Títol: **PORTANT LA REALITAT AUGMENTADA AL MAC**

Autor: JOSE MARCOS RÍOS SÁNCHEZ

Data: 12/06/2012

Director: PAU FONSECA CASAS

Departament: LABORATORI D'ESTADÍSTICA I INVESTIGACIÓ OPERATIVA DE LA FACULTAT D'INFORMÀTICA DE BARCELONA (LCFIB)

Titulació: ENGINYERIA TÈCNICA EN SISTEMES INFORMÀTICS (ETIS)

Centre: **Facultat d'Informàtica de Barcelona (FIB)**

Universitat: Universitat Politècnica de Catalunya (UPC)

BarcelonaTech

A Marta,

por toda su paciencia y amor

Agradecimientos

A Pau Fonseca, por haberme dado la oportunidad de realizar este proyecto final de carrera y aprender tanto.

A Raúl Sáez, por compartir esfuerzos, tanto en este proyecto como en el suyo. Suerte!

A todos los profesores que he tenido en mi vida. Os admiro.

A mi hermana Maria, por devolverme al camino del conocimiento.

A mi hermana Lucy, por su paciencia con lo difícil que siempre se lo pongo.

A mi hermano Alfredo, por ser tan, tan grande y cariñoso.

A mi cuñado Marcos por ilustrarme tanto.

A mi cuñada Carol por contagiarme su alegría.

A mis padres, por darme todo lo que tengo y creer en mí.

A mis sobrinas Natalia y Malena, el motivo de todo esto.

A Marta, siempre a mi lado.

Índice

1. **Introducción 5**

1.1. ¿Qué es la Realidad Aumentada? 7

1.1.1. ¿Qué funcionalidades puede ofrecer? 7

1.2. Aplicaciones para dispositivos móviles. 10

1.2.1. Aplicaciones multiplataforma. 11

1.2.2. Aplicaciones nativas versus aplicaciones *web*. 11

1.2.3. Aplicaciones híbridas para dispositivos móviles. 12

1.3. Algunos conceptos básicos. 13

1.3.1. Android. 13

1.3.2. iOS de Apple. 14

1.3.3. jQuery Mobile. 15

1.3.4. AJAX. 16

1.3.5. Document Object Model (DOM). 16

1.4. Escenario del Proyecto. 17

1.5. Organización de la Memoria 17

2. **Análisis del proyecto 19**

2.1. Resumen del proyecto. 19

2.2. Oportunidad de negocio. 20

2.3. Propósito. 20

2.4. Objetivos. 21

2.5. Conceptualización. 21

2.6. Alcance. 233

3. **Análisis de requerimientos 24**

3.1. Requerimientos funcionales. 24

3.1.1. Actores. 24

3.1.2. Requerimientos funcionales de la aplicación. 25

3.1.3. Requerimientos funcionales del dispositivo móvil. 25

3.1.4. Requerimientos funcionales de realidad aumentada. 25

3.2. Requerimientos no funcionales. 26

3.2.1. Requerimientos no funcionales del dispositivo móvil. 26

3.2.2. Requerimientos no funcionales de la aplicación. 27

4. **Análisis de las tecnologías existentes 29**

4.1. Análisis del entorno *Mono for Android*. 29
4.1.1. ¿Qué es *Mono for Android*? 29
4.1.2. Características generales de *Mono for Android.* 30
4.1.3. Ventajas e inconvenientes de *Mono for Android.* 30
4.1.4. Pruebas realizadas con *Mono for Android.* 32
4.2. Análisis del *framework PhoneGap*. 34
4.2.1. ¿Qué es *PhoneGap*? 34
4.2.2. Ventajas e inconvenientes de *Phonegap.* 34
4.3. Análisis del *framework AppCelerator Titanium*..... 37
4.3.1. ¿Qué es *AppCelerator*?..... 37
4.3.2. Ventajas e inconvenientes de *AppCelerator.* 37
4.4. Análisis de Tecnologías de Realidad Aumentada. 38
4.4.1. Navegadores de Realidad Aumentada 38
4.4.2. Ventajas y desventajas de los navegadores *AR*..... 38
4.4.3. Kits de desarrollo de Realidad Aumentada. 39
4.4.4. Pruebas realizadas con Realidad Aumentada. 41
4.4.5. Conclusiones respecto a la Realidad Aumentada. 413
5. **Tecnologías utilizadas 44**
5.1. *PhoneGap*. 44
5.2. *jQuery Mobile*. 45
6. **Especificación 47**
6.1. Esquema previo de la interfaz de usuario. 47
6.2. Modelo de casos de uso. 53
6.2.1. Mostrar Menú. 53
6.2.2. Mostrar Submenú. 54
6.2.3. Volver a menú anterior. 55
6.2.4. Mostrar Punto Interés. 55
6.2.5. Localizar Punto Interés. 56
6.2.6. Activar/Desactivar Actualizaciones al inicio. 57
6.2.7. Mostrar Mapa..... 57
6.2.8. Vista Realidad Aumentada. 58
6.2.9. Cambiar Idioma. 58
6.2.10. Buscar Actualizaciones. 59
6.2.11. Actualizar Aplicación. 60

6.2.12. Publicar Actualización. 60
6.3. Modelo de estados. 61
6.3.1. Ciclo de vida de una aplicación Android. 61
6.3.2. Diagrama de estados de arranque de la aplicación. 63
7. **Implementación 64**
7.1. Entornos para el desarrollo del sistema. 64
7.1.1. Eclipse IDE. 64
7.1.1. Adobe Dreamweaver. 65
7.2. Estrategia de actualización de la aplicación. 67
7.3. Diagramas de secuencia. 70
7.3.1. Diagrama de secuencia de arranque de la aplicación. 70
7.3.2. Diagrama de secuencia de búsqueda de actualizaciones. 73
7.3.3. Diagrama de secuencia de actualización de la aplicación. 74
7.3.4. Diagrama de secuencia Act./Desactivación de actualizaciones en el arranque. ... 76
7.3.5. Diagrama de secuencia Cambiar idioma. 77
7.4. Modificación de contenidos y publicación de actualizaciones. 78
7.5. La aplicación implementada. 789
8. **Resultados 90**
8.1. Carga de trabajo. 900
8.2. Coste del proyecto. 900
8.3. Conclusiones. 922
8.3.1. Objetivos conseguidos. 922
8.3.2. Opinión personal. 944

Capítulo 1
Introducción.

El propósito de este primer capítulo es el de proporcionar una visión general del proyecto, así como introducir los principales conceptos necesarios para la comprensión del proyecto y sus objetivos. Para ello se describen varios términos y tecnologías, como por ejemplo el concepto de Realidad Aumentada.

A modo de apoyo, al final de este capítulo se ofrece una guía donde se explican brevemente los contenidos del resto de capítulos de esta memoria.

1.1. ¿Qué es la Realidad Aumentada?

La *realidad aumentada* (RA) es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real. Esta es la principal diferencia con la *realidad virtual*, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. La información artificial sobre el medio ambiente y los objetos pueden ser almacenada y recuperada como una capa de información en la parte superior de la visión del mundo real.

1.1.1. ¿Qué funcionalidades puede ofrecer?

La realidad aumentada ofrece infinidad de nuevas posibilidades de interacción, que hacen que esté presente en muchos y varios ámbitos, como son la arquitectura, el entretenimiento, la educación, el arte, la medicina o las comunidades virtuales. A continuación se describen algunos campos en los que se aplica la Realidad Aumentada.

En proyectos educativos, actualmente la mayoría de aplicaciones de realidad aumentada se usan en museos, exhibiciones, parques de atracciones temáticos, etcétera, puesto que su coste todavía no es suficientemente bajo para que puedan ser empleadas en el ámbito doméstico. Estos lugares aprovechan las conexiones inalámbricas para mostrar información sobre objetos o lugares, así como imágenes virtuales como por ejemplo ruinas reconstruidas o paisajes tal y como eran en el pasado, además de escenarios completos en realidad aumentada, donde se pueden apreciar e interactuar con los diferentes elementos en 3D, como partes del cuerpo. Una de las primeras aplicaciones en formación es un sistema de realidad aumentada para aprender a soldar sin riesgos.

En el campo de la cirugía, la aplicación de realidad aumentada permite al cirujano superponer datos visuales como por ejemplo *termografías* o la delimitación de los bordes limpios de un tumor, invisibles a simple vista, minimizando el impacto de la cirugía.

En el ámbito del entretenimiento, teniendo en cuenta que el de los videojuegos es un mercado que mueve unos 30.000 millones de dólares al año en los Estados Unidos, es comprensible que se esté apostando mucho por la realidad aumentada en este campo puesto que ésta puede aportar muchas nuevas posibilidades a la manera de jugar. Una de las puestas en escena más representativas de la realidad aumentada es el "*Can You See Me Now?*", de *Blast Theory*. Es un juego *on-line* de persecución por las calles donde los jugadores empiezan en localizaciones aleatorias de una ciudad, llevan un ordenador portátil y están conectados a un receptor de *GPS*. El objetivo del juego es procurar que otro corredor no llegue a menos de cinco metros de ellos, puesto que en este caso se les hace una foto y pierden el juego. La primera edición tuvo lugar en Sheffield pero después se repitió en otras muchas ciudades europeas. Otro de los proyectos con más éxito es el *ARQuake Project*, donde se puede jugar al videojuego *Quake* en exteriores, disparando contra monstruos virtuales. A pesar de estas aproximaciones, todavía es difícil obtener beneficios del mercado de los juegos puesto que el *hardware* es muy costoso y se necesitaría mucho tiempo de uso para amortizarlo.

También se puede aplicar la realidad aumentada para simular vuelos y trayectos terrestres.

En lo que respecta a servicios de emergencias, la realidad aumentada puede servir para mostrar instrucciones de evacuación de un lugar. En el campo militar, puede mostrar información de mapas, localización de los enemigos...

En la arquitectura, la realidad aumentada es muy útil a la hora de resucitar virtualmente edificios históricos destruidos, así como proyectos de construcción que todavía están bajo plano.

Apoyando a tareas complejas, como el montaje, mantenimiento, y la cirugía, éstas pueden simplificarse mediante la inserción de información adicional en el campo de visión. Por ejemplo, para un mecánico que está realizando el mantenimiento de un sistema, las etiquetas pueden mostrar las partes del mismo para aclarar su funcionamiento. La realidad aumentada puede incluir imágenes de los objetos ocultos, que pueden ser especialmente eficaces para el diagnóstico médico o la cirugía. Como por ejemplo una radiografía de rayos vista virtualmente basada en la tomografía previa o en las imágenes en tiempo real de los dispositivos de ultrasonido o resonancia magnética nuclear abierta.

En los dispositivos de navegación, la realidad aumentada puede aumentar la eficacia para una variedad de aplicaciones. Por ejemplo, la navegación dentro de un edificio puede ser mejorada con el fin de dar soporte al encargado del mantenimiento de instalaciones industriales. Las lunas delanteras de los automóviles pueden ser usadas como pantallas de visualización frontal para proporcionar indicaciones de navegación y información de tráfico.

La realidad aumentada puede ser utilizada en el entorno industrial para comparar los datos digitales de las maquetas físicas con su referente real para encontrar de manera eficiente discrepancias entre las dos fuentes. Además, se pueden emplear para salvaguardar los datos digitales en combinación con prototipos reales existentes, y así ahorrar o reducir al mínimo la construcción de prototipos reales y mejorar la calidad del producto final.

En los campos de la hidrología, la ecología y la geología, la realidad aumentada puede ser utilizada para mostrar un análisis interactivo de las características del terreno. El usuario puede utilizar, modificar y analizar, tres mapas bidimensionales interactivos.

Una de las últimas aplicaciones de la realidad aumentada es la publicidad. Hay diferentes campañas que utilizan este recurso para llamar la atención del usuario.

Fiat ha lanzado una campaña en la que cualquier usuario puede crear su propio anuncio de televisión con el *Fiat 500* como protagonista a través de la página web, el usuario solo necesita tener una cámara web.

Plataformas como *Junaio* o *Layar* permiten el desarrollo de aplicaciones a terceros, prácticamente sin conocimientos técnicos, a través de sus servidores.

Esto ha fomentado la publicación de miles de aplicaciones sobre turismo, gincanas, exposiciones virtuales, etc.

La empresa austriaca *Mobilizy* ha desarrollado *Wikitude*. Al apuntar la cámara del móvil hacia un edificio histórico, el *GPS* reconoce la localización y muestra

información de la *Wikipedia* sobre el monumento. En Japón, *Sekai Camera*, de la empresa *Tonchidot*, añade al mundo real los comentarios de la gente acerca de direcciones, tiendas, restaurantes, etcétera. *Acrossair*, disponible en siete ciudades, entre ellas Madrid y Barcelona, identifica en la imagen la estación de metro más cercana. *Bionic Eye* y *Yelp Monocle*, en Estado Unidos, son ejemplos similares.

1.2. Aplicaciones para dispositivos móviles.

No mucho tiempo atrás, el diseño de aplicaciones para dispositivos móviles presentaba un escenario en el cual, a la hora de determinar las tecnologías en la implementación de las aplicaciones, era obvio que se debía utilizar el lenguaje de programación nativo de cada plataforma. Los fabricantes enfocaban los sistemas operativos de sus productos a un lenguaje de programación específico, mediante el cual se implementaban las aplicaciones para sus dispositivos. Así, por ejemplo, el desarrollo de aplicaciones para dispositivos *Apple* pasaba por la programación en el lenguaje *Objective-C*, para Palm el desarrollo era en el lenguaje *C*, y las aplicaciones para dispotivos *Blackberry* en el lenguaje *Java*. En este escenario, el mantenimiento de una aplicación que corriese en varios dispositivos móviles exigía el mantenimiento de varias aplicaciones escritas en lenguajes de programación diferentes.

Este escenario progresivamente ha ido variando con la proliferación de populares aplicaciones *web*, como *Facebook* o *Windows Live Hotmail*, las cuales no requieren ser instaladas en el dispositivo, ya que son ejecutadas en el servidor y gestionadas desde el navegador web instalado en el dispositivo. Actualmente, a grosso modo, es posible desarrollar dos tipos de aplicaciones para dispositivos móviles. En el tipo clásico clasificaríamos las aplicaciones nativas, es decir, escritas en el lenguaje nativo de la plataforma sobre la que se ejecutará la aplicación. El segundo tipo sería el de las aplicaciones web, que se ejecutan en un servidor web y se presentan en diferentes formatos a varios tipos de dispositivos móviles equipados con navegador web.

Recientemente se ha extendido un nuevo tipo de aplicaciones para dispositivos móviles. Estas aplicaciones corresponden a una combinación de ambas técnicas, aplicaciones desarrolladas en el lenguaje nativo de la plataforma, cuyo núcleo está formado en mayor o menor medida por código de lenguaje de páginas web, esto es, *HTML*, *CSS* y *Javascript*. En resumen, este tipo mixto de aplicaciones consiste en una página web envuelta en código nativo. El código web es ejecutado sobre una

plataforma de ejecución de código *HTML* que incorporan de fábrica los dispositivos móviles.

1.2.1. Aplicaciones multiplataforma.

Una plataforma es una combinación de *hardware* y *software* usada para ejecutar aplicaciones. En su forma más simple consiste únicamente en un sistema operativo, una arquitectura o una combinación de ambos. Generalmente el software está escrito de modo que depende de las características de una plataforma particular. Este tipo de software es conocido como aplicación nativa. En el mundo de las computadoras, el máximo exponente del software multiplataforma lo tenemos en la tecnología *Java*.

En lo que respecta al mundo de los dispositivos móviles, las tecnologías multiplataforma más empleadas se basan sobre todo en *HTML5*. Son varios los factores que hacen de esta tecnología la más extendida y con más proyección del escenario del software multiplataforma para dispositivos móviles, entre los cuales el principal sería el ser una evolución de uno de los lenguajes de programación más extendidos de todos los tiempos (*HTML*). Estas tecnologías multiplataforma permiten reducir parte del código dependiente de la plataforma (código nativo), lo cual reduce el coste de producción y mantenimiento de aplicaciones diseñadas para correr sobre varias plataformas de dispositivos móviles.

1.2.2. Aplicaciones nativas versus aplicaciones *web*.

En los últimos años, existe un gran debate en el mundo de las aplicaciones para dispositivos móviles, acerca de cuál será el futuro del desarrollo de aplicaciones, si se mantendrá el predominio de aplicaciones nativas que existe en la actualidad, o por lo contrario se impondrá el desarrollo de aplicaciones basadas en tecnología web. El hecho innegable es que en la actualidad, aún impera el desarrollo de aplicaciones nativas, debido a varios factores. Uno de los principales motivos por los cuales aún predominan las aplicaciones nativas es la eficiencia o rendimiento. Las aplicaciones nativas aprovechan mejor los recursos de los dispositivos sobre los que corren, debido a que fueron diseñadas específicamente para la plataforma sobre la que corren. La diferencia entre las capacidades de las aplicaciones móviles basadas en la Web y las aplicaciones nativas se está reduciendo rápidamente.

Por otra parte, los rápidos avances en *HTML5*, la nueva versión del lenguaje *HTML*, más adecuada para dispositivos móviles, y el duro trabajo por la especificación de estándares del *World Wide Web Consortium* (*W3C*) para crear interfaces estándar a través de dispositivos móviles, evidencia que las aplicaciones *web* pueden - o serán capaces - de hacer muchas de las cosas que pueden hacer las aplicaciones nativas.

1.2.3. Aplicaciones híbridas para dispositivos móviles.

En lo que respecta este proyecto, nos decantamos por un desarrollo híbrido. Se trata de un tipo de aplicación cuyo núcleo está escrito en *HTML5*, *CSS* y *Javascript*, encapsulado en un contenedor nativo.

Este tipo de aplicaciones híbridas permiten escribir parte de la aplicación en lenguaje nativo y parte de la aplicación en lenguaje *web*, aprovechando las ventajas de cada modo de programación. De este modo, se tiene acceso total al dispositivo en aquellos puntos en los que la aplicación lo requiera, como por ejemplo determinadas propiedades actualmente exclusivas de las aplicaciones nativas, como son el acceso a la cámara o el procesamiento de audio y vídeo. Otro motivo por el que sería preciso hacer uso de código nativo sería la velocidad de procesamiento. El lenguaje nativo es, por mucho, más eficiente que el lenguaje web por lo que rendimiento en ejecución se refiere, por lo que en aquellos casos en los que se requiera un alto rendimiento, se puede implementar en código nativo. En el presente proyecto, la interfaz de usuario se escribió en lenguaje *web*, mientras que la parte relativa a la realidad aumentada se delegará a código nativo, para optimizar el alto rendimiento que requieren las herramientas de realidad aumentada.

Este tipo de aplicaciones se basan en un contenedor nativo, el cual crea una instancia de una clase o componente nativo el cual se encarga de incrustar código web en la aplicación. En *iOS*, estaríamos hablando de la clase *UIWebView*, en *Android* se trataría de la clase *android.webkit.WebView*, etcétera.

Mediante código *web* (*HTML5*, *CSS3*, *Javascript*), se implementa la interfaz de usuario y la lógica de la aplicación. También se activan controladores nativos mediante mecanismos específicos del sistema operativo, atendiendo a las llamadas a través de *Javascript*.

1.3. Algunos conceptos básicos.

1.3.1. Android.

Android es un sistema operativo móvil basado en *Linux*, que está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas y otros dispositivos. Es desarrollado por la *Open Handset Alliance*, la cual es liderada por *Google*. Este sistema por lo general maneja aplicaciones como *Market* (*Marketing*) o su actualización, *PlayStore*.

Fue desarrollado inicialmente por *Android Inc*., una firma comprada por *Google* en 2005. Es el principal producto de la *Open Handset Alliance*, un conglomerado de fabricantes y desarrolladores de *hardware*, *software* y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con *Android* se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre. A nivel mundial alcanzó una cuota de mercado del 50,9% durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo con más cuota (*iOS* de *iPhone*).

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 400.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de *Android* (*Google Play*) sin tener en cuenta aplicaciones de otras tiendas no oficiales para *Android*, como pueden ser la *App Store* de *Amazon* o la tienda de aplicaciones *Samsung Apps* de *Samsung*. *Google Play* es la tienda de aplicaciones en línea administrada por *Google*, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación *Java*. No obstante, no es un sistema operativo libre de *malware*, aunque la mayoría de ello es descargado de sitios de terceros.

El anuncio del sistema *Android* se realizó el 5 de noviembre de 2007 junto con la creación de la *Open Handset Alliance*, un consorcio de 78 compañías de *hardware*, *software* y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. *Google* liberó la mayoría del código de *Android* bajo la licencia *Apache*, una licencia libre y de código abierto.

La estructura del sistema operativo *Android* se compone de aplicaciones que se ejecutan en un *framework Java* de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de *Java* en una máquina virtual *Dalvik* con compilación en tiempo de ejecución. Las bibliotecas escritas en *lenguaje C* incluyen un administrador de interfaz gráfica (*surface manager*), un *framework OpenCore*, una base de datos relacional

SQLite, una Interfaz de programación de *API* gráfica *OpenGL ES 2.0 3D*, un motor de *renderizado WebKit*, un motor gráfico *SGL*, *SSL* y una biblioteca estándar de *C Bionic*. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de *XML*, 2,8 millones de líneas de *lenguaje C*, 2,1 millones de líneas de *Java* y 1,75 millones de líneas de *C++*.

1.3.2. iOS de Apple.

iOS (anteriormente denominado *iPhone OS*) es un sistema operativo móvil de *Apple*. Originalmente desarrollado para el *iPhone*, siendo después usado en dispositivos como el *iPod Touch*, *iPad* y el *Apple TV*. *Apple, Inc*. no permite la instalación de *iOS* en *hardware* de terceros. Tenía el 26% de cuota de mercado de sistemas operativos móviles vendidos en el último cuatrimestre de 2010, detrás de *Google Android* y *Nokia Symbian*. En mayo de 2010 en los Estados Unidos, tenía el 59% de consumo de datos móviles (incluyendo el *iPod Touch* y el *iPad*).

La interfaz de usuario de *iOS* está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal).

iOS se deriva de *Mac OS X*, que a su vez está basado en *Darwin BSD*, y por lo tanto es un sistema operativo *Unix*.

iOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de “Servicios Principales”, la capa de “Medios” y la capa de “Cocoa Touch”. La versión actual del sistema operativo (*iOS* 5.0.1) ocupa más o menos 770 *megabytes*, variando por modelo.

Apple reveló la existencia de *iPhone OS* en la *Macworld Conference & Expo* del 9 de enero de 2007, aunque el sistema no tuvo un nombre oficial hasta que salió la primera versión beta del *iPhone SDK* un año más tarde, el 6 de marzo de 2008. Antes de esto se consideraba simplemente que el *iPhone* ejecutaba *OS X*. A partir de entonces se llamaría *iPhone OS*. El lanzamiento del *iPhone OS* tuvo lugar el 29 de junio de 2007.

El interés en el *SDK* aumentaría en meses siguientes debido al explosivo crecimiento de la plataforma *iPhone*, que se vio incrementado en septiembre de 2007 del *iPod Touch*, un dispositivo con las capacidades multimedia del *iPhone* pero sin la capacidad de hacer llamadas telefónicas.

El 27 de enero de 2010 Steve Jobs, CEO de *Apple*, anunció el *iPad*, un dispositivo muy similar al *iPod Touch* pero con un enfoque más orientado hacia la industria de contenidos. Este dispositivo, apoyado en una pantalla táctil de mayor dimensión, compartiría sistema operativo con sus dos exitosos hermanos, y vendría acompañado de una aplicación oficial para la compra y lectura de libros electrónicos, *iBooks*.

A fecha de abril de 2010 se estima por encima de 185.000 las aplicaciones disponibles para *iPhone OS* a través de la *App Store*. El 7 de junio de 2010, durante la presentación del *iPhone 4*, Steve Jobs anunció que *iPhone OS* pasaría a ser llamado oficialmente como *iOS*.

1.3.3. jQuery Mobile.

JQuery Mobile es un *framework web* optimizado para dispositivos táctiles (conocido además como una biblioteca *Javascript* o un *framework* para dispositivos móviles) que actualmente está siendo desarrollado por el equipo del proyecto *jQuery*. El desarrollo se centra en la creación de un marco compatible con una amplia variedad de teléfonos inteligentes y tabletas, hecha para responder al creciente pero heterogéneo mercado de tabletas y teléfonos inteligentes. El *framework* de *jQuery Mobile* es compatible con otros *frameworks* de aplicaciones móviles y plataformas de este tipo como *PhoneGap*, *Worklight* y muchos otros.

Es compatible con todas las principales plataformas móviles, así como todos los navegadores de escritorio más importantes, incluyendo *iOS*, *Android*, *Blackberry*, *Palm*, *Symbian* y *Windows Phone 7*. Construido sobre el núcleo de *jQuery*, tiene una curva de aprendizaje mínima para las personas que ya están familiarizados con la sintaxis de *jQuery*. Con temas predefinidos, aunque completamente personalizables, se muestra en escala en cualquier tamaño de pantalla. Hace uso de la tecnología *Ajax* para la navegación, con transiciones de página animadas, permitiendo limpiar las direcciones *URL*.

1.3.4. AJAX.

Acrónimo de *Asynchronous JavaScript And XML* (*JavaScript* asíncrono y *XML*), *AJAX* es una técnica de *desarrollo web* para crear aplicaciones interactivas o *RIA* (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. *JavaScript* es el lenguaje interpretado (*scripting language*) en el que normalmente se efectúan las funciones de llamada de *AJAX* mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en *XML*.

AJAX es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como *JavaScript* y *Document Object Model* (*DOM*).

1.3.5. Document Object Model (DOM).

El *Document Object Model* o *DOM* ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (*API*) que proporciona un conjunto estándar de objetos para representar documentos *HTML* y *XML*, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del *DOM*, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos *HTML* y *XML*, que es para lo que se diseñó principalmente. El responsable del *DOM* es el *World Wide Web Consortium* (*W3C*).

El *DOM* es una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como *ECMAScript* (*JavaScript*).

1.4. Escenario del Proyecto.

El proyecto surge del interés por parte del *Museu Arqueològic de Catalunya* (*MAC*) de desarrollar una aplicación para dispositivos móviles que potencie la experiencia de los visitantes de las instalaciones de la *Seu d'Olèrdola*, haciendo uso de las nuevas tecnologías de realidad aumentada. Para el *MAC* representa una buena oportunidad de acercar a un público cada vez más familiarizado con las tecnologías móviles la experiencia de visitar un museo arqueológico, ofreciendo a los clientes una nueva forma de recepción de la información en el mismo lugar de la visita.

