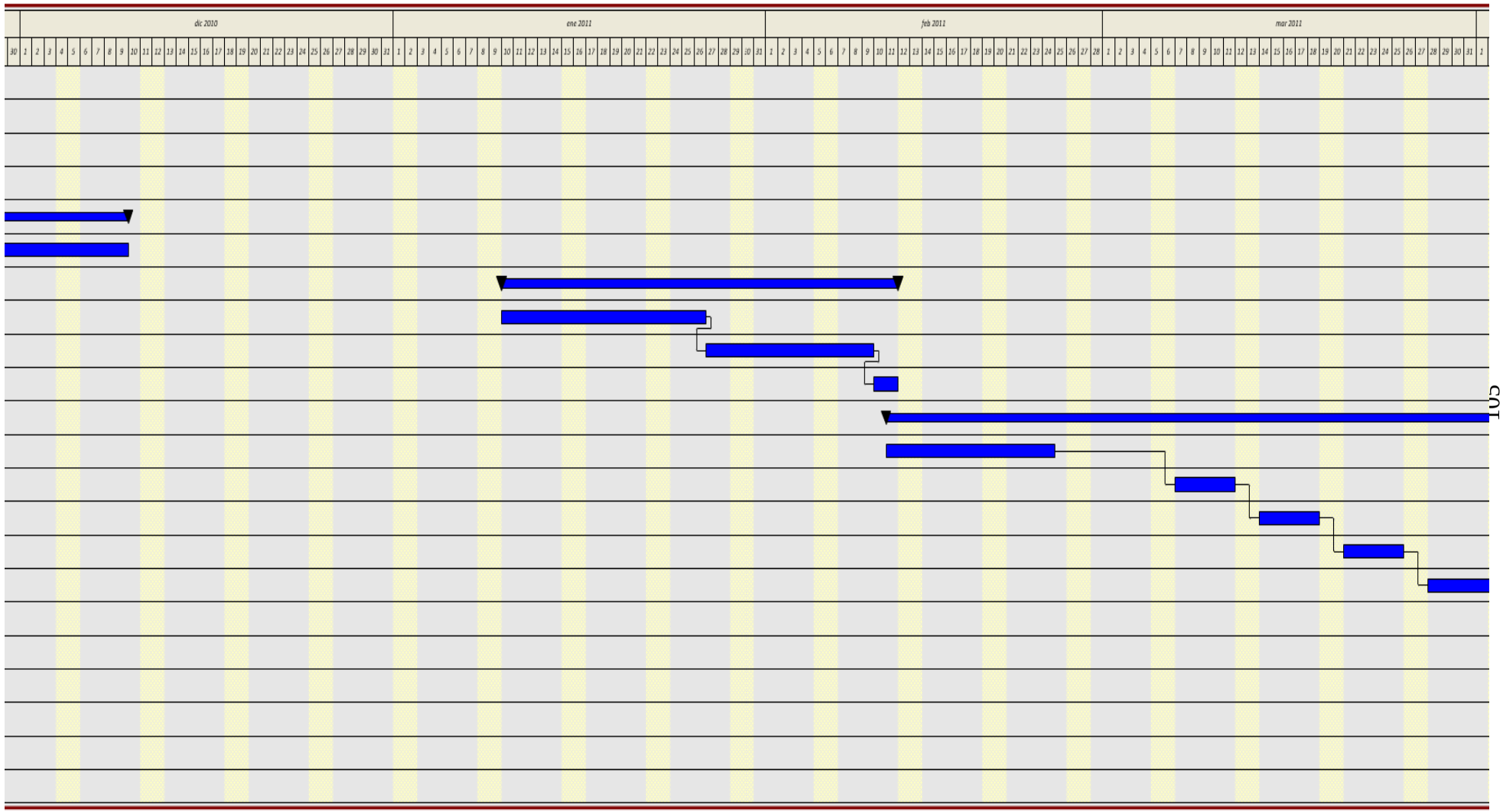
ID	Task Name	Start	Finish	Duration
1	**Introducción**	**01/09/2010**	**28/09/2010**	**20d**
2	Primera reunión	01/09/2010	01/09/2010	1d
3	Planificación, índice de la memoria	02/09/2010	14/09/2010	9d
4	Primera aproximación, objetivos e iniciación	15/09/2010	28/09/2010	10d
5	**Análisis completo**	**29/09/2010**	**26/10/2010**	**20d**
6	Primer análisis de requisitos	29/09/2010	26/10/2010	20d
7	**Diseño básico**	**27/10/2010**	**30/11/2010**	**25d**
8	Definición de clases	27/10/2010	12/11/2010	13d
9	Modelo de datos	15/11/2010	26/11/2010	10d
10	Borrador de memoria	29/11/2010	30/11/2010	2d
11	**Implementación y testing**	**01/12/2010**	**25/01/2011**	**40d**
12	Primer prototipo	01/12/2010	07/12/2010	5d
13	Segundo prototipo	08/12/2010	14/12/2010	5d
14	Tercer prototipo	15/12/2010	21/12/2010	5d
15	Cuarto prototipo	22/12/2010	28/12/2010	5d
16	Primer prototipo mejorado	29/12/2010	04/01/2011	5d
17	Segundo prototipo mejorado	05/01/2011	11/01/2011	5d
18	Tercer prototipo mejorado	12/01/2011	18/01/2011	5d
19	Cuarto prototipo mejorado	19/01/2011	25/01/2011	5d
20	Puesta en marcha y validación final	26/01/2011	01/02/2011	5d
21	Memoria final	02/02/2011	15/02/2011	10d
22	Presentación del proyecto	15/02/2011	21/02/2011	5d


dic 2010
ene 2011
feb 2011
30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
CUT

Anexo C: Planificación real



ID	Task Name	Start	Finish	Duration
1	**Introducción**	**01/09/2010**	**12/11/2010**	**53d**
2	Primera reunión	01/09/2010	01/09/2010	1d
3	Planificación, índice de la memoria	02/09/2010	14/09/2010	9d
4	Primera aproximación, objetivos e iniciación	01/11/2010	12/11/2010	10d
5	**Análisis completo**	**12/11/2010**	**09/12/2010**	**20d**
6	Primer análisis de requisitos	12/11/2010	09/12/2010	20d
7	**Diseño básico**	**10/01/2011**	**11/02/2011**	**25d**
8	Definición de clases	10/01/2011	26/01/2011	13d
9	Modelo de datos	27/01/2011	09/02/2011	10d
10	Borrador de memoria	10/02/2011	11/02/2011	2d
11	**Implementación y testing**	**11/02/2011**	**23/06/2011**	**95d**
12	Primer prototipo	11/02/2011	24/02/2011	10d
13	Segundo prototipo	07/03/2011	11/03/2011	5d
14	Tercer prototipo	14/03/2011	18/03/2011	5d
15	Cuarto prototipo	21/03/2011	25/03/2011	5d
16	Primer prototipo mejorado	28/03/2011	08/04/2011	10d
17	Segundo prototipo mejorado	11/04/2011	22/04/2011	10d
18	Tercer prototipo mejorado	25/04/2011	04/05/2011	8d
19	Cuarto prototipo mejorado	15/06/2011	23/06/2011	7d
20	Puesta en marcha y validación final	24/06/2011	30/06/2011	5d
21	Memoria final	01/07/2011	14/07/2011	10d
22	Presentación del proyecto	01/08/2011	05/08/2011	5d


dic 2010
ene 2011
feb 2011
mar 2011
30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1

abr 2011
may 2011
jun 2011
jul 2011
ago 2011

Anexo D: Test de JUnit

Nombre y descripción	**Resultado**
BaseDatos	
testSelecciona. El valor *seleccionada* se pone a true.	OK
testDeselecciona El valor *seleccionada* se pone a false.	OK
Camara	
testOnKeyDown El valor de entrada es el de la tecla pulsada.	OK
testOnKeyDown2 Llama al método compruebaUltimoFix().	OK
testOnKeyDown3 Retorna si no hay fix.	OK
testOnKeyDown4 Toma una foto si se ha conseguido un fix.	OK
testOnKeyDown5 Si la tecla es back, para el preview.	OK
testLocalizacion Asigna correctamente los escuchadores, tanto de posicionamiento como de GPS	OK
testOnResume Activa el preview.	OK
testOnResume2 Activa el sensor de orientación.	OK
testOnResume3 Llama a preguntaWiFi.	OK
testPreguntaWiFi Consigue el estado de la red inalámbrica de datos.	OK
testPreguntaWiFi2 Si hay conexión de datos, la utiliza para conseguir un fix.	OK
testPreguntaWiFi3 Sólo si es la primera ejecución y el usuario no está conectado a una red WiFi, le preguntará si se quiere conectar a una.	OK
testOnStop Desactiva el sensor de orientación.	OK
testOnPictureTaken Crea el archivo que contiene la fotografía.	OK
testOnPictureTaken2 Lanza excepción en caso de que algún parámetro no sea correcto.	OK
InfoExtra	

