Universidad Zaragoza

Proyecto Fin de Carrera

Framework basado en Realidad Aumentada en dispositivos móviles iOS y su aplicación en el ámbito turístico

Autor

Adrián Grasa Arrabal

Director y ponente

Director: Juan López de Larrínzar Galdámez

Ponente: Pedro R. Muro Medrano

Escuela de Ingeniería y Arquitectura

2012/2013

Quiero agradecer a mis profesores, compañeros, amigos y familia, la ayuda, los consejos y la comprensión que me han demostrado y que me han permitido realizar este proyecto fin de carrera lo mejor posible.

Un agradecimiento especial a:

Al IAAA y GeoSpatiumLab, por la oportunidad que me ha brindado de realizar este proyecto y poder disponer de todo el soporte necesario para el mismo, así como también darme un hueco dentro del equipo, que ha hecho posible que realizara el trabajo en un ambiente idóneo para ello.

A Juan, por su paciencia y dedicación y, por todos los consejos que me ha dado; a Javier y a Pedro que han estado a mi lado, involucrados, también, en mi proyecto y me han dado el soporte técnico necesario.

A todos los profesores del IAAA como son Pedro Muro, F. Javier Zarazaga y F. Javier López, que con sus consejos y apoyo constante me han hecho mejor como profesional y como persona, gracias.

A mi familia y amigos, que me han apoyado incondicionalmente y me han enseñado a creer en mí mismo y a enfrentarme con optimismo a cualquier dificultad.

Gracias, sin ellos no podría haber llegado hoy aquí.

Framework basado en Realidad Aumentada en dispositivos móviles iOS y su aplicación en el ámbito turístico

Resumen

La realidad aumentada (AR) es un concepto en auge en los últimos tiempos. Consiste en la definición de la vista de un entorno físico del mundo real en combinación con elementos virtuales para la creación de una realidad mixta en tiempo real. De esta manera, con la ayuda de la tecnología, la información que rodea al usuario se convierte en digital e interactiva, siendo ésta la principal diferencia entre la realidad aumentada y la realidad virtual. En la segunda modalidad, el mundo real se ve modificado. Sin embargo, en la realidad aumentada se añade una mayor riqueza de información a la realidad sin alterarla.

El objetivo principal de este proyecto ha sido la creación de un framework especializado en AR y que permita el desarrollo de aplicaciones móviles, para dispositivos con sistema operativo iOS, que requieran de esta funcionalidad con independencia de la temática y la tipología de los datos, únicamente siendo necesario que se encuentren georreferenciados. A partir de esto y una vez desarrollado, se procedió a la creación de un caso de uso muy generalista orientado a un entorno turístico. De este modo, el usuario final puede llevar a cabo visitas turísticas en las ciudades que desee sin necesidad de guías de viajes específicas de cada lugar, ya que dispone de servicios que proporcionan información de todos los lugares del mundo. Todo esto, se puede realizar de una forma distinta, innovadora y entretenida gracias a las capacidades que hoy en día proporciona la AR y la tecnología de los smartphones que aprovechan los recursos digitales disponibles a través de internet.

Otro aspecto innovador en este proyecto, como se puede ver en el análisis del estado del arte, es la inexistencia de algún framework sobre iOS de código abierto con todo el potencial y funcionalidad que eran necesarios para realizar el caso de uso desarrollado. Por ello, se trata de un framework de gran utilidad, cuyo diseño garantiza su reutilización en otros contextos temáticos para el desarrollo de nuevas aplicaciones o incluso, para su posible liberación como código abierto.

Por otro lado, como se pudo ver en la propuesta del proyecto, existía un objetivo doble, es decir, además del framework se pretendía la creación de una aplicación que debía ser un caso de uso lo más abierto y flexible posible, en el cual se refleje todo el potencial del framework implementado.

Los puntos fuertes del framework, los cuales no han sido encontrados combinados en ninguno de los frameworks de iOS de AR analizados, y por lo que se decidió que era necesario su creación, son varios. En primer lugar, la posibilidad de funcionamiento tanto en línea (con uso de internet), como fuera de ella gracias a la persistencia de datos implementada. La facilidad de integración con otros proyectos abstrayendo a otros desarrolladores de la complejidad propia de esta tecnología, todo ello con un rendimiento y consumo aceptables dado que la visualización es en tiempo real. Por último, la posibilidad de añadir varias fuentes de información distintas simultáneamente, reduciendo así la dependencia con servicios concretos, además de ofrecer una categorización común de todos los puntos de interés, con independencia de la fuente de datos.

Índice general

ANEXOS

Memoria

1. Introducción

En este capítulo se va a explicar el ámbito en el que se ha desarrollado el proyecto, tanto en el contexto profesional como en el tecnológico. Además se va a exponer la motivación del PFC junto con la estructura que va a seguir este documento.

1.1 Contexto profesional

1.1.1 *GeoSpatiumLab* (GSL)

GeoSpatiumLab[1] es una empresa de ingeniería informática que se dedica a la creación y puesta en servicio de tecnología para la gestión y difusión de información georreferenciada (también denominados generalmente datos geográficos y ahora, más comúnmente, datos espaciales) sobre un nuevo paradigma que se ha convenido en llamar Infraestructuras de Datos Espaciales (IDEs).

Nace como una *Spin-Off* de la Universidad de Zaragoza tomando como punto de partida la experiencia de más de 10 años del Grupo de Sistemas de Información Avanzados (IAAA)[2] de la Universidad de Zaragoza y perteneciente al i3A[3]. El trabajo de investigación del grupo trata de cubrir desde aspectos básicos de ingeniería de servicios hasta aspectos metodológicos de la puesta en funcionamiento de sistemas industriales que explotan la información geográfica.

Dentro de todo esto, *GeoSpatiumLab* posee como una de sus líneas de negocio, el desarrollo de soluciones que facilitan el tratamiento y la visualización de datos de base geográfica tanto en entornos web como móvil (desarrollos nativos para Android e iOS) en diversas temáticas, siendo una de ellas el sector turístico y dentro de la cual se encuadra este proyecto.

1.2 Contexto tecnológico

1.2.1 Realidad aumentada (AR)

Este término 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 datos al mundo real.

Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos o con información georreferenciada), 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 puede ser almacenada y recuperada como una capa de información en la parte superior de la visión del mundo real.

1. www.geoslab.com
2. www.iaaa.cps.unizar.es
3. www.i3a.unizar.es

1.2.2 **iOS** [1-6]

Se trata de un sistema operativo móvil de la empresa Apple Inc.[4] Originalmente desarrollado para el iPhone (iPhone OS), 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.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. La respuesta a las órdenes del usuario es inmediata, proveyendo de una interfaz fluida. 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 donde las capas más altas contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas más bajas controlan los servicios básicos. La versión actual del sistema operativo es la iOS 6.1.3, habiéndose presentado la versión 7, pero todavía por distribuir oficialmente.

A continuación se explican dichas capas (fig. 1):

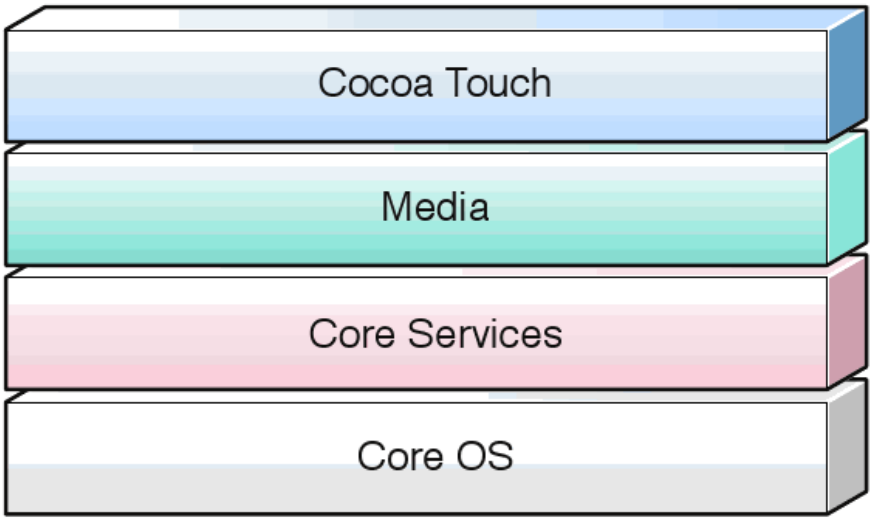


Figura 1. *Arquitectura de capas iOS*

Cocoa Touch

Es la capa más importante para el desarrollo de aplicaciones iOS. Posee un conjunto de frameworks que proporciona el API de Cocoa para desarrollar aplicaciones. Se podría decir que Cocoa Touch proviene de Cocoa, la API ya existente en la plataforma Mac.

Esta capa está formada por dos frameworks fundamentales:

- UIKit: contiene todas las clases que se necesitan para el desarrollo de una interfaz de usuario.
- Foundation Framework: define las clases básicas, acceso y manejo de objetos, servicios del sistema operativo.

4. www.apple.com

Media

Provee los servicios de gráficos y multimedia a la capa superior y consiste en los siguientes componentes:

- Core Audio
- Open GL
- Audio Mixing
- Audio Recording
- Video Playback
- JPG, PNG, TIFF
- PDF
- Quartz
- Core Animation
- OpenGL ES

Core Services

Provee una abstracción sobre los servicios que proporciona el sistema y consiste en los siguientes componentes:

- Collections
- Address Book
- Networking
- File Access
- SQLite
- Core Location
- Net Services
- Threading
- Preferences
- URL Utilities

Core OS

Contiene las características de bajo nivel: ficheros del sistema, manejo de memoria, seguridad y drivers del dispositivo. Esta capa interactúa directamente con el hardware.

En un primer momento este PFC se desarrolló en iOS 4, pero en todo momento se mantuvo la compatibilidad con las versiones posteriores, por lo que tanto la aplicación como los frameworks desarrollados funcionan en esta versión del sistema operativo y en las posteriores.

1.2.3 Objective-C

Consiste en un lenguaje de programación orientado a objetos creado como un superconjunto de C para que implementase un modelo de objetos parecido al de Smalltalk. Originalmente fue creado por Brad Cox y la corporación StepStone en 1980. En 1988 fue adoptado como lenguaje

de programación de NEXTSTEP y en 1992 fue liberado bajo licencia GPL para el compilador GCC. Actualmente se usa como lenguaje principal de programación en Mac OS X, iOS y GNUstep.

Este es el lenguaje que se ha utilizado en este PFC junto con Xcode, el entorno de desarrollo integrado (IDE, en sus siglas en inglés) de Apple Inc. y se suministra gratuitamente junto con Mac OS X. Xcode trabaja conjuntamente con Interface Builder, una herencia de NeXT, una herramienta gráfica para la creación de interfaces de usuario, añadiendo las storyboards en las últimas versiones.

Xcode incluye la colección de compiladores del proyecto GNU (GCC) y puede compilar código C, C++, Objective-C, Objective-C++, Java y AppleScript mediante una amplia gama de modelos de programación, incluyendo, pero no limitado a Cocoa, Carbón y Java. Otras compañías han añadido soporte para GNU Pascal, Free Pascal, Ada y Perl.

La versión de XCode utilizada en el desarrollo de este PFC fue la 3.2.6, última versión gratuita de este entorno que permite programar para el iOS 4.3, y para adaptarlo a las versiones posteriores se utilizaron las versiones actuales del IDE.

1.2.4 Servicio web

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C[5] son los comités responsables de la arquitectura y reglamentación de los servicios web.

En la realización del PFC se han utilizado básicamente dos servicios web de geolocalización de información distintos, Geonames[6], CloudMade[7] y un servicio propio de GSL. El primero aporta información procedente de artículos de Wikipedia[8] georreferenciados entre otros; y el segundo datos procedentes de OpenStreetMaps[9] [22]. Con esto se consigue tener un ámbito genérico y no solo local.

1.2.5 *JSON*

Se trata del acrónimo de *JavaScript Object Notation*. Es un formato ligero para el intercambio de datos. *JSON* es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de *XML*.

La simplicidad de *JSON* ha dado lugar a la generalización de su uso, especialmente como alternativa a *XML* en *AJAX*. Una de las supuestas ventajas de *JSON* sobre *XML* como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de *JSON*.

En el contexto del PFC se ha utilizado, ya que los servicios web utilizan este formato de intercambio de datos junto con el que se explica a continuación, *GeoJSON*.

5. www.w3.com
6. www.geonames.org
7. www.cloudmade.com
8. www.wikipedia.org/wiki/Wikipedia:Portada
9. www.openstreetmap.org

1.2.6 *GeoJSON*

Se trata de un formato abierto para codificar una gran variedad de estructuras de datos geográficos. Su estructura se basa en *JSON*. De hecho, cada estructura de datos *GeoJSON* es también un objeto *JSON*, y por lo tanto las herramientas de *JSON* también pueden ser utilizadas para el procesamiento de datos en formato *GeoJSON*.

1.2.7 SQLite

Es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una biblioteca escrita en C. SQLite[10] es un proyecto de dominio público.

El hecho de que se trate de un sistema tan liviano ha dado lugar a que sea un sistema muy utilizado en el ámbito de las tecnologías móviles y el modelo de persistencia de datos en el PFC se base en este sistema.

1.3 Objetivos y motivación del proyecto

Actualmente el uso de tecnologías móviles se ha extendido por todo el mundo de una manera global, dando lugar a unos usuarios cada vez más exigentes y con conocimientos especializados en el ámbito de estas tecnologías. Por otro lado, cada día existe una mayor información disponible al alcance de cualquier individuo; el problema de toda esta gran cantidad de información es el de poder suministrarla al usuario final de una manera clara y comprensible.

Así pues, *GeoSpatiumLab* posee como una de sus líneas de trabajo el desarrollo de soluciones que facilitan el tratamiento y la visualización de datos de base geográfica tanto en entornos web como móvil en diversas temáticas, entre ellas el sector turístico. Por lo que la principal motivación del proyecto ha sido abordar el problema de mostrar toda esta información, de manera clara y comprensible, por medio de la realidad aumentada, concretamente en entornos iOS, ya que se trata de una tecnología en pleno auge y con un futuro prometedor.

De este modo, el objetivo principal de este proyecto ha sido el de abordar el desarrollo de un framework para dispositivos con sistema operativo iOS, basado en los conceptos de realidad aumentada que encapsule la funcionalidad de base para la gestión de esta tecnología, siendo parametrizable y abstraíble y, por tanto, aplicable en aplicaciones futuras que requieran de esta forma de mostrar la información, algo que hoy en día no está disponible como tal en código abierto y de libre acceso en iOS.

Con el fin de mostrar todo el potencial de este framework se ha desarrollado una aplicación de ámbito turístico, que accede a fuentes de información dispares y muestra la información de tres modos distintos: el modo mapa, se presentan en este soporte los elementos geolocalizados que se encuentren próximos al usuario; el modo lista, que presenta en forma de lista los puntos de interés disponibles, y el modo cámara o modo explorador, que hace uso del framework de realidad aumentada desarrollado y en el que se muestra la información en

10. www.sqlite.org

tiempo real de los puntos que rodean y están en la misma dirección que el usuario en función de su posición y su orientación a través de la cámara. En todos estos modos se podrá extraer la información disponible de los puntos de interés al seleccionarlos.

Así pues, a lo largo de la realización de este proyecto se han abordado diferentes actividades, que partieron de la investigación y la documentación de la tecnología de la realidad aumentada y de toda la física y algoritmia que hay detrás de ésta, así como del entorno iOS y de las posibilidades gráficas que ofrecía, las cuales eran nuevas para el proyectando, o el análisis de otros proyectos en el ámbito de la tecnología móvil. Fue entonces cuando se comenzó el análisis y el diseño del proyecto que se iba a abordar, el cual siguió las fases típicas de un proyecto software pero de un modo iterativo e incremental más ágil, explicado de una manera más amplia en el primer anexo.

1.4 Estructura del documento

Este documento ha sido redactado de tal modo que se adapta a la estructura recomendada por el centro. En dicho texto se pueden encontrar tres partes bien diferenciadas. La primera parte es la memoria como tal de proyecto, en la que se muestra de un modo general el trabajo llevado a cabo a lo largo de éste. A continuación hay una serie de anexos, en los que se explica más detalladamente el trabajo realizado y el resultado final del mismo; y, por último, las referencias bibliográficas junto con la lista de acrónimos y el índice de figuras. Así pues, la estructura es la siguiente:

– Parte I: Memoria:
  - Introducción
  - Descripción del trabajo realizado
  - Conclusiones

– Parte II: Anexos:
  - Anexo A: Gestión del proyecto
  - Anexo B: Análisis de los requisitos
  - Anexo C: Análisis del estado del arte
  - Anexo D: Análisis de Mixare
  - Anexo E: Física realidad aumentada
  - Anexo F: Diseño y documentación framework realidad aumentada
  - Anexo G: Diseño y documentación framework de gestión de datos
  - Anexo H: Diseño y documentación framework de mapas
  - Anexo I: Diseño aplicación de realidad aumentada
  - Anexo J: Análisis de prestaciones de la gestión de los datos en el módulo de realidad aumentada
  - Anexo K: Pruebas de funcionalidad e integración

– Parte III:
  - Bibliografía
  - Acrónimos
  - Figuras

2. Trabajo realizado

Este capítulo detalla el trabajo realizado a lo largo de todo el período de desarrollo de este proyecto. Parte de la situación actual o estado del arte de la realidad aumentada para, posteriormente, una vez explicada la política de trabajo que se ha seguido, estudiar el sistema a nivel general y luego componente a componente, según se ha ido desarrollando.

2.1 Estado del arte

Uno de los primeros trabajos que se llevó a cabo en el comienzo de este PFC, junto con el estudio de las tecnologías y los nuevos lenguajes que se iban a utilizar, fue la búsqueda de información relevante acerca de la realidad aumentada y de diversos proyectos vinculados a estas tecnologías. Esto tenía como fin detectar las necesidades y deficiencias que pudieran tener para tratar de incluirlas o corregirlas en este trabajo, así como entender en mayor medida la problemática a la que se estaba enfrentando.

2.1.1 Realidad aumentada

La realidad aumentada consiste en una línea de investigación que trata de aunar la presencia de objetos del mundo real con objetos virtuales en tiempo real. El término aumentada se refiere precisamente a que se trata de aumentar la percepción de la realidad añadiendo información adicional generada por ordenador. Toda esta tecnología tiene su aplicación en diferentes ámbitos, como son: la simulación, el entretenimiento, la cirugía, servicios de emergencias y militares, arquitectura, aplicaciones industriales o turismo entre otras diferentes.

Métodos de registros de información

Usando marcadores (fig. 2)

Un marcador es un patrón en forma de imagen que se imprime para poder situar un objeto 3D en la realidad aumentada. El software se basa en el reconocimiento de estos objetos para determinar qué objeto 3D se representa y su posición.

Una de las ventajas del uso de marcadores es que permite situar con mucha más exactitud el objeto 3D que los otros métodos. Sin embargo, el principal inconveniente que surge al utilizar este método es que se necesita tener la imagen impresa en alguna superficie para poder ver el objeto, y si la imagen no se reconoce, ya sea porque esté oculta una parte o porque esté deteriorada, no se mostrará nada. Este inconveniente hace que no sea un método adecuado en el contexto en el que se está trabajando.

Figura 2. *Ejemplo de realidad aumentada con marcadores*

Usando reconocimiento de objetos (fig. 3)

Esta opción es la más actual y se basa en el reconocimiento de objetos conocidos, por ejemplo, el de la cara de una persona o el de la forma de un edificio u objeto.

El principal inconveniente de este método es su gran coste computacional y de almacenamiento de información, esto es debido a la necesidad de reconocer la forma del objeto, ya que en el proceso se tendría que cotejar las formas de la imagen con una base de datos, y cuantos más objetos se quieran reconocer, más grande será la base de datos.

Figura 3. *Ejemplo de realidad aumentada por reconocimiento de objetos*

Usando la posición GPS (fig. 4)

Otra opción que existe es situar los objetos en una cierta posición GPS y que al aproximarte a ella puedas ver los objetos a través del dispositivo.

El inconveniente de este método es la menor precisión con respecto a los métodos anteriores, debido a la precisión del sistema GPS. Sin embargo, con este sistema los inconvenientes que se veían en los métodos anteriores desaparecen, ya que ya no es necesario la existencia de ningún marcador físico en la realidad y, además, el coste computacional y de almacenamiento es considerablemente menor que en el caso del reconocimiento de objetos y, más en el contexto en el que se está trabajando en el que se maneja mucha información.

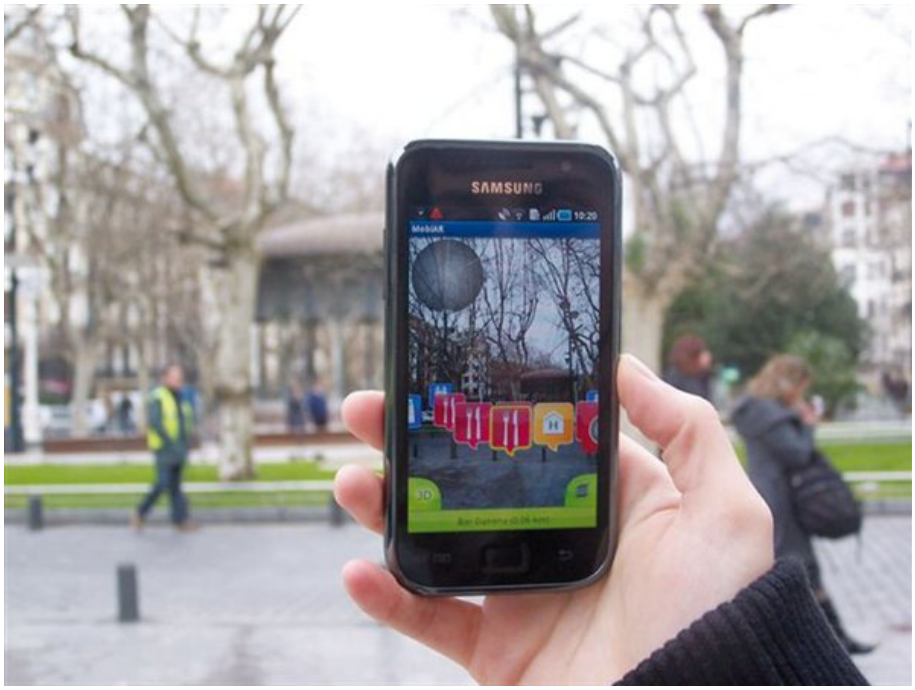


Figura 4. *Ejemplo de realidad aumentada por geoposicionamiento*

Este último método sería el que se ha seguido en el desarrollo del framework y del caso de uso de este PFC.

2.1.2 Análisis de frameworks de realidad aumentada

La empresa en la que se ha desarrollado el PFC, *GeoSpatiumLab*, ya disponía de un framework de realidad aumentada para Android (sistema operativo para dispositivos móviles creado por Google), por lo que fue analizado junto con otros proyectos que se exponen a continuación. A diferencia de Android, en iOS existen pocos proyectos libres de realidad aumentada. A pesar de ello se pudieron analizar varios ejemplos, aunque solo fuera a nivel de usuario.

Se analizaron proyectos para iOS como: Vuforia [11], Layar [12], Mixare [10] o Wikitude [13]; pero también de otras tecnologías como Android (NyARToolKit [14], ARViewer [15], LookAR [16]) o Windows Phone (Nokia City Lens [17]).

No solo se han analizado proyectos del propio iOS, puesto que de todos ellos se podía obtener ideas de navegabilidad, con el fin de mejorar la experiencia de usuario o analizar sus puntos fuertes y debilidades, los cuales se podrían incluir en el proyecto final. Tras realizar un análisis de todos ellos, el cual se puede observar más detalladamente en el anexo C y más profundamente del proyecto Mixare en el anexo D, ya que se consideró que era un proyecto del que se podría obtener mucha información acerca de los principios de la realidad aumentada, se tomó la decisión de llevar a cabo un diseño propio partiendo de las ideas obtenidas en este análisis, ya que no había ninguno que cumpliera todas las necesidades solicitadas y que se pueden ver en el análisis de los requisitos.