El objetivo del presente proyecto es cubrir una primera etapa del desarrollo del proyecto real, diseñando la interfaz de usuario de la aplicación para la plataforma *Android*, teniendo en consideración que la aplicación se desarrollará también para la plataforma *iOS*. Inicialmente el objetivo fue el proyecto real completo, pero tras las tareas de investigación se concluyó que los objetivos del proyecto real sobrepasaban en mucho las posibilidades en tiempo y esfuerzo dedicados a un Proyecto Final de Carrera para una ingeniería técnica como es la que nos ocupa. Por ello, se tomó la decisión de centrar los esfuerzos en el desarrollo de una interfaz de usuario que fuera fácilmente exportable a otras plataformas, relegando la parte de realidad aumentada de la aplicación para una segunda fase del proyecto real, quedando esta fuera del alcance del presente proyecto. Es por esta motivación inicial de cubrir la totalidad del proyecto por la que, a pesar de no haberse implementado dichas herramientas de realidad aumentada, en el presente proyecto se incluye una parte importante de investigación de las tecnologías de realidad aumentada disponibles en la actualidad.

1.5. Organización de la Memoria

Se ha intentado organizar la memoria de manera que refleje en los principales capítulos de la misma las etapas que conforman el desarrollo de una aplicación, con el objetivo de plasmar sobre el papel cómo el proyecto evolucionó y se desarrolló.

Capítulo 1 - Introducción

Explicación del marco del proyecto y las motivaciones para realizarlo.

Capítulo 2 - Análisis del proyecto

Análisis de objetivos, definición de la estructura básica de la aplicación.

Capítulo 3 - Análisis de requerimientos

Análisis de requerimientos funcionales y no funcionales del sistema.

Capítulo 4 - Análisis de las tecnologías existentes

Estado del arte de las tecnologías, análisis y pruebas realizadas.

Capítulo 5 - Tecnologías utilizadas.

Descripción de las tecnologías empleadas en el proyecto, justificación de su elección.

Capítulo 6 - Especificación

Descripción y diagramas de casos de uso, diagramas de estados.

Capítulo 7 - Implementación

Explicación de los puntos más interesantes de la etapa de implementación de la aplicación. Diagramas de secuencia de algunos casos de uso.

Capítulo 8 - Conclusiones

Costes, objetivos alcanzados y problemas o mejoras surgidas durante el desarrollo del proyecto.

Capítulo 2
Análisis del proyecto.

En este apartado se analizarán y enfocarán los objetivos, así como los límites del proyecto. Además se definirá la estructura básica de la aplicación y su funcionamiento, se concretarán las directrices que conducirán a los objetivos propuestos, a evaluar los resultados y finalmente a obtener conclusiones.

2.1. Resumen del proyecto.

El presente proyecto final de carrera está basado en un proyecto de *software* real. El proyecto real consiste en una aplicación de realidad aumentada para el *Museu Arqueològic de Catalunya*, concretamente para la *Seu d'Olèrdola*, museo arqueológico sito en aquella ciudad. La aplicación sería diseñada para las plataformas *Android* y *iOS*. La primera parte del proyecto consistió, tras un primer análisis superficial de requisitos, en investigar las tecnologías existentes actualmente en el mercado, así como aquellas que están aún en etapas previas de desarrollo o expansión, que satisfagan al máximo los dichos requisitos, prestando especial interés a dos factores básicos en la toma de decisiones, los cuales son, el desarrollo de las herramientas de realidad aumentada necesarias para la aplicación, y la búsqueda de un *framework* de desarrollo de *software* para dispositivos móviles que fuese multiplataforma.

Tras las tareas de investigación, se tomó la decisión de dividir el proyecto real en dos partes. Por una parte, se desarrollaría una interfaz de usuario multiplataforma que pudiese ser utilizada en ambas plataformas (*Android* y *iOS*) compartiendo así gran parte de código de la aplicación. Por otra parte, una vez finalizada la primera fase, se desarrollaría todo lo concerniente a las herramientas de realidad aumentada, las cuales se implementarían en lenguaje nativo para cada plataforma.

Debido a esta decisión de dividir las tareas del proyecto real en dos fases, el presente proyecto final de carrera podría dividirse a su vez en dos partes claramente diferenciadas. Una parte inicial, en la cual se realizan labores de investigación de las tecnologías disponibles para la realización del proyecto real en su totalidad, y una segunda parte en la que nos centramos únicamente en el desarrollo de una interfaz de usuario multiplataforma.

2.2. Oportunidad de negocio.

Dado que estamos hablando de un proyecto real, la oportunidad de negocio en este caso es evidente, ya que el interés por parte del *MAC* en el desarrollo de la aplicación ha sido claro desde el principio del desarrollo del proyecto.

Por otra parte, existe una gran oportunidad de negocio subyacente en el presente proyecto. El *Museu Arqueològic de Catalunya* es una red de museos formada principalmente por seis sedes: *Barcelona*, *Empúries*, *Olèrdola*, *Centre d'Arqueologia Subaquàtica de Catalunya* (*CASC*), *Girona* i *Ullastret*. La *Seu d'Olèrdola* podría ser la primera de una serie de proyectos del mismo tipo para el resto de sedes. Es de destacar que, aparte de la oportunidad de negocio que supondría un proyecto para cada sede, el reaprovechamiento de los conocimientos y desarrollos realizados en el primer proyecto haría muy rentable el desarrollo de los subsiguientes.

2.3. Propósito.

Diseñar e implementar la interfaz de usuario de una aplicación para dispositivos móviles tipo *smartphone* (principalmente *Android* e *iOS)* que permita a los usuarios de la aplicación obtener información de forma más enriquecedora, mediante tecnologías de realidad aumentada, en una visita *in situ* al *Museo Arqueológico de Olérdola*. La aplicación permitirá identificar puntos de interés en el *Castell d'Olèrdola,* así como edificios, elementos arquitectónicos, objetos y/u otros elementos. La interfaz de usuario ha de ser multiplataforma, de manera que pueda ser utilizada y mantenida en todos los dispositivos móviles más extendidos en el mercado.

En una segunda etapa del proyecto real, mediante realidad aumentada, la aplicación ha de mostrar al usuario dónde se encuentran los puntos de interés respecto a la posición actual del dispositivo, con el fin de que el usuario pueda localizarlos y visitarlos. El usuario podrá seleccionar un punto de interés a su elección, pero además, el usuario podrá acceder a una vista de realidad aumentada donde se le mostrarán los puntos de interés que se encuentren próximos a la posición actual del dispositivo.

El proyecto se considera un primer prototipo, y como tal puede no estar exento de fallos o limitado en cuanto a funcionalidades. No se pretende crear la versión definitiva de la interfaz de usuario, aunque se intenta dejar reflejadas las bases para un futuro desarrollo, de ampliación o de mejora de las capacidades de la aplicación.

2.4. Objetivos.

- Crear la interfaz de usuario para una aplicación para dispositivos móviles tipo *smartphone* con sistema operativo *Android.* Dicha aplicación:
    - Permitirá identificar puntos de interés como edificios, elementos arquitectónicos, objetos y/o otros elementos.
    - Permitirá ver en qué dirección está un punto de interés respecto a la posición del dispositivo.
    - Permitirá mostrar una lista de todos los puntos de interés existentes.
    - Permitirá al usuario acceder a una vista de realidad aumentada donde se mostrarán los puntos de interés próximos a su posición actual.
    - Permitirá al usuario ver información sobre el *Museo Arqueológico de Olérdola*.
    - Permitirá al usuario ver información sobre un punto de interés que el usuario haya seleccionado.
- Investigar en la utilización de técnicas de realidad aumentada en la aplicación.
- Facilitar la portabilidad de la aplicación a la plataforma *iOS* en la medida de lo posible, compartiendo tanto código y herramientas de software como sea viable con la aplicación para *iPhone*, teniendo en cuenta este aspecto en la toma de decisiones de todo el proyecto.
- Interfaz sencilla y de fácil utilización.

2.5. Conceptualización.

En este punto nos centraremos en obtener una visión de alto nivel del proyecto real, identificando sus elementos básicos y las relaciones de éstos entre sí y con el entorno.

El usuario final interactúa con un dispositivo móvil tipo *smartphone* a través de la aplicación. Al iniciar la aplicación por primera vez, ésta comprueba si existen actualizaciones disponibles para su descarga, en cuyo caso ofrece al usuario la posibilidad de descargarlas e instalarlas en el dispositivo. Al finalizar esta función, la aplicación muestra la pantalla inicial.

En la base de todas las pantallas se presenta una barra fija de botones que estarán disponibles siempre para el usuario mientras navega por las diferentes pantallas de la aplicación, excepto en las pantallas que se haga uso de realidad

aumentada para mostrar información. Esta barra está compuesta por cuatro botones los cuales muestran al usuario la pantalla de inicio, la pantalla *Olèrdola*, la pantalla de opciones y la pantalla de ayuda.

En la pantalla de inicio se muestran botones que dan acceso a varias pantallas de segundo nivel, las cuales muestran cierta información acerca de la *Seu d'Olèrdola*, en lo referente a localización, tarifas, horarios, etcétera. También se muestra información sobre histórica e institucional sobre el *Museu Arqueològic de Catalunya* y la *Seu d'Olèrdola*.

En la pantalla *Olèrdola* se muestran varias opciones para acceder a lo que serían las herramientas de realidad aumentada. Se muestra una lista de puntos de interés del lugar. De cada punto de interés se muestra información específica de dicho punto, así como una herramienta de localización del punto de interés en las instalaciones. También se muestra un mapa del lugar en el que se ubican todos los puntos de interés listados anteriormente. Desde esta pantalla se accede también a la herramienta de realidad aumentada que guía al usuario en la visita a las instalaciones.

En la pantalla de opciones se ofrece la posibilidad de configurar el idioma de la aplicación seleccionando entre catalán, castellano e inglés, siendo el idioma por defecto de la aplicación aquel idioma que se detecte en el dispositivo en el momento del primer arranque, o en su defecto el inglés. También se puede configurar desde esta pantalla que la aplicación compruebe o no si existen actualización en el momento de arranque. Se ofrece también un botón que el usuario puede pulsar para comprobar en el instante si existen actualizaciones disponibles.

En la pantalla de ayuda, se ofrece información para el buen entendimiento del funcionamiento de la aplicación. También se explica al usuario el concepto de realidad aumentada. Se extiende también en este apartado información sobre el *Museu Arqueològic de Catalunya* y sobre la *Seu d'Olèrdola*.

La aplicación permite que pueda ser utilizada con el dispositivo en posición vertical (retrato) o en posición horizontal (apaisado). Al girar el dispositivo, la aplicación hace rotar los elementos que conforman la pantalla, adaptándolos a la posición real del dispositivo.

2.6. Alcance.

El alcance del proyecto debe incluir todos aquellos procesos que se requieren para asegurar que el proyecto contenga únicamente el trabajo definido. Hay que definir y controlar qué está y qué no está incluido dentro del proyecto.

- La interfaz de usuario debe ser actualizable a nivel de contenidos por parte del personal del *Museu Arqueològic de Catalunya* destinado a ello, sin necesidad de modificar el código fuente de la aplicación.
- La aplicación debe mostrarse al menos en tres idiomas, estos son, catalán, castellano e inglés. La aplicación ha de detectar automáticamente el idioma configurado en el dispositivo, de manera que el usuario no tenga que seleccionar activamente el idioma.
- La interfaz de usuario, aunque diseñada para la plataforma *Android*, debe poderse exportar a otras plataformas fácilmente, de manera que el mantenimiento de la aplicación para varias plataformas sea asumible.

Capítulo 3
Análisis de requerimientos.

En este apartado se definen los requerimientos del sistema, qué debe hacer, pero no cómo hacerlo, mostrando con claridad lo que se desea obtener. Los requerimientos de dividen en dos tipos: funcionales y no funcionales. En este punto se tratan también las tecnologías y dispositivos utilizados para el desarrollo del sistema.

3.1. Requerimientos funcionales.

Los requerimientos funcionales especifican las funcionalidades que ha de implementar la aplicación, es decir, lo que la aplicación ha de ser capaz de realizar. También se describen los actores que participan en el sistema. Aunque queden fuera del alcance del diseño de la interfaz de usuario, se añade a los requerimientos aquellos que hacen referencia a las herramientas de realidad aumentada que en una fase posterior se implementarán en la aplicación

3.1.1. Actores.

Existen dos tipos de actores: el administrador de actualizaciones y el usuario de la aplicación.

Administrador de actualizaciones

- Encargado de administrar los contenidos de la aplicación en el formato adecuado para su correcta adquisición, instalación y ejecución por parte de la aplicación en el dispositivo móvil.

Usuario de la aplicación

- Interactúa con la aplicación, debe poder hacer uso de las herramientas y opciones que proporciona la misma.

3.1.2. Requerimientos funcionales de la aplicación.

- La aplicación debe poder realizar actualizaciones a través de una conexión a Internet, actualizando los contenidos sin necesidad de modificar el código fuente de la aplicación.
- La aplicación debe detectar el idioma en el que está configurado el dispositivo y aplicarlo automáticamente de manera que no tenga que ser seleccionado activamente por el usuario.
- La aplicación se ofrecerá en tres idiomas: catalán, castellano e inglés.
- La aplicación debe ofrecer al usuario contenidos acerca del *Museu Arqueològic de Catalunya* y la *Seu d'Olèrdola*.
- La aplicación debe ofrecer al usuario una lista de puntos de interés, mostrando una descripción de cada uno de ellos, y la posibilidad de ubicarlos en un mapa.
- En una fase posterior de desarrollo, la aplicación ofrecerá al usuario una herramienta de realidad aumentada mediante la cual:
    - El usuario podrá hacer una visita guiada a las instalaciones, siguiendo un itinerario que será mostrado en pantalla sobreimpresionado en la imagen real.
    - El usuario podrá identificar los puntos de interés, que se mostrarán sobreimpresionados en la imagen real.

3.1.3. Requerimientos funcionales del dispositivo móvil.

- Para que la aplicación pueda acceder a las actualizaciones, así como para acceder a los mapas de ubicación que no utilizan herramientas de realidad aumentada, el dispositivo ha de disponer de acceso a Internet.
- Para que el usuario pueda hacer uso de las herramientas de realidad aumentada, el dispositivo debe disponer de las herramientas de acelerómetro, *GPS* y cámara, así como permitir a la aplicación el acceso a las mismas.

3.1.4. Requerimientos funcionales de realidad aumentada.

Debido a que inicialmente el proyecto final de carrera incluía el desarrollo e implementación en la aplicación de herramientas de realidad aumentada, vamos a describir los requerimientos funcionales que la aplicación requiere al respecto, aunque estas funcionalidades no se hayan añadido a la aplicación a fecha del escrito, ya que como se explicó anteriormente, quedaron fuera del alcance de este proyecto, relegadas a una segunda etapa de desarrollo de la aplicación.

Se describen a continuación dichos requerimientos funcionales:

- La aplicación debe mostrar una vista de realidad aumentada, en la que se superpongan los siguientes elementos:
    - Una ruta a seguir por el usuario, consistente en una línea dibujada sobre el suelo, o una serie de flechas, las cuales indiquen al usuario el itinerario a seguir, guiándole en la visita a las instalaciones.
    - Una etiqueta por cada punto de interés, sobrepuesta en la imagen sobre el punto de interés en cuestión, indicando su nombre.
    - La etiqueta de cada punto de interés debe poder ser pulsada, tras lo cual se mostrará información acerca del punto de interés en cuestión.
- La aplicación debe ofrecer una herramienta de ubicación de puntos de interés. Al seleccionar un punto de interés en una lista, la aplicación debe ofrecer una vista de realidad aumentada, con una flecha superpuesta en la imagen, que apunte hacia el punto de interés que hemos seleccionado, así como una etiqueta sobre el mismo punto de interés.

3.2. Requerimientos no funcionales.

Los requerimientos no funcionales especifican las cualidades que deben tener los elementos del sistema, cómo deben realizar sus funciones. En estos requerimientos no funcionales, no se describen los requerimientos no funcionales referentes a las utilidades de realidad aumentada de la aplicación, ya que como se ha explicado anteriormente, en una etapa avanzada del proyecto se decidió posponer el desarrollo de esta parte de la aplicación.

3.2.1. Requerimientos no funcionales del dispositivo móvil.

- El dispositivo móvil ha de ser suficientemente potente como para procesar la aplicación con el rendimiento de proceso que ésta requiere (*smartphone)*.

- El dispositivo móvil ha de ser táctil.
- El dispositivo móvil ha de tener como sistema operativo *Android,* versión 2.2 o superior.

3.2.2. Requerimientos no funcionales de la aplicación.

- **Portabilidad**.
    - La interfaz de usuario ha de ser multiplataforma, de manera que el código fuente escrito para su implementación pueda ser reutilizado exportándolo a otras plataformas, principalmente a *iOS* y *Windows Phone 7*, minimizando las adaptaciones necesarias a la plataforma destino.
- **Mantenimientos.**
    - La aplicación debe poder realizar actualizaciones de contenidos sin necesidad de modificar el código fuente.
    - Los contenidos de la aplicación en los diferentes idiomas se guardarán en archivos en formato JSON, uno para cada idioma de la aplicación. Estos archivos estarán disponibles en un servidor remoto al cual la aplicación tendrá acceso para su descarga.
    - El servidor almacenará también un archivo JSON con varios parámetros de actualización, como la fecha de actualización disponible, número de archivos a descargar, nombre de dichos archivos, etcétera. La aplicación se basará en dicho archivo para determinar si se debe actualizar o no la aplicación, así como los archivos que ha de descargar.
- **Usabilidad.**
    - La aplicación está destinada para un público muy amplio, por lo que debe cuidar especialmente la usabilidad de la interfaz de usuario. La navegación entre pantallas debe ser clara y debe facilitar la accesibilidad a los contenidos.
- **Disponibilidad.**
    - La aplicación debe estar disponible para ser instalada en la mayoría de dispositivos *Android*.
- **Rendimiento.**

- La interfaz de usuario debe ofrecer una experiencia de uso agradable y fluida, con transiciones entre las diferentes pantallas de navegación.

- **Escalabilidad.**
  - La distribución de contenidos en las pantallas de la interfaz de usuario se diseñará teniendo en cuenta la posibilidad de añadir nuevas funcionales en etapas posteriores de desarrollo.

Capítulo 4
Análisis de las tecnologías existentes.

Antes de explicar qué tecnologías y cómo se utilizan, es necesario explicar la elección realizada de las tecnologías que se adaptan mejor a solucionar los problemas, teniendo en cuenta las necesidades y requerimientos que deseamos cubrir, pero sin dejar de lado la inversión de tiempo requerido para un rápido desarrollo del proyecto.

Para crear una aplicación de estas características, el abanico de tecnologías a elegir es muy amplio, lo cual ha requerido cierto trabajo de análisis de las tecnologías existentes. Debido a que inicialmente el presente proyecto final de carrera abarcaba tareas de realidad aumentada, se hizo todo un trabajo de investigación en busca de tecnologías que cubrieran adecuadamente los requerimientos de realidad aumentada, así como el resto de requerimientos de la aplicación. Es por ello que en el análisis de tecnologías que se describe a continuación, se contemplaron todos los aspectos a tener en cuenta, incluido el desarrollo de realidad aumentada.

4.1. Análisis del entorno *Mono for Android*.

4.1.1. ¿Qué es *Mono for Android*?

Mono for Android es un *kit* de desarrollo de *software* o *SDK* para dispositivos *Android*. *Mono for Android* es parte de un proyecto más amplio llamado *Proyecto Mono*, un proyecto libre y de código abierto, que consiste en un conjunto de herramientas basadas en *GNU/Linux*, compatibles con las especificaciones de la *plataforma .NET*.

La finalidad de *Mono for Android* es facilitar a los desarrolladores de aplicaciones para dispositivos móviles el desarrollo de aplicaciones multiplataforma, permitiendo que gran parte del código de una aplicación desarrollada con *Mono for Android* se pueda reutilizar en el desarrollo de esa misma aplicación para otras plataformas.

Para ello *Mono for Android* está constituido, entre otros componentes, por una máquina virtual o *runtime*, una serie de librerías y una serie de herramientas para el empaquetado y distribución de las aplicaciones.

4.1.2. Características generales de *Mono for Android.*

Mono for Android, en líneas generales, muestra las siguientes características:

- Reutilización de código. Utilizando *C#* y *APIs* del núcleo de *.NET*, permite compartir código entre aplicaciones para *iOS*, *Android* y *Windows Phone 7*.
- Permite reutilizar librerías escritas en cualquier lenguaje de la plataforma *.NET*, importándolas al desarrollo de la aplicación móvil.
- Permite utilizar la *.NET BCL* (*.NET Base Class Library*) en sus aplicaciones móviles.
- Soporte prácticamente para todas las *APIs* nativas de *Android*.
- Provee un IDE multiplataforma propio (*MonoDevelop*) para desarrollo en *Windows*, *Linux* y *MacOS*, así como un *plugin* para el desarrollo con *Visual Studio*.

4.1.3. Ventajas e inconvenientes de *Mono for Android.*

Las principales ventajas de este *framework* serían:

- Reutilización de parte del código. Con un buen diseño por capas, exceptuando la interfaz de usuario y parte de la capa de hardware (como explicaremos más adelante), la lógica de negocio puede ser reutilizado para desarrollar la aplicación para *iOS*, mediante el *SDK Monotouch*. *Monotouch* es otro kit de desarrollo de software que, al igual que *Mono for Android*, facilita el desarrollo de aplicaciones multiplataforma para dispositivos móviles, en este caso para desarrollar aplicaciones para el sistema operativo *iOS*. El código se puede compartir entre los *SDKs Mono for Android* y *Monotouch*.
- Potente lenguaje de programación (*C#*).
- Gran compatibilidad. Se puede desarrollar software prácticamente para todas las *APIs* existentes para *Android*.

- Muy buen tiempo de respuesta del equipo de desarrollo de *Mono for Android* ante una nueva *API* de *Android*. Como ejemplo, en el momento de escribir estas líneas, la última versión de *Android* (4.0) fue liberada el 14/11/2011; *Xamarin* lanzó *Mono for Android* con soporte para dicha versión el 5/12/2011, es decir, tres semanas después. Parece garantizado que la publicación de una nueva versión de *Mono for Android* con soporte para la última versión de *Android* es cuestión de poco tiempo.

- La aplicación que finalmente se instala en el dispositivo incorpora una máquina virtual que se instala en el dispositivo junto con la aplicación para su ejecución. Este *runtime* se encarga de interpretar el código para el dispositivo en cuestión. Debido a la evolución del mercado a dispositivos móviles cada vez más potentes, esta forma de ejecución de programas está actualmente muy próxima, en términos de rendimiento, a los programas compilados a código nativo, superándolos en algunos casos.

- Tras *Mono for Android* se encuentra el *Proyecto Mono*, un proyecto más amplio del que parte y al que pertenece. El *Proyecto Mono* se viene desarrollando desde el año 2000, prácticamente en paralelo junto a la plataforma *.NET* de *Microsoft*, hasta la actualidad por su fundador *Miguel de Icaza* y parte del equipo inicial, y en asociación con *Novell* desde el año 2003. La información que he podido recibir me transmitió sensación de fiabilidad, que detrás del proyecto hay un equipo fuertemente implicado en su desarrollo y una gran comunidad de desarrolladores que se ha ido ampliando estos últimos años con el auge de las tecnologías móviles.

- Documentación amplia y completa, bien organizada y de fácil entendimiento, muchas guías y documentos "*getting started*" que orientan bien al programador novel, así como tutoriales específicos para programadores iniciados o más avanzados.

En contra, podríamos destacar lo siguiente:

- No existe un IDE en el que se pueda desarrollar la aplicación de forma integral. La comunidad *Mono* proporciona un *IDE* libre y de código abierto, llamado *MonoDevelop*, que no incorpora un editor *WYSIWYG* para la interfaz gráfica de la aplicación. El *plugin* de *Mono for Android* para *Microsoft Visual Studio* tampoco permite trabajar de esta manera. Por lo tanto la solución pasa por buscar otros editores *WYSISYG* en los que implementar la interfaz gráfica e integrar posteriormente ese código al proyecto. Este aspecto negativo es común a todas las herramientas similares existentes actualmente.

- Licencia de pago. La licencia no expira pero sólo permite actualizaciones durante un año. Por lo tanto, a partir de ese periodo cuando salgan nuevas *APIs* de *Android* se deberá comprar una nueva licencia si se quiere desarrollar aplicaciones sobre ellas. Por otra parte, para poder desarrollar la misma aplicación con *Monotouch* para *iOS* hay que pagar una nueva licencia de actualizaciones para *Monotouch*. El código se comparte, las licencias no.

- Una parte clave del código para el proyecto no será reutilizable con otras plataformas móviles. *Xamarin* actualmente está desarrollando una librería, llamada *Xamarin Mobile*. Se trata de un conjunto multiplataforma de *APIs* que permite a los desarrolladores acceder a varias herramientas comunes en los dispositivos móviles, como por ejemplo contactos del móvil, geolocalización, brújula, acelerómetro, audio y video, y notificaciones del dispositivo. Esta API proporciona llamadas comunes para *Android* y *iPhone* (más adelante incorporará *Windows Phone*), de manera que el código sea reutilizable entre las plataformas. El problema es que actualmente, en el momento de escribir estas líneas, la API se encuentra en desarrollo, por lo que la última versión sólo incorpora acceso a la libreta de direcciones y al geolocalizador.

Algunos aspectos concretos sobre *Monotouch* que habría que considerar:

- *Monotouch* no permite depurar el proyecto sobre ningún dispositivo virtual, ni compilar el proyecto, si éste no se desarrolla en un sistema operativo *MacOS* con el *Apple iPhone SDK* instalado. Además, para instalar la aplicación en un dispositivo *iOS* real se necesita licencia de desarrollador de *Apple iPhone*.

- La aplicación que finalmente se instala en el dispositivo incorpora una máquina virtual que se instala en el dispositivo junto con la aplicación para su ejecución. Ello provoca que las aplicaciones aumenten entre cuatro y cinco *megabytes* su tamaño original.

4.1.4. Pruebas realizadas con *Mono for Android.*

Para realizar pruebas con el *framework* de *Mono for Android*, se realizaron los siguientes pasos:

- Se utilizó un ordenador portátil *HP Pavillion dv9000*, con un procesador *Intel Core2 Duo t7500* a 2.20 *GHz*, 3 *Gb* de memoria *RAM*, con sistema operativo *Microsoft Windows XP Professional, Service Pack 3*.