testDameExtras Recupera los *extras* empaquetados en la Activity anterior.	OK
testCreaLayout Crea el *layout* con los parámetros dados.	OK
testMuestra Detecta si el punto de interés tiene una URL de una fotografía.	OK
testMuestra2 Detecta si el punto de interés tiene texto adicional.	OK
testMuestra3 Si un punto de interés tiene URL, carga la fotografía en un ImageView.	OK
testMuestra4 Muestra solo un campo (URL o texto), en caso de que lo haya.	OK
testMuestra5 Muestra los dos campos(URL y texto), en caso de que los haya.	OK
testMuestra6 El punto tiene los dos campos. La URL está mal formada o con algún error, entonces cargará solo el texto.	OK
testDameDrawable Detecta error y avisa al usuario en caso de que la URL esté mal formada.	OK
testDameDrawable2 Detecta error de lectura y escritura y alerta al usuario.	OK
testCreaInputStream Obtiene el contenido de la URL.	OK
Introduccion	
testLeeBasesDeDatos Crea un *progress dialog* cuando empieza el procesado.	OK
testLeeBasesDeDatos2 Carga correctamente las bases de datos estandar.	OK
testLeeBasesDeDatos3 Carga correctamente las bases de datos externas.	OK
testLeeBasesDeDatos4 Lanza excepción en caso de que haya error de lectura de la base de datos.	OK
testLeeBasesDeDatos5 Lanza excepción en caso de que haya error durante el parsing del archivo XML.	OK
testLeeBasesDeDatos6 Lanza una excepción general en caso de que no sea ninguna de las anteriores.	OK

testLeeBasesDeDatos7 Si hay error de lectura o parsing, los datos cargados en memoria se podrán utilizar.	OK
testLeeBasesDeDatos8 Instancia un objeto PuntoInteres por cada localización correctamente leída de la base de datos.	OK
testLeeBasesDeDatos8 Pone correctamente los valores a los atributos del objeto PuntoInteres.	OK
testCompruebaBBDD Si no hay bases de datos cargadas, vuelve a la ventana anterior.	OK
testCompruebaGPS Devuelve false si no está activado, true si sí.	OK
alertaNoGPS Muestra al usuario un texto diciendo que active el servicio GPS y abre la ventana de configuración de GPS.	OK
PuntoInteres	
testCalculaCercania Comprueba si la localización actual está dentro del radio de interés del usuario.	OK
testCalculaVisible Calcula el ángulo de orientación del punto de interés respecto con el de la cámara.	OK
testCalculaVisible2 Detecta cualquier problema causado por los ángulos superiores a 360º.	OK
PuntoInteresUI	
testPonId El valor *id* se pone al valor introducido.	OK
testDameId Devuelve el valor de *id*.	OK
testOnDraw Crea un View que se superpone al fondo.	OK
testOnDraw2 Obtiene correctamente las dimensiones de la pantalla del teléfono.	OK
testOnDraw3 Pinta el texto en una posición dependiendo de su identificador y dentro de una zona acotada en altura.	OK
Preview	
testGetOptimalPreviewSize Obtiene el tamaño óptimo para el *preview*.	OK
testGetOptimalPreviewSize2	OK

Obtiene la relación de aspecto para el *preview*.	
testSurfaceChanged Llama correctamente al método getOptimalPreviewSize.	OK
testSurfaceChanged2 Arranca el *preview* con los parámetros indicados.	OK
Resultado	
testMuestraPuntos Instancia correctamente los objetos PuntoInteresUI y asigna a sus atributos los valores correspondientes.	OK
testMuestraPuntos2 En caso de que el valor GPS de la posición actual sea incorrecto el método mostrará un mensaje de advertencia.	OK
testMuestraPuntos3 Popula correctamente el vector *puntosVisibles*.	OK
testOnKeyDown Elimina correctamente el contenido del vector *puntosVisibles*.	OK
testOnDestroy Gestiona correctamente la memoria y elimina la asignación de memoria asignada a la imagen.	OK
SeleccionBBDD	
testCargaBasesDeDatosExternas Carga correctamente los nombres bases de datos existentes en la carpeta *descargas* al vector *canales*.	OK
testCargaBasesDeDatosExternas2 Detecta si no hay contenido en la carpeta *descargas* y como resultado no hace nada.	OK
testCargaBasesDeDatosStandard Carga correctamente las dos bases de datos estándar.	OK
testCreaBotones Crea los botones requeridos.	OK
testCreaBotones2 Añade correctamente todos los elementos al *layout* creado previamente.	OK
testCreaExplicacion Crea un TextView con el texto requerido.	OK
testCreaLayout Crea el *layout* con los parámetros dados.	OK
testCreaSeekBar Crea la *seekbar* con los atributos necesarios.	OK
SeleccionPI	
testCompruebaConexion Detecta si se tiene conexión a Internet por redes telefónicas.	OK

testCompruebaConexion2 Si no tiene conexión a Internet por redes telefónicas, lanza la ventana de ajustes del teléfono.	OK
testCreaBotones Crea los botones requeridos.	OK
testCreaBotones2 Añade correctamente todos los elementos al *layout* creado previamente.	OK
testCreaLayout Crea el *layout* con los parámetros dados.	OK
testOnContextItemSelected Detecta correctamente el botón pulsado y lanza la *Activity* correspondiente.	OK
testOnContextItemSelected2 Llama correctamente al método *CompruebaConexion*.	OK
testOnCreateContextMenu Crea correctamente un menú emergente por cada uno de los puntos de interés mostrados.	OK
testOnCreateContextMenu2 Muestra el botón información adicional sólo si el punto de interés contiene el campo URL o info.	OK

Anexo E: Crear un nuevo terminal Android emulado (AVD)

Para crear un nuevo terminal emulado desde Eclipse se deberán seguir los siguiente pasos:

- Abrir el menú *Android SDK and AVD Manager* y seleccionar *new*.
- Ponerle un nombre en *name*, seleccionar como *target* Android 2.1-update1.
- Poner en tamaño de tarjeta SD, 512 MB.
- En la sección de *hardware* seleccionar:
    - SD card support
    - GPS
    - Acelerómetro
    - Cámara

Una vez completados estos pasos, ya se tendrá listo un terminal Android emulado, que se arrancará cada vez que se ejecute la aplicación. También se puede dejar corriendo e ir ejecutando la aplicación ya que la reinstalará.

Para acceder a información adicional, como las trazas o los mensajes del sistema operativo, se deberá abrir la perspectiva DDMS en Eclipse. Además esta perspectiva permite obtener capturas de pantallas así como introducir archivos al terminal o explorar el contenido de su sistema de archivo.

Anexo F: Instalar una aplicación en un terminal Android.

Para instalar la aplicación en un terminal móvil Android es necesario instalar el paquete APK (Android Package). Para ello hay tres opciones: si la aplicación está publicada, es decir está en el Android Market, basta con descargársela y se instalará automáticamente, en caso de que no lo esté hay que instalarla manualmente. Otra opción, en caso de que se tenga el código fuente, es instalarla desde el Eclipse. Esta última opción es la mejor en caso de estar desarrollando puesto que permite monitorizar las trazas tanto de la aplicación como del terminal.

Para realizar una instalación manual de la aplicación existen dos posibilidades:

- Usando un instalador. Simplemente descargando la aplicación, si se pulsa sobre ella el sistema operativo llamará al instalador el cual guiará al usuario durante todo el proceso.

- Usando el Android SDK. En caso que el sistema operativo sea Linux, se podrá instalar la aplicación directamente introduciendo en el Command Prompt el comando descrito más abajo. En caso de hacerlo desde Windows hay que instalar los drivers Android USB para conectar el terminal móvil al Android SDK mediante USB.

- Una vez hecho esto deberemos entrar en:
  Settings -> Application Settings y activar Unknown Sources.
  Y también:
  Settings -> SD Card and Phone Storage y desactivar Use for USB Storage.

Habiendo completado estos pasos pasamos a usar Command Prompt e introducimos:

adb install ruta/archivo.apk

Donde ruta es la ruta completa al archivo APK y archivo es el nombre del archivo de la aplicación. Una vez hecho esto la aplicación estará instalada.

Para proceder a instalar una aplicación en desarrollo a un terminal Android habrá que seguir los siguientes pasos:

- Descargar los drivers del teléfono para el sistema operativo que se esté usando en la máquina de desarrollo, para buscarlos se utilizará la siguiente página: http://developer.android.com/sdk/oem-usb.html

- Activar en el terminal el modo *Depuración USB* de la siguiente manera:
  Settings → Application Settings y activar Unknown Sources.
  Y además:
  Settings → Application Settings → Development y activar USB Debugging.