2.2 Análisis de los requisitos

Una vez realizado el estado del arte, haber analizado otros proyectos, y haber llevado una formación previa en iOS y Objective-C (lenguaje de programación utilizado en el iOS SDK), se llevó a cabo la especificación de los requisitos de la aplicación que se iba a desarrollar y que sirvieron como guía en las fases posteriores. En este apartado se hace un resumen de estos requisitos, ya que se han explicado más detalladamente en el anexo C de la memoria.

Requisitos funcionales:

- **RF 0:** La aplicación poseerá tres modos de operación: modo cámara o de realidad aumentada, modo mapa y modo lista.
- **RF 1:** Acceso a servicios externos de información georreferenciada.
- **RF 2:** Representación de objetos virtuales y renderización de la imagen en tiempo real en dos dimensiones.
- **RF 3:** Representación de los POIs sobre una brújula en la vista de AR.
- **RF 4:** Cálculo de visibilidad de los objetos.
- **RF 5:** Localización y posicionamiento del usuario en exteriores.
- **RF 6:** Poseerá dos modos de funcionamiento on-line y off-line.
- **RF 7:** Se permitirá la categorización de los POIs.
- **RF 8:** Se debe permitir la interacción con los objetos virtuales representados.
- **RF 9:** El radio del alcance de observación de los POIs podrá ser modificado.
- **RF 10:** Navegabilidad y usabilidad.

Requisitos no funcionales:

- **RNF 0:** El dispositivo ha de disponer de cámara digital, sensor magnético y acelerómetro.
- **RNF 1:** Será necesario estar conectado a internet (Wi-Fi o 3G) para el funcionamiento en modo mapa, debido a que los servicios de mapas utilizados solo funcionan correctamente en modo online.
- **RNF 2:** El idioma de la aplicación será en inglés y castellano.
- **RNF 3:** El dispositivo para el que va a ser implementado deberá tener sistema operativo iOS.

– **RNF 4:** Los frameworks y la aplicación serán compatibles a partir de la versión 4 de iOS y posteriores.

– **RNF 5:** La localización del usuario se hará a través del dispositivo GPS, Wi-Fi y 3G.

2.3 Arquitectura general del sistema

En este apartado se presenta una visión general del sistema desarrollado, desde los componentes hardware de la capa física, hasta los diferentes módulos que se han implementado.

Los dos componentes fundamentales que se pretendía desarrollar en este proyecto eran el framework de realidad aumentada y la aplicación ARExplorer, la cual hace uso de todas las posibilidades que nos da dicho framework. En esta aplicación se observó que era necesario un módulo de gestión de los datos de los que se alimentaría el framework y de un módulo de visión de mapas, ya que se decidió añadir esta forma de representar los POIs además de la realidad aumentada.

Para llevar a cabo la comunicación entre la aplicación y los diferentes módulos desarrollados y a su vez, estos pudieran ser agregados en otras aplicaciones, se utilizó el patrón Facade. Con este patrón se establece un único punto de acceso a cada uno de los módulos de la aplicación, abstrayendo del comportamiento interno al desarrollador de la aplicación y permitiendo la utilización del módulo conociendo únicamente los métodos del Facade, lo cual puede ahorrar esfuerzos y facilitar su reutilización. Como cada módulo tiene su controlador propio, se ha añadido un controlador superior, como se puede ver en el esquema de la arquitectura, que unifica el control de todos ellos.

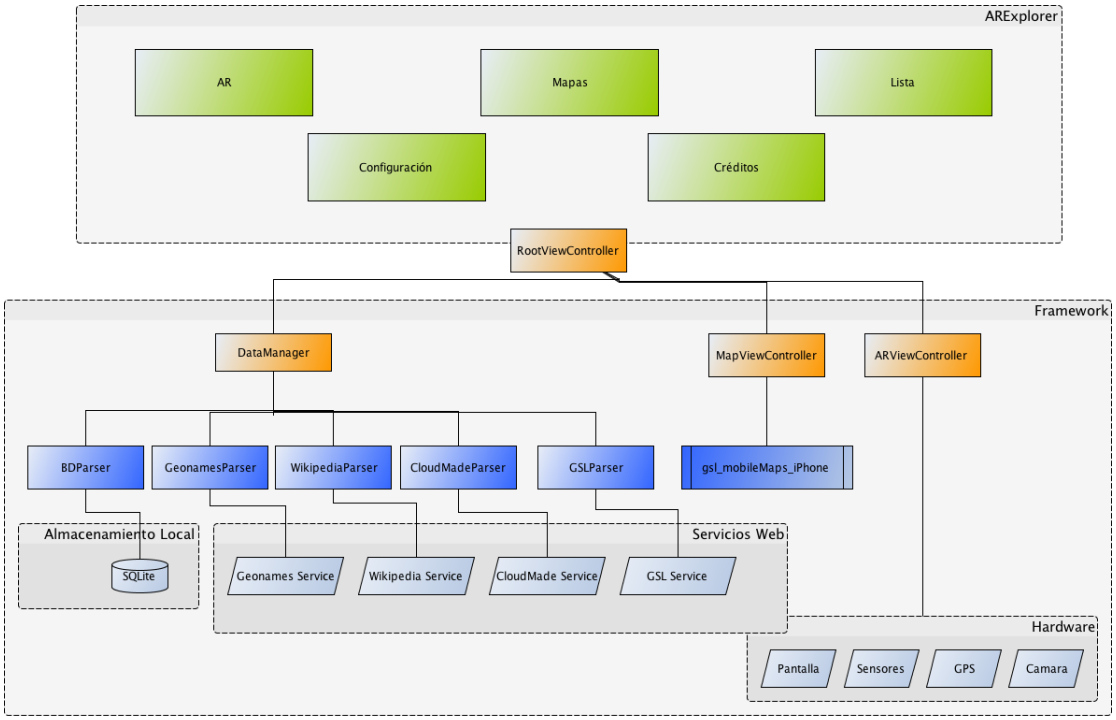


Figura 5. *Diagrama de la arquitectura del proyecto*

En el gráfico anterior (fig. 5) se puede ver claramente los diferentes componentes bien diferenciados; en la parte superior estaría la aplicación, que hace uso de los diferentes módulos por medio de RootViewController, que sería la clase facade, la cual se comunicaría con el resto de los módulos, los cuales constituyen los diferentes módulos. Además de esto, la clase RootViewController también sirve de comunicación entre los diferentes frameworks, por ejemplo, tanto el módulo de realidad aumentada como el de mapas, debe obtener datos para mostrar, que son obtenidos a través del módulo de datos, por lo que esta comunicación se hace a través de este controlador.

En la parte de la aplicación, cada uno de los módulos se corresponde con cada una de las pantallas de la aplicación o lo que es lo mismo, los controladores de las vistas.

En la parte de frameworks, estaría ARViewController (módulo de realidad aumentada), que interactúa con los diferentes sensores y componentes hardware, para controlar el renderizado y la interacción con los diferentes POIs que son mostrados a través de la realidad aumentada.

Por otra parte estaría MapViewController, el cual se encarga de la gestión del visor de mapas. Se ha realizado con la ayuda del visor de *GeospatiumLab*, el cual es sencillo de utilizar y se trata de un componente de mapas genéricos que permite la carga de éstos desde diferentes fuentes.

Por último está DataManager, encargado de la gestión de la petición de datos a los diferentes servicios utilizados y/o al almacenamiento local, según sea necesario y siguiendo las políticas explicadas en apartados posteriores.

Como conclusiones de este esquema de la arquitectura, se puede deducir que se ha tratado en todo momento de diferenciar lo que sería la parte de la aplicación de la parte de los frameworks, con el fin de que estos últimos fueran aprovechables en aplicaciones futuras sin que apenas hubiera que realizar cambios, tanto individualmente como en conjunto.

2.4 Gestión y desarrollo del proyecto

En primer lugar hay que comentar que el entorno y las tecnologías utilizadas eran totalmente nuevas para el proyectando, por lo que se partió de una fase previa de estudio y preparación. Debido al desconocimiento de estas tecnologías y con el fin de desarrollar un proyecto de mayor calidad, se evitó el uso de ciclos de vida en cascada o similares los cuales son mucho más rígidos y no son buenos si el entorno es nuevo para el desarrollador. Se llevó a cabo un desarrollo iterativo e incremental [26] en el cual el proyecto se dividió en varias partes y se les asignó una prioridad y un orden de elaboración y, además, éstas a su vez se dividieron en subtareas. Así pues, se consiguió un desarrollo mucho más ágil y, a la vez, como ya se ha dicho, de mayor calidad. Todo ello está vinculado con la búsqueda de la excelencia y la calidad de los productos, hecho por el que *GeosptiumLab* ha promovido un Sistema Integrado de Gestión de la Calidad y el Medioambiente sobre la base de los requisitos de las Normas UNE-EN-ISO 9001:2008 (noviembre de 2008) y UNE-EN-ISO 14001:2004 (diciembre de 2004), para todas las actividades que desarrolla, que se complementa con los requisitos de la norma ISO/IEC 15504 “Software Process Improvement Capability Determination” (SPICE), nivel de capacidad 2, con el objetivo primordial de satisfacer a sus clientes mediante la prestación de servicios de la máxima calidad y respeto al entorno. Este Sistema Integrado de Gestión obtuvo su certificación formal en el año 2011.

Los incrementos en que se dividió el proyecto por orden de prioridad son las siguientes:

– Módulo de realidad aumentada.
– Módulo de gestión de datos.
– Módulo de mapas.
– Integración de los módulos y elaboración de la aplicación de caso de uso.

Todos estos incrementos a su vez estaban divididos en subtareas de análisis y prototipado, diseño, implementación, integración y test, además de la elaboración de la documentación en cada una de las partes y la elaboración de una batería de pruebas de funcionalidad, independientemente de los test anteriormente comentados, para la verificación de los requisitos funcionales y no funcionales como se puede ver en el anexo K.

Cada uno de estos incrementos no se cerraban completamente ya que, como el aprendizaje también fue progresivo, en algunas ocasiones se redefinió alguna de las partes de módulos anteriores o se solventaron algunos problemas que pudieron aparecer.

Esta metodología, los recursos utilizados y el control de esfuerzos llevado a cabo, son explicados más en detalle en el anexo A.

2.5 Diseño del módulo de realidad aumentada

En este framework se quieren implementar los diferentes algoritmos relativos al cálculo de la visibilidad de los diferentes puntos de interés por medio de la realidad aumentada, es decir teniendo en cuenta la posición y la orientación del usuario para representarlos en la pantalla del dispositivo.

Las funcionalidades más importantes de este framework serían:

– Cálculo de la visibilidad:
  • Resuelve la problemática relacionada con la decisión de si los puntos de interés son visibles mediante el cálculo de la orientación y la distancia.
  • Gestiona el funcionamiento de la cámara digital del dispositivo obteniendo la máxima resolución posible.
  • Gestión de la distancia de los puntos a mostrar, limitados por la distancia que establezca el usuario, pudiendo ser ésta de hasta 20 kilómetros.
  • Filtrado de la visualización de los POIs en función de una distancia máxima o los n primeros POIs que se encuentren más cercanos.
– Representación gráfica de los puntos de interés:
  • Visualización de los puntos de interés cercanos a través de la pantalla de la cámara y en un radar en función del cálculo obtenido en el apartado anterior.
– Gestión de localización necesaria en el cálculo de la visibilidad para el correcto posicionamiento de los puntos de interés:
  • Gestión de la localización a través del GPS y del 3G y, en caso de no disponer de ésta, utilizar la última válida que ha sido obtenida.
  • Gestión de la actualización de la localización solo en el caso en el que se haya recorrido una distancia mínima o un tiempo determinado.

– Gestión de los sensores necesarios para el algoritmo de cálculo de la visibilidad:
  - Gestión de los sensores magnéticos y gravitatorios, corrigiendo la declinación existente según el Modelo Mundial Magnético válido hasta 2015 (gracias al framework nativo de iOS CoreLocation)
  - Cálculo de la orientación del dispositivo, suavizando los cambios bruscos que se obtienen de los sensores y que, en muchos casos, son datos espúreos.

– Gestión de la interacción con los puntos de interés en pantalla:
  - Control del pulsado sobre los puntos de interés.
  - Realizando acciones predefinidas para los objetos, dentro del framework, representando la información disponible.
  - Permite externalizar los eventos de pulsado indicando qué objeto fue seleccionado y facilitando el poder modificar dicho comportamiento.

2.5.1 Problemática del rumbo y de la representación en pantalla [7-9]

Cuando se trata de implementar la realidad aumentada existen dos problemáticas fundamentales, la representación en pantalla de los puntos de interés y el problema del rumbo y la orientación del usuario. Para dominar estos problemas hubo que realizar un estudio teórico de éstos y de una serie de conceptos, los cuales han sido resumidos en el anexo E.

Cálculo del rumbo y orientación

Para el cálculo de la distancia entre dos puntos de la esfera terrestre y su orientación (fig. 6), existe una trigonometría de triángulos esféricos que relaciona ángulos con la distancia de sus lados.

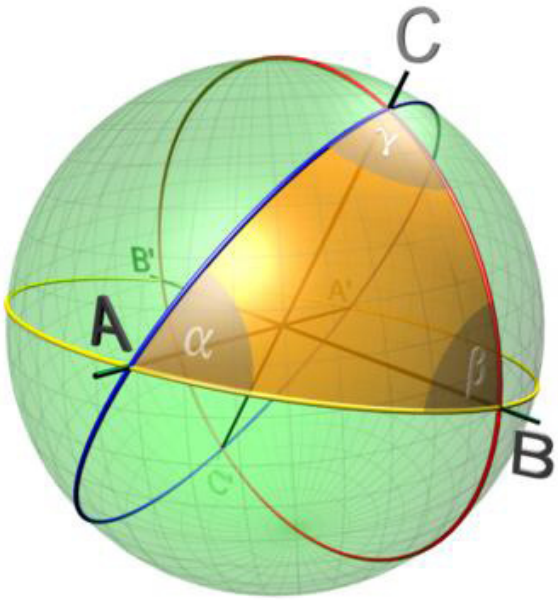


Figura 6. *Triángulo esférico entre dos localizaciones y el norte*

Como se puede ver en la imagen anterior, A sería el usuario, B el POI y C el norte geográfico, por lo que hay que saber si el usuario está orientado hacia el punto concreto, lo cual se resuelve cuando el ángulo acimut coincide con el ángulo alfa del triángulo esférico.

Todos estos cálculos podrían realizarse con trigonometría euclídea y en distancias cortas la diferencia apenas se apreciaría, pero las diferencias surgen en la media y larga distancia, por lo que se opta a esta implementación a través del ángulo acimut, empleando el modelo WGS 84 que también es usado por iOS.

La representación en pantalla

Uno de los problemas que presenta la realidad aumentada en el móvil, es que al tratarse de un sistema de realidad en 2D no existe la perspectiva de profundidad. Así pues, lo que hay que hacer es hallar las coordenadas de los pixeles en las que debemos representar los puntos sobre la pantalla del dispositivo.

Los que si que se conocen son el ángulo y la distancia que hay con respecto al observador, como se ha visto en el apartado anterior, y para dar la sensación de profundidad lo que se puede hacer es tomar como referencia que todos los puntos estén a la misma distancia, el horizonte; por tanto la distancia al punto, que es conocida, se use como factor de escalado y así poder modificar el tamaño del punto de interés en función de este valor, dibujando de menor tamaño los puntos más lejanos.

Otro aspecto a tener en cuenta es el ángulo de visión de la cámara. Esto quiere decir que puede que se tenga el dispositivo orientado hacia una dirección, pero la cámara no es capaz de representar toda la imagen, sino que tiene un ángulo de visión limitado que hay que tener en cuenta, ya que sino la representación sería errónea.

Así, para calcular las coordenadas en la pantalla de un POI se realizaría con la siguiente fórmula:

$$pixel_x = \frac{WIDTH}{2} + [\sin(\partial)/\sin(\ ANGULO_{VISION}\ I)] * \left(\frac{WIDTH}{2}\right)$$

Siendo WIDTH el ancho de la pantalla, el ángulo la diferencia entre nuestra orientación acimut, el ángulo que forma el objeto con nuestra localización y suponiendo un ángulo de visión de 30°. Otro aspecto sería que se considera el centro de coordenadas (WIDTH/2, WIDTH/2) en lugar de (0,0), por lo que se realiza una translación del punto.

Con respecto al eje Y se emplea el ángulo de inclinación del dispositivo para que los objetos no se observen siempre en el eje medio de la cámara, en lugar del ángulo del eje X.

2.5.2 Protocolo de utilización

En el anexo F de este documento se explica de manera detallada el diseño de este framework. En este apartado se va a realizar un resumen de sus componentes principales y un esquema de su funcionamiento.

En primer lugar, lo que se hizo fue elaborar un protocolo de Objective-C (comunmente conocido como interfaz) con el fin de que los futuros desarrolladores pudieran ver de manera clara

las funciones que se pueden utilizar para el correcto manejo del mismo (esta técnica también ha sido seguida en el resto de frameworks). Dicho protocolo se puede ver a continuación:

```
@protocol ARViewControllerProtocol
//Añadir POIs obtenidos de las fuentes de datos
- (void)addPOI:(PoiObject *)coordinate;
- (void)addPOIs:(NSArray *)newCoordinates;
//Borrado de datos
- (void)removePOI:(PoiObject *)coordinate;
- (void)removePOIs;

- (id)initWithDB:(sqlite3 *)contact;
- (void) startListening;
- (void) finishCamera;
@end
```

Se puede observar que se dispone de varios mensajes para añadir y borrar coordenadas (POIs) que puede ser tanto de manera individual, como por medio de un array de POIs. Además se dispone de un inicializador en el que se le pasa como parámetro un conector con una base de datos para que pueda hacer uso de la petición y gestión de la base de datos.

Además de este protocolo existen una serie de parámetros, los cuales se pueden modificar, con el fin de variar el comportamiento del framework. Los fundamentales serían:

- **maximumRotationAngle, rotateViewsBasedOnPerspective**: Estos parámetros sirven para indicar si se les debe aplicar una transformación a los puntos gráficos cuando se encuentra en los laterales de la pantalla y el ángulo de transformación, con el fin de dar una mayor sensación de profundidad.
- **onlyBasedDistance**: Sirve para indicar si en la representación solo se debe tener en cuenta la distancia o también la altura, de modo que los puntos no son fijos en el eje horizontal de la imagen. Esto es útil en el caso de que se tengan servicios que proporcionen la altura como tercer coordenada y así dar una mayor fidelidad de la representación de los datos.
- **updateFrecuency**: Frecuencia con la que es actualizada la pantalla de AR.
- **showTitleLabel**: Se trata de un booleano para indicar si se desea que se muestre el título del POI debajo del icono de la categoría.
- **dataManager:** Se utiliza para indicar el manejador de la gestión de datos (en la aplicación se trata del framework que se ha diseñado para tal efecto).
- **radarfixed:** Este parámetro se utiliza para controlar si el radar en el que se representan los POIs, además de por la realidad aumentada, se encuentra fijo o móvil. En el caso de que se deje fijo, los puntos no se mueven y es una flecha sobre el radar la que indica la orientación de usuario. Si se configura en modo "móvil", esta flecha permanece fija y son los POIs los que se van moviendo por el radar, indicando esta flecha los puntos que se están visualizando en cada momento en la realidad aumentada.

2.5.3 Principales componentes

Los principales componentes serían:

ARViewController: Ésta sería la clase principal y se encarga de controlar diversas funciones:

- Se realiza la calibración de los sensores necesarios como son el acelerómetro y el giroscopio.
- Se encarga de la localización geográfica del dispositivo, gracias al GPS y al 3G o el WiFi según esté disponible uno u otro o proporcione una mayor precisión.
- Realiza la inicialización de cámara y superpone a la vista de ésta los objetos que se quieren representar, pero también inicializa las clases relacionadas con el radar para el dibujado del mismo.

El funcionamiento de esta clase consiste en la llamada cada cierto período de tiempo parametrizable, controlado por temporizadores, al método de pintado, para realizar la actualización de la pantalla en tiempo real, recorriendo una lista en la que se almacenan los objetos a representar tanto en la realidad aumentada como en el radar. Esto se podría ver más simplemente, a través de la siguiente sección de pseudocódigo el cual sería llamado con la frecuencia establecida:

```
- (void)updateLocations:(NSTimer *)timer {
        for (poi in list_ofPOIs) {
            if (poiIsInScreen(poi)) {
                loc = calculateLocInScreen(poi);
                Marker = calculateScaleWithDistance(poi);
            }
            else {
                removeFromScreen(poi);
             }
        }
}
```

Como se puede ver el funcionamiento, una vez implementadas las funciones para conocer si un punto debe representarse, y calcular su posición en la pantalla y su escala en función de la distancia que lo separe del usuario, sería bastante básico.

RadarView: Esta clase se encarga de dibujar el radar y los POIs de su interior dibujándolos de diferente color en función de la fuente de datos y escalados en relación con el radio máximo de búsqueda.

POIObject: Se trata de un objeto geolocalizado que se encuentra cargado en memoria, del que se conoce como poco su latitud y su longitud (esta clase es la misma que aparece en el framework de gestión de datos).

También se almacena cual es la fuente de datos desde la que se ha obtenido, la URL con la información del POI, su nombre y descripción (título y subtítulo) y la distancia al mismo desde

el dispositivo, así como el acimut, que son los grados de desviación de un punto con respecto al norte geográfico y los grados de inclinación.

MarkerView: En esta clase se controlan los eventos de pulsado en la pantalla en la capa superpuesta a la vista de la cámara que se encarga de representar gráficamente los puntos de interés.

Esta clase es configurable, ya que estos eventos pueden ser manejados en la propia clase del framework, o ser enviados hacia el controlador de la aplicación que lo esté utilizando, gracias al modelo de notificaciones implementado en iOS, debido a que no se desee utilizar la representación por defecto implementada en el framework. La variable que controla esto es **touchEvent.**

2.6 Diseño del módulo de mapas

En este framework se va a implementar la representación de los diferentes puntos de interés que han sido obtenidos, sobre un mapa centrado sobre la posición actual del dispositivo.

Para su implementación se ha hecho uso de un visor de mapas desarrollado por *GeoSpatiumLab* para iOS, que satisface las necesidades de la aplicación que se quiere llevar a cabo. Ofrece algunas funcionalidades adicionales, como el acceso a diferentes servicios de mapas tileados. Pero la idea era que el cambio de un visor de mapas a otro no sea complicado y sean fácilmente intercambiables, para ello se ha creado una interfaz con las funciones básicas que el servicio en cuestión debería proporcionar.

Las funcionalidades principales serían:

- Representación de los puntos de interés sobre el mapa.
  - Representación de los puntos de interés que han sido descargados por el framework de gestión de datos.
  - Representación de la localización actual del dispositivo.
- Control de zoom y navegación por el mapa.
- Interacción con los objetos representados, captando el evento de tocado para mostrar la información que se dispone del mismo.
- Control del tipo de mapas sobre el que mostrar la información (normal, satélite e híbrida).
- Acceso a diferentes servicios de mapas como son Google Maps, Apple Maps, IDEZar, Cartociudad o PNOA.

2.7 Diseño del módulo de gestión de datos

En este framework se han llevado a cabo varios aspectos importantes de la aplicación. En primer lugar, la gestión de datos a servidores externos; en segundo lugar, la categorización de estos datos; y, por último, dar soporte al modo offline de la aplicación por medio de alguno de los sistemas de persistencia de datos posibles.

Así pues este framework se deberá encargar de los siguientes aspectos:

– Petición de datos a servicios externos:
  - Utilización de los servicios REST que nos proporcionan Geonames [20], CloudMade [21] y GSL. Servicios elegidos para la descarga de datos.
  - Parseo de los datos procedentes de dichos servicios en formato JSON [19] o XML, adaptándolos al modelo de datos de la aplicación, ya que la estructura de los datos de dichos servicios no es homogénea.
  - Soporte para la adición de un nuevo servicio futuro, implementado un protocolo.