- Se instaló el siguiente *software:*
    - *Java Standard Edition Development Kit 1.6.*
    - *Android Standard Development Kit 1.14*, instalando posteriormente el paquete de desarrollo para *Android 2.3* (*API 10*).
    - *MonoDevelop 2.8*, *IDE* proporcionado por *Mono Project* para el desarrollo de aplicaciones de sus *frameworks*.
    - *Mono for Android Standard Development Kit 1.2.*
- La versión que se instaló del *SDK* de *Mono for Android* fue la versión gratuita, la cual no permite instalar las aplicaciones que desarrollemos para dispositivos móviles en dispositivos móviles reales, sólo permite la ejecución de las aplicaciones en dispositivos virtuales *Android*. Por lo tanto, las pruebas realizadas fueron con un dispositivo virtual *Android* creado con el software *Android Virtual Device Manager* que provee el *SDK* de *Android*.
- El dispositivo virtual creado emulaba la plataforma *Android 2.3* (*API 10*) y 512 *Mb* de memoria *RAM*.
- Se realizó una primera prueba, una simple aplicación que consistía en una pantalla con un botón cuyo título mostraba el número de veces que el botón era pulsado, actualizando el número mostrado tras cada pulsación de botón.
- A continuación, se investigó acerca de la posibilidad de trabajar con imagen real de la cámara del dispositivo. La *API* que proporciona la plataforma para acceder a la cámara únicamente ofrece herramientas para realizar una fotografía y guardarla en la memoria del dispositivo. Por otra parte, *Mono for Android* soporta la *API OpenTK*, un encapsulado para acceder a librerías OpenGL.
- No se fue más allá en las pruebas debido a la dificultad añadida que implica realizar pruebas de realidad aumentada sobre un dispositivo virtual.

4.2. Análisis del *framework PhoneGap*.

4.2.1. ¿Qué es *PhoneGap*?

PhoneGap fue el nombre original del proyecto, y es el nombre con el que comúnmente se conoce este proyecto. Posteriormente se renombró a *Apache Callback* y, finalmente, el 28 de noviembre de 2011 se renombró de nuevo el proyecto de *Apache Callback* a *Apache Cordova*. Se trata de un *framework* libre y de código abierto para desarrollar software multiplataforma para dispositivos móviles. Permite crear aplicaciones para dispositivos móviles utilizando lenguajes de programación web, como *Javascript*, *HTML5* y *CSS5*, lo cual resulta en una aplicación no nativa.

4.2.2. Ventajas e inconvenientes de *Phonegap.*

La principales ventajas de este *framework* serían:

- Es libre y de código abierto, bajo licencia *BSD*, que permite el uso de código fuente en software no libre. Soporte técnico de pago.
- Lenguaje *web* (*HTML5*, *CSS3*, *Javascript*) muy extendido, conocido, documentado, de fácil uso y aprendizaje, y único para todas las plataformas.
- Existe bastante documentación y ejemplos, y una gran base de usuarios, debido al lenguaje de programación que utiliza.
- Cubre la práctica totalidad de las principales plataformas existentes en el mercado.
- Provee una plantilla implementada para el contenedor nativo de las ocho principales plataformas de dispositivos móviles.
- Implementa un puente entre *Javascript* y código nativo para al menos seis plataformas de dispositivos móviles. Esto posibilita el acceso bilateral entre *Javascript* y el lenguaje de programación nativo del sistema operativo del dispositivo.

- Proporciona una *API* en *Javascript* independiente del sistema operativo para activar diversas funciones, como muestra la *figura 3.2.2.1*.

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 5.x	OS 6.0+	WebOS	WP7	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✓	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✓	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✗	✓	✗	✓
CONTACTS	✓	✓	✓	✓	✓	✗	✓	✓	✓
FILE	✓	✓	✓	✓	✓	✗	✓	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✓	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✓	✓	✓	✓	✓	✗

Figura 3.2.2.1. Acceso a las características nativas de los dispositivos de diversas plataformas desde la *API Javascript* que proporciona *PhoneGap*.

- En el caso concreto de *Android*, los tamaños de las librerías *Java* y *Javascript* que se han de empaquetan con la aplicación, en total sobrepasan por poco los 300 *Kb*, lo cual supone un incremento más que aceptable en el tamaño final de la aplicación.

- Al contrario que otros *frameworks* de desarrollo basados en *HTML5, como AppMobi*, con *PhoneGap* tenemos control sobre la versión de la aplicación en el *market* de aplicaciones. De manera que, si subimos una nueva versión de la aplicación al *market*, los usuarios que tengan una versión anterior instalada recibirán automáticamente un aviso de que existe una nueva versión de la aplicación disponible para su descarga.

- Permite ocultar la barra de estado nativa del dispositivo móvil para que no sea visible mientras se ejecuta la aplicación.

En contra, podríamos apuntar lo siguiente:

- Una pequeña parte del código que escribamos dependerá ineludiblemente de la plataforma. Al igual que sucede en el mundo de la programación web con las hojas de estilo y los deferentes navegadores, en *PhoneGap* (y en general en

todos los *frameworks* multiplataforma para desarrollo de aplicaciones para dispositivos móviles basados en HTML5), hay determinadas características intrínsecas a cada plataforma que imposibilitan estandarizar el código a determinados niveles. Por ejemplo, el ciclo de vida de las aplicaciones en *Android* es diferente que en *iOS*. Esto provoca que tengamos que codificar ciertas ramas condicionales que discriminen en qué plataforma está corriendo la aplicación para determinar cómo se ha de comportar la aplicación.

- Al generalizar el código perdemos particularidades de cada plataforma que pueden ser ventajosas. Por ejemplo, en *Android* los objetos tipo *Intent*, que permiten enviar y recibir información desde o hacia otras actividades o servicios que se están ejecutando en el dispositivo móvil. Y aunque *PhoneGap* pusiera a disposición del desarrollador *plugins* para poder usar dichas particularidades (en el antedicho caso de los *Intents* así es), su uso sería un contrasentido en un *framework* como *PhoneGap*, cuya filosofía es el desarrollo de código independiente de la plataforma. Algo similar sucede si se desean desarrollar interfaces avanzadas, en cuyo caso se han de crear interfaces especializadas para cada entorno.

- Bajo rendimiento de computación, limitado por el componente *WebKit* nativo, que debe leer e interpretar el código *web*. En situaciones en las que la aplicación requiera un buen rendimiento computacional, tendremos que recurrir a código nativo, con lo cual perdemos la independencia de la plataforma.

- En contra de la excesiva flexibilidad de diseño que proporciona el *framework*, se echa en falta documentación orientativa acerca de cuáles son las mejores técnicas de desarrollo (*best practices*), directrices de implementación a seguir para optimizar el flujo y la efectividad de la aplicación.

- No provee *widgets* o controles a nivel de interfaz de usuario (transiciones, controles estándar, etcétera). Aunque existen paquetes de herramientas libres que proporcionan dichos controles, éstos no proporcionan la suavidad y el aspecto de los controles nativos, y requieren de bastante tiempo de diseño si se desea que la aplicación tenga un aspecto nativo.

- No implementa soporte para audio, con lo cual las aplicaciones pierden mucho potencial sin sonidos, empobreciendo la experiencia audiovisual (o hay que recurrir de nuevo a código nativo).

- No existe un IDE en el que se pueda desarrollar la aplicación de forma integral. Por lo tanto la solución pasa por buscar otros editores *WYSISYG* en los que

implementar la interfaz gráfica e integrar posteriormente ese código al proyecto.

4.3. Análisis del *framework AppCelerator Titanium*.

4.3.1. ¿Qué es *AppCelerator*?

Appcelerator Titanium es una plataforma para desarrollar aplicaciones móviles y de escritorio utilizando tecnologías *web*. *Appcelerator Titanium* está desarrollado por *Appcelerator Inc.* y fue lanzado en diciembre del 2008. En junio de 2009 se añadió soporte para el desarrollo de aplicaciones móviles para *Android* e *iPhone*. En abril de 2010 se añadió soporte para el desarrollo de aplicaciones para *Ipad*.

AppCelerator utiliza como código fuente lenguaje web (*HTML5*, *CSS3* y *Javascript*). En tiempo de compilación, el *framework* compila al lenguaje nativo de la plataforma destino. Para ello, utiliza una *API* de interfaz de usuario, que contiene asignaciones a componentes de interfaz de usuario nativos, y una *API* de dispositivo, que contiene asignaciones a las capacidades nativas del dispositivo.

4.3.2. Ventajas e inconvenientes de *AppCelerator.*

La principales ventajas de este *framework* serían:

- Es software libre y de código abierto, bajo licencia *Apache 2.0*.
- Basado únicamente en el lenguaje *Javascript* para todas las plataformas, un lenguaje muy extendido, conocido, documentado, de fácil uso y aprendizaje.
- Incluye un entorno de desarrollo basado en *Eclipse*.
- La *API* ofrece componentes independientes de la plataforma, pero también componentes específicos de cada plataforma. Por ejemplo, para la plataforma *Android*, proporciona acceso a *Intents*.

En contra, podríamos apuntar lo siguiente:

- Únicamente se puede desarrollar para las plataformas de *Apple* y *Android*.
- Su uso requiere una curva de aprendizaje bastante alargada.

4.4. Análisis de Tecnologías de Realidad Aumentada.

En este apartado, estudiaremos concretamente tecnologías de realidad aumentada enfocadas al diseño de aplicaciones para dispositivos móviles. El análisis realizado no tuvo la profundidad que requiere el proyecto real, debido a que se modificaron los objetivos del presente proyecto final de carrera. Los estudios y las conclusiones alcanzadas definidos a continuación deben por lo tanto servir de referencia aunque no deberían ser determinantes a la hora de decidir las tecnologías de realidad aumentada que se emplearán finalmente en el proyecto real.

4.4.1. Navegadores de Realidad Aumentada

En los últimos años ha surgido un tipo de solución muy popular en el mundo de la realidad aumentada: navegadores *web* de realidad aumentada. Infinidad de navegadores de realidad aumentada, como *Wikitude*, *Junaio*, *Mixare*, etcétera, están disponibles en el mercado, siendo el precursor de todos ellos *Layar*. Se trata de navegadores *web* que funcionan como una aplicación autónoma, pudiéndose integrar en una aplicación móvil, de manera que su funcionalidad está limitada a las herramientas que proporciona la plataforma sobre el navegador. Las herramientas que proporcionan estas soluciones son principalmente la creación de una capa de puntos de interés geolocalizados, que muestran cierta información, la cual se puede extender al pulsar sobre el punto de interés.

4.4.2. Ventajas y desventajas de los navegadores *AR*.

Entre las ventajas del uso de estas tecnologías, podríamos destacar:

- Coste económico: Crear nuestra propia capa de navegación es muy económico. De hecho, es gratuito, aparte del espacio *web* necesario para almacenar los datos, y el tiempo de creación de la capa de datos es relativamente bajo.

- Facilidad de uso: Son varias las formas de crear canales de realidad aumentada. Si se tienen algunos conocimientos básicos sobre *PHP* y *SQL* se puede desarrollar con bastante rapidez. Aunque la manera más fácil, especialmente para los programadores no experimentados, puede ser *Hoppala*, un sistema de gestión de contenido (*CMS*) especializado en navegadores de realidad aumentada, que permite crear un canal de datos de realidad aumentada para los tres principales navegadores de realidad aumentada, esto es, *Layar*, *Junaio* y *Wikitude*.

- *Natural Feature Tracking*: Se trata de una técnica de realidad aumentada muy avanzada, consistente en utilizar imagen real a color como marcador. No todos los navegadores soportan esta característica, *Junaio* sí lo soporta.
- Interactividad: El contenido no sólo se mostrará, sino que también permite interactuar con él.
- Diversidad de contenidos: Los contenidos que pueden mostrarse no se limitan a planos *2D*. Vídeo, audio e incluso objetos *3D* pueden ser colocados en la capa de realidad aumentada.
- Integración: *Layar* y *Wikitude* permiten integrar los canales o capas dentro de la aplicación.
- Multiplataforma: Los tres principales navegadores *web* de realidad aumentada (*Layar*, *Junaio*, *Wikitude*) están disponibles para los dispositivos *Android*, *iPhone* y *Symbian*.

Por contra, podríamos destacar los siguientes puntos:

- Visibilidad de la marca: Si no se integra la capa *AR* en la aplicación, la capa será una de las muchas dentro del creciente catálogo de capas de la plataforma. Por esto, puede suceder que el usuario se distraiga con otros contenidos mientras se busca y se carga el canal que deseamos.
- No hay modo *offline*: Debido a que los navegadores extraen datos de un servidor a través del servicio *web*, es fundamental contar con una conexión a *Internet* para poder ver cualquier contenido.
- Funcionalidades: Son muchas las funcionalidades posibles en un navegador *AR*, sin embargo, éstas tiene sus límites. Si, por ejemplo, se quisiera una interfaz de usuario más avanzada y/o personalizada con un conjunto de características más sofisticado, estas plataformas ofrecen muy poco apoyo.

4.4.3. Kits de desarrollo de Realidad Aumentada.

Es este un mercado muy reciente, por lo que no existen muchos kits de desarrollo en este sentido, y los que existen no son ni mucho menos completos. Algunos *SDKs* ofrecen ciertas características de las cuales otros carecen. A continuación enumeramos los principales kits de desarrollo más populares.

Una de las soluciones más populares es *Vuforia* de *Qualcomm. Qualcomm* es una empresa productora de *chipsets* para tecnología móvil. Es uno de los principales proveedores para todas las compañías fabricantes de telefonía móvil. En 2010, la empresa estuvo mostrando algunas aplicaciones propias que hacían uso de tecnologías de realidad aumentada. Finalmente, ese mismo año anunció en un comunicado oficial que ponía a disposición de los desarrolladores de *Android* el conjunto de herramientas con las que las desarrollaron. El *software* desarrollado, llamado finalmente *Vuforia* y disponible también para *iOS*, se basa en el reconocimiento de imágenes para aplicar renderizados, y no tanto en la información proporcionada por el sensor *GPS*, como otros kits de desarrollo.

Quizá la solución más popular y extendida sea *ARToolKit*. *ARToolKit* es una biblioteca que permite la creación de aplicaciones de realidad aumentada, en las que se sobrepone imágenes virtuales al mundo real. Para ello, utiliza las capacidades de seguimiento de vídeo, con el fin de calcular, en tiempo real, la posición de la cámara y la orientación relativa a la posición de los marcadores físicos. Una vez se conoce la posición de la cámara real, la cámara virtual se pueden colocar en el mismo punto y modelos 3d son sobrepuestos exactamente sobre el marcador real. Así *ARToolKit* resuelve dos de los principales problemas en la realidad aumentada, el seguimiento de punto de vista y la interacción objeto virtual.

ARToolKit fue desarrollado originalmente por *Hirokazu Kato* en 19992 y fue publicado por el *HIT Lab* de la *Universidad de Washington*. Actualmente se mantiene como un proyecto de código abierto alojado en *SourceForge* con licencias comerciales disponibles en *ARToolWorks*. *ARToolKit* es una librería muy utilizada, con un seguimiento de la biblioteca con más de 160.000 descargas desde el año 2004.

Los puntos fuertes de esta librería podrían ser:

- Facilidad de implementación.
- Algoritmo de seguimiento (*tracking*) no demasiado intensivo para la *CPU*.
- Puede trabajar hasta con 100 marcadores simultáneamente.
- Permite el diseño y uso de marcadores propios.
- Permite exportar directamente desde *Blender 3D*.
- Puede superponer casi cualquier contenido: imágenes, vídeo, objetos *3D*, etcétera.
- Existen multitud de adaptaciones (*ports*) de esta librería a otras tecnologías y lenguajes de programación, entre ellas *JSARToolkit*, un *port* de la librería al lenguaje *Javascript*.

En contra, podríamos destacar los siguientes aspectos de *ARToolkit*:

- La exportación desde *Blender 3D* requiere algunos ajustes para que los objetos se puedan mostrar correctamente.
- Muchas llamadas a *getElementById()* desde las estructuras de control.
- *JSARToolKit* es un *port* basado *FLARToolKit*, versión *AS3* (por lo que utiliza *Adobe Flash*) de *ARToolKit*, otro *port* de la librería original. Ello significa que *JSARTookit* no es puro *JavaScript*, sino que utiliza la tecnología *web Flash*. Conocidas por todos son las polémicas discrepancias entre *Apple* (*iOS*) y *Adobe*.

4.4.4. Pruebas realizadas con Realidad Aumentada.

Se realizaron algunas pruebas reales con aplicaciones de demostración. Todas las pruebas realizadas se hicieron utilizando las siguientes tecnologías:

- Ordenador portátil *Toshiba Satellite Pro L770*, con un procesador *Intel Core i5-2400M* a 2.50 *GHz*, 8 *Gb* de memoria *RAM*, con sistema operativo *Microsoft Windows 7 Professional, Service Pack 1*.
- Se instaló el siguiente *software:*
    - *Java Standard Edition Development Kit 1.6.*
    - *Android Standard Development Kit 1.14*, instalando posteriormente el paquete de desarrollo para *Android 2.3* (*API 10*).
    - *Eclipse Indigo* (3.7.2).
- Dispositivo móvil *Samsung Galaxy Young*, con procesador de *833 MHz.*

La primera prueba realizada fue con una aplicación de demostración de un artículo de *The Linux Journal*, *"Augmented Reality with HTML5"* (30 de abril de 2011, por *Rick Rogers*). La aplicación en cuestión, *ARCompass*, es una aplicación *Android* híbrida con *HTML5*. La aplicación usa los datos del acelerómetro para obtener la rotación que permite mostrar una imagen superpuesta de una brújula en la dirección correcta en la pantalla. Se descargó el código fuente, que consistía en un proyecto *Eclipse*, se compiló y se instaló en un dispositivo *Android*, obteniendo los resultados deseados con un rendimiento óptimo.

Seguidamente se realizaron algunas pruebas con la librería de realidad aumentada *ARToolkit*. En concreto, se utilizó la librería *AndAR*, un *port* de la librería *ARToolkit* para *Android*. Se obtuvo el código fuente de varios proyectos de la página *web* de *Google Code*, un sitio *web* de *Google* para desarrolladores interesados en desarrollos relacionados con *Google* y de código abierto, como es el caso de *AndAR*. En este repositorio se encuentran proyectos que utilizan la librería *AndAR*, desarrollados en *Eclipse*. La primera de las pruebas se realizó con una aplicación *Android* llamada *AndARSampleProject*. La aplicación consiste sencillamente en superponer un cubo en tres dimensiones dibujándolo sobre un marcador mediante *OpenGL*. Se compiló y ejecutó la aplicación en un dispositivo *Android*. Al mostrar el marcador, la aplicación dibujaba el cubo sobre el mismo, aunque en ocasiones el *software* no conseguía localizar correctamente el marcador y, por lo tanto, no dibujaba el cubo. El rendimiento era muy bueno, aunque lógicamente los requerimientos no eran excesivos debido a la simplicidad del modelo a dibujar, el cubo.

Posteriormente se hicieron pruebas con un nuevo proyecto del mismo repositorio. El proyecto, llamado *AndARModelViewer*, consiste en mostrar un modelo *3D* sobre un marcador. El modelo mostrado consiste en un archivo tipo *Wavefront OBJ*, un estándar de archivo de modelado *3D*, que se empaqueta con la aplicación. La aplicación incorpora varios modelos y el usuario puede elegir el modelo a mostrar sobre el marcador, pudiendo realizar acciones sobre el modelo, como escalarlo o rotarlo. Se compiló y ejecutó la aplicación, y los resultados a nivel de rendimiento fueron prácticamente idénticos a los obtenidos con las pruebas realizadas anteriormente.

Sobre este mismo proyecto se realizaron nuevas pruebas, que consistieron en utilizar un marcador generado por nosotros mismos. Para ello, se utilizó la herramienta de *software mk_patt*, que forma parte de la librería *ARToolkit*. Se dibujó el marcador, imprimiéndolo posteriormente en papel, y se utilizó la herramienta *mk_patt* para obtener el archivo de la marca, el cual se ha de empaquetar con la aplicación y referenciar desde el código fuente mediante el constructor de la clase *ARObject*.

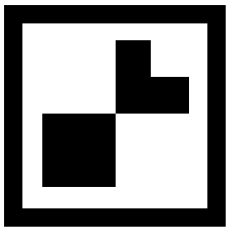

Figura 4.4.4.1. Marcador personalizado utilizado para las pruebas realizadas con la librería *ARToolkit*.

La herramienta *mk_patt* requiere de una cámara para generar el archivo del marcador. En este caso se utilizó la cámara *web* del ordenador portátil, donde se ejecutó la herramienta para obtener el archivo del marcador. Se realizaron pruebas con el nuevo marcador, obteniendo resultados idénticos a los obtenidos con los marcadores proporcionados por la aplicación de muestra.

Se realizaron otras pruebas de menor relevancia con otro *software*. Se probó la nueva etiqueta *video* de *HTML5*, para utilizar la cámara *web* del dispositivo móvil con imagen en vivo, aunque se comprobó que ningún *webkit* nativo de ninguna plataforma soporta hasta la fecha dicha característica. De hecho, el único navegador que soporta esta característica es *Opera Mini*.

4.4.5. Conclusiones respecto a la Realidad Aumentada.

Aunque la fase de implementación de las herramientas de realidad aumentada queda fuera del ámbito del presente proyecto, todo apunta a que la mejor opción para la implementación de dichas herramientas sea la librería *ARToolkit* por diversos motivos, entre los cuales destacaríamos:

- El código de reconocimiento solo usa simples cuadrados negros.
- La posibilidad de crear cualquier patrón para el código de reconocimiento.
- Fácil código de calibración para la cámara.
- Velocidad suficiente para aplicaciones de realidad aumentada en tiempo real.
- Es libre y de código abierto.
- Existe un puente para la interacción con el lenguaje *Java*: *jARToolkit*.

Los motivos que llevaron a posponer la fase de implementación de las herramientas de realidad aumentada fueron diversos, el principal de los cuales fue el tiempo. Por un lado, se deseaba tener una aplicación para poder mostrar al cliente lo antes posible. Por otro lado, el desarrollo del proyecto sobrepasaba en términos de tiempo el alcance del presente proyecto final de carrera. Por ello se optó por hacer un buen desarrollo de la interfaz de usuario de la aplicación en detrimento del desarrollo de las herramientas de realidad aumentada.

Capítulo 5
Tecnologías utilizadas.

En este apartado se enumeran y describen las tecnologías utilizadas para el desarrollo de la interfaz de usuario, justificando la elección realizada, teniendo en cuenta las necesidades y requerimientos que deseamos cubrir.

5.1. *PhoneGap*.

¿Qué es *PhoneGap?*

Como ya describimos anteriormente, *PhoneGap*, también conocido *Apache Cordova*, es un *framework* libre y de código abierto para desarrollar software multiplataforma para dispositivos móviles. Permite crear aplicaciones para dispositivos móviles utilizando lenguajes de programación web, como *Javascript*, *HTML5* y *CSS5*, lo cual resulta en una aplicación no nativa.

En el presente proyecto se utilizó la última versión disponible hasta la fecha del escrito, esta es, la versión 1.7.

¿Por qué *PhoneGap?*

El principal motivo de esta decisión fue que se trata de un *software* libre y de código abierto, bajo licencia *BSD*, que permite el uso de código fuente en software no libre, aunque el soporte técnico es de pago. La reciente adquisición de *Nitobi*, la empresa creadora de *PhoneGap*, por parte de *Adobe*, presupone que la plataforma tiene relativamente garantizada la continuidad en su desarrollo, aunque como contrapunto, es probable que en un futuro, cuando la tecnología esté más extendida, quizá pase a ser una plataforma de pago, según viene siendo últimamente la estrategia comercial habitual de esta empresa.

Otro de los principales motivos fue la característica multiplataforma que define la plataforma, además del lenguaje de programación que utiliza (*HTML5*, *CSS3*, *Javascript*), muy extendido, conocido, documentado, de fácil uso y aprendizaje, y único para todas las plataformas. La gran base de usuarios que hay detrás de la plataforma

también la hizo atractiva, porque hacía suponer que se encontraría solución con relativa facilidad a los problemas técnicos que se nos pudiesen plantear durante el desarrollo.

El hecho de que proporcione un puente entre el código *Javascript* y el código nativo también resultó clave, debido a que permitiría realizar las tareas de Realidad Aumentada desde código nativo y acceder a ellas desde *PhoneGap*, lo cual se adaptaba a la división realizada del proyecto real en interfaz de usuario y Realidad Aumentada.

5.2. *jQuery Mobile.*

¿Qué es *jQuery Mobile?*

JQuery Mobile es un *framework web* optimizado para dispositivos táctiles, cuyo desarrollo se centra en la creación de un marco compatible con una amplia variedad de teléfonos inteligentes y tabletas.

Es compatible con todas las principales plataformas móviles, así como todos los navegadores de escritorio más importantes, incluyendo iOS, Android, Blackberry, Palm, Symbian y Windows Phone 7. Está construido sobre el núcleo de jQuery, una biblioteca *JavaScript*, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos *HTML*, manipular el árbol *DOM*, manejar eventos, desarrollar animaciones y agregar interacción con la técnica *AJAX* a páginas *web*.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la *Licencia MIT* y la *Licencia Pública General de GNU v2*, permitiendo su uso en proyectos libres y privativos. *jQuery*, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Como se ha descrito anteriormente, *jQuery Mobile* está basado en la librería *jQuery*, orientada a la plataforma de ordenadores de escritorio. En el presente proyecto se utilizó la última versión de ambas librerías disponible hasta la fecha del escrito, esta es, la versión 1.7.1 de *jQuery* y la versión 1.1.0 de *jQuery Mobile.*

¿Por qué *jQuery Mobile?*

La carencia de *PhoneGap* de herramientas para el desarrollo de la interfaz de usuario hizo inevitable la búsqueda de una herramienta que complementase este aspecto de la plataforma. Existen multitud de *frameworks* similares a *jQuery Mobile*, como pueden ser *Sencha Touch*, *Dojo Mobile*, *Boilerplate* y muchos otros. Los más extendidos de entre todos ellos podríamos decir que son *jQuery Mobile* y *Sencha Touch*. Por motivos de tiempo, restringimos la decisión a estos dos *frameworks*. Los puntos fuertes de *Sencha Touch* son básicamente dos. El primero de ello es que se trata de un proyecto que lleva más tiempo en desarrollo que *jQuery Mobile*, por lo que está más madurado. Por otra parte, *Sencha Touch* es más robusto y algo más fluido en ejecución que *jQuery Mobile*. En contra de *Sencha Touch* encontramos que se trata de un *framework* mucho más difícil de programar que *jQuery Mobile*, el cual es extremadamente fácil. Otro punto fuerte de *jQuery Mobile* es la gran variedad de plataformas de dispositivo móvil que soporta, frente a *Sencha Touch* que sólo soporta *iOS*, *Android* y recientemente *Blackberry*.

Finalmente, la decisión fue *jQuery Mobile* por la facilidad de programación y la amplia variedad de plataformas que soporta.

Capítulo 6
Especificación.

La especificación o modelo de análisis, muestra cuál es el comportamiento externo del sistema, es decir, qué podrá hacer el usuario final con la aplicación. Es importante elaborar la especificación con detalles, puesto que a partir de ella se realizarán posteriormente el resto de tareas.