- Abrir el eclipse y lanzar la aplicación, cuando aparezcan los diferentes terminales objetivo, seleccionar el físico (el otro debería ser el emulado)

Una vez completados todos los pasos, la aplicación estará instalada en el terminal de igual manera que todas las demás, con la ventaja de que, mientras esté el USB conectado, podremos ver en la vista DDMS de Eclipse, las distintas trazas que va dejando tanto el programa como el sistema operativo.

Anexo G: Instalar una aplicación en el emulador.

Para instalar un paquete APK en el emulador de Android, habrá que seguir los siguientes pasos:

- Antes que nada habrá que arrancar Eclipse, y el terminal emulado. Una vez completado el arranque se proseguirá con el siguiente paso.
- Moviendo el paquete .apk en la carpeta /tools del Android sdk.
- Situar la consola de Windows o de Linux en la carpeta donde está instalado el Android adb. Introduciendo, generalmente:
  - En Windows: cd C:\Program Files\Android\android-sdk-windows\tools
  - En Linux: cd /home/<username>/android-sdk-linux_x86/platform-tools
- Una vez en la carpeta correspondiente, en la consola se debe teclear:

  `adb install <nombre_del_paquete>`

- Habrá que esperar un mensaje de **`Success.`**

Una vez completados todos los pasos, el icono de la aplicación aparecerá en el terminal emulado, basta con pulsarlo y arrancará la aplicación.

Anexo H: Nomenclatura

1. Vista general

Esta sección tiene como objetivo describir las reglas generales que se han seguido mientras se escribía el código y se desarrollaba la aplicación. El propósito es facilitar el mantenimiento y potenciar un desarrollo futuro de más calidad y más rápido.

2. Reglas seguidas durante el desarrollo

El código del software debía seguir un conjunto de reglas estrictas a la hora de nombrar los métodos, atributos y las clases. Las siguientes subsecciones se centrarán en distintos aspectos del código y su nomenclatura.

Reglas generales

Estas reglas generales deben seguirse para todos los nombres de las siguentes subsecciones, salvo que se especifique lo contrario:

- El nombre debe tener alguna relación con la función que realiza la variable, método, atributo...
- El lenguaje utilizado siempre será el castellano.
- Todos los nombres empezarán con una minúscula.
- En el caso de que el nombre sea compuesto, todas las palabras se escribirán juntas, sin separador alguno y con la primera letra de cada palabra en mayúsculas (exceptuando la primera). Por ejemplo: dameLocalizacion.

Nombres de contantes

Al nombrar constantes, el nombre entero, incluso cuando éste sea compuesto, se escribirá entero en mayúsculas. En el caso de que sea compuesto, se utilizará como separador el carácter “_”.

Ejemplos:

FIX_OBSOLETO: constante que determina el tiempo que debe de pasar para que un fix quede obsoleto.

Nombres de variables

Al nombrar las variables se deben seguir las reglas nombradas anteriormente. Una excepción son las variables locales, utilizadas normalmente para iteraciones, en las que se podrá usar una notación más simple. Por lo general: i, j, k, etc... Un ejemplo de esta excepción también se da con los Toasts, mensajes cortos que se muestran al usuario, que son siempre variables locales.

Ejemplos:

i: variable utilizada sólo para iteraciones

t: objeto Toast, que se mostrará y se destruirá de manera inmediata.

Nombres de atributos

Los atributos seguirán las reglas generales, sin ninguna excepción.

Nombres de métodos

Los métodos, al igual que los atributos, seguirán las reglas generales.

Ejemplos:

calculaCercania(): Método que calcula la cercanía de un punto de interés y la posición actual del terminal.

Nombres de las clases y otros componentes

Al nombrar clases u otros componentes, se seguirán las siguientes reglas: el nombre de la clase debe ser representativo y la primera letra debe ser mayúsculas.

Ejemplos:

Resultado: Clase que muestra una ventana con los puntos de interés cercanos, además de realizar todos los cálculos correspondientes para realizarlo.

3. Reglas al comentar

Al principio de cada clase y cada método relevante se encontrará una descripción de éstos, detallando su función. Además de eso, comentarios adicionales se añadirán cuando el código no sea fácilmente comprensible.

El formato a seguir debe cumplir con los estándares de Java, para ser capaces de generar automáticamente el documento JavaDoc. El estándar es el siguiente:

- Todos los comentarios estarán escritos en castellano.
- No debe de incluirse ningún tipo de tilde, la letra “ñ” o cualquier tipo de carácter especial.
- Ejemplos:
  - Cabeceras de clases:

```
/**
 * @nombre nombre de la clase
 * @proposito breve descripción de la clase
 *
 * @author autor de la clase
 * @version versión del código
 */
```

  - Cabeceras de métodos:

```
/**
 * Descripción del método
 *
 * @param parametros de entrada
 * @return variable de retorno
*/
```

4. Aspectos generales del desarrollo

Hay varios aspectos que se han tenido en cuenta durante el desarrollo del

software.

Mantenimiento

El código debe ser implementado de manera que cualquier desarrollador pueda realizar el mantenimiento y trabajar sobre él de manera sencilla. Los siguientes aspectos se han tenido en cuenta:

- El código debe estar bien documentado.
- Se ha separado de manera lógica los diferentes métodos en bloques.
- Se debe usar la indentación en las iteraciones, para facilitar la lectura.
- Los paréntesis se usará en operaciones lógicas o aritméticas para facilitar la lectura.

Estructura

La estructura debe ser modular, separando en diferentes clases las distintas funcionalidades.

Anexo I: Roadmap

1. Nomenclatura de versiones

La nomenclatura usada para las distintas versiones del software ha seguido las siguientes reglas:

- Nomenclatura: <Versión principal>.<Versión secundaria>.<Número de revisión>
- Explicación:
  - Versión principal: versiones que han pasado un completo análisis y revisión del software por parte del cliente. Revisión y redefinición de requisitos.
  - Versión secundaria: la funcionalidad añadida al producto no implica ningún cambio sustancial.
  - Revisión: revisión superficial hecha por el cliente o actualización de funcionalidades internas. Estas modificaciones pueden implicar una implementación mejorada, solución de errores, etc...

2. Versionado del producto

En esta subsección se detalla el proceso de versionado del producto:

- La versión inicial se denominará 0.0.0.
- Por cada nuevo requisito funcional implementado, se incrementará por uno la versión secundaria.
- Si se ha hecho alguna modificación o revisión por parte del cliente para una funcionalidad concreta (sin estar ésta completamente implementada) se incrementará por uno el número de revisión.
- Una vez que se hace una revisión completa del producto por parte del cliente, la versión primaria se incrementará por uno. Cuando sucede esto, se debe de realizar un nuevo análisis de requisitos completo así como revisar las prioridades para actualizar el roadmap.

3. Roadmap usado

La siguiente tabla resume el roadmap seguido hasta la versión 1.0.0 del software. Todas las funcionalidades han sido implementadas.

Título y descripción	**Versión**
Creación de interfaces gráficas básicas Creación de botones y escuchadores Creación de Activities y Layouts	0.1.0

Mejora de interfaces gráficas Establecimiento de fondos de pantalla Creación de gráficos a partir de archivos de imágenes Creación de alertas para usuario	0.1.1
Creación de la cámara Establecer *preview* de la cámara como fondo de pantalla Generar imagen al apretar obturador Gestión de recursos	0.2.0
Optimización de la cámara Compresión de imágenes Detección de pausa y reinicio de cámara	0.2.1
Implementación de geolocalización por GPS Adquisición de datos de GPS Asignación y gestión de escuchadores de localización Actualización de variables y atributos internos relacionados con la geolocalización	0.3.0
Implementación de geolocalización por redes Adquisición de datos de localización mediante redes Asignación y gestión de escuchadores de localización	0.3.1
Carga de datos básica Lectura de datos desde archivos Cierre de archivos del terminal Acceso a carpetas privadas y públicas del terminal	0.4.0
Carga de datos XML Lectura de archivos XML Parsing de archivos XML Creación de estructuras de datos a partir de XML	0.4.1
Mejora de carga de datos XML Creación de barra de progreso durante la carga Creación de mensajes de alerta	0.4.2
Integración de geolocalización y datos Lectura de datos desde XML y adquisición de información Cotejo de datos entre base de datos y localización actual	0.5.0
Optimización de geolocalización Ajuste de tiempos de actualización Avisos a usuario cuando se obtiene fix	0.6.0
Implementación de realidad aumentada Sobreimpresión de información en fotografía Creación del algoritmo de presentación de puntos de interés cercanos Creación de mecanismos de control y trazas	0.7.0
Obtención de información desde internet Gestión de la conexión a Internet Creación de alertas de conectividad Llamadas a métodos para usar servicios: maps, google, etc...	0.8.0
Añadir información adicional a los puntos de interés Lectura de campos extra en XML	0.9.0

Creación de nuevos atributos en clase PuntoInteres Gestión de conectividad	
Mejoras respecto a la información adicional Gestión de errores al cargar la imagen Presentación de la información disponible	0.9.1
Mejora de la interfaz gráfica de usuario Creación de menú contextual para información Uso de la misma forma(en este caso de tú) Mejorar el sistema de alertas Añadir pregunta de si usuario conoce red wifi para conectarse	0.10.0
Revisión general de pequeños detalles y modificaciones Hacer testing Generación de documentación	0.11.0
Revisión final por parte del cliente	0.11.1
Publicación de la versión última y definitiva del código, incluyendo esta documentación	1.0.0

4. Roadmap recomendado

A continuación se muestra el roadmap recomendado, que detalla todas las funcionalidades a implementar en el futuro. Las funcionalidades están ordenadas según las prioridades del cliente y del departamento de Telemática.