– Persistencia de datos:
  - Almacenamiento de datos en una base de datos SQLite, con el fin de reducir la transferencia de información entre los servicios seleccionados y la aplicación al mínimo posible.
  - Cache intermedia para la optimización de la utilización de los datos y mejora del rendimiento en la muestra de los POIs tanto en la parte de realidad aumentada como en el caso de los mapas.
  - Categorización de los puntos de interés, con independencia de la fuente de datos.
  - Soporte para el modo offline (gracias a la persistencia de datos) en el que no sea necesario la conexión a los servicios de datos, bien porque no esté disponible o porque el propio usuario no lo desee.

– Soporte para las búsquedas siguiendo diversos criterios:
  - Búsqueda por categorías en los servicios.
  - Selección de servicios de los que descargar los datos (Geonames, Wikipedia-Geonames, Cloudmade o GSL en una primera implementación, pero ampliable con gran facilidad).
  - Limitación del número máximo de puntos a mostrar en una petición.

2.7.1 Problemática de la gestión de datos

En este caso se han tomado varias decisiones de relevante importancia para el correcto funcionamiento de la aplicación y del framework de realidad aumentada, en el que es crítico que los datos estén disponibles en el momento adecuado, para no comprometer su correcto funcionamiento.

Todas estas decisiones han sido explicadas detalladamente en diferentes anexos de esta memoria, pero en este apartado se va a hacer un resumen de todas ellas, ya que, como se ha dicho, han sido decisiones muy importantes en el correcto desarrollo del proyecto.

En primer lugar, se ha tenido que decidir cual iba a ser la política de petición y de almacenamiento de datos, ya que hay que tener un compromiso entre la cantidad de peticiones que se hacen a los servidores externos, lo cual es un cuello de botella en la gestión de datos, y la necesidad de mantener los datos actualizados en todo momento.

Como existen diversas opciones, se ha llevado a cabo un estudio de tiempos y rendimiento, el cual se puede ver detalladamente en el anexo J, con el fin de tomar la mejor decisión con datos objetivos. El estudio en detalle se puede ver en el último anexo de la memoria, pero las conclusiones se van a explicar y se pueden ver en la siguiente gráfica (fig. 7):

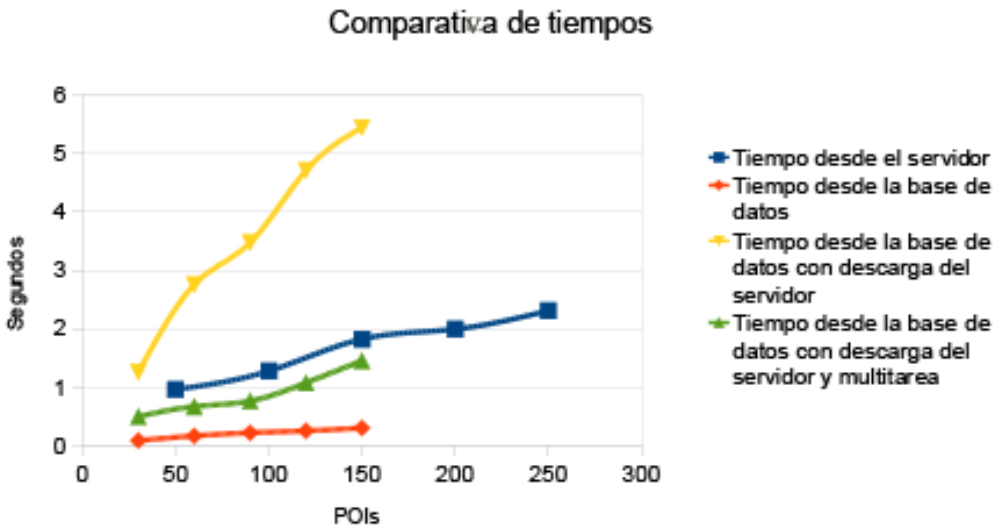


Figura 7. *Comparativa de tiempos entre los diferentes modelos de gestión de datos*

En la gráfica se pueden ver las cuatro posibilidades que se analizaron. En primer lugar, se ha analizado la opción de pedir los datos cada vez que fuera necesario, a los servidores externos, pero se descarta debido a que, como ya se ha dicho, la petición a los servidores externos es un proceso lento, como se puede ver en la gráfica azul. Por tanto, se ha tratado de reducir la cantidad de peticiones de los servidores aprovechando la base de datos local que había que añadir para implementar el modo offline, ya que como se puede ver en la gráfica naranja, las peticiones a la base de datos local era la opción más rápida de todas.

El siguiente paso que se decidió fue la combinación de la base de datos y de los servidores, pero se observó que en los casos en los que había que solicitar datos al servidor y guardarlos en la base de datos, los tiempos aumentaban considerablemente como se ve en la gráfica amarilla.

Para solucionar este problema se observó que hay procesos que se pueden realizar en paralelo, por lo que se ha optado por la multitarea. Por una parte, al pedir datos a tres servidores distintos, las peticiones a cada uno de ellos eran independientes y se podían realizar en paralelo, con el fin de agilizar el proceso. Además, una vez que los datos son recibidos en el dispositivo, estos ya pueden ser servidos a la aplicación, por tanto, el guardado en la base de datos local es realizado por una tarea en segundo plano. De este modo, se ha conseguido reducir los tiempos, en el caso crítico, a los que aparecen en la gráfica verde y siendo el caso óptimo el de la gráfica naranja.

Unido a este problema ha surgido otro distinto, ya que, aunque se han reducido los tiempos considerablemente, siguen habiendo métricas entre los dos últimos casos que se han expuesto, por lo que era necesario reducir la cantidad de peticiones a los servidores y aumentar las de la base de datos local. Así pues, se estableció una política de datos en la que solo se hacen peticiones en caso de que se ampliara el radio de búsqueda (ya que si se reduce, los puntos

recibidos serán un subconjunto de los presentes en la base de datos) o en caso de que se detecte un cambio de posición.

A partir de esto, ha habido que definir cual es la granularidad para detectar un cambio de posición. Tomando un compromiso entre la actualización de los datos y la eficiencia comentada, se ha planteado que se produce un cambio de posición cuando existe una distancia igual o mayor a la mitad del radio de búsqueda, entre la posición actual y la posición en la que se realizó la última petición, es decir:

distancia (punto actual, punto última petición) >= radio de búsqueda/2

En un primer instante puede parecer mucha distancia, pero es necesario tener un compromiso, ya que si en cada desplazamiento se produjera una actualización, el tiempo real que requiere la realidad aumentada se pondría en peligro. No obstante, si el usuario quiere más precisión, puede reducir el radio de búsqueda y la peticiones serán mucho más frecuentes.

El último aspecto a tener en cuenta era qué hacer con los datos ya existentes en la base de datos cuando se producen nuevas peticiones, puesto que ésta funciona a modo de cache y por tanto no es de gran tamaño. Así pues, se han implementado dos soluciones entre las cuales puede elegir libremente el desarrollador, en función de los datos que vaya a haber de la aplicación. Una primera en la que cada vez que se realiza una nueva petición los datos que hay en la base de datos son borrados y una segunda en la que los datos no son borrados en la base de datos hasta que no se alcanza una cantidad de POIs máxima en la misma. Dicha cantidad puede ser definida por el desarrollador. Una vez que se ha alcanzado esta cantidad máxima, se han implementado dos políticas de borrado, una en la que se borran lo datos que hace más tiempo que se descargaron (FIFO) y otra en la que se borran los datos que hace más tiempo que no se usan (política LRU).

2.7.2 Modos de operación

Combinando las políticas de datos analizadas anteriormente se ha creado en el framework una política propia con tres modos de operación y, por tanto, también en la aplicación, los cuales pueden ser elegidos por el usuario:

– El **modo internet u online puro** en el que lo datos siempre son pedidos a los servidores externos y no existe interacción con la base de datos. En este modo se consigue que los datos siempre estén actualizados con la información más reciente que haya disponible en el servidor. Por otro lado, hay que tener en cuenta que en este modo el tráfico de datos será mayor, ya que no se está teniendo en cuenta los datos que ya pudieran haber sido pedidos.

– El **modo mixto**. En este modo sí que se tienen en consideración los datos que ya han sido pedidos anteriormente. Consiste en que los datos cada vez que son pedidos al servidor también son guardados en una base de datos, lo cual permite evitar ciertas peticiones a dicho servidor externo en algunas situaciones, así como surtir de la información necesaria al modo offline que se explicará a continuación. En este modo se implementa la política optimizada que se explicó en el apartado anterior.

– Por último estaría el **modo offline**. Este modo puede ser utilizado por petición del usuario pero también entra en funcionamiento automáticamente en caso de que no se detecte conexión 3G o WiFi. En este caso, la posición del dispositivo es tomada solamente con la

ayuda de los satélites GPS, sin poder hacer uso del A-GPS (Assisted GPS) ya que como se ha dicho no hay conexión ni a 3G ni a WiFi. Esto tiene como consecuencia que la obtención de la posición será más lenta y, por otro lado, que los datos no son tomados de servidores externos sino de los datos que hubiera en la base de datos local que se encuentren dentro del radio de búsqueda.

El motivo de implementar estos tres modos de operación se debe a que al tratarse de un framework y de una aplicación generalista tenían mucho interés puesto que es posible que en aplicaciones futuras sea interesante alguno de estos modos de operación en concreto.

2.7.3 Categorización de los puntos de interés

Para darle mayor riqueza a la información de los puntos que se muestra a través de la realidad aumentada o de cualquiera de las otras formas posibles en las que puede ser integrado este framework, se ha decidido hacer una categorización de los puntos de interés en función de diferentes temáticas (en este caso orientadas al turismo).

El problema de la categorización ha surgido debido a que se han elegido varios servicios heterogéneos, los cuales proporcionan una categorización, pero ésta es distinta entre unos y otros. Por tanto, se ha decidido que la mejor solución es elaborar una categorización propia (la cual se puede ver en el anexo G) y mapearla con la de los diferentes servicios, guardando este mapeo en la base de datos. De este modo, cuando se recibe un nuevo dato, éste se puede categorizar de un modo correcto siguiendo la categorización creada. El motivo por el que se eligieron estos servicios de datos heterogéneos fue debido a la escasez de servicios que ofrezcan este tipo de información. Así que la posible eleción estaba limitada por este motivo.

2.7.4 Protocolo de utilización

Del mismo modo que en el caso del framework de AR, se ha elaborado un protocolo de Objective-C, con el fin de facilitar la utilización de este framework a los desarrolladores que deseen utilizarlo en aplicaciones futuras:

```
@protocol DataManagerProtocol

-(void) setTypeConnection:(NSString *)type;

-(NSMutableArray *) requestData: (CGFloat) lat Lon: (CGFloat) lon Alt: (CGFloat) alt radius: (CGFloat) rad Elems: (NSInteger) elem;

-(NSMutableArray *) requestDataFrom: (NSString*) source Lat: (CGFloat) lat Lon: (CGFloat) lon Alt: (CGFloat) alt radius: (CGFloat) rad Elems: (NSInteger) elem Lang: (NSString *) lang;

-(NSMutableArray *) getFromDB: (CGFloat) lat Lon: (CGFloat) lon Alt: (CGFloat) alt radius: (CGFloat) rad;

-(void) startDB: (sqlite3 *) contact;

-(void) deleteDB;

-(void) saveDB: (NSMutableArray *) listPOIs;

-(void) savePoiDB: (GeoPOIObject *) poi;

@end
```

En él se pueden ver los mensajes principales:

setTypeConnection: Selecciona el tipo de conexión con el que los datos van a ser tomados como se ha explicado en el apartado de política de datos (Internet, mixto, offline).

requestData: Realiza la petición de datos desde una posición concreta y con un radio determinado a todos los servicios que estén activos en ese momento en el modo de conexión de internet y mixto.

requestDataFrom: Igual que el anterior pero se especifica al servidor que se quiere realizar la petición.

getFromDB: Realiza la petición, en este caso, a la base de datos local.

startDB y deleteDB: El primero de ellos inicia y crea la base de datos local. El segundo borra todo el contenido que hubiera en la base de datos relativo a los puntos de interés.

saveDB y savePOIDB: El primero de ellos recibe como parámetro una lista de puntos de interés y los almacena en la base de datos y el segundo hace lo mismo pero de forma unitaria.

El framework tiene diferentes atributos que permiten parametrizar diferentes comportamientos, como son el tipo de política (**typeConnection**) de obtención de datos (internet, mixta u offline), la cantidad máxima de puntos almacenados en la base de datos local (**maxInDB**), la política de borrado (FIFO o LRU) cuando hay que borrar elementos que hay en la base de datos (**updateType**) ya que se ha alcanzado esta cantidad máxima (**maxInDB**), el listado de los servicios desde los que se quieren obtener los datos (**sources**) y el listado de las categorías sobre las que se quiere mostrar datos (**searchCategories**).

Todo lo referido a la conexión, la creación y la consulta de la base de datos se ha realizado por medio de la librería sqlite que aparece por defecto en el iOS SDK.

2.8 Caso de uso final: aplicación ARExplorer

Una vez desarrollados los diferentes módulos que se han explicado, la elaboración de la aplicación simplemente consistió en la integración de todos ellos como se puede ver en el siguiente esquema (fig. 8):

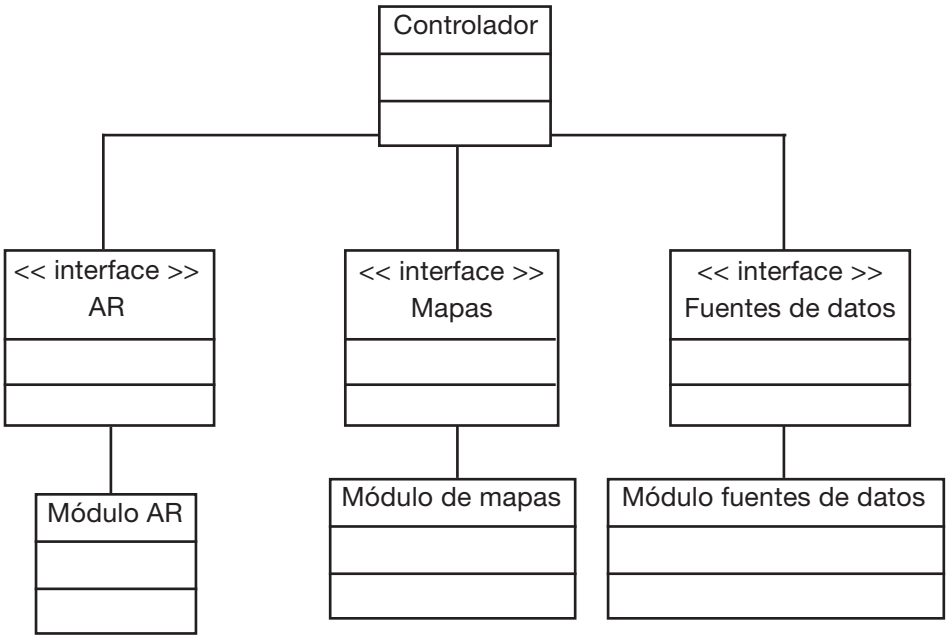


Figura 8. *Esquema conceptual de la utilización de los diferentes módulos implementados*

El patrón elegido para la elaboración de la aplicación es MVC (Modelo-Vista-Controlador). En este aspecto, no ha existido mucha elección ya que la programación de aplicaciones iOS se fundamenta en este patrón y todas ellas deben ajustarse a éste. Por un lado estaría el modelo de datos; la vista, la cual puede ser programada de diversos modos, por medio de los storyboards de iOS, o los ficheros XIB, en los que la programación es gráfica, pero también se puede programar directamente con código. Por último estaría el controlador, que comunica la vista con el modelo de datos y el caso de iOS, por regla general reciben el nombre de ViewControllers.

Como ya se ha comentado, la aplicación se ha fundamentado en ensamblar los diferentes módulos que se ha desarrollado, por medio de la clase RootViewController. Pero antes de todo esto, hubo que realizar un estudio de la navegabilidad y un prototipado de las pantallas que iban a conformar la aplicación los cuales pueden ser analizados en el anexo I.

Así pues, en el siguiente diagrama (fig. 9) se podría ver un esquema de clases simplificado de lo que sería la aplicación, con los diferentes módulos bien diferenciados, con la clase “RootViewController”, con controlador de la aplicación y de la interacción entre los diferentes módulos:

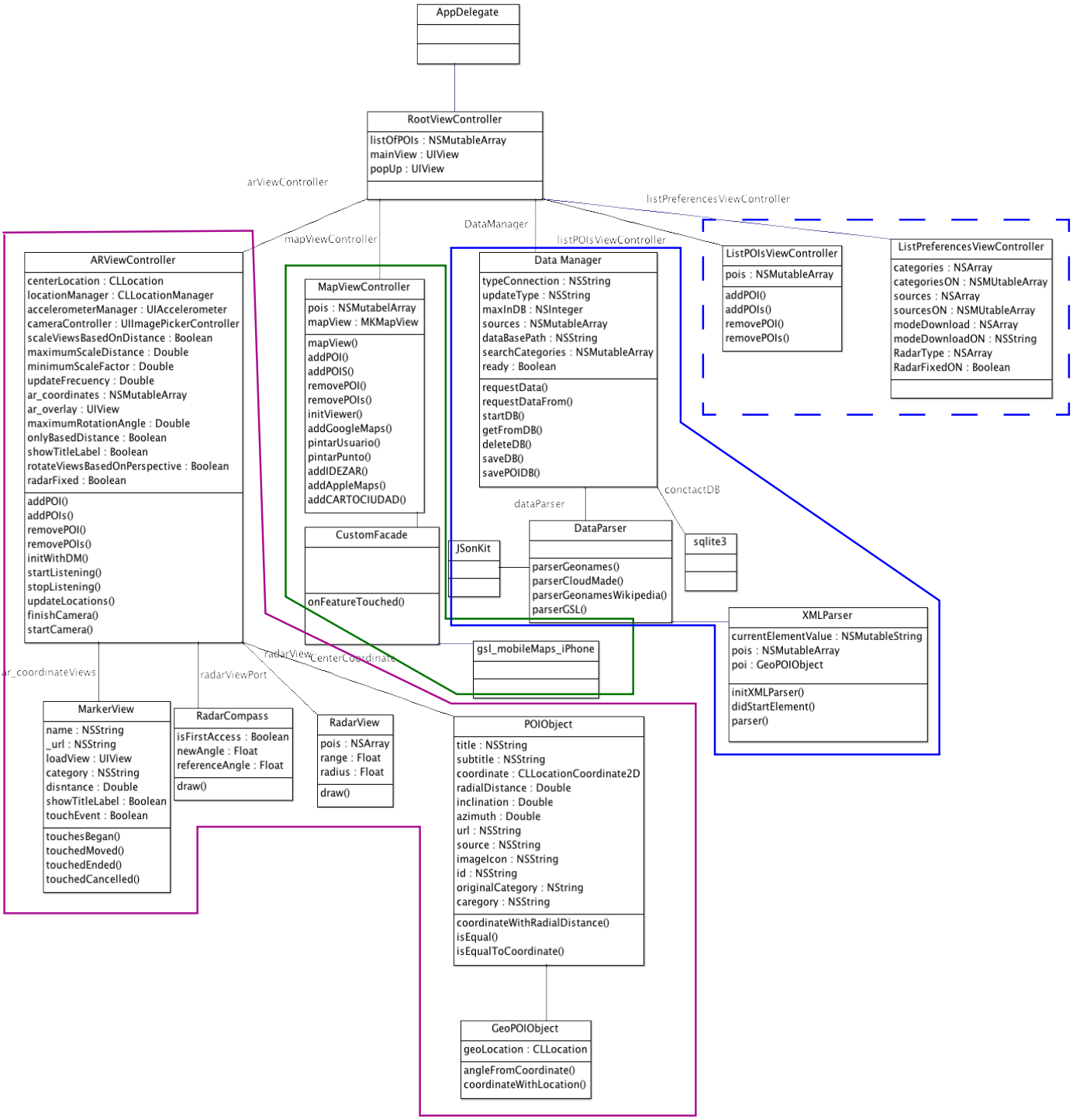


Figura 9. *Diagrama de clases simplificado de la aplicación ARExplorer*

2.8.1 Resultado final

Una vez explicado toda la estructura del sistema, se van a mostrar una serie de imágenes de la aplicación que se ha desarrollado.

En primer lugar, al abrir la aplicación aparece la portada de la misma (fig. 10):

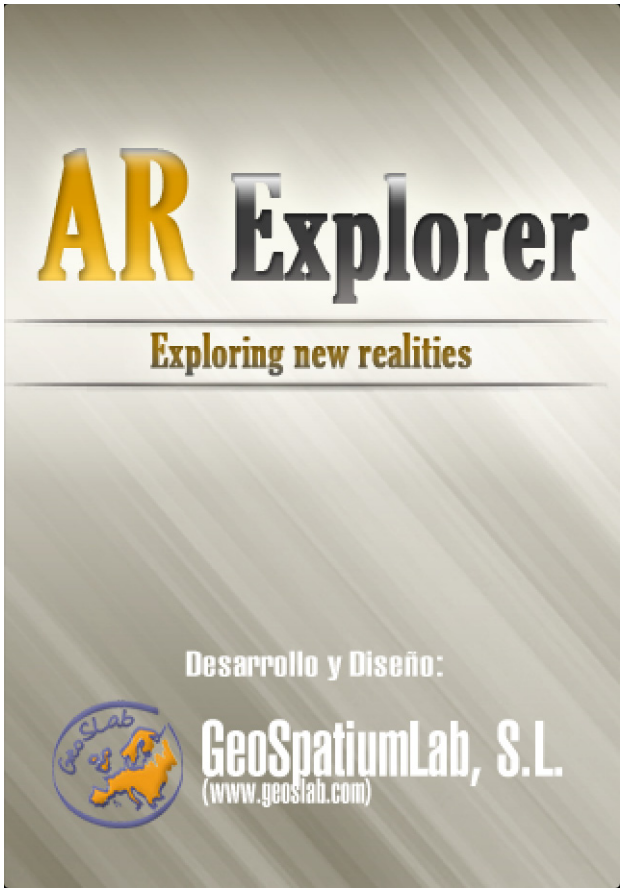


Figura 10. *Portada de la aplicación*

Una vez cargados los datos de los diferentes servicios se inicia automáticamente el visor de realidad aumentada con los POIs correspondientes (fig. 11), pudiéndose interactuar con ellos (fig. 12) o cambiar el radio de búsqueda mediante el botón que aparece en la esquina superior derecha:

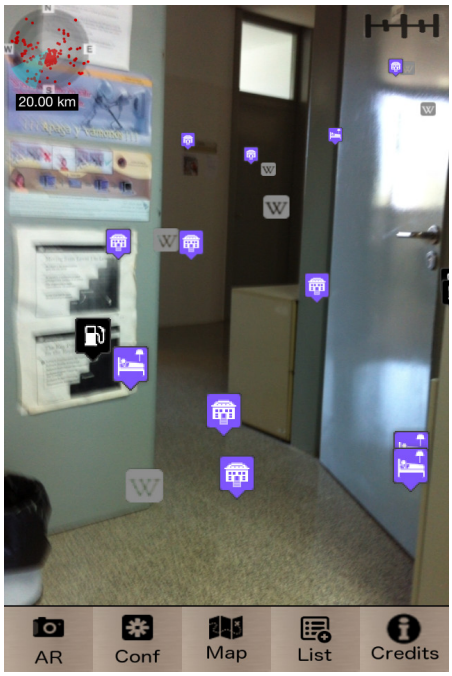


Figura 11. *Visor AR*

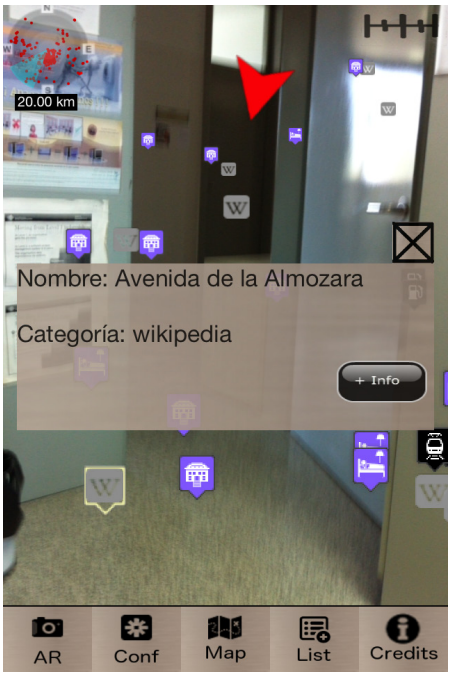


Figura 12. *Interacción con POIs*

Como ya se ha comentado en apartados anteriores existen tres modos de representar los puntos de interés que rodean al usuario, es decir, el modo listado (fig. 13) y la representación sobre un mapa, habiéndose elegido para esta aplicación la posibilidad de elegir la fuente de este mapa: Apple (fig. 14), IDEZar (fig. 15) y Cartociudad (fig. 16):