6.1. Esquema previo de la interfaz de usuario.

Previamente a la especificación final de la interfaz de usuario, se realizaron una serie de reuniones con el cliente final, el resultado de las cuales fue un primer diseño de la interfaz de usuario. Dicho diseño fue plasmado gráficamente en un documento generado mediante una herramienta de *software* para el desarrollo de esquemas llamada *Balsamiq Mockups*. El documento que mostramos a continuación fue desarrollado por Raúl Sáez, también poniente del mismo proyecto final de carrera pero en la versión para *iOS*. El documento fue presentado al cliente final para concluir en el diseño conceptual de las pantallas que debía incorporar la aplicación.

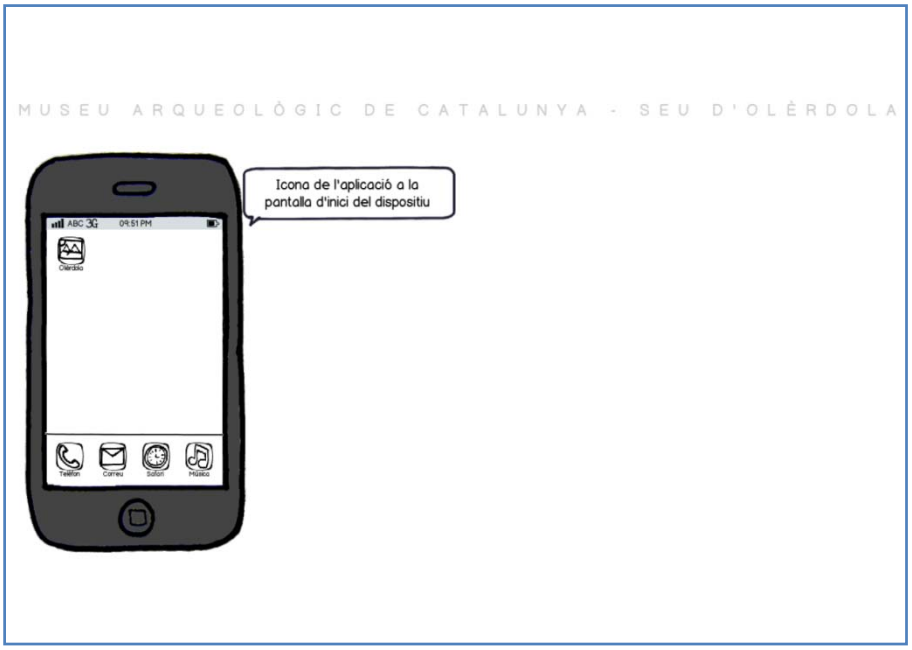

Figura 6.1.1. Página 1 del documento presentado al cliente para la especificación de la interfaz de usuario.

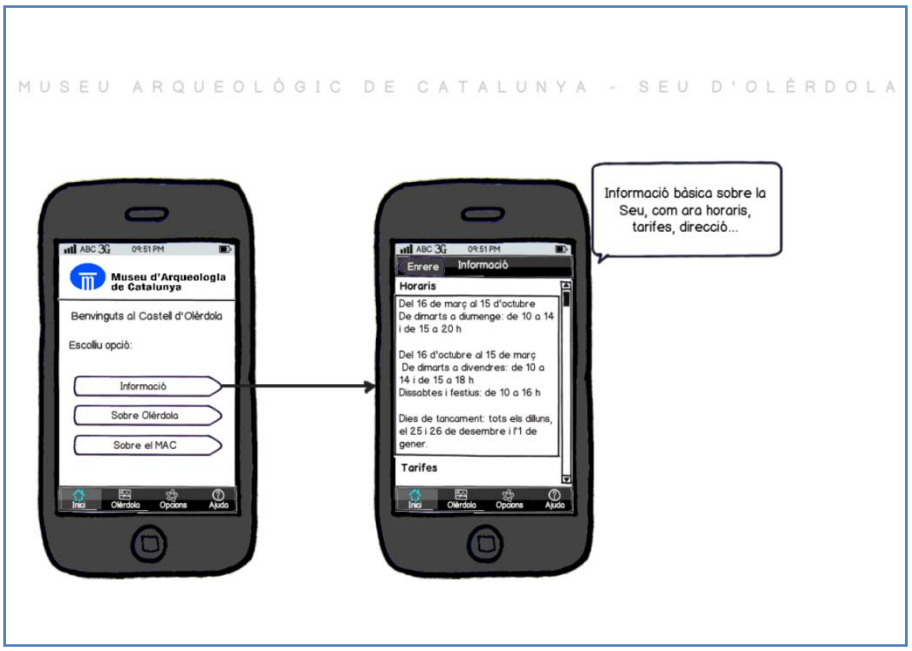

Figura 6.1.2. Página 2 del documento presentado al cliente para la especificación de la interfaz de usuario.

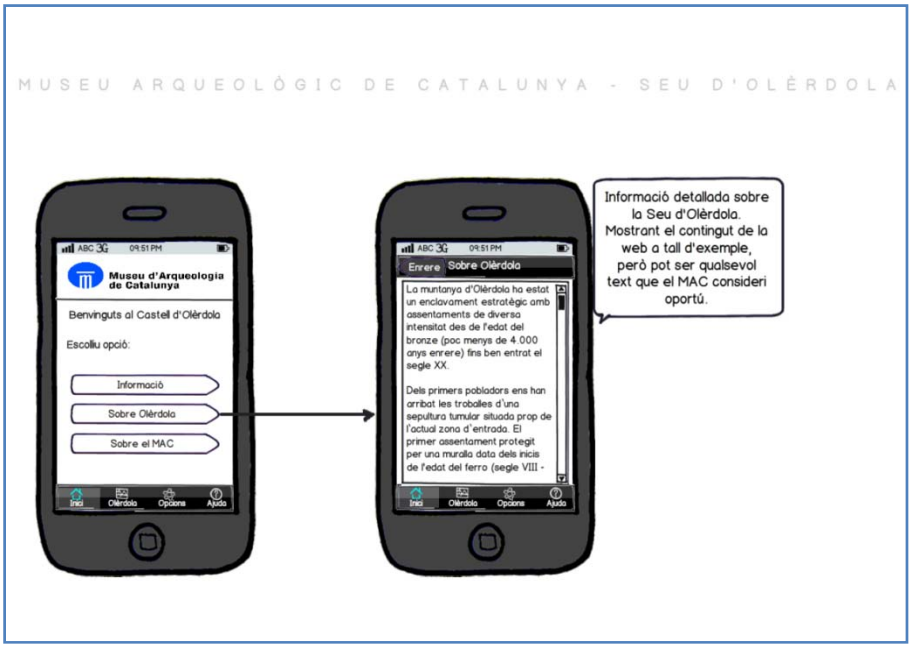

Figura 6.1.3. Página 3 del documento presentado al cliente para la especificación de la interfaz de usuario.

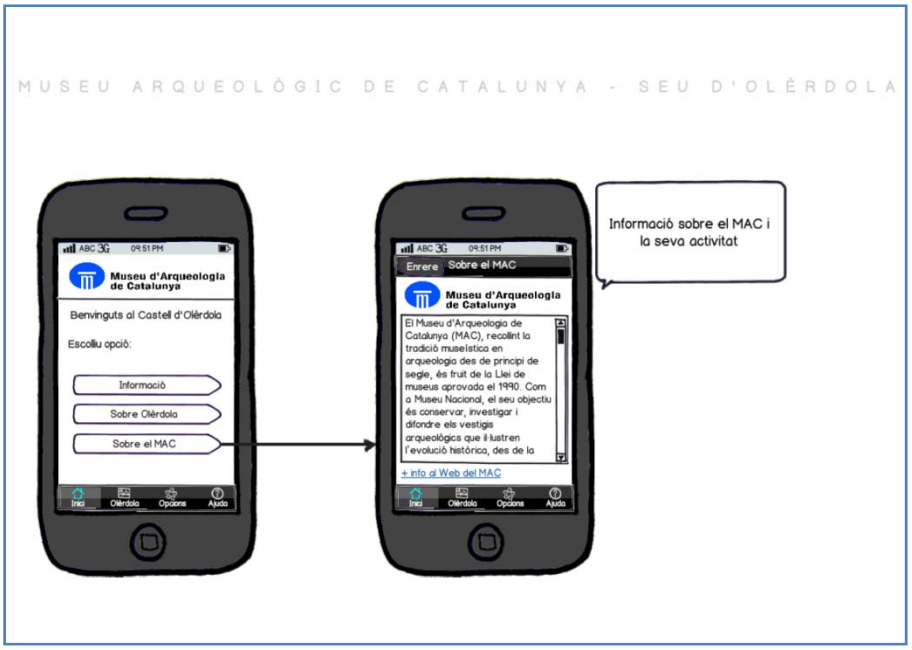

Figura 6.1.4. Página 4 del documento presentado al cliente para la especificación de la interfaz de usuario.

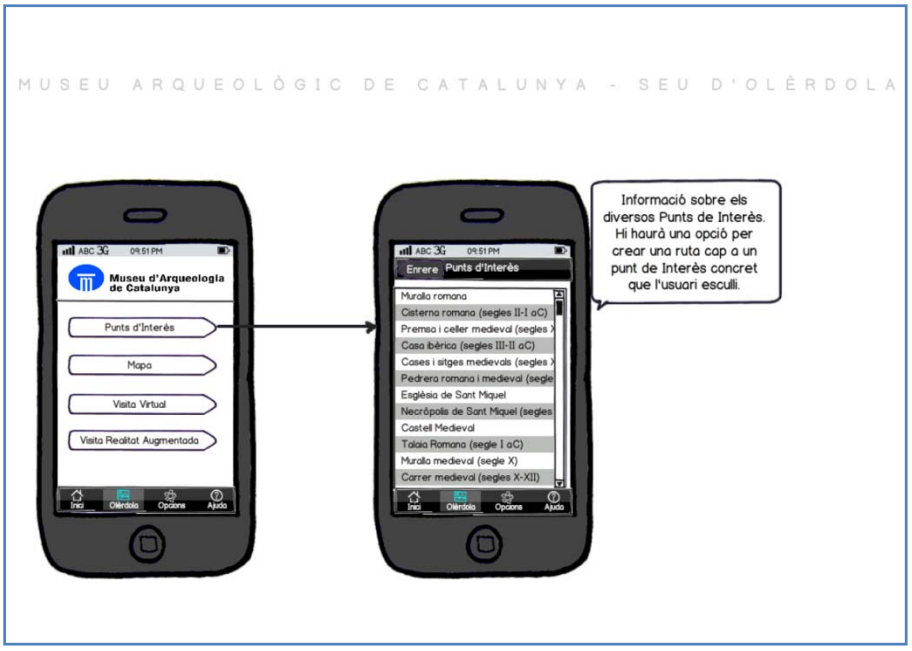

Figura 6.1.5. Página 5 del documento presentado al cliente para la especificación de la interfaz de usuario.

Figura 6.1.6. Página 6 del documento presentado al cliente para la especificación de la interfaz de usuario.

Figura 6.1.7. Página 7 del documento presentado al cliente para la especificación de la interfaz de usuario.

Figura 6.1.8. Página 8 del documento presentado al cliente para la especificación de la interfaz de usuario.

Figura 6.1.9. Página 9 del documento presentado al cliente para la especificación de la interfaz de usuario.

Figura 6.1.10. Página 10 del documento presentado al cliente para la especificación de la interfaz de

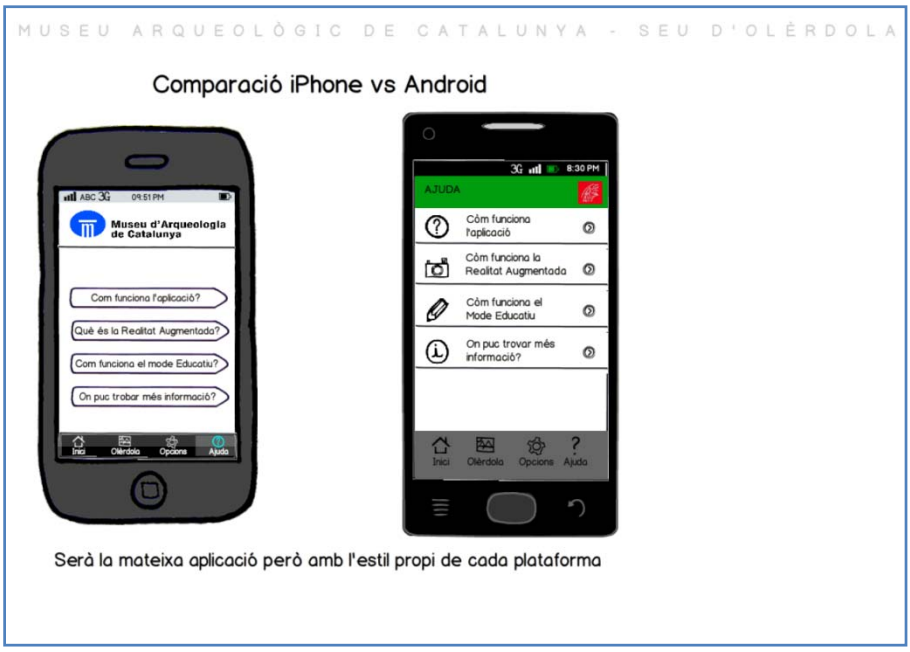

Figura 6.1.11. Página 1 del documento presentado al cliente para la especificación de la interfaz de usuario.

6.2. Modelo de casos de uso.

El modelo de casos de uso es una representación de las funciones principales del sistema. Cada caso de uso describe una secuencia de eventos que realiza un actor que utiliza el sistema para obtener algún resultado. En el caso de nuestro sistema existen dos actores: el usuario de la aplicación (ver figura 6.2.1) y el administrador de actualizaciones (ver figura 6.2.2).

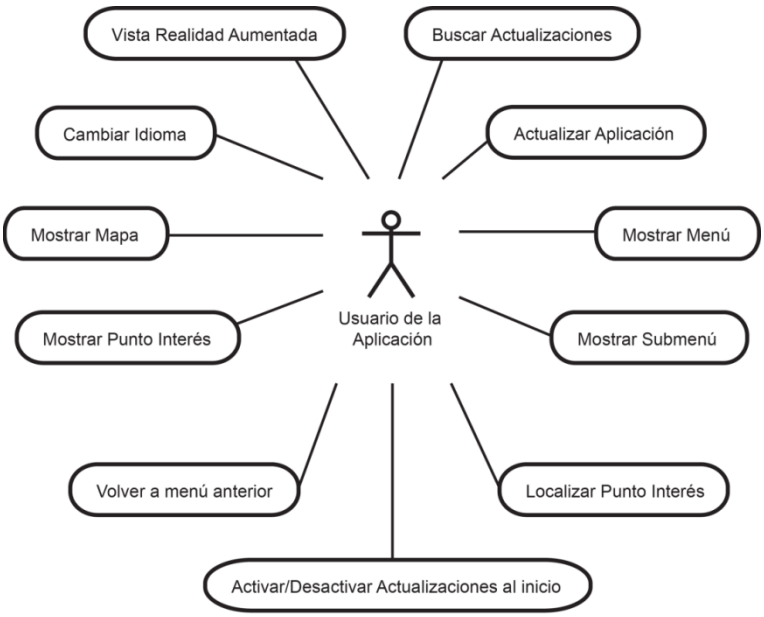

Figura 6.2.1. Casos de uso del usuario de la aplicación.

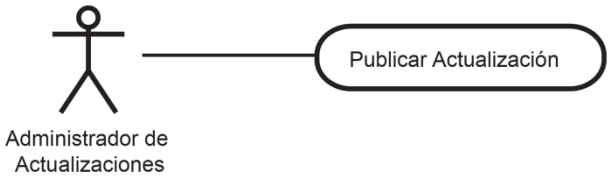

Figura 6.2.2. Casos de uso del administrador de actualizaciones.

6.2.1. Mostrar Menú.

Caso de uso: Mostrar Menú.

Actores: Usuario de la aplicación.

Propósito: Mostrar la pantalla correspondiente a la opción de menú seleccionada por el usuario. Esta opción es una de las que se listan a continuación: "*Inici", "Olèrdola", "Opcions*" y "Ajuda".

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa un botón de menú de la aplicación.	2. El sistema muestra la pantalla correspondiente al botón de menú pulsado por el usuario.

6.2.2. Mostrar Submenú.

Caso de uso: Mostrar Submenú.

Actores: Usuario de la aplicación.

Propósito: Mostrar la pantalla correspondiente a la opción de menú de segundo nivel seleccionada por el usuario. Esta opción es una de las que se listan a continuación. Para el menú "*Inici*": "*Informació", "Sobre Olèrdola*" y "*Sobre el MAC".*

Para el menú "*Olèrdola*": "*Punts d'Interès", "Mapa", "Visita Realitat Augmentada".*

Para el menú "Ajuda": "*Còm funciona l'aplicació?*", "*Què és Realitat Augmentada?*" y "*Més informació*".

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa un botón de la opción de menú de segundo nivel de la aplicación.	2. El sistema muestra la pantalla correspondiente al botón de menú de segundo nivel pulsado por el usuario.

6.2.3. Volver a menú anterior.

Caso de uso: Volver al menú anterior.

Actores: Usuario de la aplicación.

Propósito: Mostrar al usuario la pantalla correspondiente a la opción de menú padre ó de nivel inmediatamente anterior a la pantalla que se está mostrando actualmente.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa el botón de retorno a la opción anterior.	2. El sistema muestra la pantalla correspondiente al menú padre de la pantalla actual.

6.2.4. Mostrar Punto Interés.

Caso de uso: Mostrar Punto de Interés.

Actores: Usuario de la aplicación.

Propósito: Mostrar al usuario información sobre un punto de interés concreto, así como un botón para realizar la acción de localizar el punto de interés.

El punto de interés puede ser cualquiera de la siguiente lista de puntos de interés:

- Muralla Romana.
- Cisterna Romana.
- Premses i Celler Medieval.
- Casa ibèrica.
- Cases medievals al peu de la muralla.
- Pedrera romana i medieval.
- Església preromànica.
- Església romànica.
- Necròpois de Sant Miquel.

- Castell medieval.
- Torre romana.
- Muralla medieval.
- Cases i sitges medievals.
- Carrer medieval.
- Muralla ibèrica.
- Tintoreria ibèrica.
- Edifici medieval amb tallers.
- Estructures medievals fora muralles.
- Pla dels Albats.
- Església de Santa Maria fora muralles.
- Necròpolis del Pla dels Albats.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa una opción de la lista de puntos de interés.	2. El sistema muestra una descripción del punto de interés seleccionado. Además, el sistema muestra un botón para que el usuario pueda realizar la acción de localizar el punto de interés.

6.2.5. Localizar Punto Interés.

Caso de uso: Localizar Punto de Interés.

Actores: Usuario de la aplicación.

Propósito: Mostrar al usuario un mapa su ubicación actual y la ubicación de un punto de interés determinado por el usuario, para que el usuario pueda localizarlo y acceder o aproximarse al punto de interés en las instalaciones. El punto de interés puede ser cualquiera de la lista de puntos de interés mostrada en el apartado anterior 6.2.4.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa el botón de localización del punto de interés en cuestión.	2. El sistema muestra un mapa de las instalaciones en el que aparecen dos etiquetas: la primera marca la posición actual geolocalizada del dispositivo; la segunda marca la posición correspondiente al punto de interés en cuestión.

6.2.6. Activar/Desactivar Actualizaciones al inicio.

Caso de uso: Activar/Desactivar Actualizaciones al inicio.

Actores: Usuario de la aplicación.

Propósito: Activar ó desactivar la búsqueda de actualizaciones al inicio de la aplicación. Activar esta opción implica que se realizará una búsqueda de actualizaciones al iniciar la aplicación.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario habilita/deshabilita la opción de actualizaciones al inicio.	2. El sistema guarda la opción establecida por el usuario. El sistema consulta la opción guardada al iniciar la aplicación para actuar conforme estableció el usuario.

6.2.7. Mostrar Mapa.

Caso de uso: Mostrar Mapa.

Actores: Usuario de la aplicación.

Propósito: Mostrar un mapa de las instalaciones, en el que se podrán localizar todos los puntos de interés listados en el apartado 6.2.4.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa la opción "*Mapa*" del menú "*Olèrdola*".	2. El sistema muestra un mapa de las instalaciones del museo. Por cada punto de interés, se muestra una etiqueta marcando en el mapa la posición en la que se ubica el punto de interés en cuestión.

6.2.8. Vista Realidad Aumentada.

Caso de uso: Vista Realidad Aumentada

Actores: Usuario de la aplicación.

Propósito: Lanzar la herramienta de Realidad Aumentada para su uso en las instalaciones del museo.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa la opción "*Vista Realitat Augmentada*" del menú "*Olèrdola*".	2. El sistema lanza la herramienta de Realidad Aumentada para su uso en las instalaciones del museo.

6.2.9. Cambiar Idioma.

Caso de uso: Cambiar Idioma

Actores: Usuario de la aplicación.

Propósito: Cambiar el idioma en el que se muestran todos los textos de la aplicación.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario selecciona una opción de idioma de la lista de idiomas disponibles del menú "*Opcions*", esto es, catalán, castellano ó inglés.	2. El sistema guarda la opción de idioma seleccionado para mostrar, a partir de ese momento en adelante, todos los textos de la aplicación en el idioma seleccionado. Se vuelve a mostrar la misma pantalla "*Opcions*" en el nuevo idioma seleccionado.

6.2.10. Buscar Actualizaciones.

Caso de uso: Buscar Actualizaciones.

Actores: Usuario de la aplicación.

Propósito: Comprobar si existen actualizaciones disponibles para la aplicación.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario pulsa el botón *Buscar Actualizaciones ahora*.	2. El sistema lanza un mensaje de confirmación de la acción al usuario. El usuario puede seleccionar si desea realizar la acción ó cancelarla.
3. El usuario selecciona la opción de realizar la acción.	4. El sistema comprueba si existe alguna actualización disponible. En caso afirmativo, el sistema lanza un mensaje al usuario. El usuario puede

	seleccionar si desea realizar la actualización en este momento ó no realizar la actualización.

Cursos alternativos de acontecimientos:

Línea 3: El usuario selecciona la opción de no realizar la acción. El sistema oculta el mensaje de confirmación y no realiza ninguna acción.

Línea 4: El sistema comprueba que no existe ninguna actualización disponible. El sistema lanza un mensaje de información al usuario conforme no hay ninguna actualización disponible para la aplicación.

6.2.11. Actualizar Aplicación.

Caso de uso: Actualizar Aplicación.

Actores: Usuario de la aplicación.

Propósito: Actualizar la aplicación a la nueva versión disponible.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El usuario recibe en pantalla el mensaje del sistema, informando de existe una nueva actualización, y selecciona la opción de actualizar la aplicación.	2. El sistema realiza la actualización de la aplicación, mostrando en pantalla los avances del proceso. Al finalizar el proceso, el sistema muestra la pantalla de inicio.

6.2.12. Publicar Actualización.

Caso de uso: Publicar Actualización.

Actores: Administrador de actualizaciones.

Propósito: Publicar una actualización de la aplicación.

Tipo: Primario y esencial.

Curso típico de acontecimientos:

Acción del Actor	Respuesta del sistema
1. El administrador de actualizaciones publica en el servidor de actualizaciones los archivos necesarios para la actualización de la aplicación.	2. El sistema consulta los datos de actualización, ya sea automáticamente al iniciar la aplicación, si así está configurada, ó por orden directa del usuario de la aplicación, si éste pulsa el botón de buscar actualizaciones en el menú "*Opcions*".

6.3. Modelo de estados.

Este modelo es una representación de los diferentes estados de la aplicación a lo largo de su ciclo de vida, así como de las acciones necesarias para realizar un cambio de estado. Para realizar esta representación se hace uso de las siguientes entidades:

Estado: Condición de un objeto o de un caso de uso en un instante determinado en el tiempo. Tiene un cierto estado y/o está esperando algún tipo de evento.

Evento: Es una ocurrencia que puede hacer que el objeto realice una transición de un estado a otro.

Transición: Se trata de una relación entre estados como consecuencia de un evento.

6.3.1. Ciclo de vida de una aplicación Android.

El diagrama de estados de la figura 6.3.1.1. representa los posibles estados del ciclo de vida de una aplicación *Android* estándar.

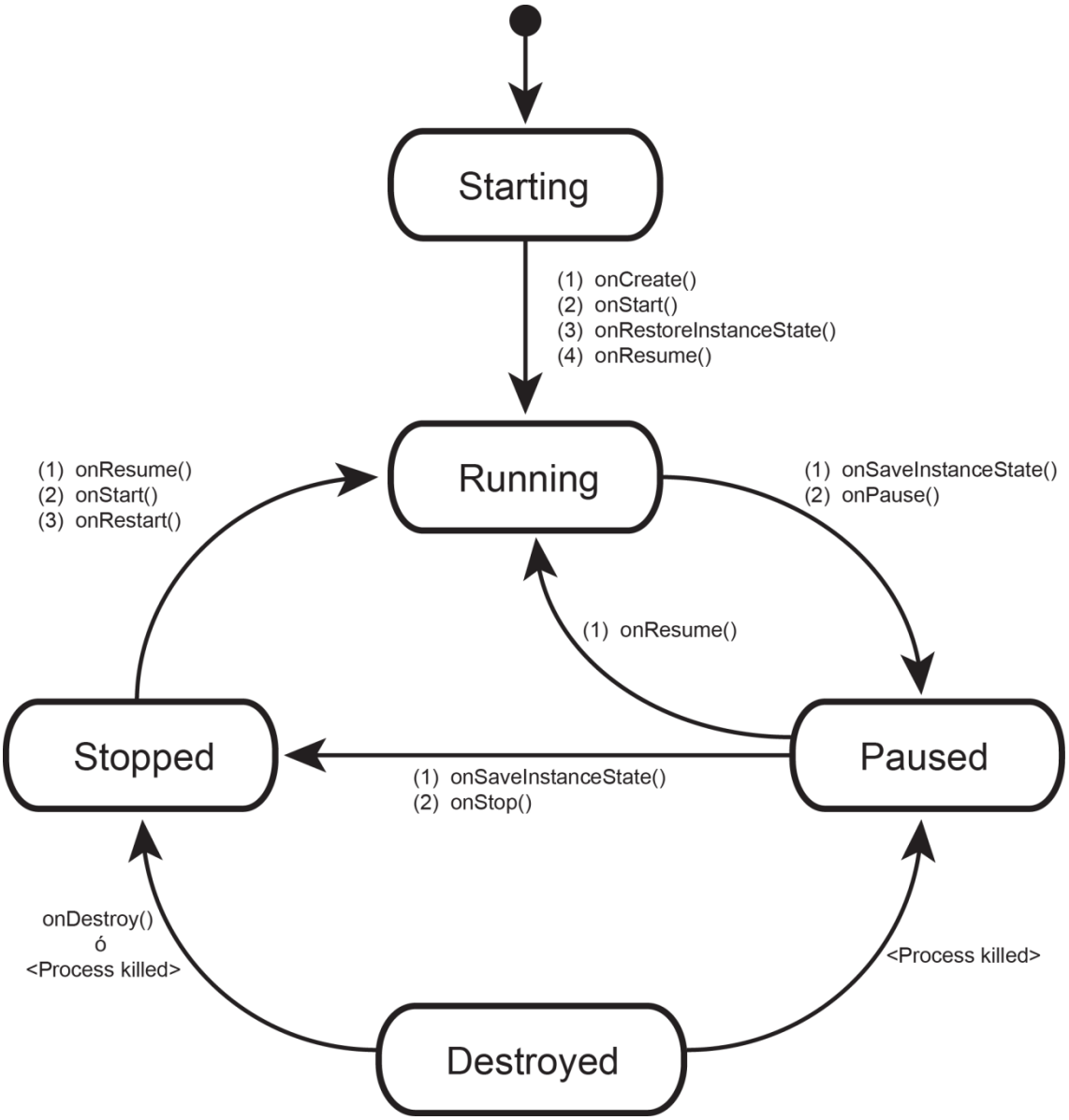

Figura 6.3.1.1. Diagrama de estados del ciclo de vida de una *Activity Android.*

El estado de *Running* es el primero en la pila de ejecución. En este estado, el usuario ve en pantalla la aplicación y puede interactuar con ella.