Título y descripción	**Versión**
Mejora de la gestión de batería Establecer conexión de datos en modo espera Moderar el uso del GPS Servirse de información sobre el estado del teléfono para ahorrar batería, por ejemplo: si se está cargando, usar el mismo fix.	1.1.0
Mejoras generales Añadir trazas y un log para diagnóstico Optimizar el método de parsing de XML Comprobar que los archivos XML son bases de datos	1.2.0
Ampliar bases de datos Integrar con Google Geocoding API Traductor de Google maps (BigTable) Añadir nuevas ciudades	1.3.0
Mejorar algoritmo de geolocalización Optimizar código de adquisición de datos GPS Implementar ángulo frontera variable dependiente de radio	1.4.0
Añadir puntos de interés Adaptar interfaz gráfica para que el usuario pueda añadir sus propios puntos de interés Cotejar con bases de datos existentes para evitar duplicados	1.5.0
Hacer aplicación multi-idioma	1.6.0

Establecer idioma cuando se arranque por primera vez Eliminar datos de las bases de datos correspondientes a los otros idiomas	
Añadir publicidad Mostrar publicidad en tiempos de carga Implementar algoritmo dependiente de la localización	1.7.0
Evitar al usuario pasar por la ventana de instrucciones Introducir lógica de carga de datos en otra ventana Añadir botón para mostrar las instrucciones	1.8.0
Revisión por parte del cliente	1.9.0
Publicación de la versión última y definitiva del código, incluyendo su documentación	2.0.0

Anexo J: Manual de usuario

Instalar la aplicación

Para instalar la aplicación en un terminal móvil Android es necesario instalar el paquete APK(Android Package). Para ello hay tres opciones: si la aplicación está publicada, es decir está en el Android Market, basta con descargársela y se instalará automáticamente, en caso de que no lo esté puede instalarse con el instalador de Android o puede hacerse manualmente. Para utilizar el instalador de Android basta con hacer click en el paquete APK y se ejecutará automáticamente.

Para realizar una instalación manual de la aplicación existen dos posibilidades:

- Usando un instalador. Simplemente descargando la aplicación, si se pulsa sobre ella el sistema operativo llamará al instalador el cual guiará al usuario durante todo el proceso.
- Usando el Android SDK. En caso que el sistema operativo sea Linux, se podrá instalar la aplicación directamente introduciendo en el Command Prompt el comando descrito más abajo. En caso de hacerlo desde Windows hay que instalar los drivers Android USB para conectar el terminal móvil al Android SDK mediante USB.
- Una vez hecho esto deberemos entrar en:
  Settings -> Application Settings y activar Unknown Sources.
  Y también:
  Settings -> SD Card and Phone Storage y desactivar Use for USB Storage.

Habiendo completado estos pasos pasamos a usar Command Prompt e introducimos:

adb install <ruta>/<archivo.apk>

Donde ruta es la ruta completa al archivo APK y archivo es el nombre del archivo de la aplicación. Una vez hecho esto la aplicación estará instalada.

Todas las instalaciones requerirán que el usuario de permiso a la aplicación para:

- Conocer la ubicación del terminal (GPS)
- Conectarse a Internet
- Modificar o eliminar contenido de la tarjeta SD
- Leer el estado del teléfono
- Hacer fotografías

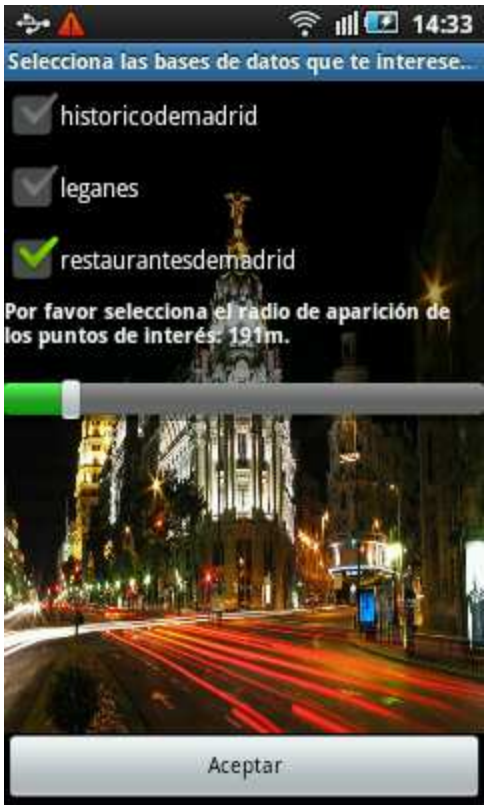
Utilizar la aplicación

Para que el usuario tenga una visión general del programa debe saber que se subdivide en *ventanas* (en el argot de Android, *Activities*) y que, por norma general, la aplicación mostrará una ventana nueva por cada interacción del usuario.

Una vez encendido el terminal, con la aplicación instalada y funcionando normalmente, los pasos a seguir para usar la aplicación son:

Icono de la aplicación GuiaInteractiva

1. Arrancar la aplicación desde el menú de aplicaciones (o desde el acceso directo en el escritorio), pulsando en el icono titulado GuiaInteractiva.



Primera ventana de la aplicación

2. Aparecerá, como ventana de arranque, una lista en la que podremos seleccionar cualquiera de las bases de datos que se tengan instaladas en el terminal móvil. Además aparecerá una barra en la que el usuario podrá seleccionar el radio de interés en el que aparezcan los puntos de interés. Una vez elegidas las bases de datos de interés y el radio de interés, pulsar *Aceptar*.

3. Si hay algún problema al cargar o leer alguna de las bases de datos se mostrará una ventana emergente avisando de la situación en la segunda ventana. Además tiene como objeto simplemente informar al usuario de cómo funciona la aplicación. Para continuar pulsar el botón con el texto “continuar”. En caso de que el usuario no tenga activado el adaptador GPS, se le avisará y entrará a la pantalla de ajustes para que lo active.

Alerta por falta de fix

4. La primera vez que se acceda a la tercera ventana se preguntará si conoce alguna WiFi para conectarse (si es que no lo está ya), puesto que conseguir un fix de GPS es mucho más fácil si se está conectado. En la tercera ventana aparecerá el *preview* de la cámara. El usuario, cuando crea conveniente y quiera obtener información de algún lugar cercano, deberá pulsar el botón de menú para tomar una foto, lo que provocará que se pase a la siguiente pantalla si hay un fix. En caso de que no lo haya tendrá que esperar a que aparezca un mensaje avisando que se ha conseguido uno.

Presentación de puntos de interés cercanos

5. La cuarta ventana tendrá como fondo la foto tomada en la anterior. Aparecerán, en caso de que hubiese, los nombres de los puntos de interés cercanos. Si el usuario estuviese interesado en obtener más información de cualquiera de los puntos de interés, debe pulsar en cualquier lugar de la pantalla. Esto último abrirá la siguiente ventana.

Opciones disponibles para un punto de interés(sin información adicional)

6. En la quinta ventana aparecerán todos los puntos de interés cercanos, en el caso de que los hubiera, en modo lista. Si el usuario quisiera obtener información de alguno de ellos debe mantener pulsado el botón del punto de interés en concreto, por lo que aparecerá una ventana emergente mostrando tres opciones. Estas tres opciones son: *buscar en Google*, *buscar en Google Maps* y leer la entrada de la *Wikipedia*. También, en caso de que el punto de interés tenga alguno o ambos de los campos opcionales, aparecerá un botón llamado *Información adicional*. El usuario deberá elegir la opción que quiera. En caso de que el teléfono no disponga de conexión a Internet, mostrará una ventana emergente y entrará a la pantalla de configuración de redes inalámbricas del terminal, para que el usuario lo active.

En cualquiera de las ventanas anteriores, si el usuario pulsase el botón retorno volvería a la pantalla anterior. Si el usuario pulsase la tecla de HOME volvería al escritorio, lo que no terminaría la aplicación si no que la pasaría a un segundo plano. La aplicación sólo la finaliza el sistema operativo, en situaciones de baja memoria disponible. Así mismo, si el usuario desease volver a la ventana anterior, bastaría con pulsar el botón BACK.

En el caso de haber descargado una base de datos de puntos de interés de la página web, o haberla hecho, ésta debe encontrarse en la carpeta de descargas de la tarjeta SD para que el programa la lea correctamente. Normalmente, el usuario no debería mover la base de datos de lugar si se la ha descargado con el navegador, puesto que la carpeta de descargas es donde se almacenan los archivos descargados.