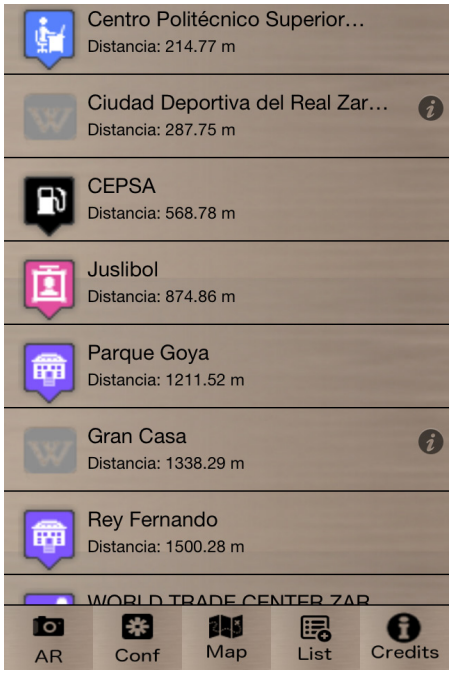


Figura 13. *Listado de POIs*

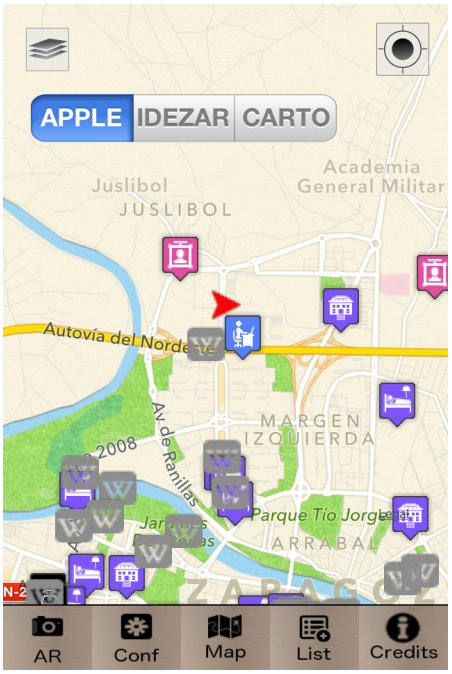


Figura 14. *Visor de mapas Apple*

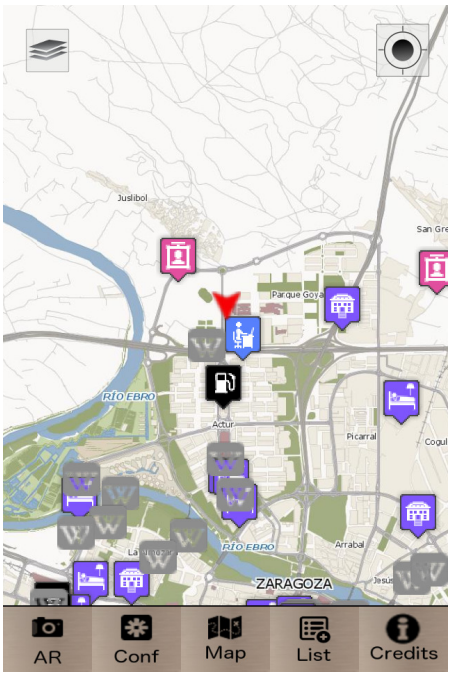


Figura 15. *Visor de mapas IDEZar*

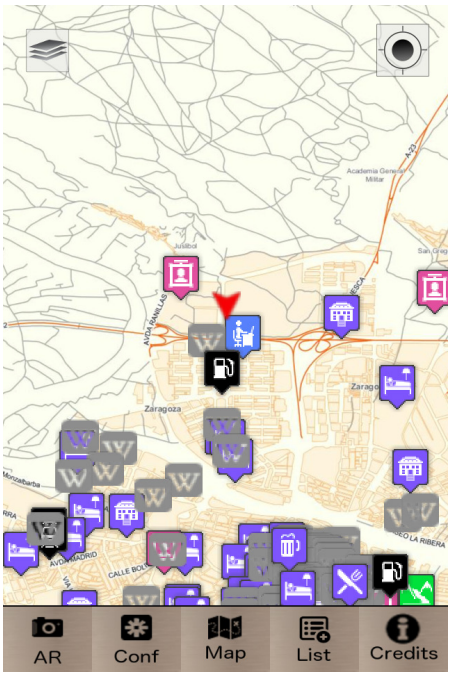


Figura 16. *Visor de mapas Cartociudad*

Así pues, estas serían las pantallas más importantes de la aplicación. Con el fin de dar una mayor riqueza y usabilidad al usuario se ha elaborado una pantalla de configuración (figs. 17 y 18) en la que éste puede modificar los diferentes parámetros y así mostrar todo el potencial de la aplicación y del framework de AR implementado:

Figura 17. *Pantalla de configuración I*

Figura 18. *Pantalla de configuración II*

Por último estaría la pantalla de créditos de la aplicación con la información del lugar de desarrollo (fig. 19):

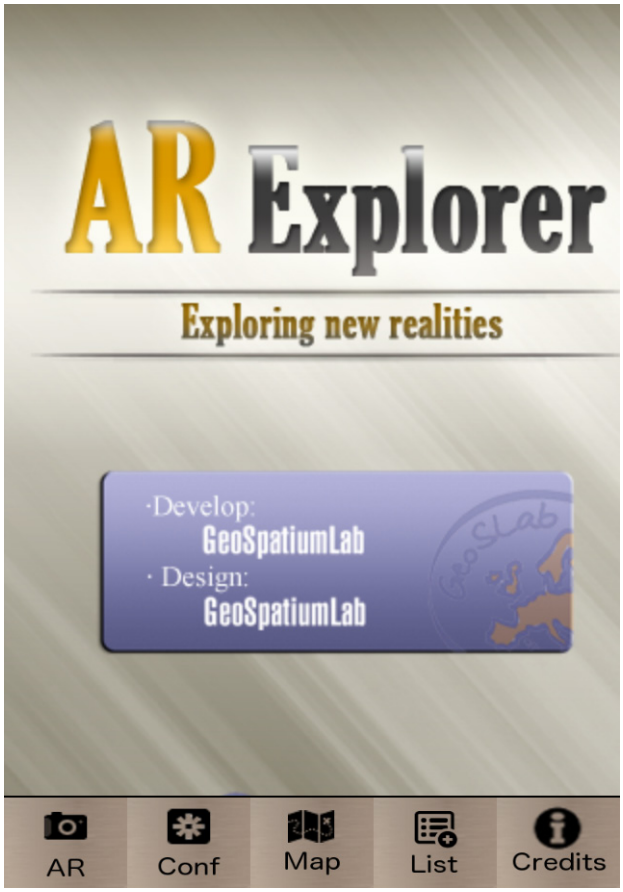


Figura 19. *Pantalla de créditos*

3. Conclusiones

En este capítulo se va a explicar las conclusiones obtenidas, algunas posibles líneas futuras del proyecto y una valoración personal del mismo.

3.1 Valoración del trabajo realizado

El proyecto ha partido del análisis en profundidad de los problemas que presentaba la realidad aumentada relacionados con la localización y la orientación, obteniendo la mejor solución posible a través del análisis y la documentación de ésta y de otras posibles con el fin de transmitirlo a proyectos futuros. De este modo, se ha desarrollado y explicado una solución, la cual sería fácilmente transferible a otras tecnologías. Por otro lado, también ha habido que tener en cuenta la limitación existente de trabajar con smartphones (aunque cada vez ésta es más pequeña) intentando aprovechar en cada momento todo su potencial para conseguir unos objetivos muy ambiciosos.

A pesar de que el objetivo principal era el de desarrollar una aplicación de realidad aumentada, se ha tratado de ser en todo momento proactivo y se han desarrollado mecanismos para que las soluciones desarrolladas en el proyecto puedan ser utilizadas en proyectos futuros. Por este motivo se han desarrollado, dentro de la aplicación, dos módulos independientes de ésta para que puedan ser reaprovechados en futuras aplicaciones. Se trata de los frameworks de realidad aumentada y de gestión de los datos.

En primer lugar, se ha creado un framework que permite representar puntos georreferenciados por medio de la realidad aumentada en dispositivos iOS, que tiene una API muy sencilla que permite ser utilizada con independencia de la aplicación y la temática de la misma.

Por otro lado, se ha desarrollado un framework de gestión de datos, el cual maneja información de tres fuentes distintas en el caso de esta aplicación. Con esto se pretende que se manejen datos lo más heterogéneos posibles y demostrar que la adición de otras fuentes de datos georefenciados de otro tipo no supondría ningún problema en el futuro en otra aplicación que utilice dicho framework. Esta solución presenta algunas limitaciones, como se presentan en las líneas futuras de desarrollo, por ejemplo con la categorización de los puntos de interés, ya que se podría mejorar utilizando otras metodologías como, por ejemplo, con el uso de algoritmos de inteligencia artificial. Por otro lado, se ha obtenido una solución que presenta una homogeneización de los datos y gestión de nuevas fuentes de datos, abstrayendo a los desarrolladores de los problemas que ello conlleva.

Por último, a modo de conclusión, se cree que los objetivos han sido satisfechos y los requisitos han sido cumplidos, analizándose todos los puntos fuertes y otros puntos débiles los cuales se pueden observar como líneas futuras del proyecto en el siguiente apartado.

3.2 Líneas futuras

Las líneas futuras de este proyecto son muy amplias y alguna de ellas muy ambiciosas. Varias son abordables a corto y medio plazo añadiendo funcionalidades al framework desarrollado. Por otro lado, existen otras más ambiciosas que se podrían llevar a cabo tomando como base el desarrollo realizado.

Algunas de las ideas podrían ser, en vez de hacer la descarga de puntos de interés, utilizar algoritmos de reconocimiento de imágenes en conjunto de la localización y la orientación del dispositivo. El problema de esta solución actualmente estaría en la capacidad hardware de los dispositivos, ya que estos algoritmos son mucho más complejos que los utilizados, pero es posible que sea abordable a medio plazo. Esta solución tendría como ventaja la precisión cuando las distancias son pequeñas, ya que éste es un punto débil del proyecto implementado, por otro lado solo serían representables puntos visibles por el usuario.

Otra solución ambiciosa sería aumentar la interacción de los usuarios con el sistema, a modo de plataforma en la que ellos mismos fueran los que introdujeran los puntos de interés y pudieran ser compartidos con el resto de usuarios a modo de red social. Con esto, se eliminaría totalmente la dependencia con servicio de terceros. No obstante, habría que tenerse en cuenta muchos otros aspectos extra, ya que podría tener un crecimiento exponencial y habría que tener algún tipo de control adicional de la veracidad y la calidad de los puntos introducidos por los usuarios.

Asociado a esta última solución, una utilidad que se podría añadir a corto plazo y no tendría una gran complejidad, sería la posibilidad de que los usuarios pudieran almacenar puntos de interés propios localmente.

Otra opción, la cual se está llevando a cabo en estos momentos en otro proyecto tomando como base éste, sería el renderizado de los puntos de interés en tres dimensiones en lugar de dos. Con esto se obtendría visualización de mayor calidad de los puntos de interés.

Una posibilidad adicional sería la del funcionamiento del proyecto en interiores ya que, actualmente, no es factible debido a la mala precisión del GPS en este medio, por tanto habría que tener otras fuentes adicionales de obtención del posicionamiento. Actualmente, existen proyectos que están tratando de abordar este problema a través de cálculos de cobertura con las antenas wifi que rodean al usuario o realizando cálculos inerciales.

Por último, otra posibilidad que se ha planteado sería la gamificación de la aplicación. Por ejemplo, por la búsqueda de puntos de interés los cuales se activaran cuando se detecta que el usuario se encuentra a una determinada distancia del punto, por medio de pistas.

Como ya se ha visto, las líneas futuras y las posibilidades son amplias y muy variadas, existiendo otras muchas como la utilización de algoritmos de inteligencia artificial para la categorización de los puntos de interés. Esto se debe a que se trata de una tecnología en continuo crecimiento.

3.3 Valoración personal

La valoración personal de este proyecto ha sido muy positiva, ya que se han utilizado y estudiado dos tecnologías con un futuro a corto y medio plazo bastante prometedor. Por un lado, la tecnología de realidad aumentada, la cual existe desde hace bastante tiempo, pero es en la actualidad cuando ha aumentado considerablemente con los smartphones y con proyectos como las Google Glass y otros distintos de otras compañías. Todo esto presenta un futuro prometedor de esta tecnología.

Por otro lado, está el uso de tecnologías móviles, en especial en el caso de iOS, en el que el aprendizaje no es sencillo, debido a las grandes necesidades de hardware y software, y por lo que estoy muy agradecido al grupo y la empresa en la que he desarrollado el proyecto, los cuales me han proporcionado en todo momento el soporte necesario y la facilidad para la asistencia a cursos de formación vinculados con el trabajo que he estado desarrollando. Del mismo modo, ésta también es una tecnología en auge en los últimos años y con un futuro alentador.

Así pues, desde el primer momento en el que se planteó el proyecto, la motivación ha sido muy alta, pese a que ha supuesto un camino largo y costoso, ya que no se había trabajado nunca con estas tecnologías y su desconocimiento era total. De todos modos, la motivación no decayó en ningún momento ya que los resultados, a nivel personal, han merecido la pena y han sido muy buenos. Ha sido muy enriquecedor el poder desarrollar este proyecto, ya que se han utilizado técnicas y metodologías profesionales que han servido como una buena transición entre el mundo académico y profesional. Además de verificarse y validarse mucho de lo aprendido a lo largo de la carrera y poder analizar mis capacidades.

Anexos

A. Gestión del proyecto

En este anexo se va a mostrar la planificación y la metodología de la gestión del proyecto que se han seguido para su elaboración, así como los diferentes recursos que han sido necesarios utilizar para la elaboración del proyecto.

A1) Metodología

En primer lugar, hay que comentar que el entorno en el que se ha desarrollado el proyecto y las tecnologías que se han utilizado en el mismo, eran totalmente nuevos para el proyectando, por tanto, se partió de una primera fase, previa a la elaboración propia del proyecto de estudio y documentación acerca de la realidad aumentada, y de algunos otros proyectos existentes que se basan en esta tecnología, como el estudio y la formación en el entorno iOS el cual, como ya se ha dicho, era totalmente nuevo.

Por otro lado, debido al desconocimiento de estas tecnologías y con el fin de desarrollar un proyecto de mayor calidad, se ha evitado el uso de ciclos de vida en cascada o similares, los cuales son mucho más rígidos y no son buenos si el entorno es nuevo para el desarrollador. Se ha llevado a cabo un desarrollo iterativo o incremental, en el cual el proyecto se ha dividido en varias partes y se les ha asignado una prioridad y un orden de elaboración y, además, éstas a su vez se han dividido en subtareas. En cada uno de estos incrementos se han ido añadiendo funcionalidades al proyecto ya que se iban cerrando algunos de los requisitos funcionales. Otro de los motivos de que se siguiera esta metodología, fue precisamente que los requisitos no estaban cerrados desde el comienzo, ya durante la evolución de proyecto se fueron añadiendo o modificando algunos de ellos, y la utilización de esta metodología facilita el trabajo en estas situaciones. Así pues, se consiguió un desarrollo mucho más ágil y a la vez, como ya se ha dicho, de mayor calidad. Todo ello está vinculado con la búsqueda de la excelencia y la calidad de los productos, hecho por el que *GeosptiumLab* ha promovido un Sistema Integrado de Gestión de la Calidad y el Medioambiente sobre la base de los requisitos de las Normas UNE-EN-ISO 9001:2008 (noviembre de 2008) y UNE-EN-ISO 14001:2004 (diciembre de 2004), para todas las actividades que desarrolla, que se complementa con los requisitos de la norma ISO/IEC 15504 “Software Process Improvement Capability Determination” (SPICE), nivel de capacidad 2, con el objetivo primordial de satisfacer a sus clientes mediante la prestación de servicios de la máxima calidad y respeto al entorno. Este Sistema Integrado de Gestión obtuvo su certificación formal en el año 2011.

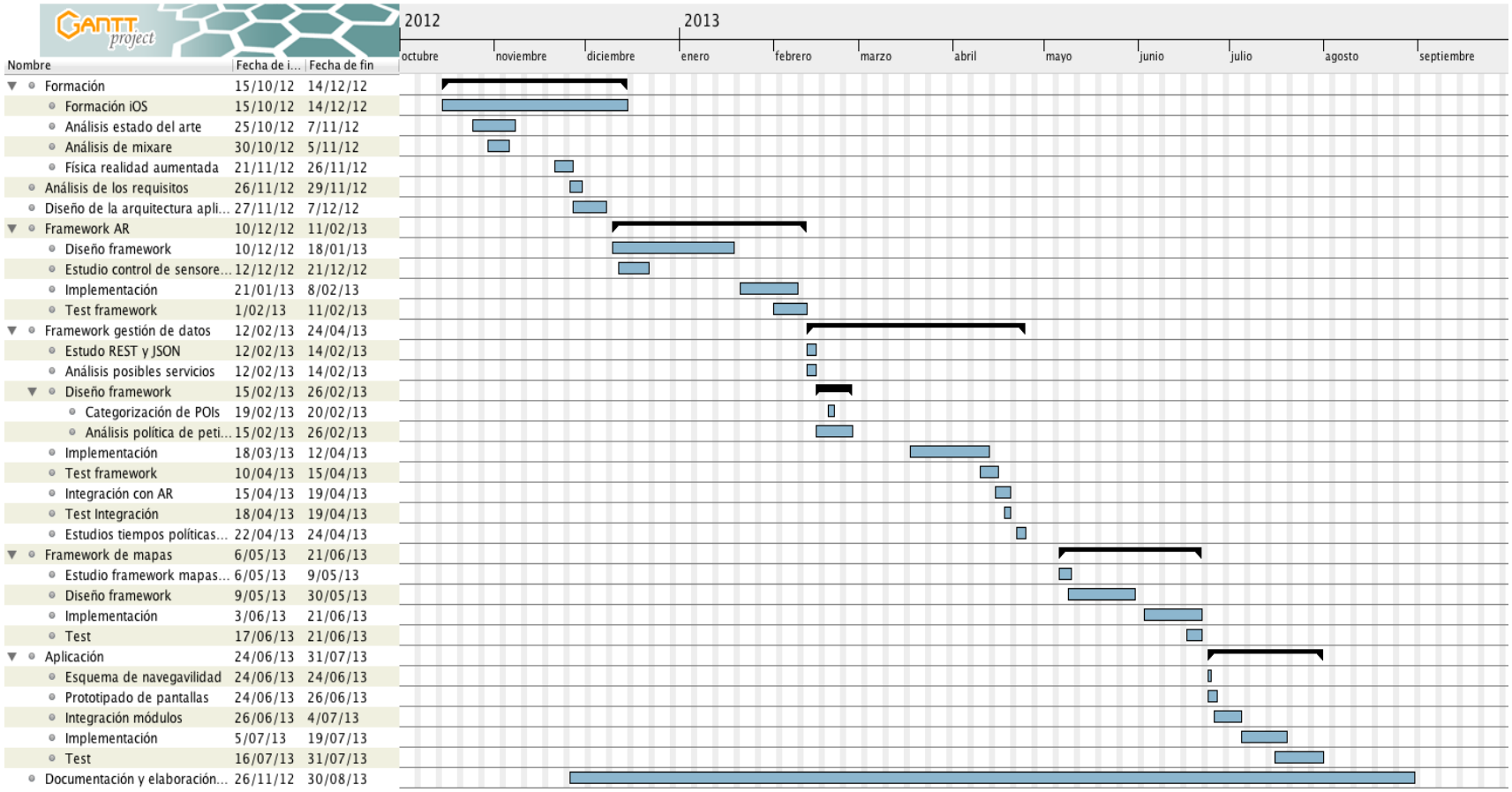
Los incrementos en que se ha dividido el proyecto por orden de prioridad son los siguientes:

- Módulo de realidad aumentada.
- Módulo de gestión de datos.
- Módulo de mapas.
- Integración de los módulos y elaboración de la aplicación de caso de uso.
- Documentación final y elaboración de la memoria del proyecto.

Todos estos incrementos a su vez se han dividido en subtareas de análisis y prototipado, implementación, integración y test, además de la elaboración de la documentación en cada una de las partes y la elaboración de una batería de pruebas de funcionalidad, independientemente de los test anteriormente comentados, para la verificación de los requisitos funcionales y no funcionales como se puede ver en el anexo K.

Cada uno de estos incrementos no se cerraban completamente ya que, como el aprendizaje también era progresivo, en algunas ocasiones se han redefinido alguna de las partes de módulos anteriores o se han solventado algunos problemas que han podido aparecer. Además, como cada uno de los incrementos era independiente entre sí, no era necesario haber terminado completamente la anterior para poder comenzar la siguiente. Con esto se ha conseguido reducir mucho los tiempos ociosos, en el sentido de que si en una de las iteraciones no se ha podido avanzar en un momento determinado (por ejemplo, en la fase de test no se disponía del dispositivo físico, iPhone, para realizarlas) se han podido comenzar tareas del siguiente incremento sin que esto afectara a la calidad y al resultado final; ya que, como se ha dicho, cada una de las iteraciones se ha tratado que fueran completamente independientes y de corta duración, ya que dentro de los módulos, cada funcionalidad que había que implementar se ha considerado una iteración distinta.

Así pues en el siguiente diagrama de Gantt se puede ver como ha evolucionado el proyecto a lo largo del tiempo y ver claramente diferenciados el desarrollo de los diferentes módulos del proyecto (fig. 20):



Nombre	Fecha de i...	Fecha de fin
Formación	15/10/12	14/12/12
Formación iOS	15/10/12	14/12/12
Análisis estado del arte	25/10/12	7/11/12
Análisis de mixare	30/10/12	5/11/12
Física realidad aumentada	21/11/12	26/11/12
Análisis de los requisitos	26/11/12	29/11/12
Diseño de la arquitectura apli...	27/11/12	7/12/12
Framework AR	10/12/12	11/02/13
Diseño framework	10/12/12	18/01/13
Estudio control de sensore...	12/12/12	21/12/12
Implementación	21/01/13	8/02/13
Test framework	1/02/13	11/02/13
Framework gestión de datos	12/02/13	24/04/13
Estudo REST y JSON	12/02/13	14/02/13
Análisis posibles servicios	12/02/13	14/02/13
Diseño framework	15/02/13	26/02/13
Categorización de POIs	19/02/13	20/02/13
Análisis política de peti...	15/02/13	26/02/13
Implementación	18/03/13	12/04/13
Test framework	10/04/13	15/04/13
Integración con AR	15/04/13	19/04/13
Test Integración	18/04/13	19/04/13
Estudios tiempos políticas...	22/04/13	24/04/13
Framework de mapas	6/05/13	21/06/13
Estudio framework mapas...	6/05/13	9/05/13
Diseño framework	9/05/13	30/05/13
Implementación	3/06/13	21/06/13
Test	17/06/13	21/06/13
Aplicación	24/06/13	31/07/13
Esquema de navegavilidad	24/06/13	24/06/13
Prototipado de pantallas	24/06/13	26/06/13
Integración módulos	26/06/13	4/07/13
Implementación	5/07/13	19/07/13
Test	16/07/13	31/07/13
Documentación y elaboración...	26/11/12	30/08/13

Figura 20. *Diagrama de Gant del proyecto*

Las tareas más costosas fueron:

- El análisis en detalle de otros proyectos de realidad aumentada.
- El control de la orientación del usuario necesario para la correcta representación de los puntos de interés.
- La visualización correcta de los puntos de interés en la realidad aumentada.
- La gestión de categorización de los puntos de interés procedentes de diferentes fuentes de datos.
- El almacenamiento en segundo plano de los datos procedentes de los servicios, en la base de datos, para mejorar la eficiencia y el tiempo de carga en la aplicación.
- Verificaciones estéticas de la aplicación final.