En el estado de *Paused*, la aplicación ha pasado a segundo plano pero aún está visible, porque otra actividad se muestra en primer plano sobre la aplicación. En este estado, la aplicación puede ser cerrada por el sistema si éste necesita liberar recursos para la nueva actividad.

En el estado de *Stopped*, la aplicación ha pasado a segundo plano y está completamente oculta por la nueva actividad que ha motivado el cambio de estado. En este estado, el sistema puede también optar por cerrar la aplicación si necesita liberar recursos.

El estado *Destroyed* implica que la aplicación ya no está disponible. Se han liberado todos sus recursos y, en caso de ser llamada de nuevo, será necesario comenzar un nuevo ciclo de vida.

6.3.2. Diagrama de estados de arranque de la aplicación.

El diagrama de la figura 6.3.2.1. representa los posibles estados en los que se puede encontrar la aplicación desde el momento de arranque hasta que se muestra la pantalla inicial de la aplicación (pantalla "*Inici*").

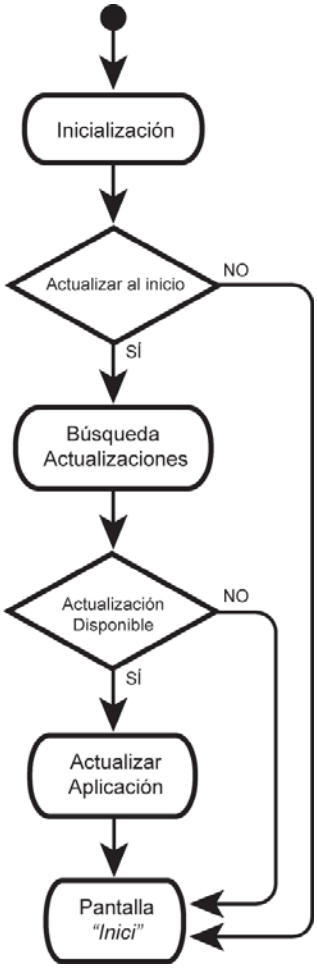

Figura 6.3.2.1. Diagrama de estados de arranque de la aplicación.

Si la aplicación está configurada para buscar actualizaciones en el momento de arranque, realiza la búsqueda de actualizaciones, si no, muestra directamente la pantalla *Inici*. Si realiza la búsqueda de actualizaciones y encuentra una actualización disponible, lanza el proceso de actualización de la aplicación, tras el cual muestra la pantalla *Inici.*

Capítulo 7
Implementación.

El objetivo de este capítulo es presentar los entornos de desarrollo utilizados, así como detalles, ventajas y problemas encontrados a lo largo de la implementación de las distintas tecnologías utilizadas.

7.1. Entornos para el desarrollo del sistema.

7.1.1. Eclipse IDE.

¿Qué es *Eclipse?*

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés *IDE*), como el *IDE* de *Java* llamado *Java Development Toolkit* (*JDT*) y el compilador (*ECJ*) que se entrega como parte de *Eclipse* (y que son usados también para desarrollar el mismo *Eclipse*).

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado *Eclipse Modeling Project*, cubriendo casi todas las áreas de *Model Driven Engineering*.

Eclipse fue desarrollado originalmente por *IBM* como el sucesor de su familia de herramientas para *VisualAge*. *Eclipse* es ahora desarrollado por la *Fundación Eclipse*, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la *Common Public License*, pero después fue re-licenciado bajo la *Eclipse Public License*. La *Free Software Foundation* ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con *Licencia pública general de GNU* (*GNU GPL*).

En el presente proyecto se utilizó la versión 3.7.2 del *IDE*, también conocida como *Eclipse Indigo*.

¿Por qué *Eclipse?*

Los dos principales *IDE* para desarrollo de proyectos *Java* más populares son históricamente *Eclipse* y *NetBeans*. Básicamente, la gran diferencia entre ambos *IDE* es que, mientras *Eclipse* es mucho más eficiente a la hora de gestionar el código, *NetBeans* es mejor en el diseño de componentes gráficos. De hecho, algunos desarrolladores emplean ambos *IDE* en paralelo en sus proyectos, utilizando el primero para escribir el código fuente de las aplicaciones y el segundo para diseñar la interfaz de usuario. Debido a que en el presente proyecto el diseño de la interfaz de usuario se basa en lenguaje *web*, no iba a ser necesario el *IDE* de desarrollo de *Java* para este fin, por lo que descartamos *NetBeans* en beneficio de *Eclipse.*

Un tercer *IDE* muy popular es *JBuilder*, aunque en muchos aspectos, *Eclipse* lo supera. *Eclipse* es completamente gratuito, está soportado por *IBM*, está construido siguiendo una arquitectura de *plugins* que lo hace extremadamente extensible, etcétera.

7.1.1. Adobe Dreamweaver.

¿Qué es *Adobe Dreamweaver?*

Adobe Dreamweaver es una aplicación en forma de suite (basada en la forma de estudio de *Adobe Flash*) que está destinada a la construcción, diseño y edición de sitios y aplicaciones *web* basados en estándares. Creado inicialmente por Macromedia (actualmente producido por *Adobe Systems*) su principal competidor es *Microsoft Expression Web* y tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras. Hasta la versión *MX*, fue duramente criticado por su escaso soporte de los estándares de la *web*, ya que el código que generaba era con frecuencia sólo válido para *Internet Explorer* y no validaba como *HTML* estándar. Esto se ha ido corrigiendo en las versiones recientes.

La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización del mismo, puesto que en este programa, sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en *Javascript-C*, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de *C++* sino rutinas de *Javascript* que

hace que sea un programa muy fluido, que todo ello hace, que programadores y editores *web* hagan extensiones para su programa y lo ponga a su gusto.

Las versiones originales de la aplicación se utilizaban como simples editores *WYSIWYG*. Sin embargo, versiones más recientes soportan otras tecnologías *web* como *CSS*, *JavaScript* y algunos *frameworks* del lado servidor. Esta aplicación está disponible tanto para la plataforma *MAC* como para *Windows*, aunque también se puede ejecutar en plataformas basadas en *UNIX* utilizando programas que implementan las *API*'s de *Windows*, tipo *Wine*.

Como editor *WYSIWYG* que es, *Dreamweaver* permite ocultar el código *HTML* de cara al usuario, haciendo posible que alguien no entendido pueda crear páginas y sitios web fácilmente sin necesidad de escribir código.

Algunos desarrolladores *web* criticaban esta propuesta ya que crean páginas *HTML* más largas de lo que solían ser al incluir mucho código inútil, lo cual va en detrimento de la ejecución de las páginas en el navegador *web*. Esto puede ser especialmente cierto ya que la aplicación facilita en exceso el diseño de las páginas mediante tablas. Además, algunos desarrolladores web han criticado *Dreamweaver* en el pasado porque creaba código que no cumplía con los estándares del *consorcio Web (W3C)*. No obstante, *Adobe* ha aumentado el soporte *CSS* y otras maneras de diseñar páginas sin tablas en versiones posteriores de la aplicación, haciendo que se reduzca el exceso de código.

Dreamweaver permite al usuario utilizar la mayoría de los navegadores *web* instalados en su ordenador para previsualizar las páginas *web*. También dispone de herramientas de administración de sitios dirigidas a principiantes como, por ejemplo, la habilidad de encontrar y reemplazar líneas de texto y código por cualquier tipo de parámetro especificado, hasta el sitio *web* completo. El panel de comportamientos también permite crear *JavaScript* básico sin conocimientos de código.

Con la llegada de la versión *MX*, *Macromedia* incorporó herramientas de creación de contenido dinámico en *Dreamweaver*. En lo fundamental de las herramientas *HTML WYSIWYG*, también permite la conexión a bases de datos como *MySQL* y *Microsoft Access*, para filtrar y mostrar el contenido utilizando tecnología de *script* como, por ejemplo, *ASP*, *ASP.NET*, *ColdFusion*, *JSP* (*JavaServer Pages*) y *PHP* sin necesidad de tener experiencia previa en programación.

Un aspecto de alta consideración de *Dreamweaver* es su funcionalidad con extensiones. Es decir, permite el uso de "Extensiones". Las extensiones, tal y como se conocen, son pequeños programas, que cualquier desarrollador web puede escribir (normalmente en *HTML* y *Javascript*) y que cualquiera puede descargar e instalar, ofreciendo así funcionalidades añadidas a la aplicación. *Dreamweaver* goza del apoyo de una gran comunidad de desarrolladores de extensiones que hacen posible la

disponibilidad de extensiones gratuitas y de pago para la mayoría de las tareas de desarrollo *web*, que van desde simple efectos *rollover* hasta completas cartas de compra.

También podría decirse que, para un diseño más rápido y a la vez fácil, podría complementarse con *Fireworks* en donde se podría diseñar un menú u otras creaciones de imágenes (*gif web*, *gif websnap*, *gif* adaptable, *jpeg* calidad superior, *jpeg* archivo más pequeño, *gif* animado *websnap*) para un sitio *web* y después exportar la imagen creada y así utilizarla como una sola en donde ya llevará los vínculos para dicho sitio.

En el presente proyecto se utilizó *Adobe Dreamweaver CS5.5* versión 11.5.

¿Por qué *Dreamweaver?*

Dreamweaver es el programa más utilizado en el sector del diseño y la programación *web*, por sus funcionalidades, su integración con otras herramientas como *Adobe Flash* y, recientemente, por su soporte de los estándares del *World Wide Web Consortium*. El *IDE* ha tenido un gran éxito desde finales de los años 1990 y actualmente mantiene el 90% del mercado de editores *HTML*. La gran cantidad de herramientas que pone a disposición del usuario, así como su facilidad de uso, hacen de este el *IDE* por excelencia para el desarrollo web.

7.2. Estrategia de actualización de la aplicación.

En este punto justificaremos las decisiones de diseño y los motivos que llevaron a emprender la estrategia de actualización de la aplicación. Así mismo, explicaremos en qué consiste dicha estrategia, y cómo se implementó finalmente.

La primera condición y más importante a la hora de tomar las siguientes decisiones de diseño fue la necesidad, por parte del cliente, de poder actualizar los contenidos de la aplicación sin tener que modificar el código fuente de la aplicación. Modificar dicho código fuente implica varias situaciones:

- Para poder realizar el proceso de publicación de la aplicación en el *App Store* de *Apple* es necesaria una licencia de desarrollador de *Apple*. Si el cliente se viera obligado a modificar el código fuente de la aplicación para modificar los contenidos, aparte de necesitar ciertos conocimientos técnicos en programación, el cliente necesitaría una licencia de desarrollador de *Apple*.

- El proceso de publicación de la aplicación pasa, en el caso de *Apple*, por un proceso de revisión de la aplicación, para comprobar que la misma cumple una serie de requisitos que la marca impone a todas las aplicaciones que se publican en su tienda *online*. Ello retrasa y complica el proceso de actualización de la aplicación.

Por otra parte, la actualización de contenidos requerida por la aplicación en este caso no es un requerimiento excesivamente complicado, debido a que se trata de una aplicación mixta. La aplicación en sí es básicamente una página *web* integrada en una aplicación nativa, la cual corresponde completamente a la interfaz de usuario. Por este motivo, la actualización se definió como se explica a continuación:

- Entre los datos que se copian en el dispositivo al instalar la aplicación, se encuentra la página *web* integrada como parte de la aplicación.
- Mientras no se realice ninguna actualización, la aplicación utiliza los archivos *web* integrados en la aplicación.
- La actualización consiste en descargar de un servidor remoto una nueva versión completa de la página *web* integrada, guardándola en memoria remanente del dispositivo, típicamente la tarjeta de memoria.
- Tras la actualización, y en adelante, la aplicación no utiliza más la página *web* integrada, sino la nueva versión descargada. La aplicación guarda en un parámetro tipo *String* la ruta al archivo de inicio de la página *web* en memoria remanente para iniciar la aplicación desde ese punto.

El proceso de actualización comienza con la descarga de un archivo de parámetros desde el servidor remoto. Dicho archivo debe cumplir las siguientes condiciones:

- El nombre de archivo debe ser "*updparams.json*".
- El archivo debe tener formato de texto *JSON*.
- El primer parámetro del archivo debe llamarse "*timestamp*", y su valor debe ser una fecha en el siguiente formato: "YYYY/mm/dd hh:mm:ss" (por ejemplo, "2012/06/18 12:30:00").
- El segundo parámetro del archivo debe llamarse "*numarch*", y su valor debe corresponder con el número de archivos a descargar.

- El número de parámetros restantes debe corresponder con el valor del segundo parámetro del archivo. Cada uno de estos parámetros debe llamarse "*archXXXX*", donde "*XXXX*" indica un valor numérico entre el "*0000*" y el "*9999*". Los parámetros deben estar ordenados por este valor numérico. El valor de cada uno de estos parámetros corresponderá al nombre absoluto del archivo a descargar desde el directorio de donde se descarga el archivo de parámetros. El símbolo para separar los directorios en el valor será la barra simple (carácter "/"). Un ejemplo del parámetro descrito podría ser "*css/jquery-mobile-1.0/images/ajax-loader.png*".

Para facilitar la tarea de generación de este archivo y asegurar su corrección, se diseñó e implementó una sencilla aplicación *Java* de escritorio. La aplicación proporciona una herramienta mediante la cual el usuario puede seleccionar un directorio del ordenador donde se ejecuta la aplicación. A partir del directorio indicado por el usuario, la aplicación de escritorio genera un archivo de texto como el descrito, con la fecha del momento de creación del archivo, el número total de archivos en el directorio indicado, y la lista formateada de parámetros con los nombres absolutos de archivo.

A partir de este archivo, la aplicación conoce la fecha de la actualización disponible, el número total de archivos a descargar y los nombres absolutos de todos ellos. La aplicación genera un directorio temporal en la memoria remanente del dispositivo, descarga en él los archivos, y renombra el directorio con el nombre de "*macolerdola*", sustituyendo si fuera necesario el antiguo directorio *web* de ejecución de la aplicación. La aplicación guarda en un parámetro la ruta al archivo *web* inicial, para utilizarlo de ahora en adelante.

Por otra parte, todos los textos de la página *web* integrada en la aplicación, es decir, de la interfaz de usuario, están traducidos a tres idiomas: catalán, castellano e inglés. Estos textos se encuentran en tres archivos con formato *JSON*. Estos archivos son *localize-ca.json* para las traducciones en catalán, *localize-es.json* para las traducciones en castellano y *localize-en.json* para las traducciones en inglés. Este formato de archivo facilita mucho las tareas de modificación de contenidos al cliente, debido a la sencillez de su formato, que lo hace accesible y comprensible incluso para personal inexperto en programación.

Durante la fase de implementación de la aplicación, se realizaron las pruebas de actualización de la aplicación utilizando un servidor *web* gratuito, en el que se subían los archivos vía *FTP.*

7.3. Diagramas de secuencia.

Los diagramas de secuencia permiten mostrar las relaciones entre los diferentes objetos. Estas interacciones entre objetos se pueden producir por un evento externo, invocación de operaciones en un objeto, ejecución de un caso de uso, etcétera. A continuación, se muestran algunos diagramas de secuencia, los cuales muestran algunos de los casos de uso descritos en el capítulo anterior y otras interacciones de la aplicación.

7.3.1. Diagrama de secuencia de arranque de la aplicación.

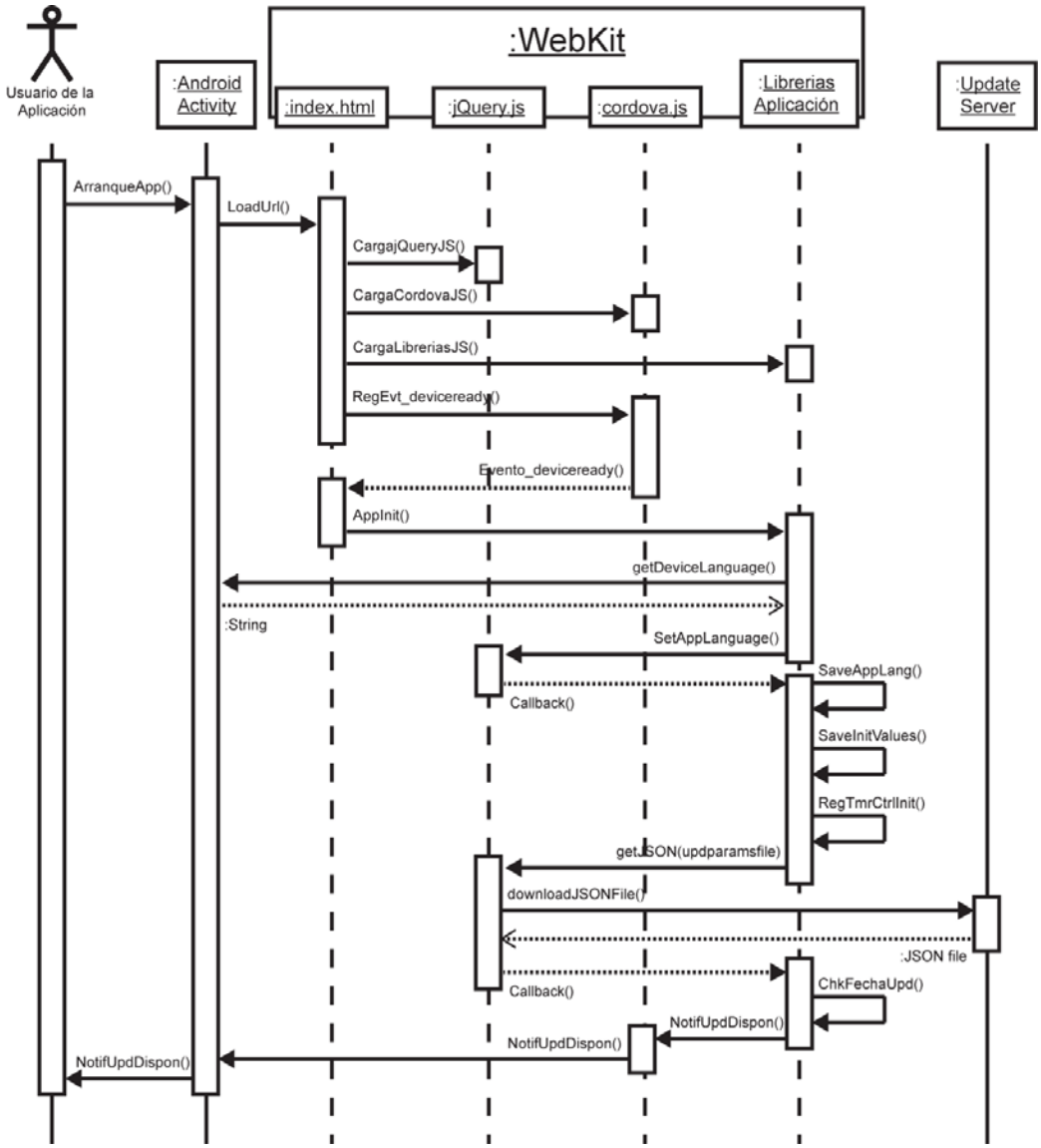

Figura 7.3.1. Diagrama de secuencia de arranque de la aplicación.

El diagrama de la figura 7.3.1 representa la secuencia de arranque de la aplicación. En este punto explicaremos la secuencia de arranque de la aplicación, ampliando algunos aspectos de la secuencia que el diagrama no especifica.

La secuencia comienza en el mismo momento en que el usuario de la aplicación arranca la aplicación. *Android* arranca la aplicación, creando la instancia *Activity* que gestionará la interacción con el usuario. Al crear dicha instancia, se lanza un evento que llama a la función *onCreate(Bundle)*, desde donde se inicializa *Phonegap*. En esta función se realizan otras acciones como la eliminación de la caché del webkit o la creación de las funciones puente entre código nativo y código *web*. La última de las acciones que se realizan en esta función es la llamada a *loadUrl(String)*, la cual lanza el webkit con la página *web* indicada como parámetro, que en nuestro caso es el archivo *index.html*, añadido como un *asset* más a la aplicación *Android*.

El *webkit* procesa la página *web*, cargando las librerías J*avascript* de *jQuery Mobile*, *Cordova (PhoneGap)*, así como otros archivos *Javascript,* implementados específicamente para la aplicación, que contienen la lógica de negocio. Por último, registra el evento *deviceready* para la función *PhoneGapReady()*. Dicho evento indica unívocamente que se han cargado todos los módulos necesarios para el funcionamiento de *PhoneGap*. Al lanzarse el evento, se llama automáticamente a la función *PhoneGapReady*, la cual realiza tareas menores antes de, finalmente llamar a la función *initApp()* del archivo *appinit.js*.

La función *initApp()* establece el idioma de la aplicación. Si el idioma fue establecido en un arranque anterior recupera dicho valor y establece el idioma correspondiente. En caso contrario, establece automáticamente el idioma de la aplicación en función de si el lenguaje establecido en el dispositivo móvil es catalán, castellano ó inglés. En caso que no sea ninguno de ellos establece el inglés como lenguaje de la aplicación.

A continuación, se inicializan algunos parámetros de la aplicación con sus valores por defecto, como por ejemplo la fecha de última actualización, la búsqueda de actualizaciones al iniciar la aplicación (por defecto activada), etcétera.

El siguiente paso a realizar es comprobar si se requiere una búsqueda de actualizaciones. La búsqueda es requerida si la aplicación tiene activada la opción de búsqueda de actualizaciones en el arranque, ó si el usuario ha pulsado el botón de búsqueda de actualizaciones (como se verá más adelante, se puede lanzar esta línea de ejecución desde ambos casos). En caso afirmativo, se activa un temporizador que ejecutará periódicamente la función *ctrlInit*(), la cual se encarga de monitorizar si ha finalizado o no el proceso de actualización y si lo ha realizado con éxito. También se encarga de controlar el tiempo de actualización transcurrido, pudiendo cancelar la

actualización si éste sobrepasa un determinado valor, el cual se establece en función del número de archivos a descargar.

El siguiente paso consiste en la descarga de un archivo ubicado en un servidor remoto. El archivo contiene los parámetros de la actualización, entre los que se encuentra la fecha de la actualización. Los detalles de dicho archivo, como el formato y la descripción detallada de dichos parámetros, se explican más adelante en el apartado correspondiente a la actualización de la aplicación. El archivo se descarga mediante la función *getJSON()* de la librería *jQuery*. Se trata de una función asíncrona, a la cual se le pasa como parámetro la función que ha de ser llamada al finalizar la descarga del archivo remoto.

Al finalizar la descarga del archivo, *jQuery* realiza la llamada de *callback* a la función que se pasó como parámetro. En esta función, se recopilan los parámetros de actualización del archivo en dos vectores de tipo *String*; uno comprende los nombres de los parámetros, el otro los valores correspondientes a dichos parámetros. A continuación, se consulta el parámetro de fecha de actualización recibido en el archivo remoto, y si compara con el parámetro local de fecha de última actualización de la aplicación en el dispositivo móvil. En caso de que la fecha indicada en el archivo remoto sea más actual que la fecha guardada en el dispositivo móvil, se determina que la versión de la aplicación del dispositivo es anterior a la versión de la aplicación disponible en el servidor remoto. Se lanza pues una notificación por pantalla, mediante la cual se informa al usuario de que existe una actualización disponible para la aplicación. En la notificación se ofrece al usuario la opción de proceder o no a la actualización.

7.3.2. Diagrama de secuencia de búsqueda de actualizaciones.

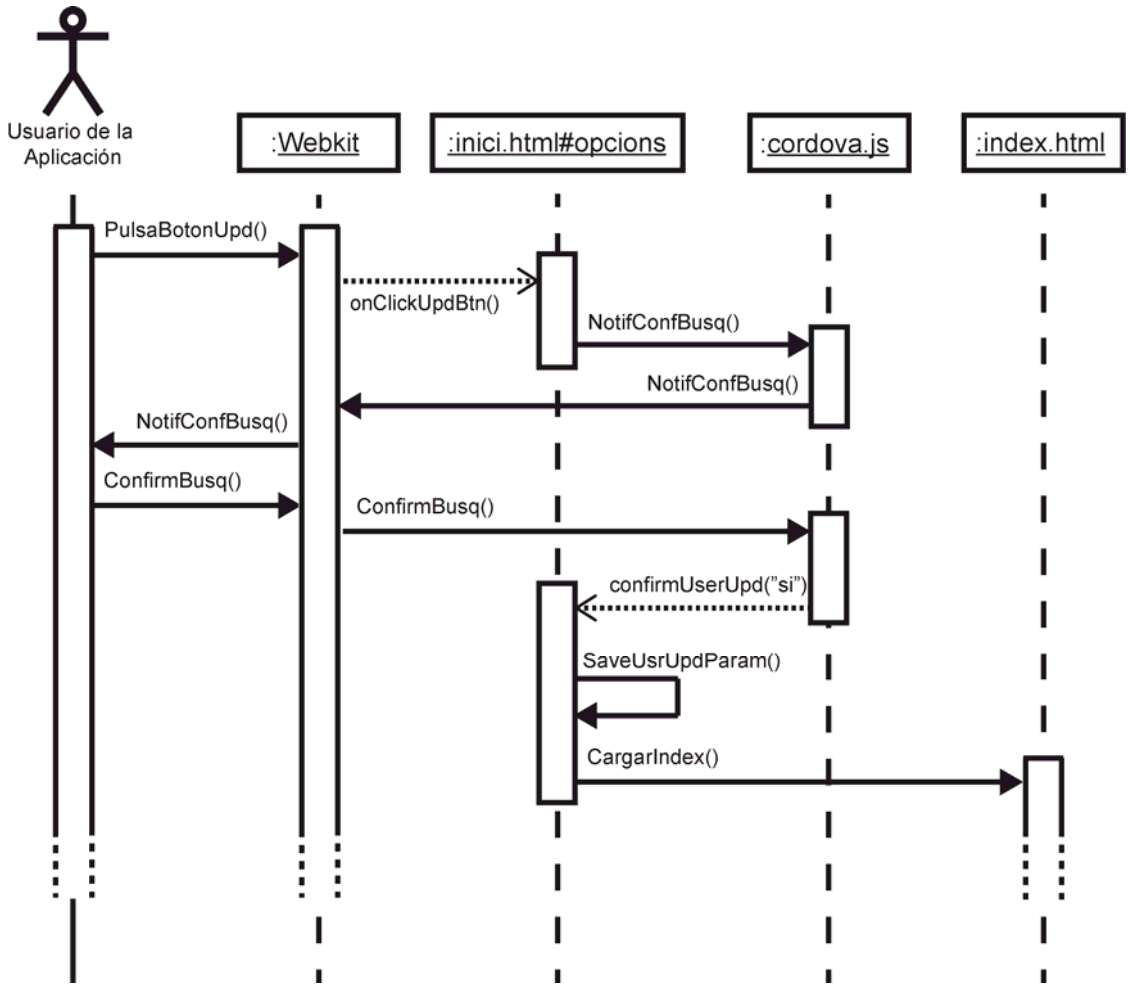

Figura 7.3.2. Diagrama de secuencia de búsqueda de actualizaciones.