Problemas comunes

En caso de que se haya dado un error en la lectura de las bases de datos XML, puede que se den dos casos: error total y error parcial. El error total significa que no se

ha podido cargar ningún punto de interés en la base de datos de la aplicación. Por otro lado si el error no haya sido total y se hayan podido cargar algunos puntos de interés, éstos estarán disponibles. Por ejemplo: una base de datos contiene 20 puntos de interés, se empiezan a cargar los puntos de interés y cuando se llega al décimo, hay algún tipo de fallo en la lectura. En ese caso, para el usuario estarán disponibles los diez primeros puntos de interés.

En caso de que el servicio GPS no funcione correctamente, intente conectarse a una red WiFi, de esta manera será más fácil conseguir un fix. En caso de que no pueda conectarse a una red WiFi, deberá esperar a que el GPS consiga un fix, lo que puede llevar hasta un minuto en ciertos terminales.

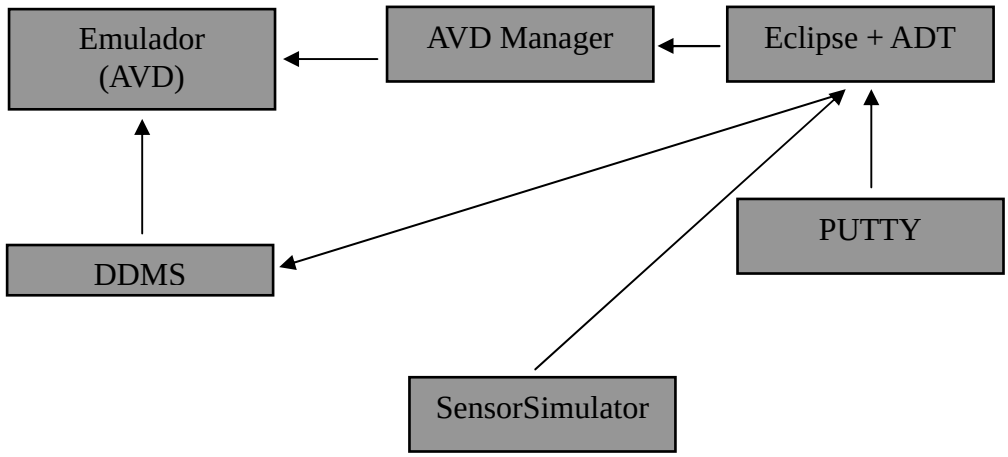
Anexo K: Manual de referencia

Aspectos generales

Este manual servirá de guía para cualquier desarrollador que en un futuro, quiera continuar con el desarrollo de esta aplicación. Para hacer la explicación lo más clara posible se empezará por explicar el cómo debe configurarse el entorno de desarrollo. Para después pasar a explicar el código desarrollado. Al final se encontrará información de cómo realizar cosas relativamente frecuentes durante el desarrollo.

Para saber cómo utilizar la aplicación a nivel de usuario, consulte el manual de usuario, el cual contiene una guía pormenorizada de los pasos a seguir al ejecutar la aplicación. Además, en dicho manual se explica como solventar ciertos problemas comunes.

A continuación, se muestra un diagrama en el que se representa el entorno de desarrollo que se ha utilizado y en el que aparecen todos los módulos que interactúan entre sí. De esta manera el desarrollador puede hacerse una idea de cómo funciona el entorno de desarrollo.



Arquitectura general del entorno de desarrollo

Para continuar con el desarrollo de esta aplicación, lo más sencillo es utilizar Eclipse con el plug-in de Android. Basta con importar el proyecto a Eclipse, para hacer esto seguir los siguientes pasos:

- En menú *File*, seleccionar *Import*.
- Seleccionar como fuente *General->Existing Projects into Workspace*.
- En el cuadro de *Select root directory* introducir la ruta de la carpeta raíz del proyecto, en este caso GuiaInteractiva y pulsar *Finish*.

Una vez importado se utilizando la ventana *Package Explorer* se podrá visualizar la estructura completa del software. Aquí se mencionan las carpetas más importantes y que, más probablemente, quieran modificar los futuros desarrolladores.

- *src*: contiene el código fuente. Bajo el paquete com.guiaInteractiva se encuentran todas las clases implementadas.

- *gen*: contiene información generada automáticamente durante el desarrollo. Bajo ningún concepto se deberá modificar el contenido de R.java.

- *Android 2.1-update1*: contiene las librerías puestas a disposición del desarrollador por Android.

- *res*: contiene los recursos que utiliza la aplicación, como archivos XML o fotos. Dentro de esta carpeta se encuentran las siguientes:

    - *drawable-hdpi*, *drawable-ldpi*, *drawable-mdpi*: contiene las imagenes que utiliza la aplicación. En este caso, el fondo de pantalla de la ventana de Introducción y SeleccionBBDD, además del icono de la aplicación.

    - *layout*: contiene información en XML de layouts de las diferentes Activitys. Aquí se encuentra solamente *main.xml*, que es utilizada por Introducción.

    - *values*: contienen variables qu pueden ser utilizadas como recurso. La variable *app_name* define el nombre de la aplicación (aparece en el icono).

    - *xml*: contiene archivos XML. Pueden utilizarse con cualquier finalidad, pero en este caso contiene las bases de datos estándar, éstas son: *historicodemadrid.xml*, *leganes.xml* y *restaurantesdemadrid.xml*.

- *AndroidManifest.xml*: se trata de un archivo XML que contiene propiedades de la aplicación, tales como los permisos que necesita y de las Activitys que se compone.

Cabe comentar también que el software compilado y empaquetado en el fichero APK no tiene por qué compilarse ni modificarse para instalarlo de un terminal Android a otro a excepción de que dicho terminal corra con una versión de Android anterior a la 2.1.

A continuación se muestra el diagrama de clases, para que el desarrollador obtenga una visión global de la estructura del proyecto.