Con el fin de llevar a cabo el control de tiempo, tanto por parte de la empresa como por parte del proyectando, diariamente se ha llevado un control de las horas invertidas y la temática de las mismas en una tabla como la que se muestra a continuación (fig. 21). Con esto se puede verificar y controlar que los tiempos estimados se iban cumpliendo.

GP	P	F	N	Descripción de la actividad	Total	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
81	11	81	10	Formacion	11			1		3		2		2						1	2															
81	11	81	10	Problemas conexión dispositivo	11			4		4							3																			
©	11	81	10	Propuesta PFC	1				1																											
81	11	81	10	Análisis de los requisitos	1				1																											
81	11	81	10	Análisis casos de uso	5				1																			3							1	
81	11	81	10	Diseño Librería AR	19										2	2													2		3	3		3	2	2
81	11	81	10	Visor Cámara	13												4	2	2			3			2											
81	11	81	10	Posicionamiento(Localización y orientación	12																	1		5	2	4										
81	11	81	10	Hands on	1																				1											
81	11	81	10		0																															
					0																															
					0																															
				Diseño Librería AR	4		2		1	1																										
				Análisis esquema navegabilidad	2		2																													
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
					0																															
				Esfuerzos totales mes	80	0	4	5	4	8	0	2	0	2	2	2	7	2	2	1	2	4	0	5	5	4	0	3	2	0	3	3	0	3	3	2

Figura 21. *Tabla de control de esfuerzos diarios*

Como resultado de este control de esfuerzos, al finalizar el proyecto se ha elaborado un estudio con la distribución de las horas totales del proyecto en las diferentes tareas propias de un proyecto software, como se puede ver en la siguiente tabla y gráfico (fig. 22):

	HORAS	PORCENTAJE
Estudio y estado del arte	118	11,98%
Análisis	107	10,86%
Diseño	185	18,78%
Implementación	298	30,25%
Integración	101	10,25%
Test	71	7,21%
Documentación y memoria	105	10,66%
HORAS TOTALES	**985**	

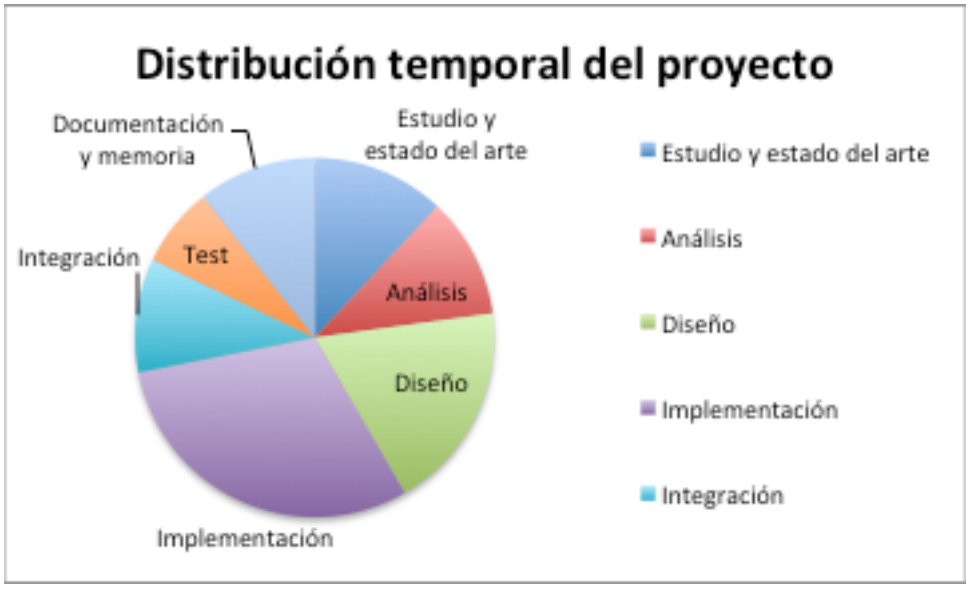


Figura 22. *Gráfico de la distribución temporal del proyecto*

A2) Recursos utilizados

En una primera fase de análisis y desarrollo se han utilizado diferentes herramientas CASE, para la elaboración de los diagramas clases, trazas de eventos, propios del paradigma de programación orientada a objetos o diagramas entidad-relación para el diseño de la base de datos, entre otros muchos que aparecen también a lo largo de la memoria. Por otro lado, también se han utilizado herramientas para la programación de tareas y el control de esfuerzos durante el proyecto; y, por último, para la realización de test también se dispuso de dispositivos iPhone para la realización de los mismos.

Así, pues, las herramientas que se han utilizado son:

Gestión y organización del proyecto:

- **Gantt Proyect**: Planificación de tareas.
- **Microsoft Office Excel**: Control de esfuerzos.
- **Bugzilla**: Asignación de tareas.
- **Dropbox**: Copias de seguridad.
- **Extensión XCode SVN**: Control de versiones.

Análisis y diseño:

- **ArgoUML**: Diagramas de clases y de despliegue.
- **VisualParadigm**: Diagramas entidad-relación base de datos, diagramas de secuencia y algunos diagramas de clases de otras arquitecturas analizadas.
- **yEd Graph Editor**: Diagrama de arquitectura.
- **Adobe Photoshop CS5**: Creación de botones.

Desarrollo y test:

- **XCode 3.2.6 y 4.5**: IDE para dispositivos iOS y Mac OS en Objective-C.
- **Simulador iPhone**: Simulador instalado junto con XCode.
- **iPhone:** Realización de pruebas.
- **Mac Mini**: Equipo necesario para el desarrollo en iOS.
- **Instruments 4.5:** Test de consumo de memoria y cpu.
- **Microsoft Office Excel**: Análisis de los datos obtenidos del análisis del rendimiento de la gestión de datos en el framework desarrollado.

Documentación y presentación:

- **Microsoft Office Word**: Elaboración de la documentación y la memoria del proyecto.
- **Adobe InDesign CS5**: Maquetación de la documentación y la memoria.
- **Microsoft Office Powerpoint**: Presentación del proyecto.

B) Análisis de los requisitos

B1) Requisitos funcionales

- **RF 0:** La aplicación poseerá tres modos de operación:
  - Modo cámara o de realidad aumentada: El usuario a través de la cámara y con la ayuda de otros sensores como el acelerómetro, puede observar los POIs que le rodean y acceder a la información asociada a ellos.
  - Modo mapa: Mostrará los elementos geolocalizados en un mapa alrededor a la posición del usuario el cual también estará representado por medio de una brújula que representará su orientación, permitiendo el zoom de las diferentes zonas así como la interacción con los POIs.
  - Modo lista: Los POIs de interés que rodean al usuario se presentan en forma de lista ordenados de mayor a menor, permitiendo el acceso directo a los contenidos asociados a cada uno de ellos.
- **RF 1:** Debe tener acceso a servicios externos y propios de información georreferenciada.
- **RF 2:** Representación de objetos virtuales y renderización de la imagen en tiempo real en dos dimensiones.
- **RF 3:** En la interacción con los POIs en el caso del modo cámara, al acceder a la información de uno de los puntos, parecerá una brújula orientada hacia el punto accedido el cual será señalado.
- **RF 4:** Cálculo de visibilidad de los objetos en función de la distancia y la orientación en la que se encuentre el POI.
- **RF 5:** Debe permitir la localización y posicionamiento del usuario en exteriores.
- **RF 6:** Poseerá de modos de funcionamiento on-line haciendo uso de los servicios web para proveer POIs, modo off-line almacenando la información entre diferentes ejecuciones o en mixto de acceso a la información a modo de cache de datos.
- **RF 7:** Se permitirá la categorización de los POIs, por tanto se podrá filtrar los resultados en función de su categoría, en cualquiera de los tres modos de operación disponibles.
- **RF 8:** Se debe permitir la interacción con los objetos virtuales representados; al pulsar sobre uno de ellos se mostrará su información asociada.
- **RF 9:** El radio del alcance de observación de los POIs podrá ser modificado por el usuario y la representación se verá afectada en los tres modos de operación.
- **RF 10:** Navegabilidad y usabilidad. Existirá una pantalla asociada a cada uno de los modos de funcionamiento y otra asociada a las preferencias de la aplicación, todas ellas accesibles en todo momento a través de un menú.

B2) Requisitos no funcionales

- **RNF 0:** El dispositivo tiene que disponer de cámara digital, sensor magnético y acelerómetro.
- **RNF 2:** Será necesario estar conectado a internet (Wi-Fi o 3G) para el funcionamiento en modo mapa, debido a que los servicios de mapas utilizados solo funcionan correctamente en modo online.
- **RNF 4:** El idioma de la aplicación será en inglés y castellano.
- **RNF 5:** El dispositivo para el que va a ser implementado deberá tener sistema operativo iOS.
- **RNF 6:** Los frameworks y la aplicación serán compatibles a partir de la versión 4 de iOS y posteriores.
- **RNF 7:** La localización del usuario se hará a través del dispositivo GPS, Wi-Fi y 3G.

B3) Casos de uso

En el desarrollo del PFC se ha especificado que se iba a implementar no solo una aplicación de realidad aumentada, sino también un framework que permitiera la implementación de la misma en aplicaciones futuras, por lo que se ha diseñado un diagrama de casos de uso (fig. 23) con las tareas que van a poder realizar de manera muy general los dos principales agentes del sistema, el usuario y el desarrollador que haga uso del framework, siendo este último una especialización del primero. En el diagrama se ha desarrollado especialmente la parte de realidad aumentada y gestión de datos que ha sido en la que se ha centrado el proyecto. La parte de mapas no ha sido desarrollada tan ampliamente, ya que *GeoSpatiumLab* empresa donde se desarrolla este PFC, posee motores y visualizadores de mapas propios y, por tanto, no se ha centrado en este aspecto.

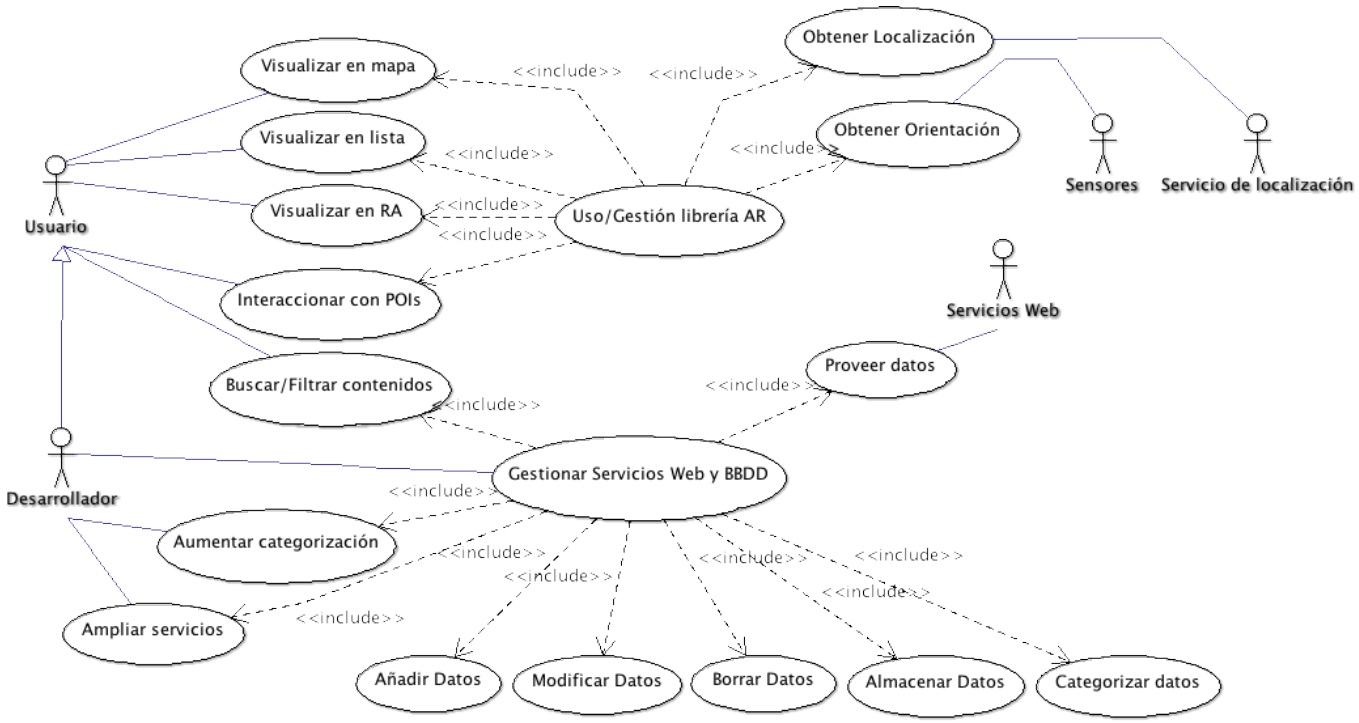


Figura 23. *Diagrama de casos de uso de la arquitectura*

C) Análisis del estado del arte

En primer lugar se ha procedido a la elaboración del estado del arte acerca de la realidad aumentada, es decir, un resumen del estado actual de la tecnología a desarrollar.

Se han analizado, en la medida de lo posible, diversos proyectos en el ámbito de la realidad aumentada, más concretamente para la plataforma iOS. Con todo esto se pretende tener una visión global de la tecnología en proyectos ya existentes, con el fin de aportar diferentes ideas y de tratar de detectar necesidades que no habían sido tratadas anteriormente, comprendiendo mejor el problema que se está tratando de afrontar.

C1) Definición de realidad aumentada

La realidad aumentada consiste en una línea de investigación que trata de aunar la presencia de objetos del mundo real con objetos virtuales en tiempo real. El término aumentada se refiere precisamente a que se trata de aumentar la percepción de la realidad añadiendo información adicional generada por ordenador.

Actualmente habría dos definiciones mayoritariamente aceptadas:

Una de ellas sería dada por Ronald Azuma en 1997, en la que afirma que la realidad aumentada combina elementos reales y virtuales, debe ser en tiempo real y estar registrada en 3 dimensiones.

Por otro lado, Paul Milgram y Fumio Kishino definieron en 1994 la realidad de Milgram-Virtuality Contimuum (fig. 24), como un continuo que abarca desde el entorno real a un entorno virtual puro. En medio se encontraría la realidad aumentada (más cerca del entorno real) y la realidad virtual (más cerca del mundo virtual).

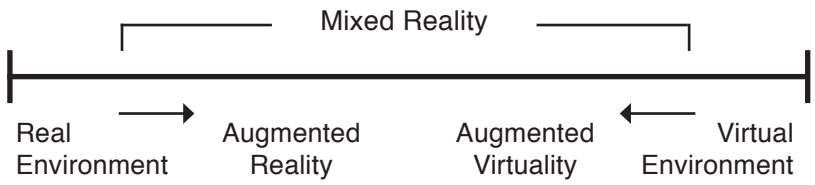


Figura 24. *Modelo de Paul Milgram y Fumio Kishino*

Toda esta tecnología tiene su aplicación en diferentes ámbitos, como son: la simulación, el entretenimiento, la cirugía, servicios de emergencias y militares, arquitectura, aplicaciones industriales o turismo, entre otras diferentes.

C2) Métodos de registros de información

Usando marcadores

Un marcador es un patrón en forma de imagen que se imprime para poder situar un objeto 3D en la realidad aumentada.

El software se basa en el reconocimiento de estos objetos para determinar que objeto 3D se representa y su posición.

Una de las ventajas del uso de marcadores es que permite situar con mucha más exactitud el objeto 3D que los otros métodos. Sin embargo, el principal inconveniente que surge al utilizar este método es que se necesita tener la imagen impresa en alguna superficie para poder ver el objeto 3D, y si la imagen no se reconoce, ya sea porque esté oculta una parte o porque esté deteriorada, no se mostrará nada. Este inconveniente hace que no sea un método adecuado en el contexto en el que se está trabajando.

Usando reconocimiento de objetos

Esta opción es la más actual y se basa en el reconocimiento de objetos conocidos, por ejemplo, el de la cara de una persona o el de la forma de un edificio u objeto.

El principal inconveniente de este método es su gran coste computacional y de almacenamiento de información, esto es debido a la necesidad de reconocer la forma del objeto, ya que en el proceso se tendría que cotejar las formas de la imagen con una base de datos, y cuantos más objetos se quieran reconocer, más grande será la base de datos.

Usando la posición GPS

Otra opción que existe es situar los objetos en una cierta posición GPS, y que al aproximarte a ella puedas ver los objetos a través del dispositivo.

El inconveniente de este método es la menor precisión con respecto a los métodos anteriores, debido a la precisión del sistema GPS. Sin embargo con este sistema, los inconvenientes que se veían en los métodos anteriores desaparecen, ya que ya no es necesario la existencia de ningún marcador físico en la realidad y además el coste computacional y de almacenamiento es considerablemente menor que en el caso del reconocimiento de objetos y más en el contexto en el que está trabajando en el que se maneja mucha información.

C3) Análisis frameworks realidad aumentada para iOS

Vuforia

Vuforia es un framework de realidad aumentada basada en marcadores, que permite crear imágenes tanto en 2D como en 3D a partir de los mismos. Es soportado por iOS, Android y Unity3D. Al estar soportado por este último se pueden desarrollar apps en las que el mismo código fuente

puede servir tanto para iOS como para Android. El desarrollo se realiza en Java (Android), Objetive-C (iOS) o en algún lenguaje que soporte .NET si se desarrolla sobre Unity3D.

Layar (fig. 25)

Se trata de un navegador de realidad aumentada de pago, por lo que no es posible, como en el caso anterior, acceder a su código fuente. Permite su integración en aplicaciones nativas a través de Layar Player SDK disponible para iOS, pudiendo añadir POIs por parte de los usuarios/desarrolladores. Al ser de pago no se dispone de mucha información. Un aspecto que se ha podido ver a través de su uso, es que usando una aplicación que hacía uso de este framework se precisaba la instalación de la app de Layar, algo no muy deseable más aún siendo que la versión completa de la app es de pago (aunque también hay una versión gratuita).

La idea de esto último reside en que Layar está basado en un sistema de capas. Por si solo no ofrece información, lo que lo hace son las capas que son añadidas por los desarrolladores. Así, pues, Layar solo sería un mapa en blanco y al abrir una capa se muestra información adicional sobre el mismo.

Debido a que estas capas son creadas por los diferentes desarrolladores, no tienen una interfaz común, véase ejemplo de Layar con una capa que muestra información acerca de los artículos de Wikipedia de elementos que se encuentran a nuestro alrededor.

Figura 25. *Ejemplo sobre Layar*

Mixare (figs. 26 y 27)

Consiste en un framework de realidad aumentada, basado en el posicionamiento GPS (modelo que en principio era el que mejor se adapta a las necesidades). Se trata de un desarrollo de código abierto con licencia GPLv3, por lo que es posible modificar cualquier sección de su código fuente para ajustarlo a las necesidades específicas, lo que nos aporta mucha más información. Está disponible tanto para iOS como para Android. Como inconveniente está la imposibilidad de modo offline y la falta de categorización de los puntos de interés.

En cuanto a la interfaz, como se puede ver en las imágenes, resulta muy sencilla; en ella los puntos de interés se reflejan con una simple circunferencia y una breve descripción bajo ésta en la que pulsando sobre ella, abre una pantalla del navegador que nos dirige a la fuente original con la información más extendida del punto de interés. También es posible modificar el radio de búsqueda, así como las fuentes de información de estos puntos. Esto sería con respecto a la parte de realidad aumentada, pero también existe la posibilidad de mostrar los puntos directamente sobre un mapa.

En este caso, el problema comentado de la visibilidad si que se ha tenido en cuenta, ya que como vemos, los puntos de interés se localizan en diferentes posiciones de la pantalla (diferentes alturas) en función de la distancia, colocando más bajos los más cercanos y así dar la sensación de profundidad.

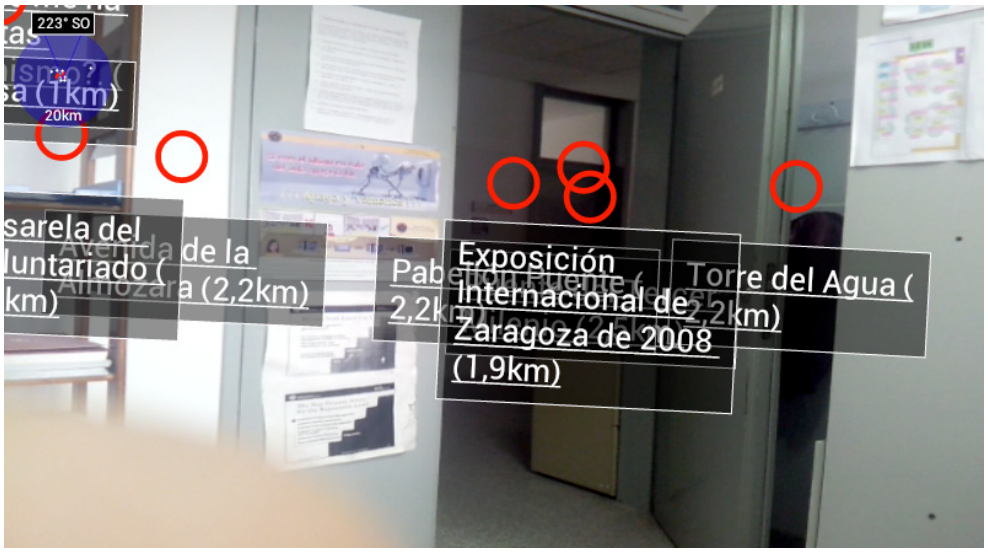


Figura 26. *Ejemplo de aplicación realizada con Mixare*

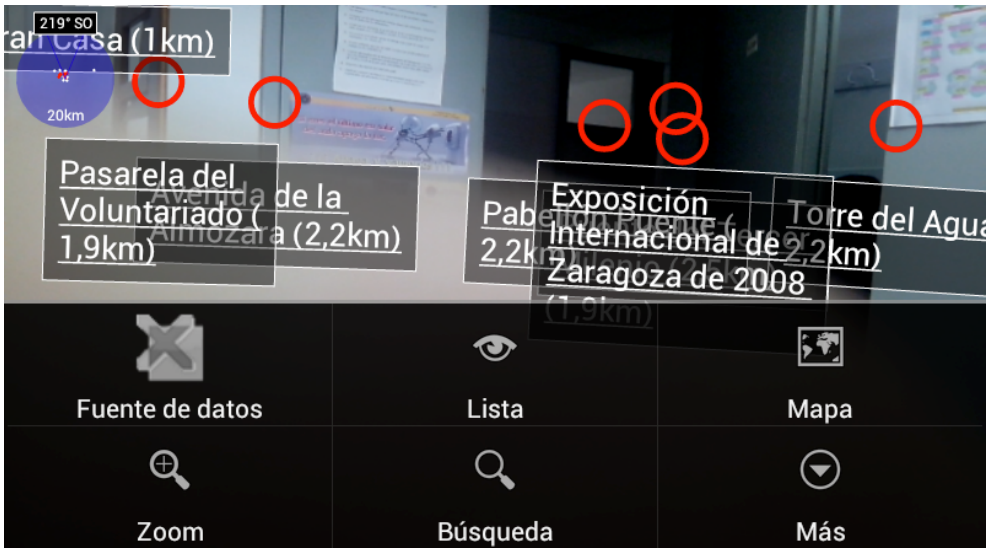


Figura 27. *Menú de aplicación realizada con Mixare*

Wikitude (fig. 28)

Se trata de un framework desarrollado por una empresa miembro del Open Geospatial Consortium.