La búsqueda de actualizaciones puede ser realizada de dos maneras. La primera manera, como se ha descrito en el apartado anterior, se realiza en el momento de arranque de aplicación. La segunda manera de realizar la acción es por acción directa del usuario de la aplicación, pulsando el botón "*Buscar Actualizaciones*" en el menú de la pantalla *Opcions*, como se muestra en el diagrama de la figura 7.3.2.

Al pulsar el usuario de la aplicación el botón "*Buscar Actualizaciones*", se lanza el evento *onClick()* del botón. En la correspondiente función, se muestra una notificación por pantalla, en la que el usuario de la aplicación debe confirmar la realización de la acción. En caso afirmativo, se guarda un testigo que indica que el usuario desea buscar actualizaciones y, a continuación, se procesa la página *index.html*. Dicha página, como se ha explicado en el apartado anterior, discernirá si hay que realizar una búsqueda de actualizaciones basándose en el testigo guardado, y continuará con el proceso hasta ofrecer la actualización al usuario, en caso de que haya alguna disponible.

7.3.3. Diagrama de secuencia de actualización de la aplicación.

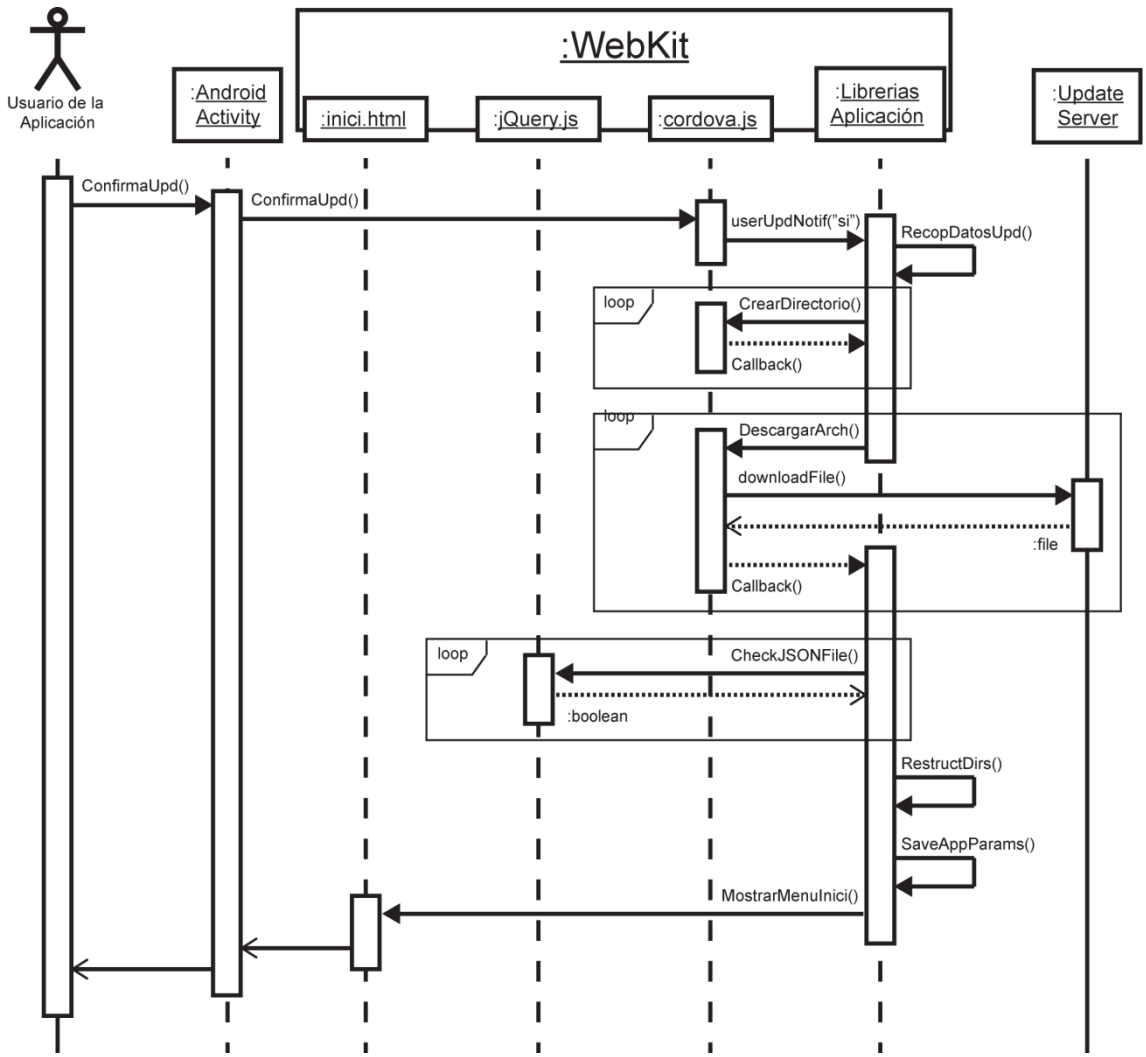

Figura 7.3.3. Diagrama de secuencia de actualización de la aplicación.

El diagrama de la figura 7.3.3 representa la secuencia de actualización de la aplicación. El diagrama muestra el proceso de actualización comenzando en el momento en que el usuario ha sido notificado de la existencia de una actualización y confirma la acción de actualización.

Tras la pulsación del botón, *PhoneGap* realiza la llamada a la función que fue programada para ello al lanzar la notificación al usuario. Dicha función recoge la decisión del usuario. En caso afirmativo, la función *InitApp()* descrita anteriormente llama a la función *UpdApp()*, la hace efectiva la actualización de la aplicación.

La función *UpdApp()* consiste en un bucle principal del cual sólo se sale si se agota el tiempo destinado a la actualización (*timeout*), o si finaliza la actualización con

éxito o error. Dicho bucle, anida una estructura de control tipo *select-case*, la cual se encarga de controlar la secuencia de actualización. Al realizarse multitud de llamadas a funciones asíncronas durante el proceso de actualización, esta estructura de control es extremadamente útil. Estas llamadas asíncronas, como por ejemplo la descarga de archivos o la creación de directorios en la tarjeta de memoria del dispositivo, reciben como parámetro una función a la que llaman al finalizar su ejecución (*callback*). La estrategia utilizada en estos casos fue la siguiente:

1. Situación: estando en un estado determinado en la estructura de control *select-case*, se quiere realizar una llamada a una función asíncrona.

2. A la variable de control de estado se le asigna el valor de control del estado donde se quiere continuar la ejecución al finalizar la llamada asíncrona.

3. A la variable de control del bucle principal se le asigna el valor adecuado para salir del bucle.

4. Se realiza la llamada asíncrona pasando como parámetro de *callback* la propia función *UpdApp()*, y se sale de la función, a la espera de la llamada de vuelta de la función asíncrona.

5. Al finalizar, la función asíncrona llama a la función *UpdApp()*, la cual continúa la ejecución en el estado indicado en el punto 2.

En el primer estado de la función *UpdApp()* se lee el parámetro que indica el número de archivos a descargar. El archivo remoto de parámetros indica también los nombres absolutos (directorio y nombre de archivo) de todos los archivos que hay que descargar. Así, en el siguiente estado, se comprueba que todos los archivos que hay que descargar están disponibles en el archivo remoto. Esta función aún no está implementada internamente, por lo que de momento siempre devuelve cierto, dando por hecho que los archivos están disponibles en el servidor remoto. En el siguiente estado, se recopilan todos los directorios que hay que crear, a partir de los nombres de los archivos.

A continuación, se crea un directorio temporal en la raíz de la memoria remanente del dispositivo. Este directorio servirá de directorio raíz para crear la estructura de directorios y ubicar los archivos *web* de la aplicación que se descarguen. En el siguiente paso se crea el árbol de directorios según los nombres de archivo. Una vez creada la estructura de directorios, se descargan los archivos remotos, cada uno en la ubicación correspondiente según el nombre absoluto del archivo. Cada vez que se descarga un archivo con extensión *json*, se memoriza el nombre del archivo y, una vez

descargados todos los archivos de la actualización, se comprueba que la sintaxis de los archivos con extensión *json* corresponde con el formato de texto *JSON*.

Seguidamente se procede a la reestructuración de los directorios. Se renombra el directorio raíz en el que se han realizado las descargas, para pasar a ser el nuevo directorio *web* que la aplicación utilizará en adelante como interfaz de usuario. Antes de esto, se comprueba si existía previamente un directorio de ejecución de una actualización previa. En caso afirmativo, el directorio de ejecución previo se renombra temporalmente, el nuevo directorio se renombra como actual y, en caso de éxito, se elimina el antiguo directorio de ejecución de la aplicación.

Finalmente, se guarda la fecha de actualización indicada en el archivo remoto de parámetros, y se finaliza con éxito la tarea de actualización, para que la tarea de monitorización de la actualización lance la versión actualizada de la aplicación.

7.3.4. Diagrama de secuencia Activación/Desactivación de actualizaciones en el arranque.

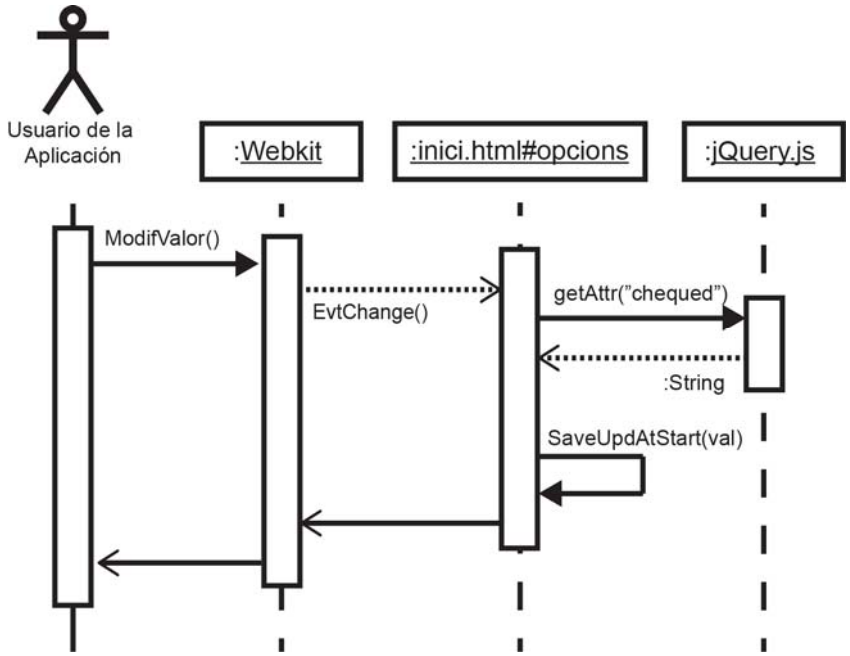

Figura 7.3.4. Diagrama de secuencia de Activación/Desactivación de actualizaciones en el arranque de la aplicación.

El diagrama de la figura 7.3.4 representa la secuencia de activación y desactivación de la opción de búsqueda de actualizaciones en el momento de arranque de la aplicación. Esta opción sólo es modificable desde la pantalla *opcions* de la interfaz de usuario. En el momento

en que el usuario modifica el valor de la opción, se lanza un evento que es capturado por una función, la cual se encarga primero de comprobar el valor del componente *jQuery*, que se trata de una típica casilla de verificación. En función del valor de dicho componente, se guarda la opción seleccionada por el usuario en la memoria del dispositivo y se devuelve el control a la interfaz de usuario.

7.3.5. Diagrama de secuencia Cambiar idioma.

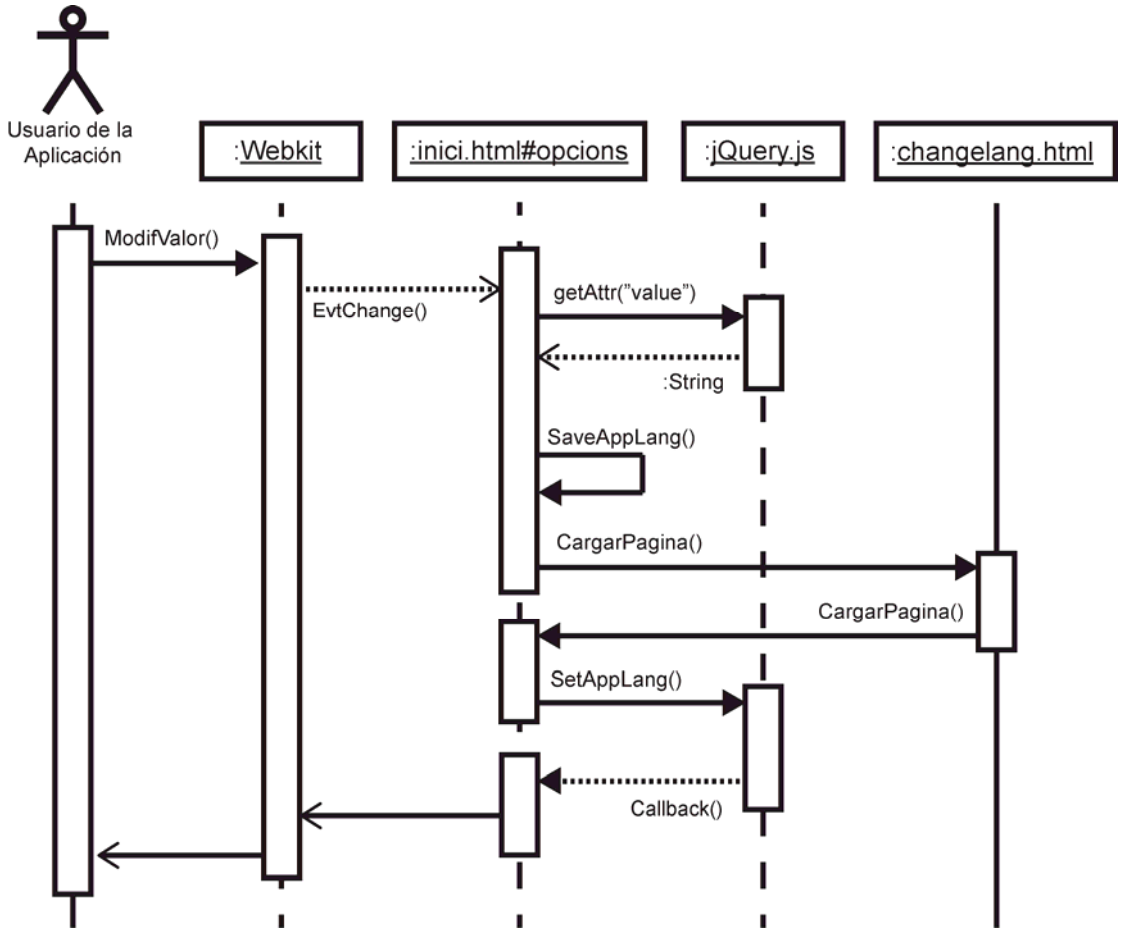

Figura 7.3.5. Diagrama de secuencia del cambio de idioma de la aplicación.

El diagrama de la figura 7.3.5 representa la secuencia de cambio de idioma de la aplicación. En la fase de implementación de la aplicación nos topamos con un problema a la hora de realizar el cambio de idioma de la aplicación. Al aplicar un idioma a la interfaz de usuario, la librería utilizada para este fin sustituye los textos de la interfaz de usuario por los textos del idioma seleccionado. Ello provocaba que el formato de los diversos componentes *web* de la interfaz de usuario se vieran alterados, sufriendo modificaciones como cambios de tamaño de fuente, tamaño de botones, etcétera. Esto se resolvió volviendo a cargar la página

tras aplicar el cambio de idioma. De esta manera, *jQuery* vuelve a aplicar los estilos sobre el *DOM* ya generado con el idioma deseado. Para conseguir este efecto se creó una pantalla en blanco en un archivo *HTML* independiente, sobre el que la aplicación "pivotaría" en el proceso de cambio de idioma.

El cambio de idioma sólo se puede realizar desde la pantalla *opcions* de la aplicación. En el momento en que el usuario selecciona un valor de la lista desplegable de idiomas, se lanza un evento que es capturado por una función, la cual se encarga primero de recoger el valor de la lista desplegable y guardar el idioma seleccionado por el usuario en la memoria del dispositivo. A continuación, se carga la página en blanco del archivo independiente *changelang.html*. Dicha página vuelve a cargar inmediatamente la interfaz de usuario directamente en la pantalla *opcions*. Finalmente, desde la pantalla *opcions* se aplica a la interfaz de usuario el idioma seleccionado.

7.4. Modificación de contenidos y publicación de actualizaciones.

En este apartado explicaremos las opciones de las que dispone el cliente para modificar los contenidos de la aplicación, así como el proceso de publicación de actualizaciones.

Como se explicó anteriormente, la aplicación *Android* finalmente implementada es una aplicación mixta, consistente en un contenedor escrito en lenguaje nativo (*Java*) con una página *web* integrada en la aplicación. El proceso de actualización consiste en la descarga por parte de la aplicación de una nueva página *web*, que sustituirá por completo la página *web* originalmente integrada en la aplicación. Esta estrategia de actualización de la aplicación expone completamente el código fuente correspondiente a la página *web*, esto es, la interfaz de usuario, de manera que el cliente final dispone de control total sobre ésta para realizar las modificaciones que considere oportunas, con todas las ventajas e inconvenientes que ello implica.

Por otra parte, los textos de la aplicación, como se dijo en apartados anteriores, están contenidos en tres archivos independientes para los tres idiomas de aplicación disponibles. De esta manera, el cliente final es capaz de modificar el contenido de los textos de todas las pantallas de la interfaz de usuario con un simple editor de texto, sin tener que modificar ni profundizar de ninguna manera en el código fuente original de la página *web* integrada en la aplicación. El contrapunto es que, aunque sólo modifique los textos de los archivos de idiomas, la aplicación necesita descargar, además del archivo *updparams.json*, todos los archivos que conforman la página *web* completa.

El proceso de publicación de actualización es muy sencillo. El cliente final únicamente debe ubicar en el servidor remoto el archivo de parámetros de actualización (*updparams.json*) en la raíz de la nueva página *web* actualizada, junto con el resto de archivos de actualización. Para facilitar la generación del archivo *updparams.json* se proporciona una herramienta de *software,* la cual genera automáticamente el archivo con el formato y parámetros correctos.

7.5. La Realidad Aumentada en la aplicación.

Aunque aún no se ha implementado ninguna herramienta de realidad aumentada, la aplicación está preparada para incorporar, en una segunda fase de desarrollo del proyecto, las herramientas de realidad aumentada que se implementen. Con este fin, la interfaz de usuario proporciona dos componentes desde los que se podrá acceder a las herramientas de realidad aumentada que se incorporen a la aplicación en una fase posterior del proyecto.

Desde la pantalla de puntos de interés, el usuario puede seleccionar un punto de interés de la lista, tras lo cual se muestra una pantalla con información sobre el punto de interés seleccionado, así como un botón de localización del punto de interés. Este botón está destinado a ser enlazado con una herramienta de realidad. Dicha herramienta mostrará al usuario la ubicación del punto de interés respecto a la posición actual del dispositivo, utilizando técnicas de realidad aumentada.

En la pantalla "*Olèrdola*", la opción "*Visita Realitat Augmentada*" está destinada a ser enlazada con la herramienta de realidad aumentada que el usuario utilizará en la visita a las instalaciones. Dicha herramienta, como se especifica en el presente documento, mostrará la ruta a seguir por el usuario visitando los puntos de interés en las instalaciones, pudiendo ubicarlos en la vista de realidad aumentada.

7.6. La aplicación implementada.

En este apartado presentamos figuras en las que se muestran capturas de pantalla la aplicación ejecutada en un dispositivo virtual *Android* generado con el *software* del *kit* de desarrollo de *Android*. El idioma detectado en el dispositivo virtual es el inglés, por lo que todas las capturas son mostradas en dicho idioma. Alguna pantallas se muestran sesgadas debido a que son demasiado largas para poder ser mostradas en una sola imagen.

En la figura 8.1 se muestra el menú de aplicaciones instaladas en el dispositivo virtual, entre las que se encuentra nuestra la aplicación "*MAC-Olerdola*".

Figura 8.1. Menú de aplicaciones instaladas en el dispositivo virtual *Android*.

La aplicación está configurada por defecto para comprobar si existen actualizaciones en el arranque de la aplicación. En la figura 8.2 se muestra la pantalla de búsqueda de actualizaciones.

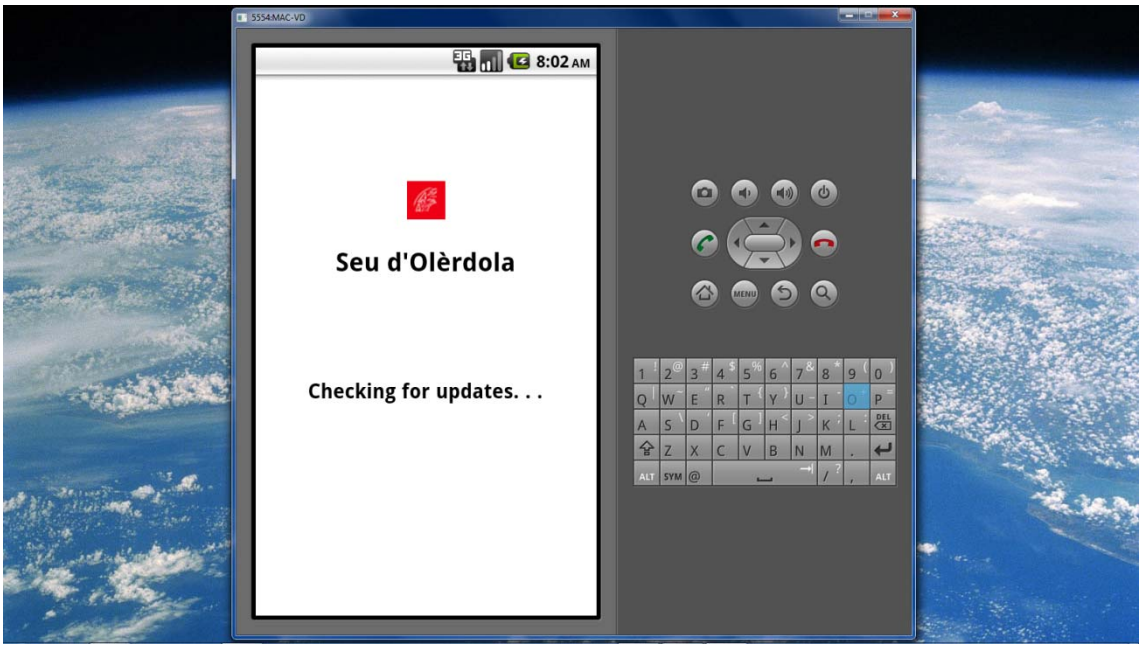

Figura 8.2. Pantalla de actualizaciones de la aplicación.

En caso de encontrar actualizaciones, la aplicación ofrece al usuario la posibilidad de actualizar la aplicación. En la figura 8.3 se puede ver la pantalla que se muestra al usuario.

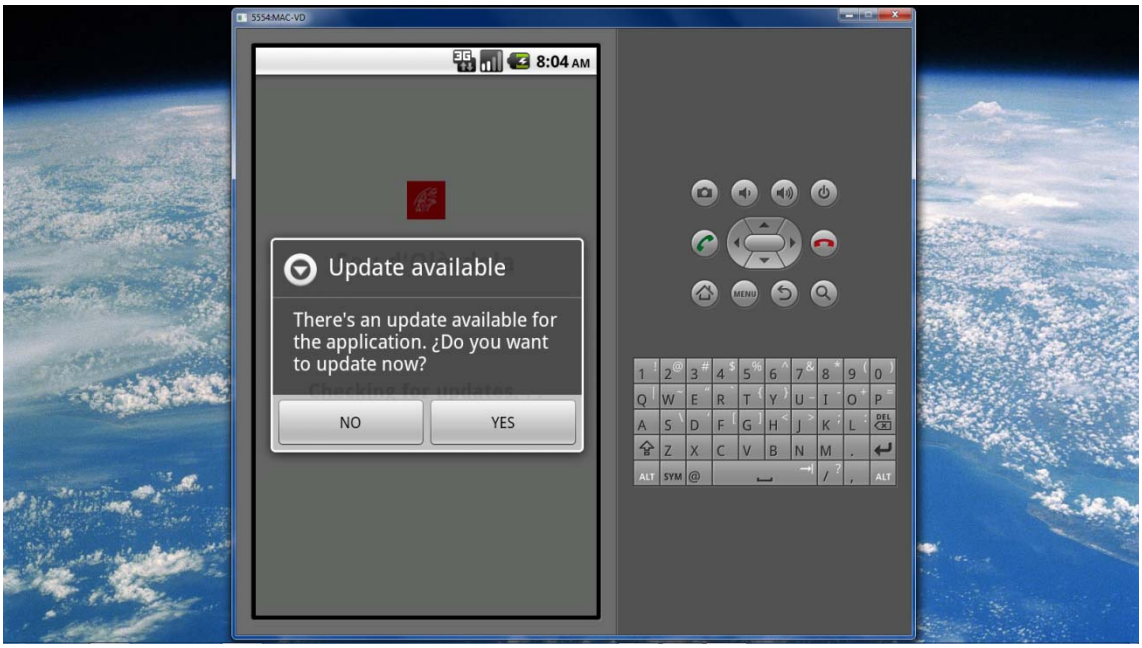

Figura 8.3. Mensaje de notificación al usuario para la instalación de actualizaciones.