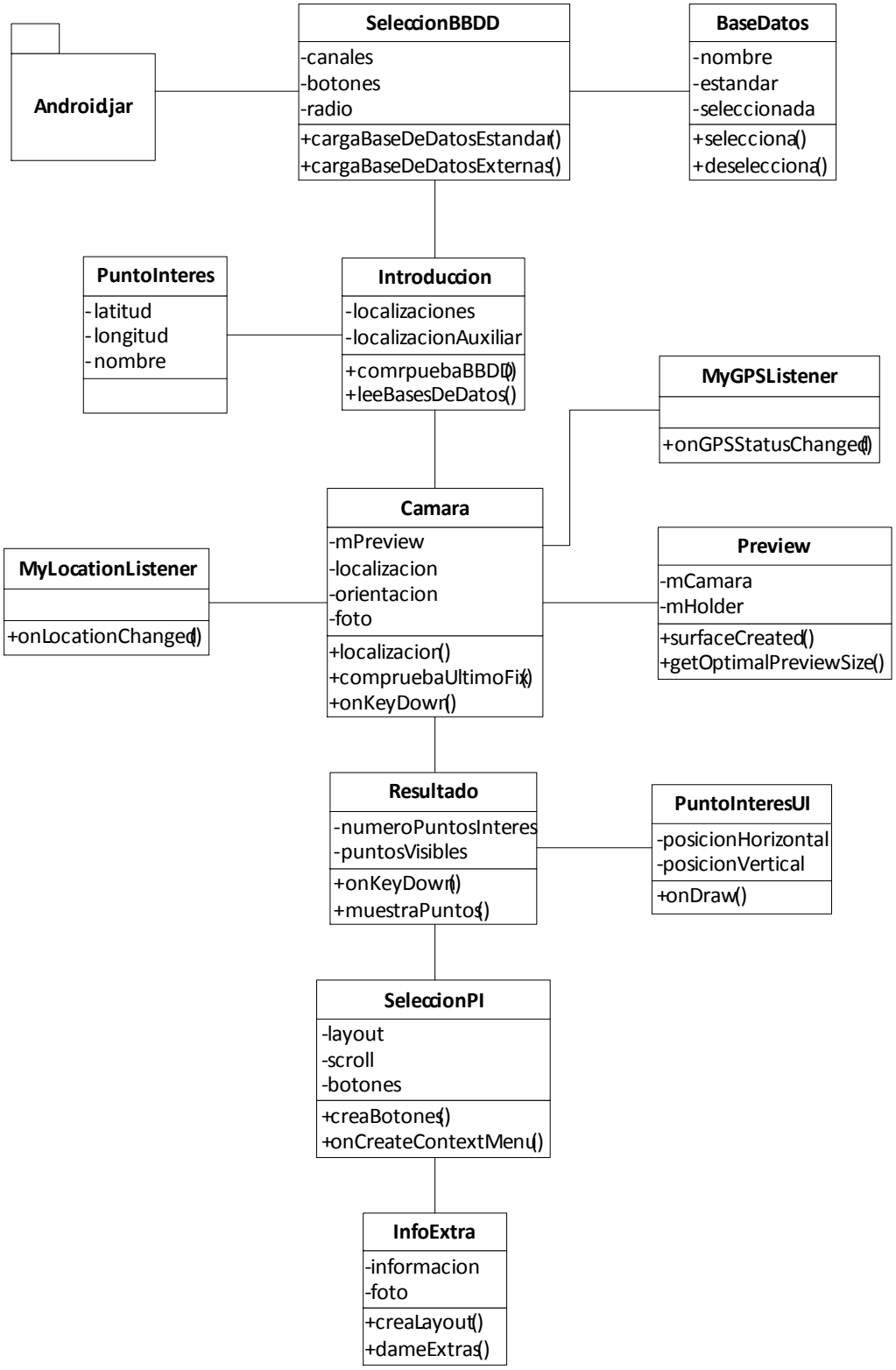


Diagrama de clases de la aplicación

Las clases que son Activitys se muestran en el eje principal del diagrama, es decir en el medio. Como se puede apreciar, aparte de algunas clases auxiliares, la mayoría de clases son Activitys que se corresponde a la interfaz gráfica de usuario. Esto es debido a la arquitectura de Android, que está basada en crear Activitys por cada ventana en la que el usuario puede interactuar. Si se sigue el hilo principal, se ve que cada Activity crea otra, hasta llegar al final de la aplicación, por lo que se parece

bastante a un diagrama de flujos.

Además de las clases Activity, existen algunas auxiliares que se encargan principalmente de adquisición de datos, representación de éstos y llamadas al sistema operativo para obtener alguna funcionalidad (como la cámara o los datos GPS).

Aunque sólo la primera Activity está conectada a la librería Android.jar, todas las clases de este proyecto están conectadas a ella. Se ha suprimido esta conexión por motivos de simplicidad y de facilidad de lectura.

Bases de datos

La base de datos de la aplicación está escrita en el formato XML y viene precargada con unos pocos puntos de interés de Madrid capital. En caso de que el desarrollador o usuario quisiera engrosar esta base de datos o crear una nueva, introduciendo sus propios sitios de interés, debe introducir los datos correspondientes en este archivo XML. Estos datos son:

1. Latitud, en grados decimales.
2. Longitud, en grados decimales.
3. Nombre
4. Información adicional (opcional)
5. URL de la fotografía (opcional)

En el caso de que se quiera introducir una nueva base de datos deberá contener todos los campos descritos en el párrafo anterior. Además, para que la lectura del archivo por parte del programa funcione sin ningún tipo de fallo, será conveniente que se eliminen todos los saltos de línea. Es decir, todos los puntos de interés estarán en la misma linea, de esta manera se evitará cualquier tipo de incompatibilidad entre los sistemas (los saltos de linea se codifican de distinta manera en Linux y Windows).

El archivo debe tener la extensión .xml. Además el nombre de la base de datos debe de ser todo en minúsculas, sin caracteres especiales y sin espacios. Además el nombre deberá ser algo representativo de lo que es la base de datos, puesto que saldrá luego en el menú de selección de bases de datos con ese nombre.

El formato del archivo XML será el siguiente:

<CATALOGO><LOC><LATITUD>latitud_en_grados_decimales</LATITUD><LONGITUD>longitud_en_grados_decimales</LONGITUD><NOMBRE>nombre_del_punto_de_interes</NOMBRE><URL>URL_de_la_fotografía</URL></LOC><LOC><LATITUD>latitud_en_grados_decimales</LATITUD><LONGITUD>longitud_en_grados_decimales</LONGITUD><NOMBRE>nombre_del_punto_de_interés</NOMBRE><INFO>información_adicional</INFO></LOC></CATALOGO>

En el caso de que se quiera incluir esa base de datos como estándar en el paquete, se deberá introducir en la carpeta GuiaInteractiva\res\xml dentro del workspace de Eclipse. Además se deberá incluir la base de datos en el vector *canales*, para ello hace falta introducir la siguiente instrucción en el método cargaBasesDeDatosStandard:

```
canales.addElement(new BaseDatos("<nombre_bbdd>",true));
```

Testear nuevas funcionalidades o bases de datos de la aplicación

Tanto si se trata de publicar una base de datos nueva como si se trata de implementar una nueva funcionalidad, se debe realizar un testing previo a la publicación definitiva en la página web. Para realizar este testing se debe instalar la aplicación en un terminal físico o en el emulador de terminales. Para utilizar la primera opción, basta con crear un nuevo AVD y lanzar la aplicación. Si se quiere probar en un terminal físico se podrá utilizar Eclipse, para hacerlo habrá que seguir los siguientes pasos:

- Descargar los drivers del teléfono para el sistema operativo que se esté usando en la máquina de desarrollo, para buscarlos se utilizará la siguiente página: http://developer.android.com/sdk/oem-usb.html
- Activar en el terminal el modo *Depuración USB* de la siguiente manera:
  Settings → Application Settings y activar Unknown Sources.
  Y además:
  Settings → Application Settings → Development y activar USB Debugging.
- Abrir el eclipse y lanzar la aplicación, cuando aparezcan los diferentes terminales objetivo, seleccionar el físico (el otro debería ser el emulado)

Una vez completados todos los pasos, la aplicación estará instalada en el terminal de igual manera que todas las demás, con la ventaja de que, mientras esté el USB conectado, podremos ver en la vista DDMS de Eclipse, las distintas trazas que va dejando tanto el programa como el sistema operativo.

Crear un nuevo AVD

Para crear un nuevo terminal emulado desde Eclipse se deberán seguir los siguientes pasos:

- Abrir el menú *Android SDK and AVD Manager* y seleccionar *new*.
- Ponerle un nombre en *name*, seleccionar como *target* Android 2.1-update1.
- Poner en tamaño de tarjeta SD, 512 MB.
- En la sección de *hardware* seleccionar:
  - SD card support
  - GPS
  - Acelerómetro
  - Cámara

Una vez completados estos pasos, ya se tendrá listo un terminal Android emulado, que se arrancará cada vez que se ejecute la aplicación. También se puede dejar corriendo e ir ejecutando la aplicación ya que la reinstalará.

Agregar una nueva base de datos externa a un terminal virtual Android (AVD)

En el caso de que sea una nueva base de datos lo que se quiere probar se deberá subir el archivo .xml a la carpeta de descargas de la tarjeta SD. En el caso de que la aplicación se haya instalado en un dispositivo físico esto se puede hacer mediante un cable USB conectado al ordenador que la contenga. En caso contrario, si se ha utilizado el emulador, habrá que introducir el archivo al sistema de ficheros emulado.

En el caso de que se quiera introducir archivos externos, simulando que han sido descargados de internet por el terminal emulado, deberán introducirse en la tarjeta sd. Para ello, en el emulador habrá que montar el sistema de archivos para simularla. Si el terminal se ha creado siguiendo las instrucciones dadas en la sección anterior, no habrá que hacer nada para montar el sistema de archivos, más que continuar siguiendo las instrucciones que muestran cómo crear una carpeta en la tarjeta SD.

Montar la sdcard por comando

En el caso de que no se haya montado el sistema de archivos, habrá que montarlo expresamente (y cada vez que se arranque de nuevo el emulador). Para ello se introducirá en el terminal, situándolo en la carpeta “tools” del sdk de Android(normalmente: C:\Program Files\Android\android-sdk-windows\tools), el siguiente comando:

mksdcard 512M sdcard.iso

Con este comando montamos una tarjeta sd de 512MBytes. Una vez montado el sistema de archivos se arrancará el terminal desde la consola, para que cargue los cambios, con el siguiente comando:

emulator -avd <nombre_del_terminal> -sdcard sdcard.iso

Crear una carpeta en la tarjeta sd por comando

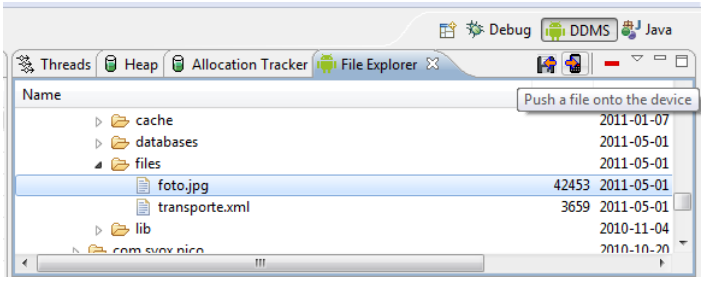
Para crear una carpeta por linea de comandos (terminal o consola) se debe introducir la siguiente instrucción estando en la carpeta de instalación de Android SDK:

adb shell "mkdir /sdcard/download"

De esta manera ya se tendría la tarjeta montada y una carpeta llamada download, que simula la carpeta donde se descargarían los archivos descargados en un terminal real.