Entre las características se encuentran:

- Desarrollo en HTML, JavaScript y CSS (es necesario tener instalado el plugin PhoneGap).
- Permite el desarrollo tanto para iOS como para Android.
- Existe la posibilidad de combinarlo con el Framework Vuforia SDK de Qualcomm para el reconocimiento de imágenes.
- Posee modo offline.
- Posibilidad de añadir múltiples objetos geolocalizados.
- Localización absoluta o relativa al usuario.
- Bien documentado.
- Framework de pago (ofrece una versión de prueba que no ofrece la posibilidad de desarrollar aplicaciones comerciales).

En este caso, se puede ver una interfaz muy similar al anterior, solo que algo más elaborada en el sentido de que también ofrece información acerca de la fuente de datos desde la que se está obteniendo la información mostrando su logotipo. El resto de funcionalidades es similar al caso anterior solo que éste permite seleccionar la temática de los puntos de interés que se van a mostrar.

En este framework, en función de las pruebas realizadas con el mismo, se deduce que el problema que se ha comentado de la visibilidad no se tiene en cuenta, ya que la representación de los puntos de interés se realizan todos sobre la línea del horizonte con independencia de la distancia a la que se encuentren, pudiendo quedar muchos de ellos superpuestos.

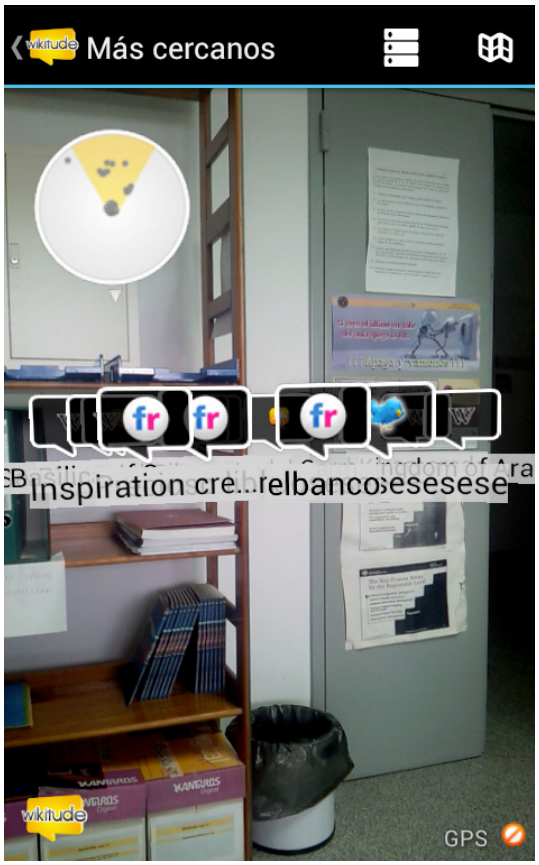


Figura 28. *Ejemplo realizado con Wikitude*

C4) Análisis framework realidad aumentada para otras plataformas

En este apartado se van a analizar frameworks y visores para otras plataformas, los cuales pese a que no van a poder ser utilizados en iOS, si que pueden servir como modelos para la obtención de posibles ideas e inspiración para el desarrollo de un framework propio.

NyARToolKit

Se trata de un framework de realidad aumentada basada en marcadores que surgió a partir de una primera librería llamada ARtoolKit para el desarrollo de aplicaciones para ordenadores usando la cámara web de los mismos. La principal dificultad que tiene este framework y todos aquellos basados en marcadores es localizar los marcadores físicos con respecto a la vista del usuario en tiempo real, para lo cual utiliza algoritmos de visión por computador.

NyARtoolKit está disponible con licencia comercial, pero también puede descargarse gratuitamente bajo licencia GPLv2.

ARviewer

Es un framework de realidad aumentada basada en la geolocalización, fácilmente integrable en aplicaciones propias. La ventaja con respecto a otros frameworks de este tipo es que funciona tanto en exteriores como en interiores y permite a los usuarios añadir nuevos POIs, editándolos y cargar el resultado en el servidor desde el propio dispositivo, permitiendo que otros usuarios los vean. El inconveniente de este proyecto es la carencia de funcionamiento offline, y la inexistencia de categorización de los POIs.

LookAR

Este framework ha sido creado como proyecto de final de carrera en la Universidad Complutense de Madrid, que permite la localización en interiores por medio de señales wifi y localización inercial, es decir con sistema de coordenadas relativo al espacio que rodea al usuario. Como se puede ver, se trata de un proyecto enfocado a la localización en interiores, algo que no es el objetivo de este PFC.

Nokia City Lens (fig. 29)

Se trata de una aplicación de Nokia para sus modelos Lumia con Windows Phone 8, en la que la información es mostrada de tres maneras distintas:

Vista Listado: Al girar el teléfono en modo vertical, se ve un listado de los POIs ordenados desde el más cercano al más lejano, junto a ellos se muestra un flecha que indica la orientación de los mismos.

Vista Mapa: Al posicionar el teléfono en modo horizontal como si se estuviera leyendo, se muestra un mapa con los POIs que rodean al usuario.

Vista RA: Al orientarlo en posición horizontal, tendríamos el módulo de realidad aumentada basado en geoposicionamiento con los POIs más próximos.

Dichos POIs pueden ser compartidos con otros usuarios por medio de email, sms o por redes sociales.

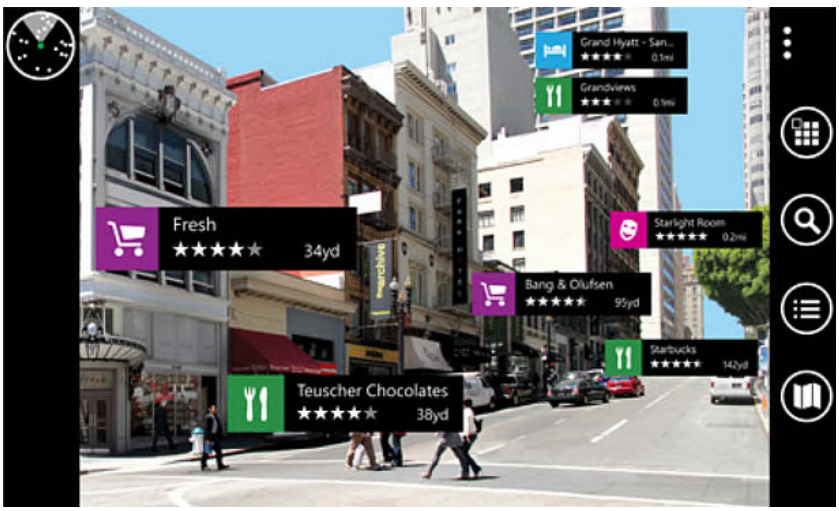


Figura 29. *Nokia City Lens*

D) Análisis de Mixare

A lo largo del desarrollo del proyecto, se ha ido realizando un seguimiento constante del estado del arte y, más en concreto, de Mixare, ya que se trata de un proyecto de código abierto, del que se han tomado algunas ideas aplicables al proyecto desarrollado, siendo por otro lado independientes uno del otro. Este apartado se ha organizado de forma que se ha elaborado un primer análisis completo y posteriormente, al tratarse de un proyecto en desarrollo, se han ido añadiendo las modificaciones que se han ido observando.

D1) Análisis versión 0.9

Es la primera versión que se ha analizado cuando se comenzó el proyecto, siendo ésta la última disponible.

En primer lugar se muestra una imagen del diagrama completo de todo el framework desarrollado por Mixare (fig. 30):

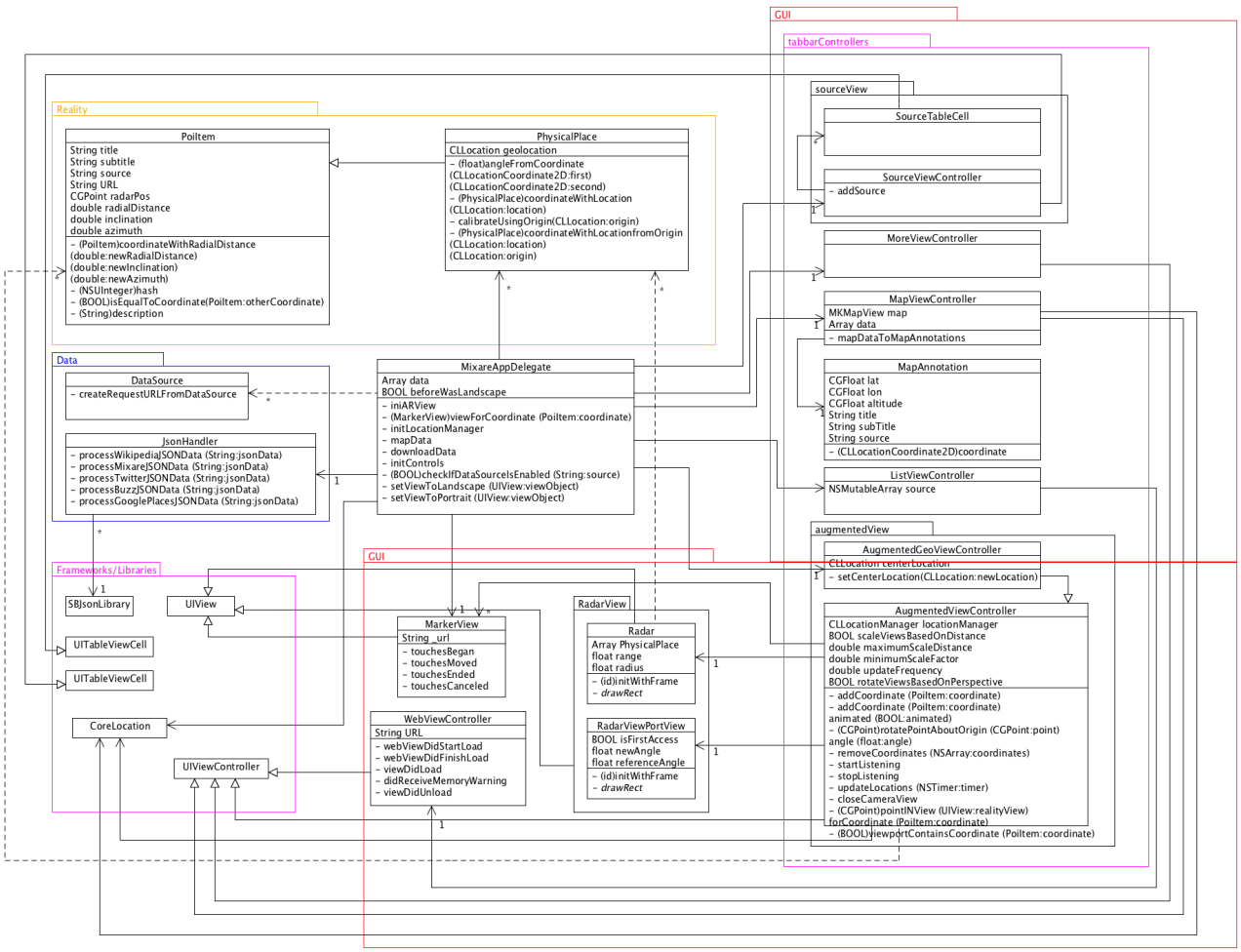


Figura 30. *Diagrama de clases completo de Mixare*

Analizando el diagrama de clases del framework Mixare se puede observar que está dividido en tres partes bien diferenciadas, las cuales son llamadas: Data, Reality y GUI. Sus nombres nos dan una idea de los aspectos de los que se ocupa en cada una de ellas. A continuación se va a proceder a explicarlas más en profundidad.

Reality (fig. 31)

Es la parte más pequeña en cuanto a número de clases pero no menos importante. Se encarga de toda la parte de la representación de los puntos de interés, y de todo lo relacionado con la física que se comentó en el estado del arte, acerca de la correcta localización en la pantalla de estos puntos en función de la distancia a la que se encuentre el observador, para así dar sensación de profundidad en un medio 2D.

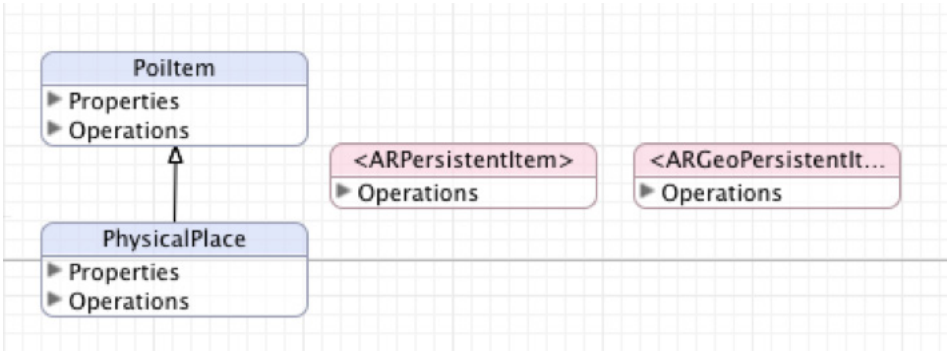


Figura 31. *Diagrama de clases módulo Reality*

Tenemos dos clases:

PoiItem: Esta clase se encarga de representar un POI con toda la información necesaria, como es: el título, el subtítulo, su coordenada, la fuente desde donde ha sido obtenido, su url, la distancia radial desde el dispositivo, la inclinación y los grados acimut.

PhysicalPlace: Esta clase hereda de la anterior y es usada para representar un lugar físico en un mundo virtual como es el dispositivo. Esto quiere decir que proporciona métodos que implementan toda la física comentada anteriormente acerca de la correcta visualización de los puntos de interés para dar una sensación de profundidad.

Data (fig. 32)

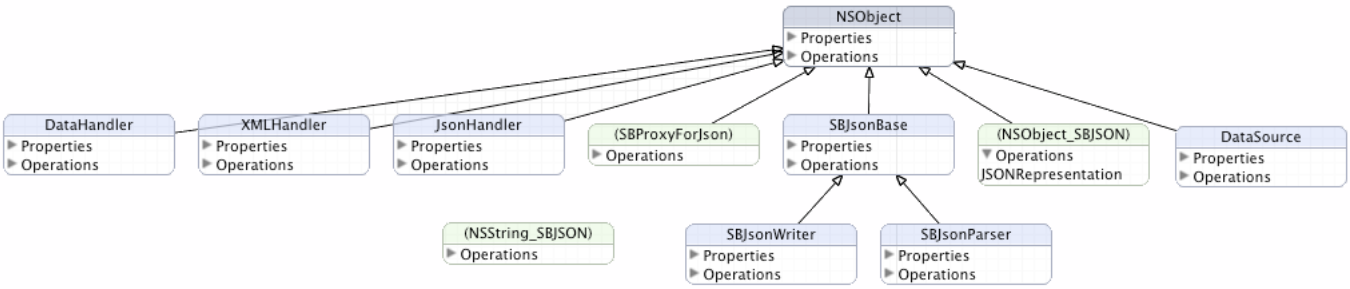


Figura 32. *Diagrama de clases módulo Data*

Esta parte se encarga del tratamiento y la petición de los datos de los POI necesarios en cada momento. Sus clases más importantes son:

DataSource: Se encarga de crear los strings de las urls con las peticiones a los diferentes servidores de los puntos de interés que se han definido, que son Wikipedia, Buzz, Twitter, Open Street Maps y el servidor propio de Mixare, a partir de la latitud, la longitud, la altitud entre otros parámetros del POI que se quiere obtener.

El tráfico y el tratamiento de información recibida se realiza en formato *JSON*, por lo que serán las clases que se van a explicar a continuación. También existen declaradas otras clases que se encargarían del tratamiento de información en formato XML u otro formato (**XmlHandler** y **DataHandler**); pero, como se ha dicho, solo están declaradas, no implementadas, además, la información es solicitada en formato *JSON*.

JsonHandler: Se encarga del procesamiento de la información en formato *JSON* recibida desde los diferentes servidores citados anteriormente. Para ello se ayuda de las clases **SBJsonBase**, **SBJsonWriter** y **SBJonParser** que no son propias del framework Mixare sino que pertenecen al framework Json-framework en Objetive-C que permite un tratamiento más sencillo de la información que se recibe en ese formato.

GUI (fig. 33)

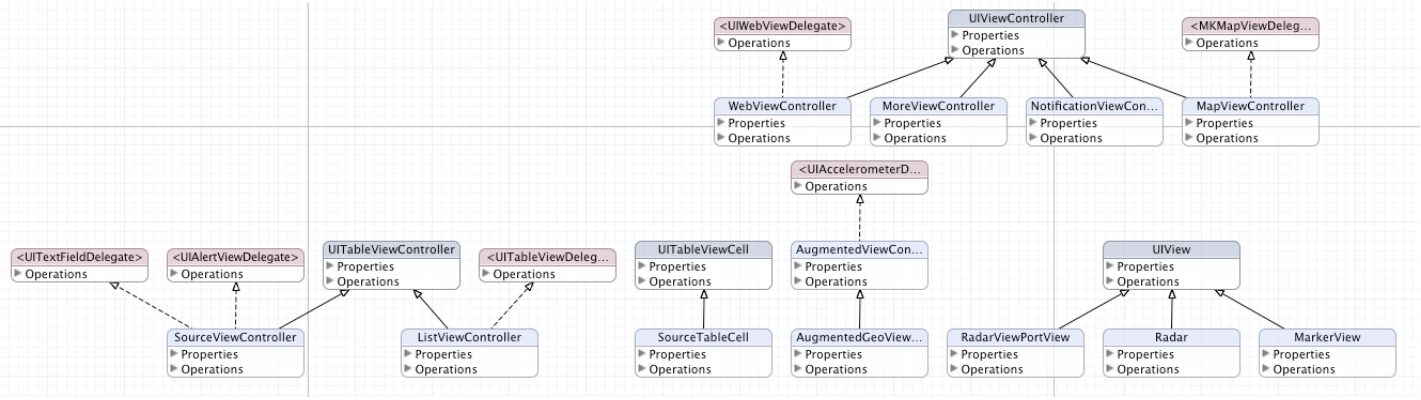


Figura 33. *Diagrama de clases módulo GUI*

En esta parte se encuentran todas las clases relacionadas con el manejo de la GUI, las cuales se van a explicar por separado y detalladamente las más importantes.

MarkerView: Esta clase se encarga de mostrar la información de un punto de interés cuando es seleccionado, mediante la apertura de un webview. Lo que hace es tomar la url del POI seleccionado y como se ha comentado antes, es un parámetro del POI y lo abre a través del webview mencionado.

WebViewController: Se trata del controlador del webview creado por la clase anterior para mostrar la información de un POI deseado.

Radar y RadarViewPortView: Estas clases se encargan del control y la representación del radar que aparece en la esquina superior izquierda cuando se está visualizando los POIs con realidad aumentada. Se tiene en cuenta la orientación del dispositivo y la distancia real al punto de interés, para así representarse como si de un radar real se tratara y mostrar solo los POIs que se encuentren en un determinado radio, el cual puede ser modificado en función de las necesidades.

ListViewController: Se trata del controlador de la vista en el caso de que los puntos de interés se quieran representar en forma de lista, en lugar de en un mapa o por medio de la realidad aumentada.

MapViewController: También se trata de un controlador de la vista solo que en este caso administra la representación de los puntos de interés en un mapa.

MoreViewController: Esta clase es propia de la aplicación, no da ninguna utilidad extra al framework, solo se utiliza como controlador para mostrar información a la licencia del producto o enlaces a la página web de Mixare.

SourceTableCell: Es una pequeña clase que únicamente se utiliza para redefinir la clase por defecto UITablaViewCell de modo que tenga un logo y un label más grande que el de por definición.

SourceViewController: Se trata de la clase controladora que se encarga de la vista en la que figura una lista con los diferentes servidores que se pueden seleccionar para obtener la información de los POIs (Wikipedia, Twitter, OpenStreetMap y el servidor de Mixare).

AugmentedViewController: Sería una de las clases más importante en la parte de realidad aumentada, ya que es la encargada del control de toda ésta. Utiliza para ello la información de la cámara, del acelerómetro y del GPS, así como de otras clases que se han explicado anteriormente, como son Radar (para el radar que aparece en la esquina superior izquierda) y PoiItem (para representar los puntos de interés correctamente, en función de la orientación del móvil y de que se encuentre a una distancia determinada (la cual es fácilmente modificable) de los puntos de interés y que solo muestre estos últimos.

En esta clase se encuentra la parte gráfica para la correcta colocación en la pantalla de los puntos de interés para dar la sensación de profundidad gracias a los métodos físicos explicados en el estado del arte.

AugmentedGeoViewController: Se trata de una clase que hereda de la anterior, la cual tiene toda su funcionalidad solo que también añade la localización actual del dispositivo.

D2) Revisión del análisis

Posteriormente, durante la realización de este proyecto se liberó una nueva reléase del visor, por lo que se procedió a analizar los cambios que habían surgido en el mismo.

Junto con el código también se añadió un diagrama de clases simplificado (fig. 34) con las líneas de desarrollo presentes en este reléase y con ideas futuras, como se muestra a continuación:

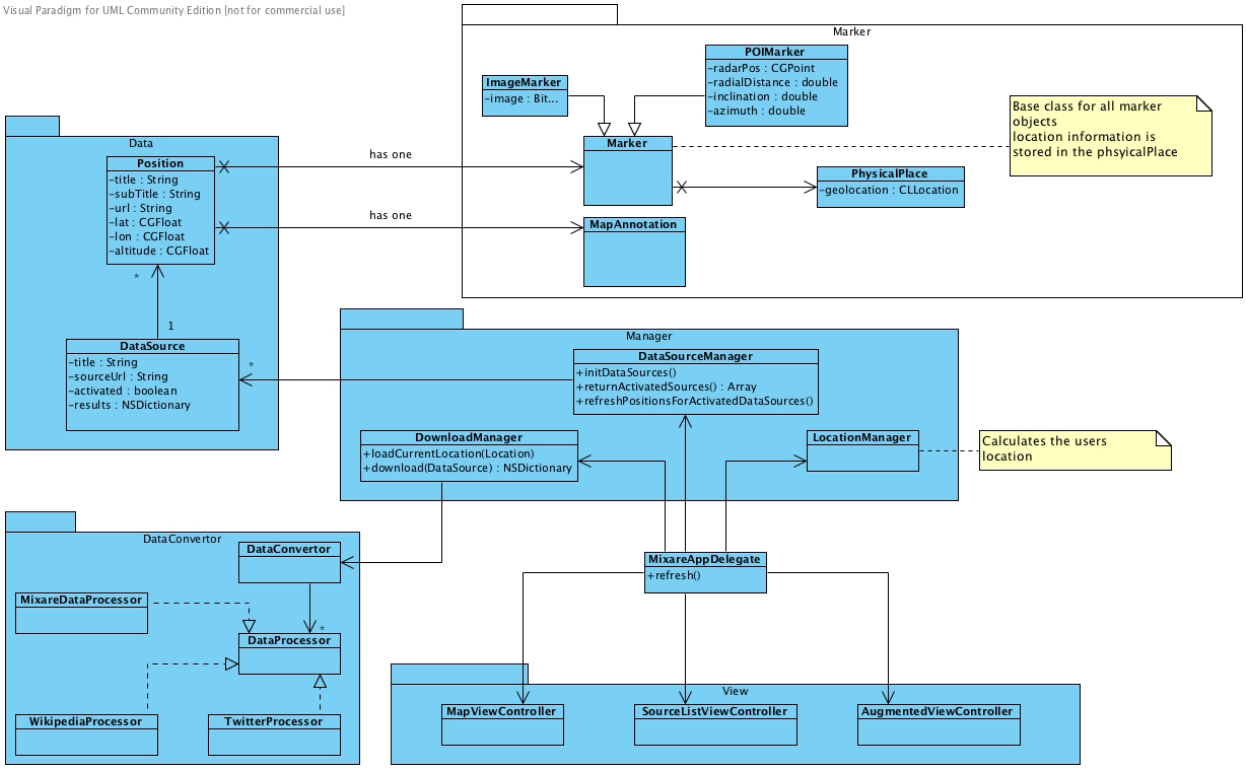


Figura 34. *Diagrama de clases simplificado de la nueva versión*

Al ver el diagrama anterior se pueden observar básicamente dos cambios; hay dos partes nuevas, por un lado el módulo DataConvertor, que contiene las clases de tratamiento de datos, los cuales en la versión anterior se encontraban dentro de Data, relacionadas con *JSON* y algunas clases nuevas que se analizarán más en detalle. Por otro lado, estaría Manager, que se encarga de la obtención de los datos de los servidores a través de las clases de DataConvertor y Data.