Si el usuario decide actualizar la aplicación, el proceso de actualización muestra los detalles del proceso por pantalla como se puede observar en la figura 8.4.

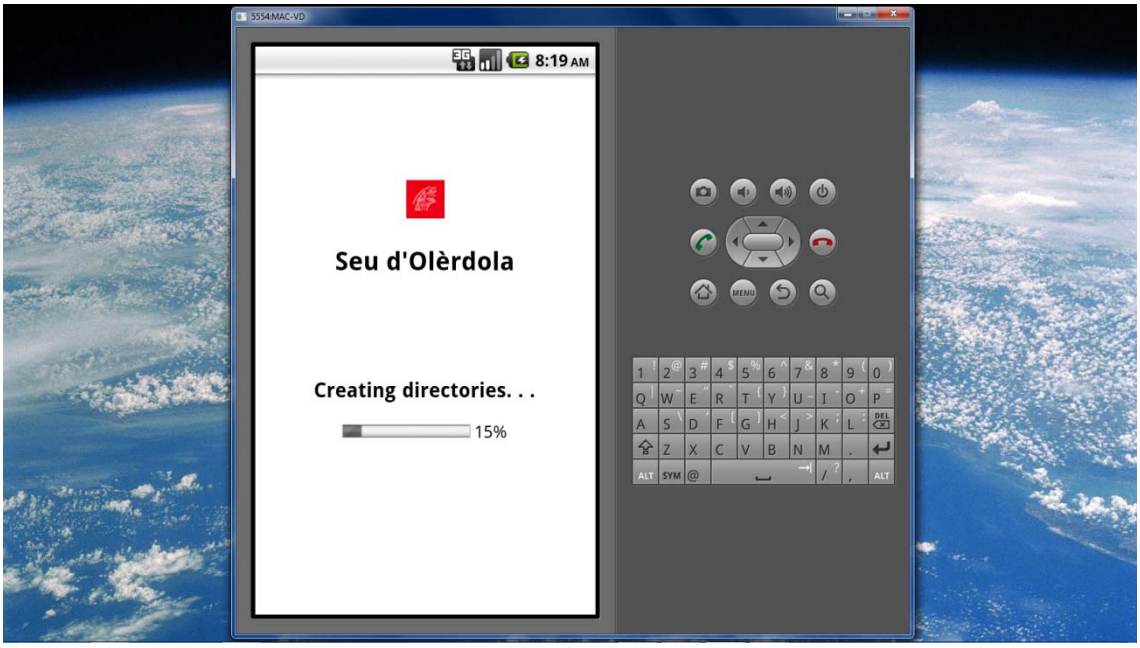

Figura 8.4. Pantalla de actualizaciones, mostrando las tareas de actualización de la aplicación.

En caso de que no existan actualizaciones, ó tras el proceso de actualización (decida o no el usuario realizar la actualización), la aplicación muestra el menú "*Inici*", como se puede ver en la figura 8.5.

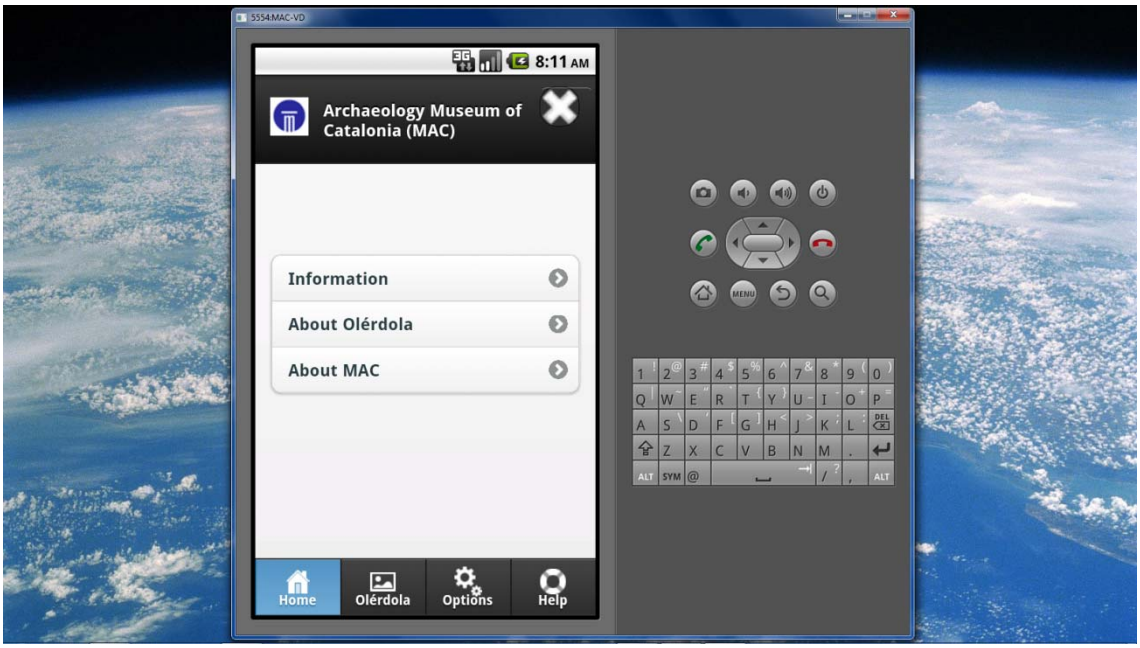

Figura 8.5. Pantalla inicial de la aplicación, donde se ofrece un menú con tres opciones.

Desde la pantalla "*Inici*", se pueden seleccionar tres menús de segundo nivel. En la figura 8.6 se muestra la pantalla "*Informació*", al final de la cual se muestra un mapa de *Google Maps* donde el usuario puede encontrar la ubicación de la *Seu d'Olèrdola*, como se muestra en la figura 8.7.

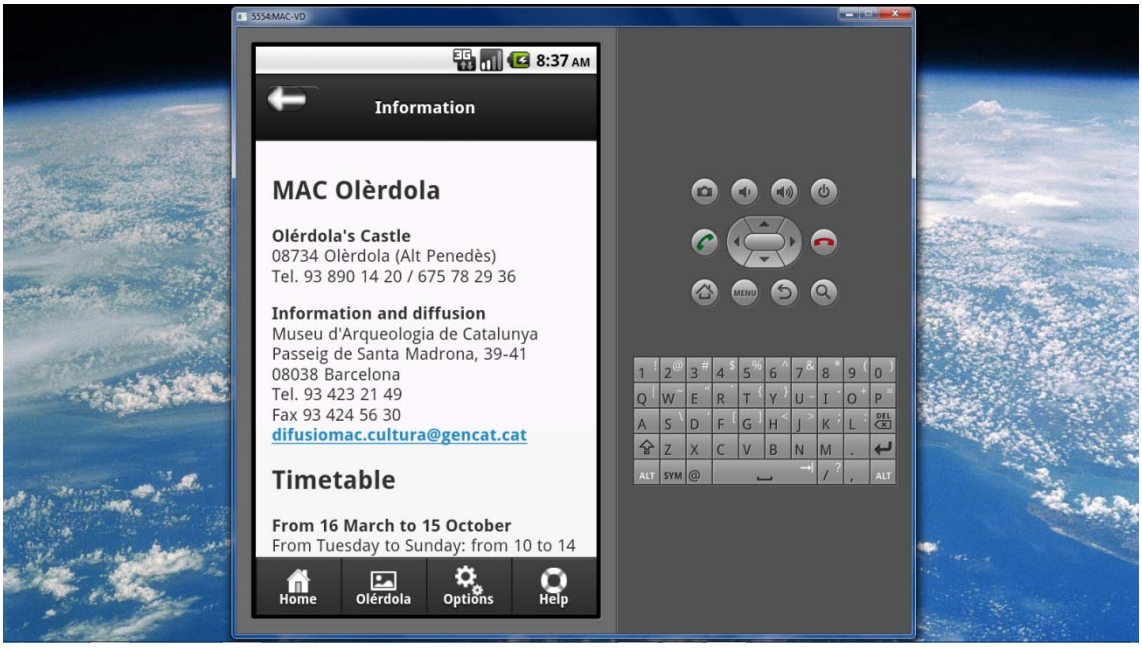

Figura 8.6. Vista de la parte superior de la pantalla "*Informació*" .

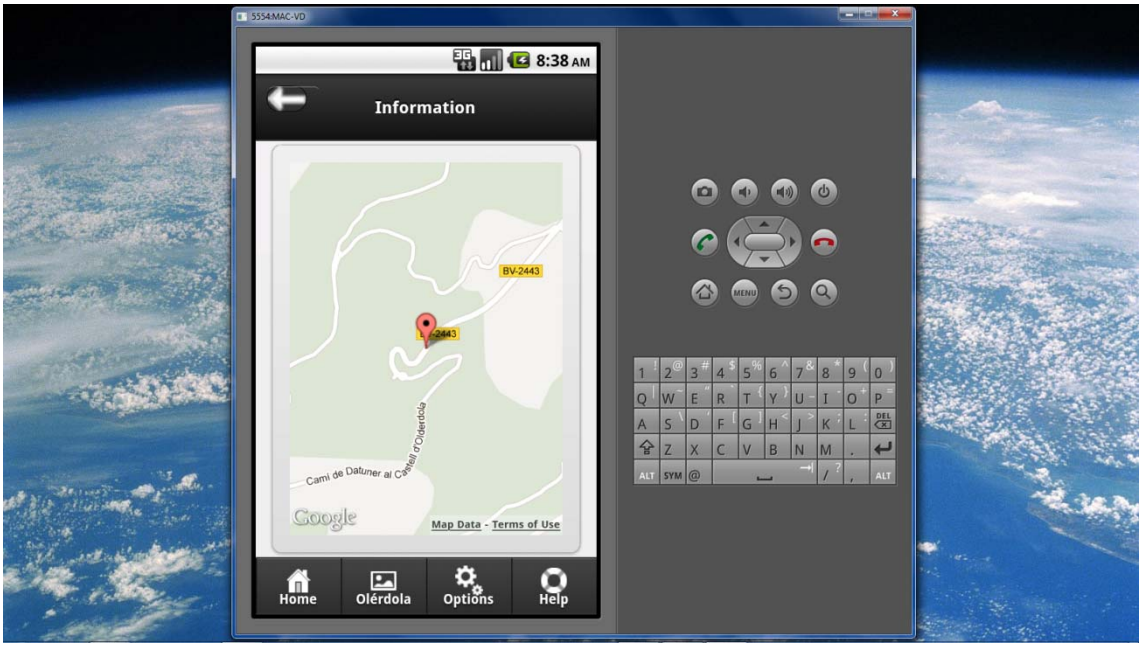

Figura 8.7. Vista de la parte inferior de la pantalla "*Informació*", donde se muestra el mapa con la localización del museo .

La figura 8.8 corresponde a la pantalla "*Sobre Olèrdola*", la segunda opción del menú de la pantalla "*Inici*". En esta pantalla se muestra información descriptiva sobre la *Seu d'Olèrdola*.

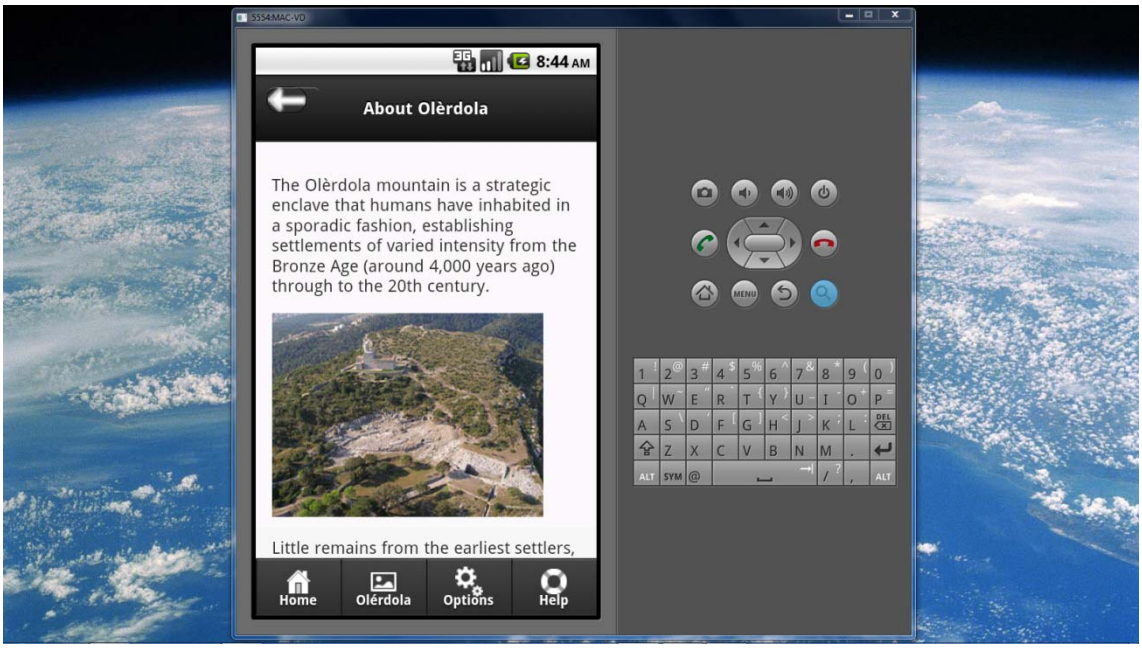

Figura 8.8. Vista de la pantalla "*Sobre Olèrdola*", con alguna información descriptiva sobre el museo .

La figura 8.9 muestra la pantalla "*Sobre el MAC*", la tercera opción del menú de la pantalla "*Inici*". En esta pantalla se muestra información descriptiva sobre el *Museu Arqueològic de Catalunya*.

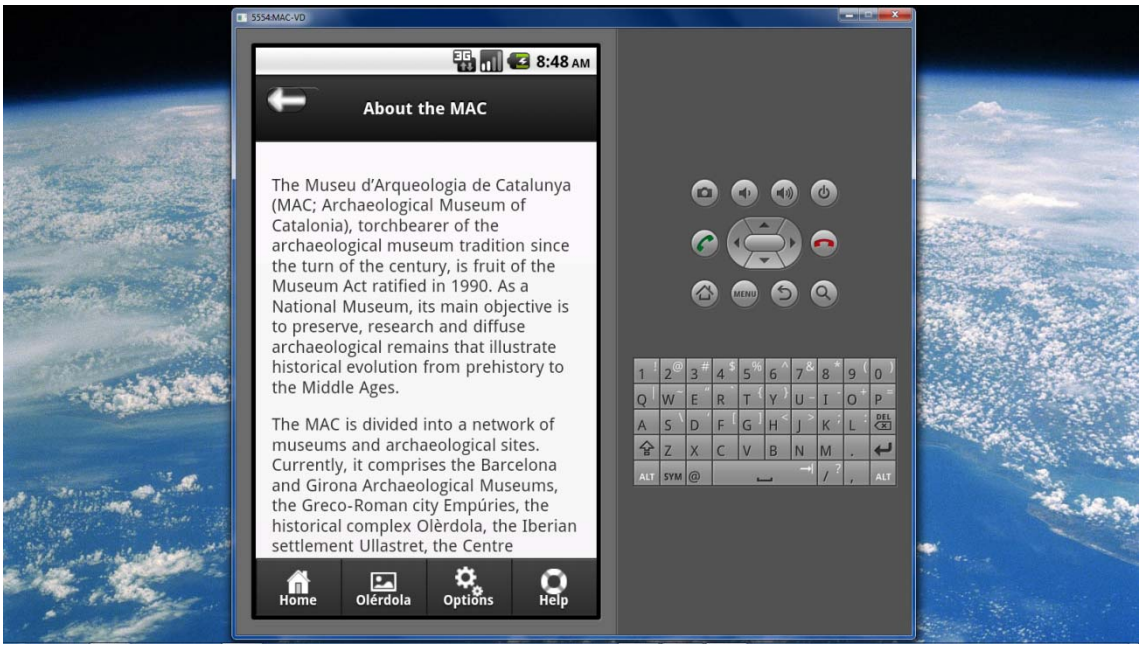

Figura 8.9. Vista de la pantalla "*Sobre el MAC*", donde se ofrece información sobre la institución.

Pulsando el botón "*Olèrdola*" de la barra inferior de botones, mostramos la pantalla "*Olèrdola*", que contiene un menú con tres opciones, como se muestra en la figura 8.10.

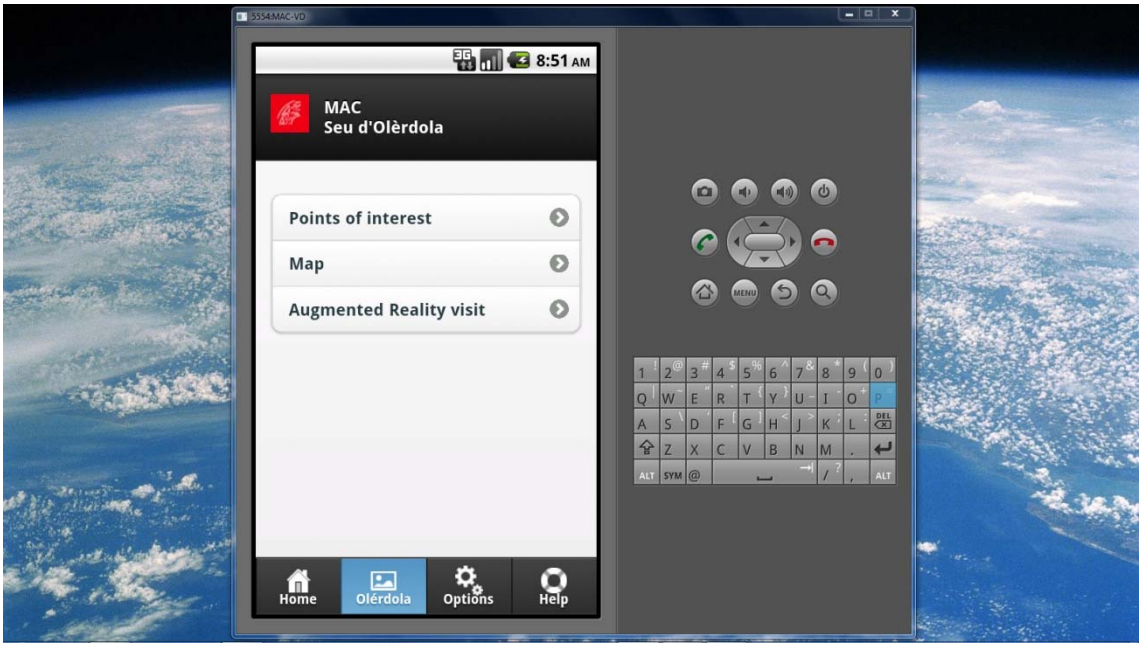

Figura 8.10. Vista de la pantalla "*Olèrdola*", mostrando el menú con las tres opciones.

Seleccionando la primera opción, se muestra la pantalla "*Punts d'Interès*", donde aparece la lista de puntos de interés disponibles, como muestra la figura 8.11.

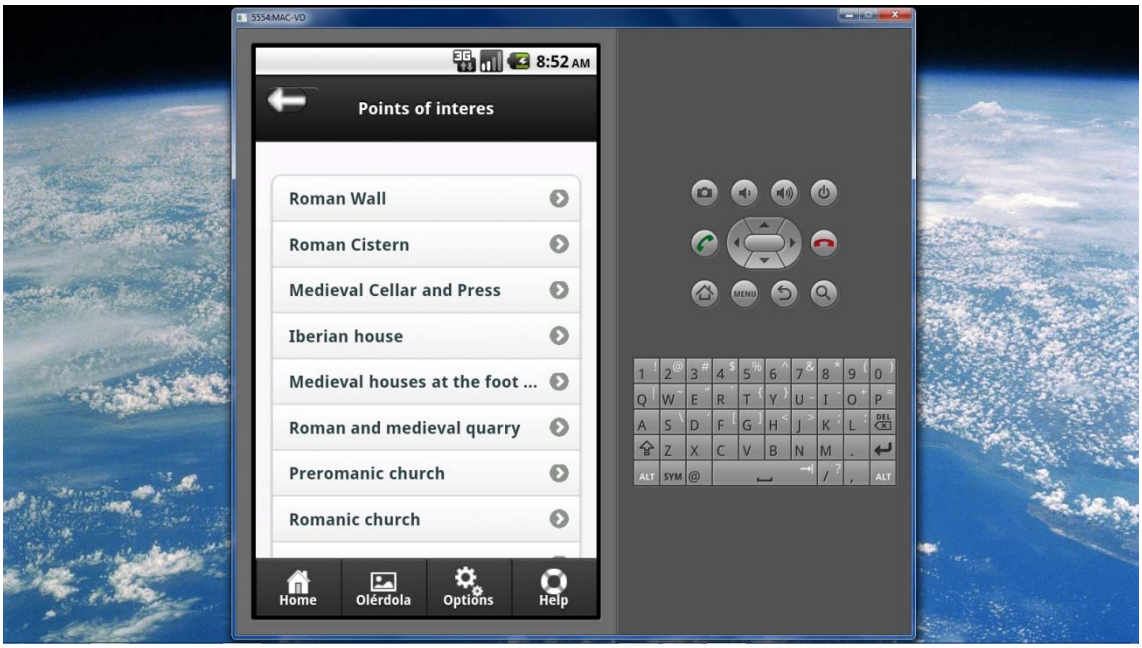

Figura 8.11. Lista de puntos de interés de la pantalla con el mismo nombre.

Seleccionando un punto de interés de la lista, se muestra una pantalla (ver figura 8.12) en la que se ofrece información sobre el punto de interés seleccionado, así como un botón destinado a lanzar una herramienta de realidad aumentada aún por implementar, la cual guiará mostrará al usuario la ubicación del punto de interés.

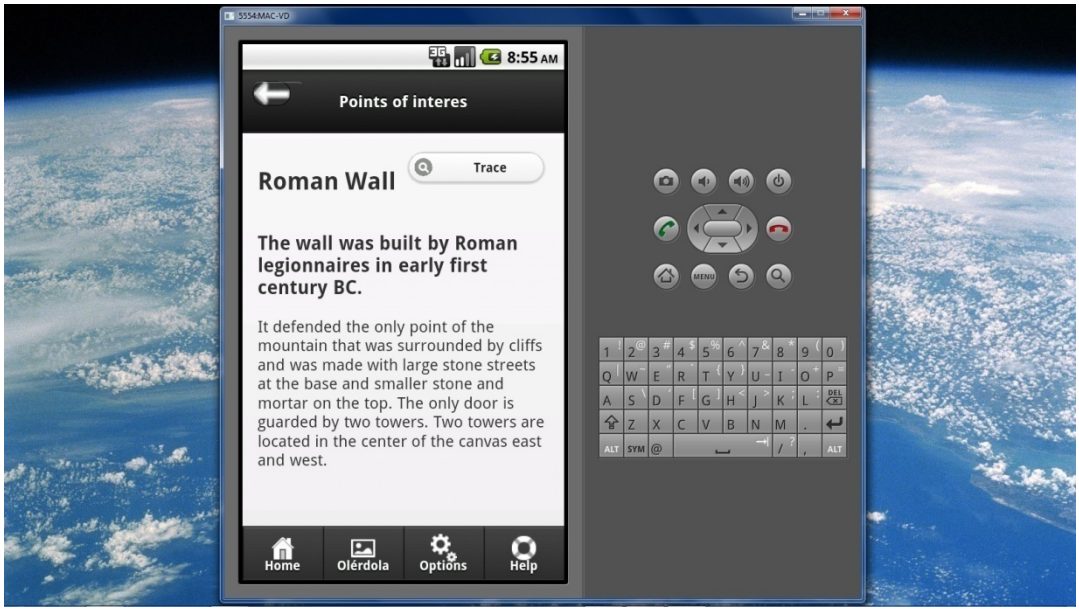

Figura 8.12. Pantalla de punto de interés, mostrando el punto de interés "Muralla romana".

La opción "*Mapa"* de la pantalla "*Olèrdola*" está destinada a mostrar un mapa en el que aparecerá la ubicación actual del dispositivo junto con las ubicaciones de todos los puntos de interés. Esta función aún no está implementada. Actualmente, al seleccionar esta opción, se muestra un mensaje por pantalla como muestra la figura 8.13. Este mensaje no se mostrará una vez se implemente esta herramienta.

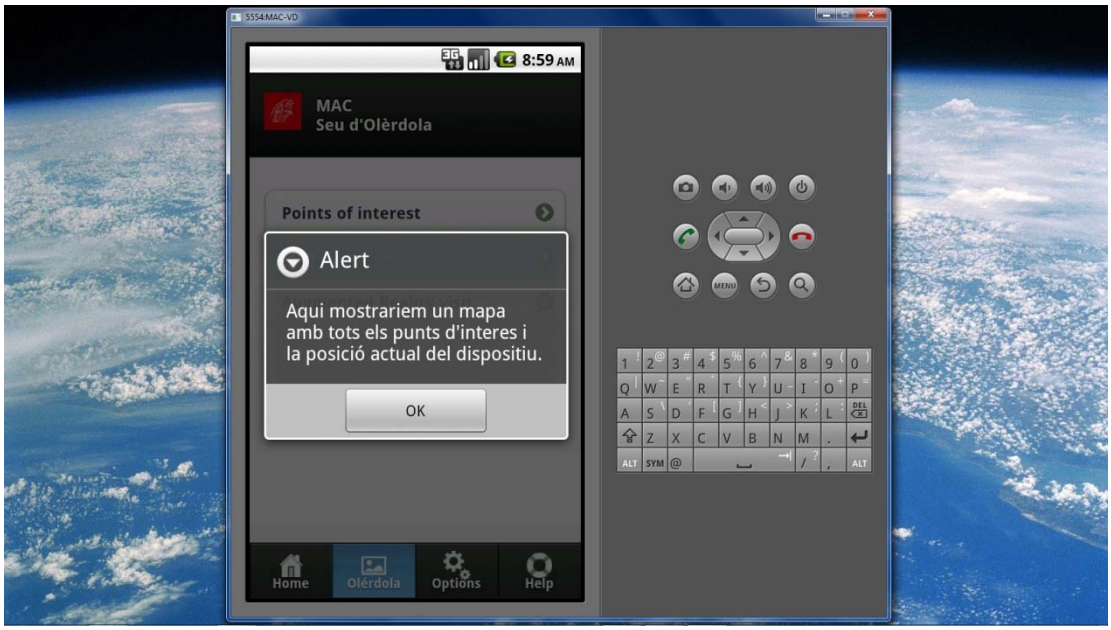

Figura 8.13. Pantalla de "*Mapa*", mostrando un mensaje de aviso por pantalla.