Introducir archivos en la carpeta privada de la aplicación

Para introducir archivos en el teléfono inteligente virtual, se ha utilizado la Dalvik Debug Monitor Server (DDMS) de Eclipse. DDMS es una herramienta de depuración que proporciona varios servicios muy útiles a la hora de desarrollar aplicaciones para Android. En este proyecto se ha utilizado para guardar las bases de datos en el teléfono virtual. Para hacerlo debemos abrir la perspectiva DDMS en Eclipse, seleccionar el botón “Push a file onto the device” que abrirá una ventana en la que se elige el archivo que se quiere subir al terminal.



Perspectiva DDMS para copiar archivo en carpeta interna de la aplicación

Crear un paquete APK instalable

Para obtener el paquete APK de un proyecto que está siendo desarrollado en

Eclipse basta con seguir la siguiente ruta:

<ruta del workspace>\GuiaInteractiva\bin\GuiaInteractiva.apk

Se presupone que el nombre del proyecto no se ha cambiado. El nombre del paquete APK será el mismo que el del proyecto.

Anexo L: Factory Acceptance Test

Este documento, escrito por el desarrollador Elias Pardo de Donlebún Matilla, y firmado por su tutor de proyecto, certifica que:

1. Se ha desarrollado y se encuentra disponible la siguiente documentación:

    Documentación del desarrollo

    Documentación del código

    Documentación de validación

    Documentación de mantenimiento y desarrollo futuro

    Memoria del proyecto

    Manual de usuario y administrador

2. Los elementos de hardware y software utilizados para realizar las prruebas durante el desarrollo del proyecto son las siguientes:

    Samsung Galaxy 3 con las siguientes características:

        Android 2.1 update-1, oficial

    Samsung Galaxy S con las siguientes características:

        Android 2.2, oficial

    Samsung Galaxy S con las siguientes características:

        Android 2.3, no oficial

    HTC Desire HD con las siguientes características:

        Android 2.3, oficial

    HTC Wildfire con las siguientes características:

        Android 2.1, oficial

3. El software funciona de acuerdo con los requerimientos fijados por el usuario. Éstos requisitos se validan mediante prototipos.

4. La batería de pruebas realizada, incluida en la documentación de validación, se han pasado satisfactoriamente.

Madrid, a 1 de Septiembre de 2011.

Elias Pardo de Donlebún Matilla Mario Muñoz Organero, tutor

BIBLIOGRAFÍA

Butler, J. “Rapid Application Development in Action”. Managing System Development, Applied Computer Research, 14(5). 6 a 8 de Mayo de 1994.

Dourish, Paul. Seeking a Foundation for Context-Aware Computing. University of California, Irvine.

Weiser; Gold; Brown. "Ubiquitous computing". http://www.research.ibm.com/journal/sj/384/weiser.html. 11 de Mayo de 1999.

Ishii, H. and Ullmer, B. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. Proc. ACM Conf. Human Factors in Computing Systems CHI’97 (Atlanta, GA). New York: ACM. 1997

William Noah Schilit, A system architecture for context-aware mobile computing, Columbia University, New York, NY, 1995

Azuma, Ronald T. A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments 6, 4 (Agosto de 1997), 355 - 385.

Milgram, Paul, and Fumio Kishino. A Taxonomy of Mixed Reality Virtual Displays. EICE Transactions on Information and Systems E77-D, 9 (Septiembre de 1994), 1321-1329.

Schilit, B., Adams, N. Want, R. Context-Aware Computing Applications. 1st International Workshop on Mobile Computing Systems and Applications. 1994. pp 85-90

Steven Feiner, Blair MacIntyre, Tobias Hollerer, Anthony Webster, "A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment," iswc, pp.74, First International Symposium on Wearable Computers (ISWC '97), 1997

Fuchs, Henry. Evolution of Graphics Hardware: 40 years since Sutherland’s HMD. University of North Carolina at Chapel Hill. Graphics Hardware 2008, Sarajevo. (http://www.google.es/url?sa=t&source=web&cd=10&ved=0CGYQFjAJ&url=http%3A%2F%2Fwww.graphicshardware.org%2Fpresentations%2Ffuchs-evolution_of_graphics_hardware.ppt&ei=hsilTdDAN4ms8gOShsm5Dw&usg=AFQjCNENxX-tF8q3_BCt-f_N3AWprVxCyA&sig2=DbRqUP2bNNKQc7-hmZGiTg)

Höllerer, Tobias H., Feiner, Steven K. Mobile Augmented Reality. Telegeoinformatics: Location-Based Computing and Services. H Karimi and A. Hammad (eds.). Taylor & Francis Books Ltd., 01/2004. Capítulo 9.

J. Kjeldskov, “Lessons From Being There: Interface Design For Mobile Augmented Reality” to appear in L. Qvortrup (ed.) Virtual Applications: Applications With Virtual Inhabited 3D Worlds, Berlin, Springer-Verlag, 2003.

Reinhard Oppermann and Marcus Specht. Adaptive support for a mobile museum guide. In Proceedings of Workshop on Interactive Applications of Mobile Computing

(IMC '98), Rostock, Alemania, Noviembre de 1998. Neuer Hochschulschriftverlag.

Benjamin B. Bederson. Audio augmented reality: A prototype automated tour guide. In Proceedings of Conference on Human Factors and Computing Systems, CHI '95, pages 210-211, Denver, CO, Mayo de 1995. ACM Press.

George W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. Communications of the ACM, 36(7):39-49, Julio de 1993.

G. Papagiannakis, G. Singh and N. Magnenat-Thalmann, "A survey of mobile and wireless technologies for augmented reality systems," Comput.Animat.Virtual Worlds, vol. 19, pp. 3-22, Febrero de 2008.

A. Cheok, S. Fong, K. Goh1, X. Yang, W. Liu, F. Farzbiz, Y. Li, "Human Pacman: A Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction over a Wide Outdoor Area", Human-Computer Interaction with Mobile Devices and Services, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003.

G. Klein and T. Drummond, "Sensor Fusion and Occlusion Refinement for Tablet-based AR", in Proc. Third IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR'04, Arlington, 2004

David Mertz, XML Matters: Putting XML in context with hierarchical, relational, and object-oriented models, 2001. (http://www.ibm.com/developerworks/library/x-matters8/index.html)

Kimbro Staken, Introduction to Native XML Databases, 2001. (http://www.xml.com/pub/a/2001/10/31/nativexmldb.html)

O'Connell, S. Advanced Databases Course Notes, Southampton, University of Southampton, 2005.

Hewitt, C., C. Manning, "Joint Information Services Architecture for Mobile Distributed Telecomputing,". Transparencias, 1993.

Clint Boulton, Google, Apple to Spar in Mobile Augmented Reality. (http://www.eweek.com/c/a/Mobile-and-Wireless/Google-Apple-to-Spar-in-Mobile-Augmented-Reality-534060/)

Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavári, Zs., Encarnação, M., Gervautz, M., Purgathofer, W., "The Studierstube Augmented Reality Project", PRESENCE -Teleoperators and Virtual Environments, MIT Press, 2002.

Hughes, C. E., Stapleton, C. B., Hughes, D. E. Smith, E., "Mixed Reality in Education, Entertainment and Training: An Interdisciplinary Approach," IEEE Computer Graphics and Applications, 26(6), Noviembre y Diciembre de 2005, 24-30.

Dey, A. (2000) Providing Architectural Support for Building Context-Aware Applications, PhD Thesis, College of Computing, Georgia Institute of Technology, pp. 170.

Dey, A. K., Salber, D., and Abowd, G. D. (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human-Computer Interaction, 16, pp, 1-67. 2001.

Dey, A. (2001) Understanding and Using Context, Personal Ubiquitous Computing, 5(1), pp. 4-7.

Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K. (2002) Gaia: A middleware infrastructure for active spaces. IEEE Pervasive Computing, Special Issue on Wearable Computing 1, pp. 74–83. 2002.

Mikalsen, M., Paspallis, N., Floch, J., Stav, E., Papadopoulos G. A., Ruiz, P. A. (2006) Putting Context in Context: The Role and Design of Context Management in a Mobility and Adaptation Enabling Middleware, International Workshop on Managing Context Information and Semantics in Mobile Environments (MCISME'06), Nara, Japón. IEEE Computer Society Press, pp. 76-83. 9 al 12 de Mayo de 2006

Satyanarayanan, M. (2001) Pervasive Computing: Vision and Challenges, IEEE Personal Communications Magazine, pp. 10-17. 2001.

Korkea-Aho, M. (2006). Context-Aware Applications Survey. Reportaje Técnico. 3.6 Summary of Existing Applications. 2006.

Jonietza, E., "10 Emerging Technologies – Augumented Reality", Technology Review,

Marzo/Abril de 2007.
(http://www.technologyreview.com/read_article.aspx?ch=specialsections&sc=emerging&id=18291)

Banâtre, M., Couderc, P., Pauty, J. and Becus, M.. "Ubibus: Ubiquitous Computing to Help Blind People in Public Transport", Proc. Mobile HCI 2004 – 6th International Symposium, Glasgow, Reino Unido. pp:310-314. 13 a 16 de Septiembre de 2004.