Así pues, se va a realizar un análisis, más detalladamente de estas dos nuevas partes. Como se puede ver, los cambios de esta nueva versión no están vinculados a la realidad aumentada, es decir, al modo en que los POIs son mostrados por pantalla y el modo de calcular su orientación y demás parámetros. Toda esta parte fue analizada en las dos versiones y no había cambiado. Los cambios están orientados hacia el modo en el que se obtiene la información. En esta versión se centra en la obtención de datos en formato *JSON*, las clases para datos XML ya no están ni declaradas, sino que han sido eliminadas.

DataConvertor

Este módulo, como ya se ha comentado antes, se encarga del tratamiento de los datos que llegan en formato *JSON*. Siguen estando las clases pertenecientes al *JSON* framework, pero además se han creado clases para el tratamiento de datos procedentes de los diferentes servidores de los que se obtiene la información: Twitter, Wikipedia, Google y un servidor propio de Mixare. Para cada una de estas fuentes se ha creado una clase diferente, pero la idea de todas ellas es la misma (está la clase genérica DataProcessor que es heredada por todos ellos).

En estas clases se encargan básicamente de convertir los datos en formato *JSON* al tipo NSMutableArray, para poder ser manejados por el framework y de obtener los diferentes campos de información que obtenemos de los servidores, los cuales pueden cambiar de unos a otros.

Manager

Este módulo tiene dos clases: DownloadManager y DataSourceManager. La primera se encarga del control de la descarga de la información necesaria en función de la posición y del radio de búsqueda que el usuario haya seleccionado, al comienzo y cuando sea necesario debido a que estos valores hayan cambiado. Para ello se sirve del otro módulo nuevo DataConvertor.

La segunda de las clases simplemente se encarga de controlar la conexión con los diferentes servidores que el usuario ha seleccionado para obtener la información de los POIs.

D3) Conclusiones

Como resultado del análisis del framework de Mixare más en profundidad a nivel de implementación, se deduce que se han sacado ideas bastante buenas de cómo llevar a la práctica los diferentes aspectos de la realidad aumentada y así conocerlo un poco más en detalle.

Sin embargo, creo que se trata de un desarrollo demasiado pensado para la aplicación que los propios desarrolladores de Mixare han implementado y que, por otro lado, sería un poco complicado reutilizarlo para una aplicación propia, ya que hay diferentes aspectos que están muy relacionados entre la aplicación y el framework. Se podría decir que no está clara la separación entre lo que sería la aplicación y el framework, simplemente sería un visor de realidad aumentada.

Así pues, este estudio ha servido para obtener diferentes ideas acerca de cómo implementar la realidad aumentada que pueden servir como inspiración de un framework propio.

E) Física realidad aumentada

En este apartado se van a presentar los diferentes problemas y conceptos físicos presentes en la realidad aumentada, así como la algoritmia creada para la misma, dado que se trata de un problema muy estudiado.

En primer lugar se van a describir algunos conceptos relacionados.

E1) Coordenadas geográficas

Se trata de un sistema de referencia que utiliza las dos coordenadas angulares, latitud (norte y sur) y longitud (este y oeste) y sirve para determinar los ángulos laterales de la superficie terrestre (o en general de un círculo o un esferoide). Estas dos coordenadas angulares medidas desde el centro de la Tierra son de un sistema de coordenadas esféricas que están alineadas con su eje de rotación.

La latitud mide el ángulo entre cualquier punto y el ecuador. Las líneas de latitud se denominan paralelos. La latitud es el ángulo que existe entre un punto cualquiera y el Ecuador, medida sobre el meridiano que pasa por dicho punto. La distancia en km a la que equivale un grado depende de la latitud, a medida que la latitud aumenta disminuyen los kilómetros por grado.

La longitud mide el ángulo a lo largo del ecuador desde cualquier punto de la Tierra. Se acepta que Greenwich en Londres es la longitud 0 en la mayoría de las sociedades modernas. Las líneas de longitud son círculos máximos que pasan por los polos y se llaman meridianos. Para los meridianos, sabiendo que junto con sus correspondientes antimeridianos se forman circunferencias de 40.007 km de longitud, 1° equivale a 111,131 km.

Combinando estos dos ángulos, se puede expresar la posición de cualquier punto de la superficie de la Tierra.

Junto con estos conceptos están los meridianos y los paralelos. Los meridianos son los semicírculos máximos del geoide terrestre que pasan por los polos, líneas imaginarias que sirven para determinar la hora. Por extensión, son también los semicírculos máximos que pasan por los polos de cualquier esfera o esferoide de referencia. Por otro lado estarían los paralelos, es decir, los círculos formados por la intersección de la esfera terrestre con un plano imaginario perpendicular al eje de rotación de la Tierra.

E2) Campo magnético terrestre (fig. 35)

Este concepto también es conocido como campo geomagnético. Se trata de un campo magnético que se extiende desde el núcleo interno de la Tierra hasta su confluencia con el viento solar, una corriente de partículas de alta energía que emana del Sol. Es aproximadamente el campo de un dipolo magnético inclinado en un ángulo de 11 grados con respecto a la rotación del eje, como si hubiera un imán colocado en ese ángulo en el centro de la Tierra. Sin embargo, a diferencia del campo de un imán de barra, el campo de la Tierra cambia con el tiempo, porque

en realidad es generado por el movimiento de las aleaciones de hierro fundido en el núcleo externo de la Tierra (la geodinámica). Pese a esto, estas variaciones son lo suficientemente pequeñas y lentas como para que la brújula sea útil para la navegación.

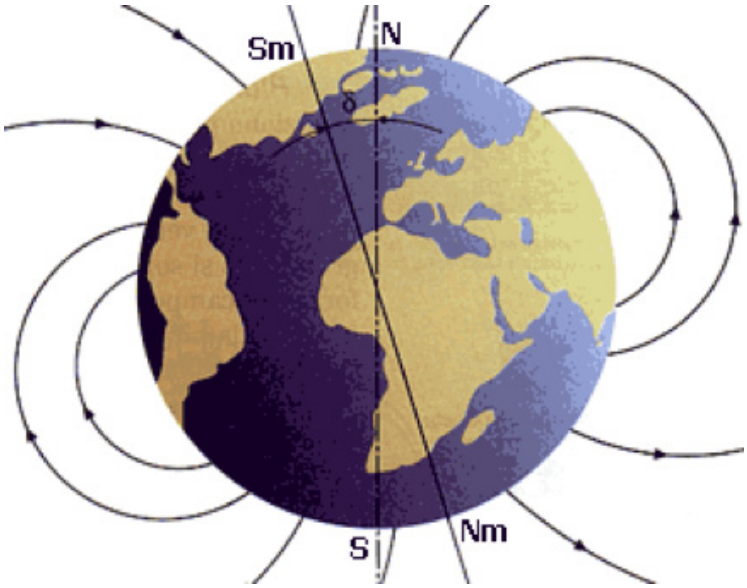


Figura 35. *Desviación del norte magnético y el norte geográfico*

E3) Ángulo Acimut

Es el ángulo de una dirección contado en el sentido de las agujas del reloj a partir del norte geográfico. El acimut de un punto hacia el este es de 90 grados y hacia el oeste de 270 grados sexagesimales. El término acimut solo se usa cuando se trata del norte geográfico. Cuando se empieza a contar a partir del norte magnético, se suele denominar rumbo o acimut magnético. En la geodesia o la topografía geodésica, el acimut sirve para determinar la orientación de un sistema de triangulación.

Debido a las distintas localizaciones del campo magnético que se han comentado en el apartado anterior, existe una desviación entre este norte magnético y el geográfico que se debe corregir, por lo que para el cálculo del acimut hay que tener el modelo magnético mundial válido hasta 2015 (WMM).

Así pues, este ángulo formado entre el norte magnético y el norte geográfico (o norte verdadero), recibe el nombre de declinación magnética.

E4) Medición geodésica

Se trata de un preciso método para la medición de distancias; en el que se considera a la tierra como una elipsoide y no esférica (como en la realidad) y por ello el modelo WGS-84 realiza correcciones introduciendo constates como son: el radio ecuatorial, el radio polar, el achatamiento y la excentricidad.

E5) Círculo máximo o Gran Círculo

Se trata de un método similar a la medición geodésica solo que supone que la tierra es un esferoide en lugar de un elipsoide, por lo que es un poco menos preciso que el método anterior para distancias muy grandes, pero utiliza cálculos más simples y, por tanto, consume menos recursos.

La intersección de una esfera con un plano que contenga su centro genera un círculo máximo y una circunferencia máxima sobre la superficie de la esfera. Un círculo máximo divide a la esfera en dos hemisferios iguales. La distancia entre dos puntos de la superficie de la esfera, unidos por un arco de círculo máximo, es la menor entre ellos y se denomina distancia ortodrómica.

Como ejemplos de círculos máximos en la superficie de la Tierra tenemos los meridianos o la línea del ecuador.

Así pues, esto se puede utilizar para el cálculo de la distancia entre dos lugares con las ecuaciones asociadas:

$$d = k \cdot \arccos \{ \sin \varphi_a \cdot \sin \varphi_b + \cos \varphi_a \cdot \cos \varphi_b \cdot \cos (\lambda_a - \lambda_b) \}$$

Siendo K el radio de la tierra y las latitudes y longitudes expresadas en radianes.

E6) Triángulo esférico

Consiste en la región de la superficie limitada por tres arcos de circunferencia máxima que se cortan dos a dos, siendo los lados del triángulo, los arcos formados y los vértices de los ángulos esféricos los vértices del triángulo esférico.

E7) El problema del rumbo

Como se ha visto en los apartados anteriores, existe una trigonometría de triángulos esféricos que relaciona a sus ángulos con la distancia de sus lados.

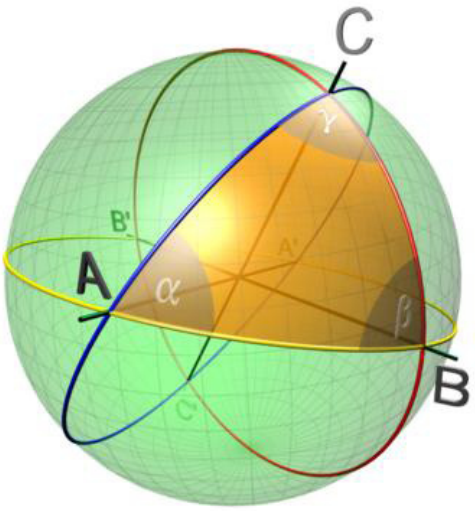


Figura 36. *Triángulo esférico entre dos localizaciones y el norte*

Como se puede ver en la imagen anterior (fig. 36), A sería el usuario, B el POI y C el norte geográfico, por lo que hay que saber si el usuario está orientado hacia el punto concreto, lo cual se resuelve cuando el ángulo acimut coincide con el ángulo alfa del triángulo esférico.

Todos estos cálculos podrían realizarse con trigonometría euclídea y en distancias cortas la diferencia tan apenas se hubiera apreciado, pero las diferencias surgen en la media y larga distancia, por lo que se opta por esta implementación empleando el modelo WGS 84 que también es usado por iOS.

Aunque también existen otros métodos de cálculos de acimut y distancia como son: el algoritmo de Vicenty (algoritmo que se encuentra bajo licencia GPL), el modelo sobre elipsoide de revolución o el uso de coordenadas UTM entre otros.

E8) La representación en pantalla

Uno de los problemas que presenta la realidad aumentada en el móvil es que, al tratarse de un sistema de realidad en 2D, no existe la perspectiva de profundidad. De esta manera, lo que hay que hacer es hallar las coordenadas de los pixeles en las que debemos representar los puntos sobre la pantalla del dispositivo.

Lo que si que se conoce es el ángulo y la distancia que hay con respecto al observador y para dar la sensación de profundidad lo que se puede hacer es tomar como referencia que todos los puntos estén a la misma distancia, el horizonte; por tanto, la distancia al punto, que es conocida, se use como factor de escalado y así poder modificar el tamaño del punto de interés en función de este valor.

Otro aspecto a tener en cuenta es el ángulo de visión de la cámara. Esto quiere decir que puede que se tenga el dispositivo orientado hacia una dirección pero la cámara no es capaz de representar toda la imagen, sino que tiene un ángulo de visión limitado que hay que tener en cuenta ya que sino la representación sería errónea.

Así pues, para calcular las coordenadas en la pantalla de un POI se realizaría con la siguiente fórmula:

$$pixel_x = \frac{WIDTH}{2} + [\sin(\partial)/\sin(\ ANGULO_{VISION}\ I)] * \left(\frac{WIDTH}{2}\right)$$

Siendo WIDTH el ancho de la pantalla, el ángulo la diferencia entre nuestra orientación acimut, el ángulo que forma el objeto con nuestra localización y suponiendo un ángulo de visión de 30°. Otro aspecto sería que se considera el centro de coordenadas (WIDTH/2, WIDTH/2) en lugar de (0,0), por lo que se realiza una translación del punto.

Con respecto al eje Y se emplea el ángulo de inclinación del dispositivo para que los objetos no se observen siempre en el eje medio de la cámara, en lugar del ángulo del eje X.

F) Diseño y documentación framework realidad aumentada

F1) Esquema general

En este framework se quiere implementar los diferentes algoritmos relativos al cálculo de la visibilidad de los diferentes puntos de interés por medio de la realidad aumentada; es decir, teniendo en cuenta la posición y la orientación del usuario para representarlos en la pantalla del dispositivo con sistema operativo iOS.

Como es de esperar en este framework, se hace uso de diferentes sensores del dispositivo como son el GPS para la localización, la brújula y el acelerómetro para la orientación y la cámara para obtener la realidad que rodea al usuario.

Las funcionalidades más importantes de este framework serían:

– Cálculo de la visibilidad:
  - Resuelve la problemática relacionada con la decisión de si los puntos de interés son visibles mediante el cálculo de la orientación y la distancia.
  - Gestiona el funcionamiento de la cámara digital del dispositivo obteniendo la máxima resolución posible.
  - Gestión de la distancia de los puntos a mostrar, limitados por la distancia que establezca el usuario, pudiendo ser ésta de hasta 20 kilómetros.
  - Filtrado de la visualización de los POIs en función de una distancia máxima o los n primeros POIs que se encuentren más cercanos.

– Representación gráfica de los puntos de interés:
  - Visualización de los puntos de interés cercanos a través de la pantalla de la cámara y en un radar en función del cálculo obtenido en el apartado anterior.

– Gestión de localización necesaria en el cálculo de la visibilidad para el correcto posicionamiento de los puntos de interés:
  - Gestión de la localización a través del GPS y del 3G y, en caso de no disponer de ésta, utilizar la última válida que ha sido obtenida.
  - Gestión de la actualización de la localización solo en el caso en el que se haya recorrido una distancia mínima o un tiempo determinado.

– Gestión de los sensores necesarios para el algoritmo de cálculo de la visibilidad:
  - Gestión de los sensores magnéticos y gravitatorios, corrigiendo la declinación existente según el Modelo Mundial Magnético válido hasta 2015 (gracias al framework nativo de iOS CoreLocation).
  - Cálculo de la orientación del dispositivo, suavizando los cambios bruscos que se obtienen de los sensores y que en muchos casos son datos espúreos.

– Gestión de la interacción con los puntos de interés en pantalla:
  - Control del pulsado sobre los puntos de interés.
  - Realizando acciones predefinidas para los objetos, dentro del framework, representando la información disponible.
  - Permite externalizar los eventos de pulsado indicando qué objeto fue seleccionado y facilitando el poder modificar dicho comportamiento.

F2) Diagrama de clases (fig. 37)

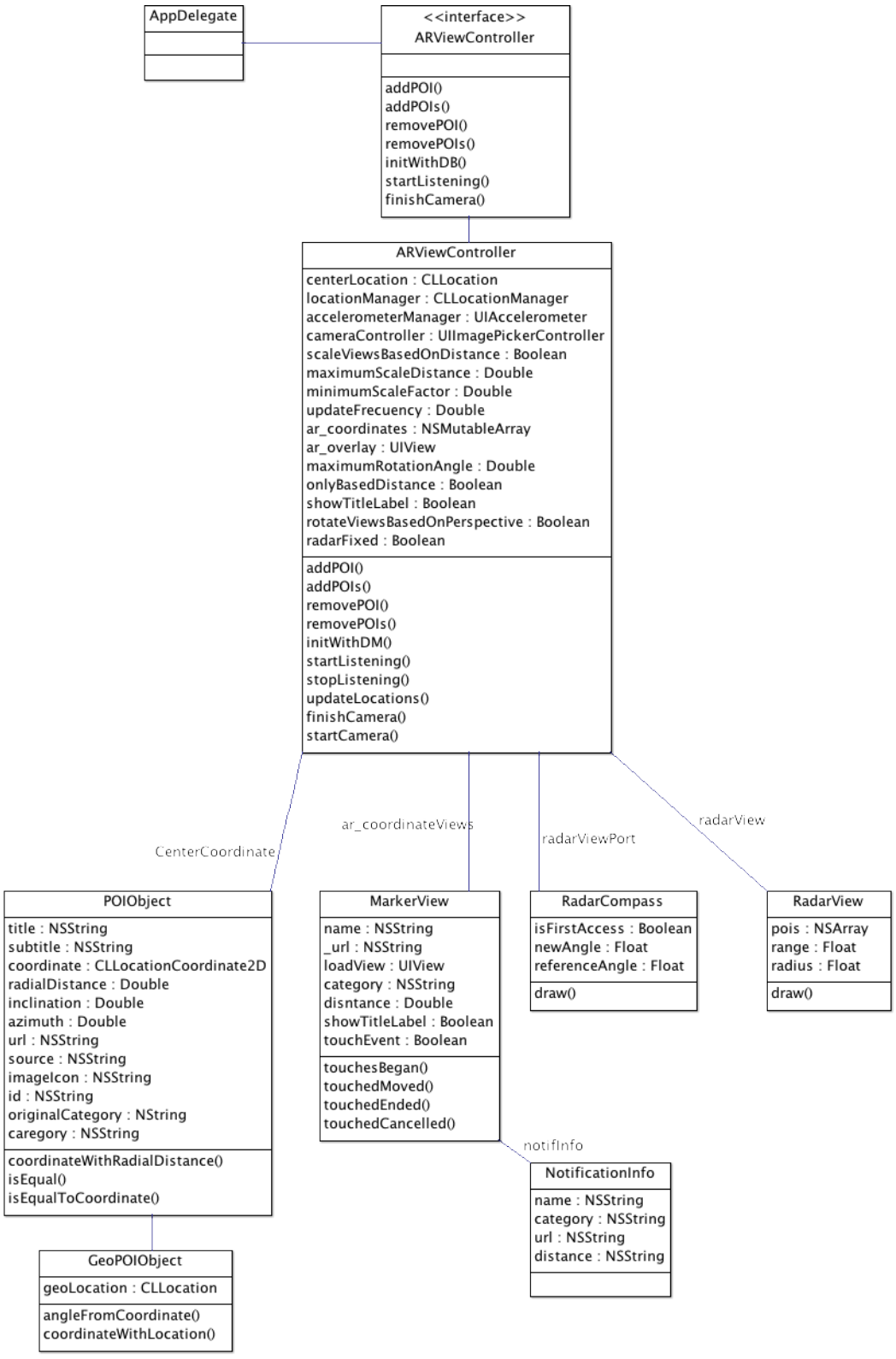


Figura 37. *Diagrama de clases módulo de realidad aumentada*

En este apartado se va a explicar el diagrama de clases que se ha implementado para la elaboración del framework de realidad aumentada, el cual se basa en un patrón de diseño facade o fachada, en el que el acceso al módulo se realiza a través de ARViewController, la clase principal que controla todo el funcionamiento, con el fin de minimizar las comunicaciones y puntos de acceso de un subsistema a otro, de modo que la dependencia entre subsistemas se reduzca o lo que es lo mismo, se disminuya el acoplamiento entre sistemas.

En el gráfico no se representan las clases nativas de Objective-C ni las específicas de Cocoa-MultiTouch como, por ejemplo, las gestoras de la cámara y de los sensores, para mostrar con una mayor claridad lo que se ha implementado y que es propio del framework.

En primer lugar tenemos la interfaz del framework (en Objective-C conocidas como protocolos) que será, a través de la cual, el framework se puede unir al resto de otra aplicación. El protocolo llamado **ARViewControllerProtocol** quedaría del siguiente modo:

```
@protocol ARViewControllerProtocol

//Añadir POIs obtenidos de las fuentes de datos

- (void)addPOI:(PoiObject *)coordinate;

- (void)addPOIs:(NSArray *)newCoordinates;

//Borrado de datos

- (void)removePOI:(PoiObject *)coordinate;

- (void)removePOIs;


- (id)initWithDB:(sqlite3 *)contact;

- (void) startListening;

- (void) finishCamera;

@end
```

Se puede observar que se dispone de varios mensajes para añadir y borrar coordenadas (POIs) que puede ser tanto de manera individual, como por medio de un array de POIs. Además se dispone de un inicializador en el que se le pasa como parámetro un conector con una base de datos para que pueda hacer uso de la petición y gestión de la base de datos.

ARViewController: Ésta sería la clase principal en la que se implementan los mensajes del protocolo entre otros y se encarga de controlar diversas funciones:

- Se realiza la calibración de los sensores necesarios como son el acelerómetro y el giroscopio, para conocer la inclinación vertical y horizontal, así como la brújula para obtener la orientación del mismo corrigiendo la declinación existente según el Modelo Mundial Magnético.
- Se encarga de la localización geográfica del dispositivo, gracias al GPS y al 3G o el WiFi según esté disponible o no, o proporcione una mayor precisión, obteniendo la última localización válida en caso de no estar disponible ninguno de los servicios.
- Realiza la inicialización de cámara y superpone a la vista de ésta los objetos que se quieren representar, es decir, los iconos de los POIs, pero también inicializa las clases relacionadas con el radar para el dibujado del mismo.

– Otro aspecto importante es la existencia del vector *ar_coordinates* que funciona como una cache en la que se guardan en memoria todos los POIs que han sido pedidos por última vez al framework de gestión de datos con el fin de ser gestionados con una mayor eficiencia.

El funcionamiento de esta clase consiste en la llamada cada cierto período de tiempo parametrizable, controlado por temporizadores, al método de pintado, para realizar la actualización de la pantalla en tiempo real, recorriendo una lista en la que se almacenan los objetos a representar tanto en la realidad aumentada como en el radar.

Existen algunos aspectos que se pueden parametrizar:

– **maximumRotationAngle, rotateViewsBasedOnPerspective**: Estos parámetros sirven para indicar si se les debe aplicar una transformación a los puntos gráficos cuando se encuentra en los laterales de la pantalla y el ángulo de transformación, con el fin de dar una mayor sensación de profundidad.

– **onlyBasedDistance:** Sirve para indicar si en la representación solo se debe tener en cuenta la distancia o también la altura, de modo que los puntos no son fijos en el eje horizontal de la imagen. Esto es útil en el caso de que se tengan servicios que proporcionen la altura como tercer coordenada y así dar una mayor fidelidad de la representación de los datos. Sin embargo, los servicios elegidos para realizar esta aplicación no lo hacían; no obstante queda implementado en el código para futuras aplicaciones.

– **updateFrecuency**: Frecuencia con la que es actualizada la pantalla de AR. Se ha establecido 10 Hz por defecto, ya que se considera que el funcionamiento con este valor es fluido y no es necesario que sea mayor, ya que sino la cantidad de operaciones por unidad de tiempo aumentan considerablemente. Esto conllevaría una sobrecarga sobre el dispositivo innecesaria, ya que el proceso de actualización de la pantalla es una tarea crítica para el correcto funcionamiento de la AR en tiempo real.