Del mismo modo, la opción "*Visita Realitat Augmentada*" está aún por implementar, por lo que al ser seleccionada, actualmente se muestra temporalmente un mensaje por pantalla, como muestra la figura 8.14.

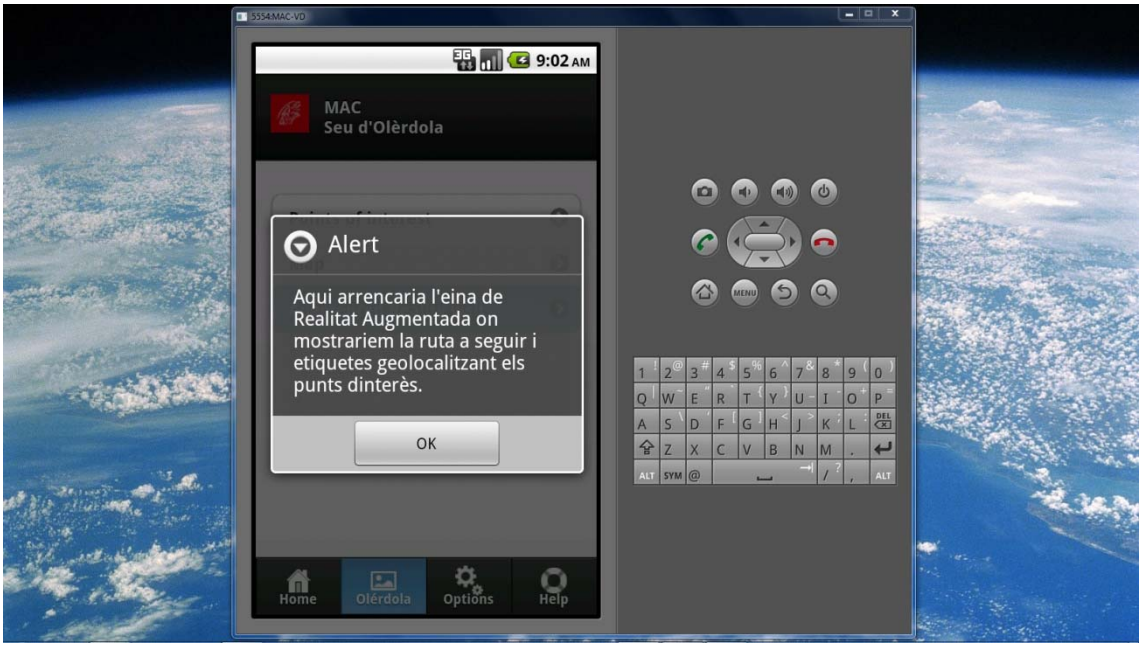

Figura 8.14. Pantalla de "*Mapa*", mostrando un mensaje de aviso por pantalla.

Pulsando el botón "*Opcions*" de la barra inferior de botones, mostramos la pantalla "*Opcions*", la cual ofrece las opciones de idioma y de actualización de la aplicación, como muestra la figura 8.15.

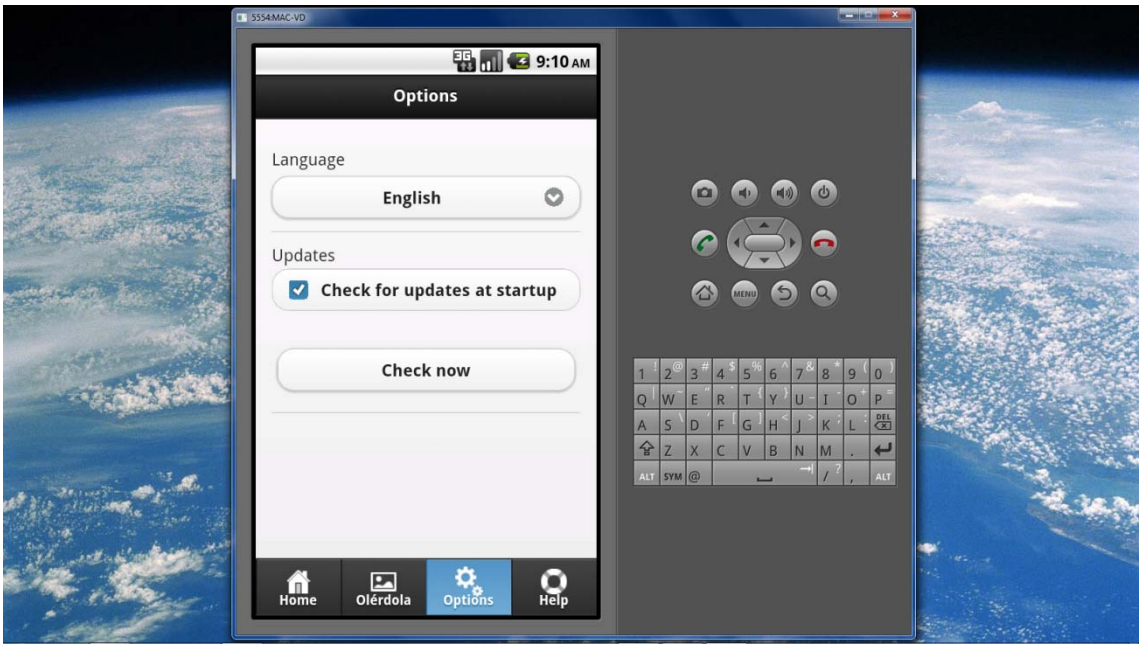

Figura 8.15. Pantalla "*Opcions*", con las opciones disponibles para el usuario.

Pulsando el botón "*Ajuda*" de la barra inferior de botones, mostramos la pantalla "*Ajuda*", que ofrece un menú con tres opciones, como muestra la figura 8.16.

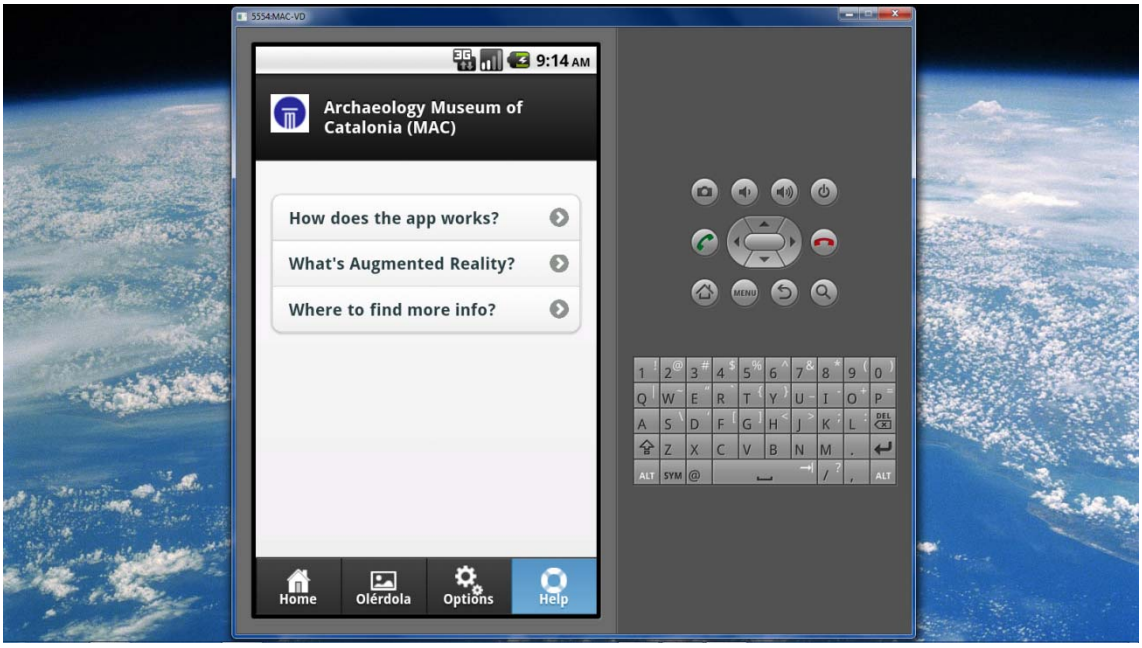

Figura 8.16. Pantalla "*Ajuda*", mostrando el menú con las opciones disponibles.

La pimera opción del menú muestra una pantalla donde se explicará cómo funciona la aplicación. La finalidad de esta opción es explicar el funcionamiento de las herramientas de Realidad Aumentada de la aplicación. Actualmente, se muestra un texto como muestra la figura 8.17.

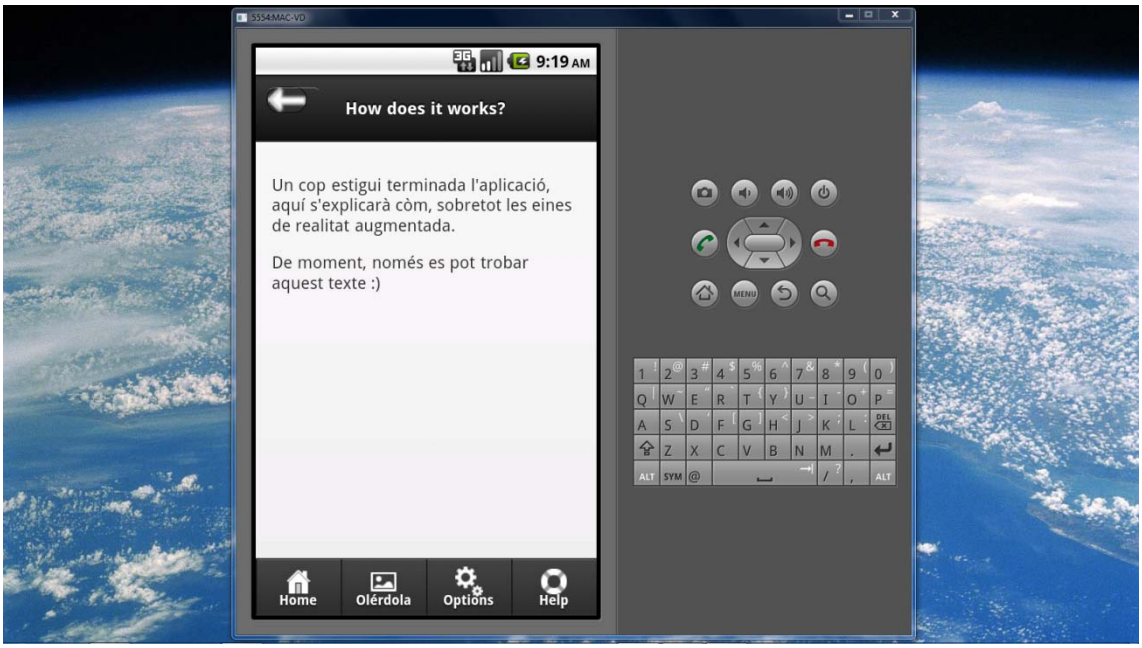

Figura 8.17. Pantalla donde se explicará el funcionamiento de las herramientas de Realidad Aumentada.

La segunda opción del menú de la pantalla "*Ajuda*" muestra una breve explicación sobre el concepto de Realidad Aumentada, como ofrece la figura 8.18.

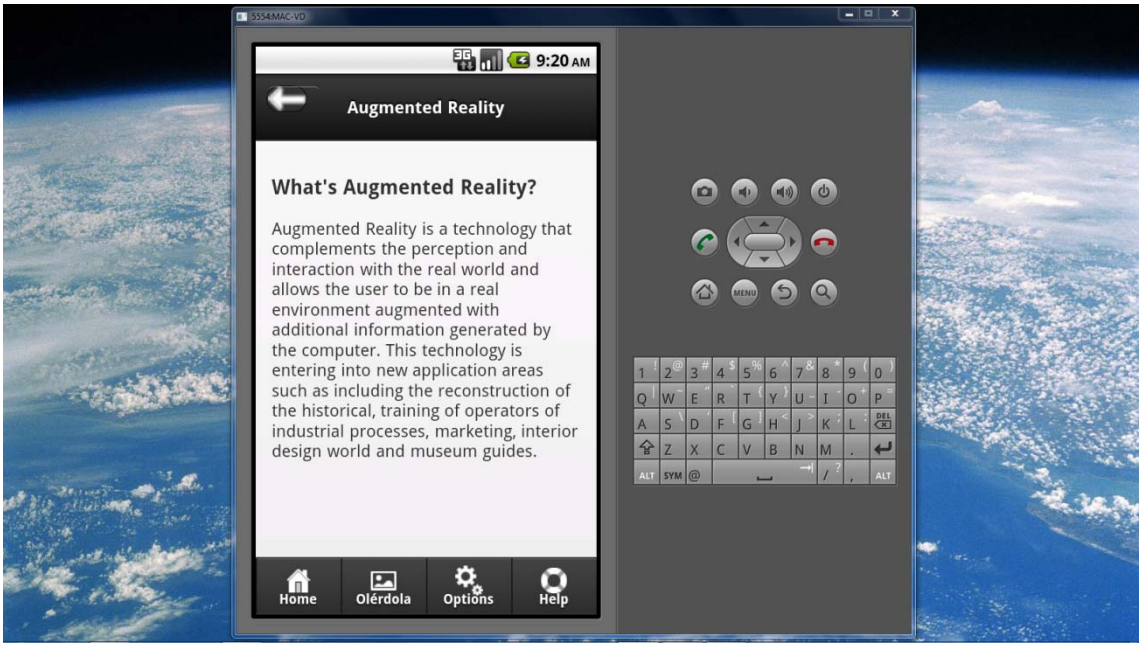

Figura 8.18. Pantalla donde se explicará el concepto de Realidad Aumentada al usuario.

La tercera opción del menú de la pantalla "*Ajuda*" ofrece algunos enlaces donde encontrar más información sobre el *Museu Arqueològic de Catalunya* y la *Seu d'Olèrdola*, como muestra la figura 8.19. Al pulsar un enlace de esta pantalla, se abre el navegador por defecto del dispositivo, mostrando la página web destino del enlace.

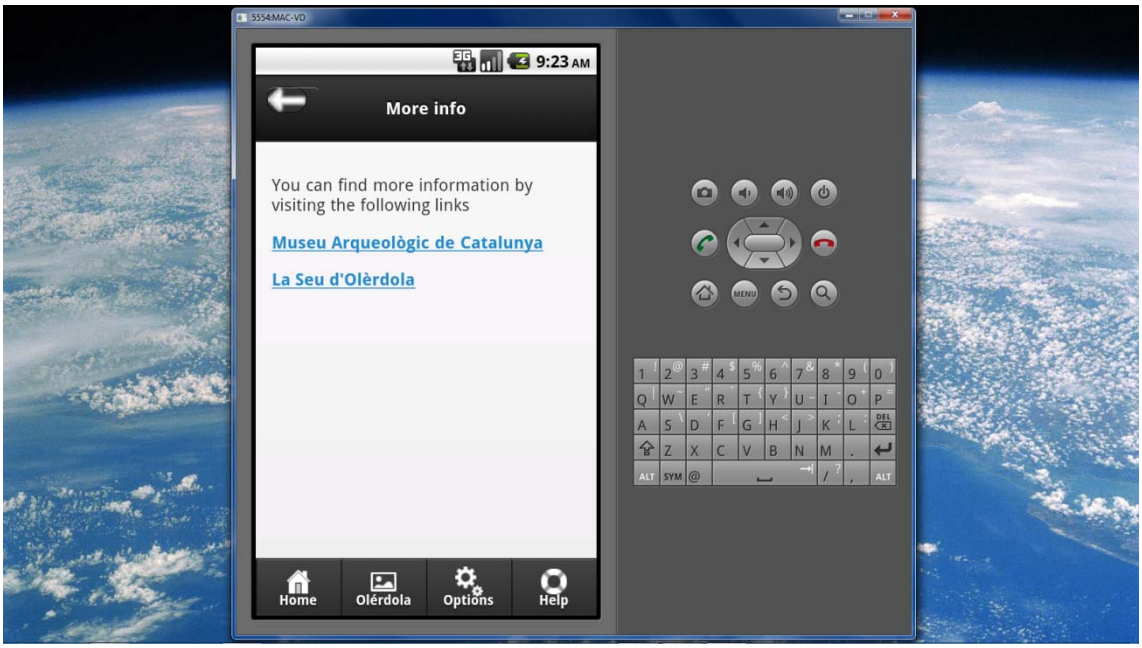

Figura 8.19. Pantalla "*Més informació*", ofreciendo algunos enlaces que extienden la información.

Capítulo 8
Resultados.

En este capítulo se exponen diferentes valoraciones del proyecto. Por un lado, se razona la valoración del coste temporal del proyecto y la valoración del coste financiero. Así mismo, se presentan las conclusiones extraídas tras la finalización del presente proyecto final de carrera

8.1. Carga de trabajo.

El proyecto se ha ido desarrollando desde el mes de noviembre de 2011 hasta mediados de mayo de 2012. Descontando los periodos festivos y días de asuntos personales, el total es aproximadamente de 105 días. La dedicación diaria media ha sido entorno a las 5 horas, lo cual hace un total aproximado de 525 horas.

La elaboración de la memoria comienza a inicios del mes de abril de 2012, y finaliza aproximadamente a inicios de junio. El tiempo dedicado al desarrollo de la memoria es aproximadamente de unas 160 horas (8 semanas * 20 horas semanales).

Concepto	**Horas**
Proyecto	525
Memoria	160
TOTAL	790

8.2. Coste del proyecto.

La valoración económica del proyecto se ha basado en el tiempo empleado por el personal que ha hecho posible el desarrollo del proyecto hasta la fecha. Por tanto, el coste de cada una de las partes implicadas se presenta de forma desglosada.

Coste del desarrollo

El coste que ha supuesto el desarrollo del proyecto ha sido medido con las horas de trabajo dedicado según las tareas realizadas. Las horas de trabajo se han calculado de forma aproximada. Es de destacar en este punto que mis conocimientos en los lenguajes de programación empleados eran de un nivel básico. No tenía conocimientos previos de desarrollo de aplicaciones para *Android*, ni había desarrollado anteriormente una página *web* tan compleja como la desarrollada para este proyecto. Esto ha supuesto que las horas empleadas, tanto en el desarrollo del código como en la resolución de problemas encontrados, han sido bastante más de las que se podría esperar en un proyecto de estas características. El precio/hora se ha basado en un precio estándar de consultorías.

Tarea	**Recurso**	**Precio/Hora**	**Horas**	**Total**
Analista de requisitos	Analista	48 €	90	4320 €
Especificación	Analista	48 €	40	1920 €
Implementación	Programador	35 €	375	13125 €
Pruebas	Programador	35 €	20	700 €
			TOTAL	20065 €

Coste del *software*

La mayor parte del *software* empleado es gratuito, como *PhoneGap* o el entorno *Eclipse*. Aunque otra parte del *software* tiene un coste. A continuación se detallan los costes del *software* de pago empleado en el desarrollo del proyecto.

Software	**Precio**
Adobe Dreamweaver CS5	270 €

Coste del *hardware*

El único *hardware* adquirido exclusivamente para el desarrollo del proyecto fue el dispositivo móvil *Android* empleado para realizar las pruebas en tiempo de implementación del proyecto.

Hardware	**Precio**
Samsung Galaxy Young	125 €

Coste total

El coste total del proyecto, sumando los costes de desarrollo, *software* y *hardware*.

Concepto	Coste
Desarrollo	20065 €
Software	270 €
Hardware	125 €
TOTAL	20460 €

8.3. Conclusiones.

8.3.1. Objetivos conseguidos.

La interfaz gráfica ha sido terminada con éxito, aunque aún existen algunos aspectos que sería deseable mejorar y pulir. Se puede decir que se han completado todos los requerimientos establecidos, obviando los requerimientos referentes a la parte correspondiente a Realidad Aumentada. La aplicación realiza actualizaciones a través de la conexión a Internet del dispositivo. La estrategia de actualización finalmente implementada permite la actualización de contenidos sin necesidad de modificar el código fuente de la aplicación.

Por otra parte, la aplicación se ofrece en los tres idiomas que se especificaron, esto es, catalán, castellano e inglés. La aplicación detecta correctamente el idioma en el que está configurado el dispositivo y aplica automáticamente dicho idioma sin que el usuario deba seleccionarlo activamente en el menú de opciones. Si el idioma detectado en el dispositivo no es ninguno de los tres idiomas en los que se ofrece la aplicación, se establece como idioma por defecto el inglés.

Se han integrado en la aplicación los contenidos suministrados por el cliente final, respecto a los puntos de interés de las instalaciones. Los contenidos, que fueron suministrados en catalán, se tradujeron al resto de idiomas de la aplicación. El resto de contenidos han sido recogidos de la página *web* oficial del *Museu Arqueològic de Catalunya*.

En la lista de puntos de interés, la aplicación describe los puntos de interés y ofrece un botón de localización disponible para implementar su funcionalidad en la fase de desarrollo de Realidad Aumentada.

La fluidez de respuesta de la interfaz de usuario a las acciones del dispositivo es aceptable. Por otra parte, sería deseable que las transiciones entre las pantallas, y al desplazar el texto verticalmente, fuesen algo más fluidas.

Aunque se ha implementado la interfaz de usuario para poder ser integrada en una aplicación para la plataforma *iOS*, aún no se han realizado pruebas reales sobre esta plataforma.

Problemas detectados

Se ha detectado un problema en la pantalla en la que se muestra la lista de puntos de interés. La pantalla muestra, como el resto de pantallas, una barra horizontal en la parte superior con el título de la pantalla y un botón para volver a la pantalla anterior. Esta barra debería permanecer inmóvil mientras se desplaza verticalmente la lista de puntos de interés. En cambio, al desplazarnos hacia abajo en la lista de puntos de interés, la barra se mueve hacia arriba solidariamente junto con la lista de puntos de interés, hasta desaparecer de la pantalla.

Otro problema detectado sucede en el proceso de actualización. Se ha detectado que, en ocasiones, el proceso de actualización al finalizar no renueva el parámetro de fecha de actualización en el dispositivo con la fecha de actualización instalada. Esto provoca que tras la actualización, si se realiza una nueva búsqueda de actualizaciones, se detecta la actualización ya instalado como una nueva actualización.

Posibles mejoras

La principal mejora que se podría implementar afectaría a la estrategia de actualización de la aplicación. Como se ha explicado en el apartado al respecto, el proceso de instalación de actualización pasa por descargar del servidor remoto de actualizaciones la página *web* que conforma la interfaz de usuario de la aplicación. Actualmente se descargan en este proceso todos y cada uno de los archivos que conforman la página *web*. La modificación consistiría en descargar únicamente los archivos que han sido modificados. Otra versión de esta misma modificación podría ser

restringir los archivos que puede modificar el cliente final, de manera que únicamente pueda modificar los archivos en formato *JSON* referentes a los textos de idiomas.

8.3.2. Opinión personal.

El presente proyecto representa mi primer contacto con el diseño de aplicaciones a cierto nivel. La titulación que estoy cursando es la de Ingeniería de Sistemas Informáticos, la cual está orientada más hacia la vertiente *hardware* que hacia la vertiente *software* de la informática. Por ello, la fase de especificación y diseño ha sido algo complicada para mí, debido a que nunca había cursado asignatura alguna relacionada con la arquitectura del *software* en general (especificación formal de proyectos, patrones de diseño, modelos arquitectónicos, etcétera).

Por otra parte, el proyecto ha representado todo un reto para mí. No solamente se ha tratado de aplicar gran parte de los conocimientos adquiridos a lo largo de la carrera, sino que ha supuesto el aprendizaje desde cero de una serie de tecnologías totalmente desconocidas para mí, además de otras tareas de investigación, experimentación, etcétera. Nunca antes había desarrollado una aplicación para *Android*, ni conocía tampoco *JQuery*. Así mismo, mis conocimientos de los lenguajes *web* como *HTML*, *CSS* y *Javascript* no eran demasiado extensos, sino básicamente nociones. Todo esto sumado ha supuesto un sobreesfuerzo, añadido a algunas situaciones personales que particularmente he tenido que afrontar en los últimos meses.

A todo ello me he enfrentado con muchas ganas e ilusión. El hecho de tratarse de un proyecto real ha supuesto, por un lado, una presión añadida aunque, por otro lado, ha sido un aliciente y una motivación extra.

El momento más complicado del proyecto fue, sin duda, la fase previa a la división del proyecto en dos etapas. Los días pasaban y cada vez se hacía más evidente que el alcance del proyecto real sobrepasaba en mucho los esfuerzos y requerimientos de un proyecto final de carrera de una titulación de ingeniería técnica como la que tratamos. Ello provocaba en mí cierta angustia de ver que el proyecto se iba a alargar mucho más de lo previsto inicialmente por mí. Finalmente, la división del proyecto en dos etapas fue clave para poder afrontar con garantías el presente proyecto final de carrera.

Otra parte del proyecto que también me resultó especialmente costosa fue el desarrollo de la presente memoria. Varios han sido los motivos que han confluido haciendo algo más complicada la tarea de documentación de la memoria. El primero de ellos ha sido el hecho de que en los proyectos en los que he participado en los cinco

años que llevo en el mercado laboral, ninguno ha requerido por mi parte ninguna tarea de arquitectura del *software*, ni tareas de documentación tan laboriosas como esta. Este es sin duda el proyecto que más esfuerzo de documentación ha requerido por mi parte, cosa a la que no estaba habituado, y que en futuros proyectos cuidaré con más detalle en algunos aspectos, como por ejemplo la bibliografía.

Por supuesto, el momento más interesante del proyecto ha sido la fase de implementación, la cual siempre es muy gratificante, debido a que es la fase en la que finalmente toma forma todo aquello en lo que se ha trabajado en todas las fases previas.

En líneas generales, estoy bastante satisfecho con los resultados obtenidos. Sobre todo porque el trabajar con técnicas de ingeniería del *software* proporciona una nueva visión en el desarrollo del *software*, con herramientas tan extremadamente potentes como el *UML*, con las cuales nunca antes había tenido la ocasión de experimentar. Aunque por otra parte, me queda la sensación de que si volviera a realizar el proyecto de nuevo, seguramente cambiaría la forma de desarrollarlo, debido a los conocimientos adquiridos durante todo el proyecto.

Bibliografía

[1] Definiciones. Real Academia Española:

http://www.rae.es

[2] Definiciones. Wikipedia:

http://es.wikipedia.org

[3] UML. "Aprendiendo UML en 24 horas", Joseph Schmuller, Ed. Prentice Hall.

http://www.pearson.es

[4] PhoneGap. "Beginning PhoneGap", Thomas Myer, Ed. Wrox.

http://www.wrox.com

[5] HTML, CSS, Javascript. "Building Android apps with HTML, CSS and JavaScript", 2ª edición, Jonathan Stark with Brian Jepson, Ed. O'Reilly.

http://www.oreilly.com

[6] jQuery Mobile. "jQuery Mobile", Jon Reid, Ed. O'Reilly.

http://www.oreilly.com