Franco. Smartphones Popularity Increase Leads to More Mobile Media Consumption. 8 de Marzo de 2011 (http://www.pragermicrosystems.com/blog/technology-news/smartphones-popularity-increase-leads-to-more-mobile-media-consumption/)

Cassavoy, Liane. What Makes a Smartphone Smart?. About.com. Sin fechar. (http://cellphones.about.com/od/smartphonebasics/a/what_is_smart.htm)

comScore. European Smartphone Market Grows 41 Percent in Past Year. 14 de Septiembre de 2010(1). ttp://www.comscore.com/Press_Events/Press_Releases/2010/9/European_Smartphone_Market_Grows_41_Percent_in_Past_Year

Clarke, Peter. Android wins 35% of Q1 smartphone market. 9 de Mayo de 2011. (http://www.eetimes.com/electronics-news/4215816/Android-captures-35--of-Q1-smartphone-market)

Analysys Mason. 1.7 billion smartphones by 2014, says Analysys Mason. 28 de Abril de 2010. (http://www.analysysmason.com/About-Us/News/Press-releases/17-billion-smartphones-by-2014-says-Analysys-Mason/)

Whitney, Lance. Smartphones to dominate PCs in Gartner forecast. 14 de Enero de 2010.(http://news.cnet.com/8301-1001_3-10434760-92.html)

Small Business Lab. Smartphone Sales to Over Take Feature Phone Sales in 2011.12 de Mayo de 2010. (http://www.smallbizlabs.com/2010/05/smartphone-sales-to-over-take-feature-phone-sales-in-2011.html)

Meyers, Jason. Mobile app downloads hit 8 billion in 2010 . 14 de Marzo de 2011. (http://www.mobiledevproonline.com/article/mobile-apps/mobile-app-downloads-hit-8-billion-in-2010)

Gartner. Gartner Says Consumers Will Spend $6.2 Billion in Mobile Application Stores in 2010 18 de Enero de 2010. (http://www.gartner.com/it/page.jsp?id=1282413)

comScore. Social Networking Ranks as Fastest-Growing Mobile Content Category. 2 de Junio de 2010(2). (http://www.comscore.com/Press_Events/Press_Releases/2010/6/Social_Networking_Ranks_as_Fastest-Growing_Mobile_Content_Category)

Racoma, J. Angelo. Android Smartphone Sales Saw 1,580% Growth in Europe. 10 de Marzo de 2011.(http://nexus404.com/Blog/2011/03/10/android-smartphone-sales-saw-1580-growth-in-europe-android-is-now-dominant-smartphone-in-europe-with-1580-year-on-year-growth-according-to-idc/)

Sharkey, Jeffrey. Coding for Life -- Battery Life, That Is. 27-28 Mayo de 2009. (http://www.google.com/intl/es-ES/events/io/2009/sessions/CodingLifeBatteryLife.html)

Pérez Rodriguez, Iván. Estudio de la plataforma de software Android para el desarrollo de una aplicación social. 28 de Febrero de 2009

MIThril Inference Engine: http://www.media.mit.edu/wearables/mithril/phone.html

Datos de la tabla comparativa:
http://www.abiresearch.com/press/3651-Android+Will+Seize+45%25+of+Smartphone+Market+by+2016%2C+Says+ABI+Research
http://www.quesabesde.com/noticias/smartphones-2016-abi-research-android-ios-blackberry-bada-windows-phone,6_7478
http://www.research2guidance.com/smartphone-application-market-to-reach-us15-65-billion-in-2013/

GLOSARIO DE TÉRMINOS

PDA: Personal Digital Assistant. Es un dispositivo móvil que gestiona información personal. Hoy en día es común que tengan acceso a Internet. Pueden tener también capacidades multimedia, como reproductor de archivos de audio o reproductor de video.

GPS: Global Positioning System. Es un sistema satelital de navegación global que proporciona fecha y localización mientras se tenga visión con al menos a cuatro satélites.

RAD: Rapid Application Development. Es una metodología de desarrollo de software que minimiza la planificación, en aras de la obtención de prototipos en fases iniciales del desarrollo.

ADT: Android Development Tool. Es un plugin para el entorno de desarrollo Eclipse que permite desarrollar aplicaciones Android.

QEMU: Quick EMUlator. Es un software genérico y de código abierto que emula y virtualiza máquinas. Permite correr aplicaciones diseñadas para máquinas específicas (por ejemplo una placa ARM) en otras máquinas (un PC).

AVD: Android Virtual Device. Es una configuración para el emulador de Android que permite modelar el terminal basado en ciertos parámetros modificables. Para crear un AVD se utiliza el AVD Manager.

APK: Android Package. Es el formato de los ficheros utilizados para distribuir aplicaciones Android. Contiene las siguientes carpetas en la raíz: META-INF y res. Además contiene los siguientes ficheros: AndroidManifest.xml classes.dex y resources.arsc.

Telnet: Es un protocolo de redes utilizado para mantener comunicaciones textuales bidireccionales usando una conexión terminal virtual, tanto en redes de área local como Internet.

Android SDK: Android Software Development Kit. Es un conjunto de herramientas que permiten el desarrollo de software para terminales Android.

RAM: Random Access Memory. Es una manera que tienen los ordenadores de almacenar información. Permiten un acceso mucho más rápido que al disco duro.

WiFi: Es un mecanismo que permite la interconexión de dispositivos electrónicos. Un dispositivo con capacidad WiFi puede conectarse a Internet de manera inalámbrica.

GPS fix: Es la posición obtenida gracias al sistema GPS. En caso de obtener un fix del GPS se sabrá la posición actual.

Booleano: es un tipo de dato que sólo admite dos valores, true o false. Se utiliza sobretodo para realizar operaciones de algebra booleana.

Activity: Una Activity es una sola cosa que el usuario tiene que hacer y que además tiene el foco. Casi todas interactúan con el usuario, de manera que la clase se encarga de crear una ventana donde plantar la interfaz de usuario.

Plug-in: Se trata de una serie de componentes de software que añaden funcionalidades o capacidades a un programa mayor.

Display: Pantalla de visualización.

Debug: Proceso de identificar y corregir errores de programación.

Backend: Es el módulo que se encarga de recibir la información del frontend y procesarla.

MIT: Massachusetts Institute of Technology. Instituto Tecnológico de Massachusetts.

View: Paquete dentro de la librería Android que permite el acceso a las clases

que manejan en la disposición de las ventanas e interacción con el usuario.

NASA: National Aeronautics and Space Administration. Es una agencia del gobierno de los Estados Unidos responsable del programa espacial civil y de la investigación aerospacial y aeronáutica.

IVA: Impuesto de Valor Añadido. Impuesto indirecto sobre el consumo.

API: Application Programming Interface. Es un conjunto de reglas y especificaciones que pueden ser seguidas por los programas para comunicarse entre ellos. Sirve como interfaz entre los distintos programas.

UMPC: Ultra-Mobile Personal Computer. Es una versión reducida y portátil de un PC. Son más pequeños que los netbooks y se pueden considerar como los hermanos pequeños de éstos.

WLAN: Wireless Local Area Network. Permite la interconexión de dos o más dispositivos mediante una conexión inalámbrica y que normalmente se hace mediante un punto de acceso, que permite acceso a Internet.

Punto de acceso: Es un dispositivo que permite conectar otros dispositivos a una red cableada mediante conexiones inalámbricas. Normalmente están conectados a, o integran, un router, que permite la conexión entre las dos subredes (la cableada y la inalámbrica).

Punto de interés: Es un punto o localización donde puede haber algo relevante o de interés. Un punto de interés GPS, como poco, debe especificar latitud y longitud. Suelen incluir un nombre o una descripción.

SMS: Short Message Service. Es un componente, normalmente integrado en los teléfonos móviles, que permite el intercambio, mediante un protocolo estandarizado, de mensajes de texto cortos. Es la aplicación de datos más extendida en el mundo.

QWERTY: Es la disposición más común de los teclados de ordenadores y máquinas de escribir. Se diseñó para que las teclas de las máquinas de escribir no se trabaran al escribir, además de conseguir que los usuarios usaran las dos manos para escribir, de manera que lo hicieran más rápido.

Bluetooth: Es un estándar que permite el intercambio de información con corto alcance sobre enlaces entre los 2400-2480MHz. Al ser de corto alcance, crean PAN (Personal Area Network) de con alto nivel de seguridad.

Middleware: Es software que se encarga de interconectar softwares o éstos y los usuarios. Permite una mayor interoperabilidad sobretodo para aplicaciones distribuidas.

Netbook: Son ordenadores portátiles pequeños y de bajo costo. Suele usarse para tareas básicas como navegar por Internet o procesar textos.