– **showTitleLabel**: Se trata de un booleano para indicar si se desea que se muestre el título del POI debajo del icono de la categoría.

– **dataManager:** Se utiliza para indicar el manejador de la gestión de datos (en la aplicación se trata del framework que se ha diseñado para tal efecto).

– **radarfixed:** Este parámetro se utiliza para controlar si el radar en el que se representan los POIs, además de por la realidad aumentada, se encuentra fijo o móvil. En el caso de que se deje fijo, los puntos no se mueven y es una flecha sobre el radar la que indica la orientación de usuario. Si se configura en modo (móvil), esta flecha permanece fija y son los POIs los que se van moviendo por el radar, indicando esta flecha, los puntos que se están visualizando en cada momento en la realidad aumentada.

RadarView: Esta clase se encarga de dibujar el radar y los POIs de su interior dibujándolos de diferente color en función de la fuente de datos y escalados en relación con el radio máximo de búsqueda.

POIObject: Se trata de un objeto geolocalizado que se encuentra cargado en memoria, del que se conoce como poco su latitud y su longitud (esta clase es la misma que aparece en el framework de gestión de datos).

También se almacena cual es la fuente de datos desde la que se ha obtenido, la URL con la información del POI, su nombre y descripción (título y subtítulo) y la distancia al mismo desde el dispositivo, así como el acimut, que son los grados de desviación de un punto con respecto al norte geográfico y los grados de inclinación.

Como se puede observar (fig. 38), este objeto puede estar en diferentes estados y para explicar mejor su funcionamiento se va a realizar el diagrama de flujo de eventos o diagrama de estados del mismo.

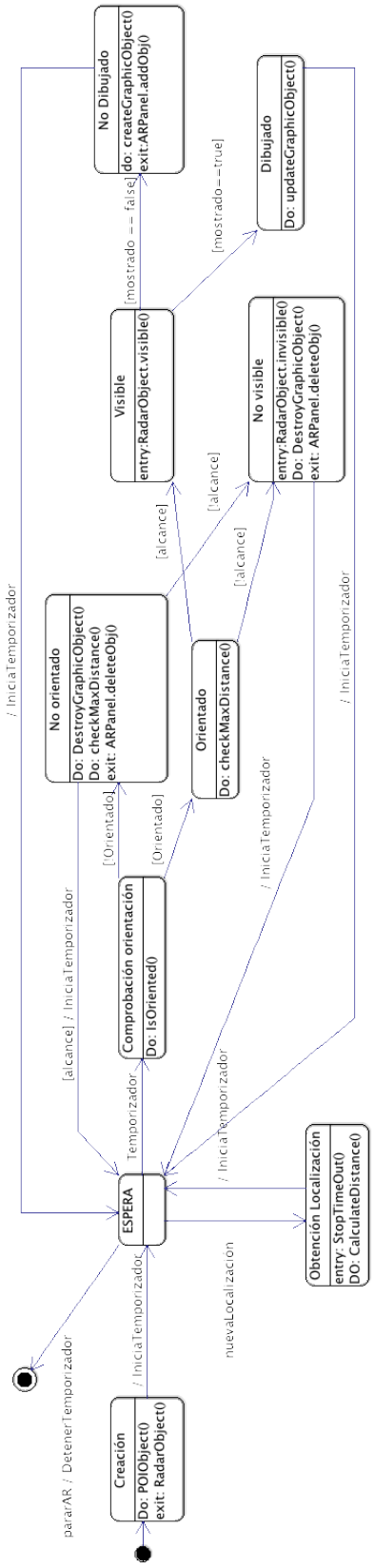


Figura 38. *Diagrama de flujo del objeto POIObject*

El gráfico presenta un modelo conceptual por lo que el nombre de las funciones no se adapta del todo con el nombre de las funciones reales implementadas con el fin de mostrar el funcionamiento de una manera más clara.

Como se puede ver en el gráfico hay un estado de espera, ya que se hace uso de un temporizador, en el que cada vez que actúa, se debe comprobar la orientación del dispositivo y si la distancia al POI que se está analizando es menor al máximo establecido. En el caso de que esté orientado y sea visible, si el objeto gráfico y el del radar no estaban creados, se crean y se añaden al ARPanel. En caso de que ya lo estuvieran, se deberá actualizar su dibujado.

Cada cierto tiempo se recibe un evento con una nueva localización y es entonces cuando se debe parar el temporizador y recalcular la distancia, para luego reactivarlo de nuevo.

GeoPOIObject: Se trata de una clase que hereda de la anterior, siendo muy similar a ésta, solo que en este caso se añade la geolocalización del POIs, implementándose algunos métodos específicos.

MarkerView: En esta clase se controlan los eventos de pulsado en la pantalla en la capa superpuesta a la vista de la cámara que se encarga de representar gráficamente los puntos de interés.

Lo que realmente hace es que cuando un POI es representado sobre la cámara y es pulsado, ésta se encarga de abrir una vista web con la información de la url disponible de este punto.

Esta clase es configurable, ya que estos eventos pueden ser manejados en la propia clase del framework, o ser enviados hacia el controlador de la aplicación que lo esté utilizando, debido a que no se desee utilizar la representación por defecto implementada en el framework. La variable que controla esto es touchEvent. En la aplicación desarrollada se optó por la segunda opción, con el fin de verificar su correcto funcionamiento.

NotificacitionInfo: Esta clase representa el uso de las notificaciones existentes en iOS para el caso en el que los eventos de pulsado se envíen hacia el controlador de la aplicación. En la notificación se envía la información del POI pulsado. Para que el controlador pueda recibir estas notificaciones deberá registrarse, como se muestra a continuación (fig. 39), para indicar que desea ser receptor de este tipo de notificaciones:

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(eventListenerDidReceiveNotification:)
                                             name:@"NotificationInfo"
                                           object:nil];
```

Figura 39. *Registro para recepción de notificaciones en una clase*

F3) Modelo dinámico

En este apartado se va a explicar como interactúan los objetos una vez explicado el modelo de datos. Para explicarlo gráficamente se ha creado un diagrama de secuencia (fig. 40):

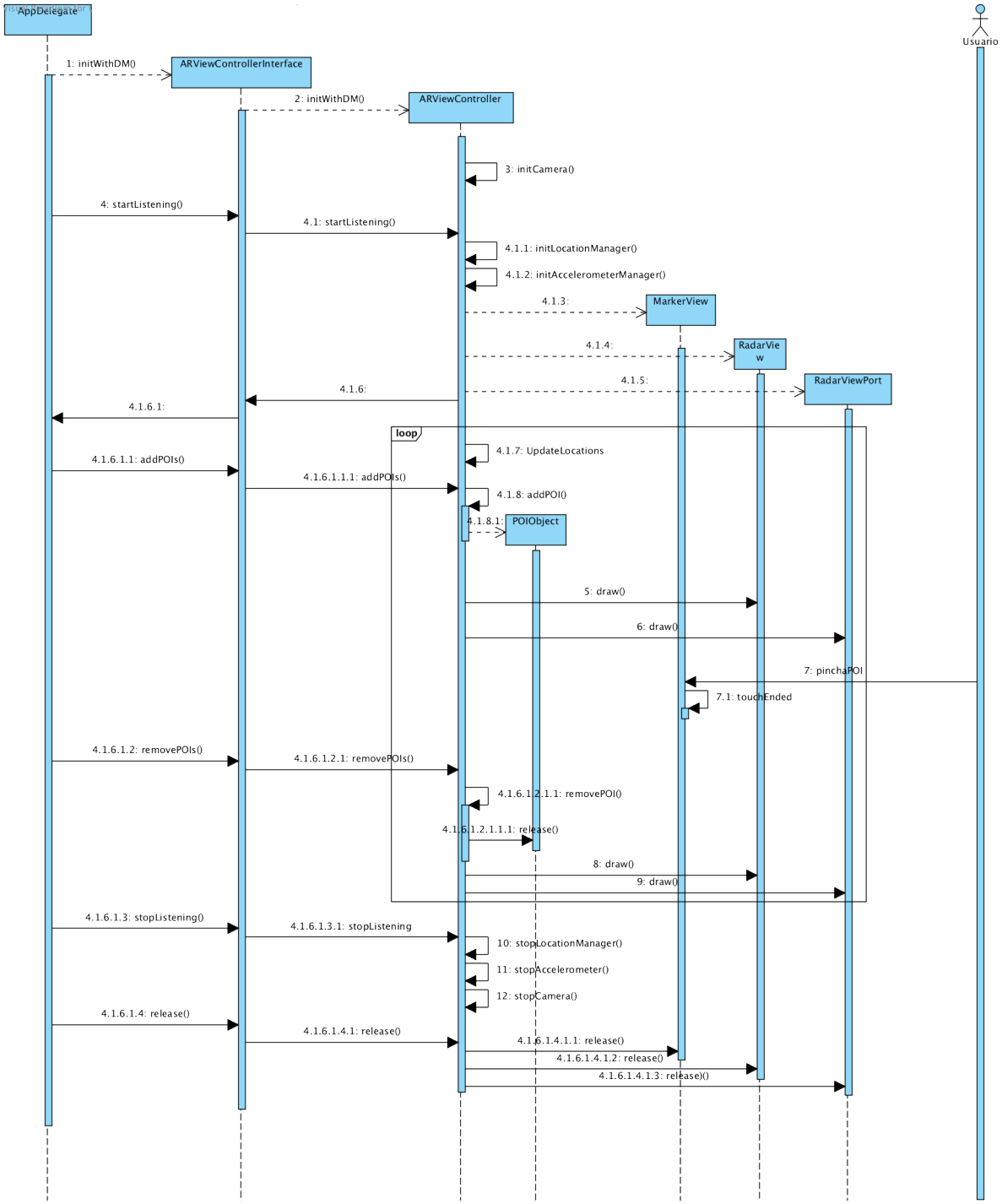


Figura 40. *Diagrama de secuencia del framework de AR*

En el diagrama se observan cuatro partes claramente diferenciadas, las cuales van a ser explicadas por separado:

Fase de instanciación (fig. 41)

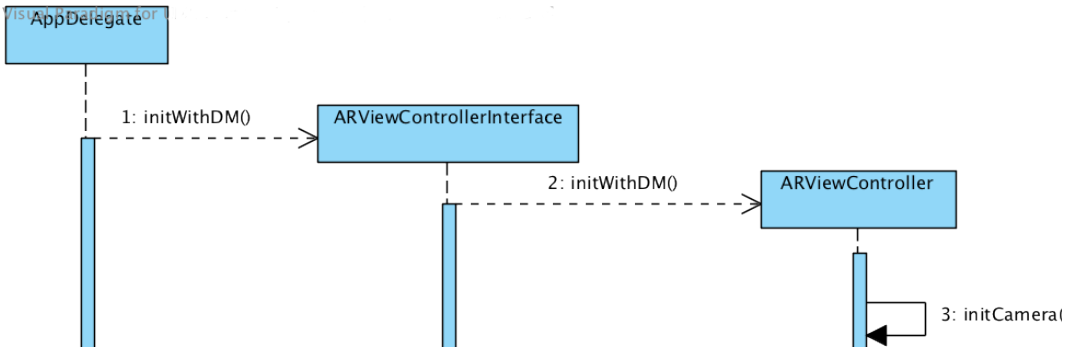


Figura 41. *Fase de instanciación*

En esta fase solo se crean instancias de los diferentes agentes que intervienen en la realidad aumentada.

Fase de creación (fig. 42)

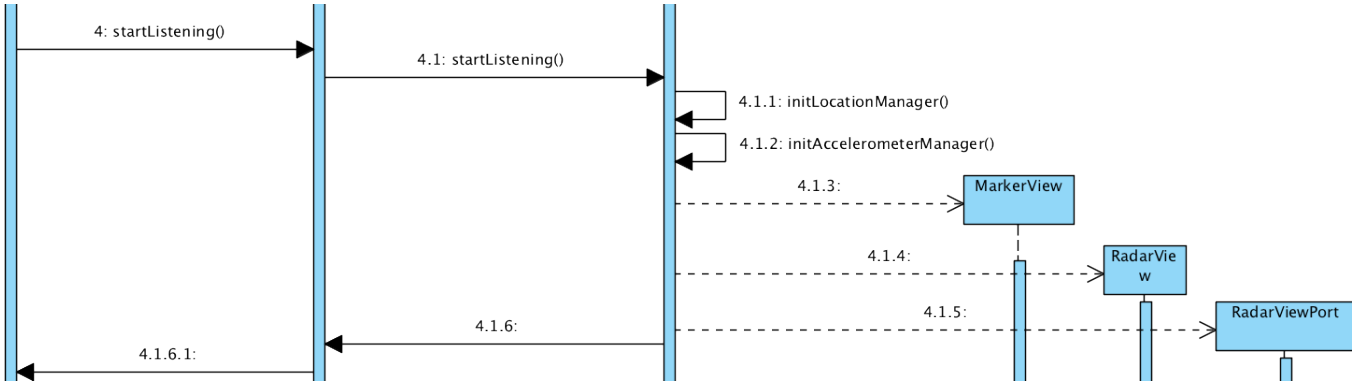


Figura 42. *Fase de creación*

Ahora lo que habría que hacer es obtener la localización, la orientación del usuario y la posición del dispositivo, por lo que se inicializan los sensores de localización y del acelerómetro. También se crean las vistas del radar junto con la brújula del mismo y el MarkerView utilizado para controlar las pulsaciones del usuario sobre un POI.

Fase de ejecución (fig. 43)

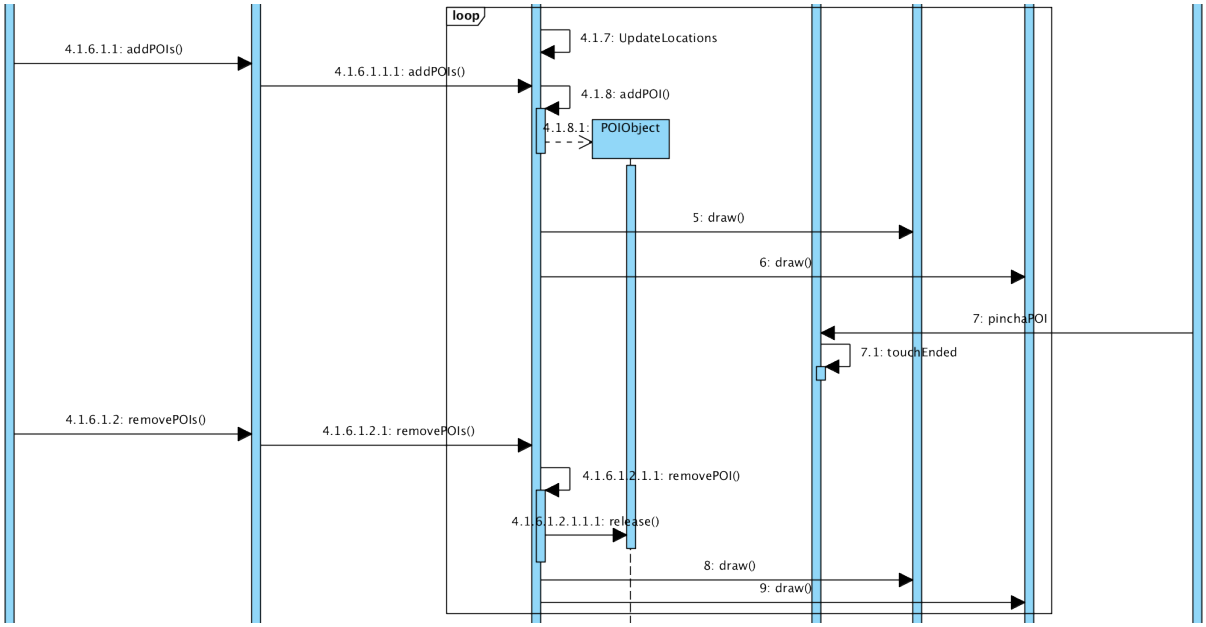


Figura 43. *Fase de ejecución*

Sería la fase fundamental del módulo, en el que se comienza añadiendo POIs en dos vectores con el listado de los POIs a dibujar y mostrar en el radar. Por medio de una se va analizando cada cierto tiempo si el POI está orientado y dentro del límite de distancia prefijado, haciendo en cada caso repintados, tanto de los puntos en la capa superpuesta de la cámara de realidad aumentada como en el radar.

Fase de desactivación (fig. 44)

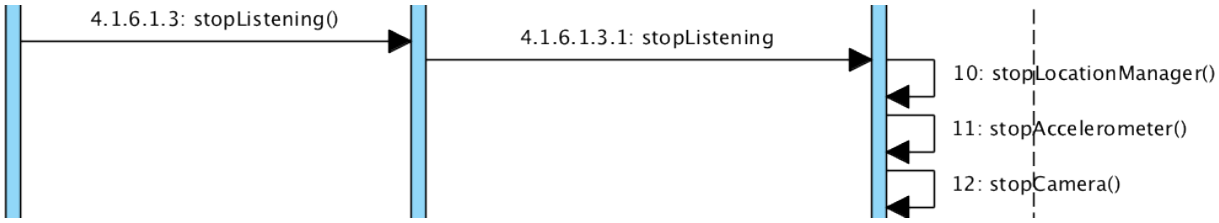


Figura 44. *Fase de desactivación*

Simplemente se trata de una fase en la que se desactivan todos los sensores y el visor de la cámara.

Fase de finalización (fig. 45)

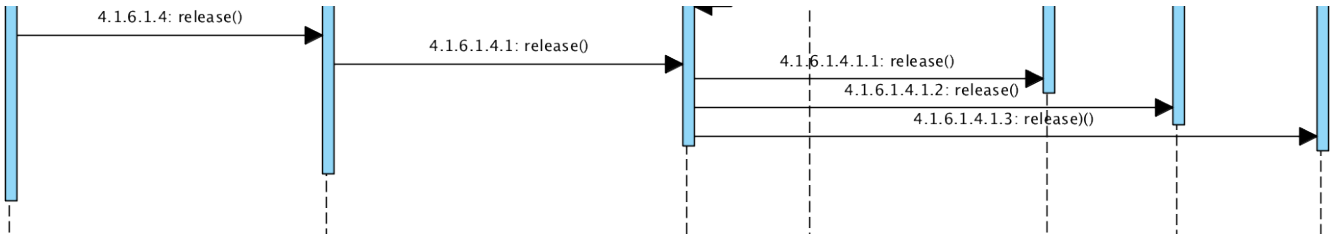


Figura 45. *Fase de finalización*

En esta fase lo único que se hace es la destrucción de las instancias de los objetos creados en la fase de instanciación.

G) Diseño y documentación framework de gestión de datos

G1) Esquema general

En este framework se han llevado a cabo varios aspectos importantes de la aplicación. En primer lugar, gestionar la petición y parseo de datos a servidores externos, también tiene que categorizar dichos puntos y, por último, dar soporte al modo offline de la aplicación por medio de alguno de los sistemas de persistencia de datos posibles.

Así pues, este framework se deberá encargar de los siguientes aspectos:

– Petición de datos a servicios externos:
  - Utilización de los servicios REST que nos proporcionan Geonames, CloudMade y GSL. Servicios elegidos para la descarga de datos.
  - Parseo de los datos procedentes de dichos servicios en formato JSON o XML, adaptándolos a modelo de datos de la aplicación, ya que la estructura de los datos de dichos servicios no es homogénea.
  - Soporte para la adición de un nuevo servicio futuro, implementado un protocolo.

– Persistencia de datos:
  - Almacenamiento de datos en una base de datos SQLite, con el fin de reducir la transferencia de información entre los servicios seleccionados y la aplicación al mínimo posible.
  - Cache intermedia para la optimización de la utilización de los datos y mejora del rendimiento en la muestra de los POIs tanto en la parte de realidad aumentada como en el caso de los mapas.
  - Categorización de los puntos de interés, con independencia de la fuente de datos.
  - Soporte para el modo offline (gracias a la persistencia de datos) en el que no sea necesario la conexión a los servicios de datos, bien porque no esté disponible esta conexión o porque el propio usuario no lo desee.

– Soporte para las búsquedas siguiendo diversos criterios:
  - Búsqueda por categorías en los servicios.
  - Selección de servicios de los que descargar los datos (Geonames, Wikipedia-Geonames, Cloudmade o GSL en una primera implementación, pero ampliable con gran facilidad).
  - Limitación del número máximo de puntos a mostrar en una petición.

G2) Política de petición de datos

Se ha diseñado una política de petición de datos que fuera eficiente y que redujera la cantidad de peticiones a servidores externos, de modo que el tráfico de datos disminuya, pero además en todo momento se tenga toda la información actualizada. Esta política se ha añadido de un

modo que fuera parametrizable por parte del usuario o que el propio desarrollador pueda elegir en todo momento la opción que más le interese.

En primer lugar se han definido tres modos distintos de actualización:

– El **modo internet u online puro** en el que lo datos siempre son pedidos a los servidores externos y no existe interacción con la base de datos. En este modo se consigue que los datos siempre estén actualizados con la información más reciente que haya disponible en el servidor. Por otro lado, hay que tener en cuenta que en este modo el tráfico de datos será mayor, ya que no se está teniendo en cuenta los datos que ya pudieran haber sido pedidos. Esto se ve reflejado en el tiempo que se tarda en mostrar los POIs en cualquiera de los modos posibles de la aplicación. Ya que cada vez que se produce un cambio de estado (cambio de radio, de posición o cambio de modo de representación) hay que pedir datos al servidor externo lo cual es mucho más lento que hacer la petición a la cache o a la base de datos local, como se puede ver en el estudio realizado y expuesto en el anexo J.

– El **modo mixto**. En este modo si que se tienen en consideración los datos que ya han sido pedidos anteriormente. Consiste en que los datos cada vez que son pedidos al servidor también son guardados en una base de datos, lo cual permite evitar ciertas peticiones a dicho servidor externo en algunas situaciones, así como surtir de la información necesaria al modo offline que se explicará a continuación.

  Como se ha explicado, en este modo hay situaciones en las que no habría que realizar la petición al servidor externo, ya que solo se realizaría en el caso de que hubiera un cambio de posición y si se quisiera hacer una búsqueda con un radio mayor al radio máximo que hay guardado en la base de datos. En caso de que el radio de búsqueda sea reducido por el usuario lo único que habrá que hacer es realizar una consulta a la base de datos, puesto que el resultado, será un subconjunto de lo almacenado en la misma.

  Para realizar esta política, en primer lugar hubo que definir en qué consistía un cambio de posición y, por otro lado, qué hacer con los datos que ya existieran en la base de datos cada vez que se realizaba una nueva petición.

  Como solución del primer problema, se planteó que se produce un cambio de posición cuando existe una distancia igual o mayor a la mitad del radio de búsqueda, entre la posición actual y la posición en la que se realizó la última petición, es decir:

  distancia (punto actual, punto última petición) >= radio de búsqueda/2

  Un ejemplo podría ser si el radio de búsqueda fuera de un kilómetro, entonces cada quinientos metros de desplazamiento con respecto al punto de origen (distancia medida en línea recta), se produciría una nueva petición. En un primer instante parecería mucha distancia, pero lo cierto es que hay que tener un compromiso entre la reducción de la cantidad de peticiones al servidor y la precisión de los datos, ya que si en cada desplazamiento se produjera una actualización, el tiempo real que requiere la realidad aumentada se pondría en peligro. No obstante si el usuario quiere más precisión, puede reducir el radio de búsqueda y la peticiones serán mucho más frecuentes (ya que si quiere precisión el radio debe ser pequeño porque en caso contrario la cantidad de POIs aumenta considerablemente y no es viable para la representación en una pantalla de un dispositivo móvil).

El segundo aspecto es el referente a los datos ya existentes en la base de datos una vez que se realiza una nueva petición. Se ha decidido que se le va a dar al desarrollador dos posibles soluciones; una primera, en la que cada vez que se realiza una nueva petición los datos que hubiera en la base de datos son borrados y, una segunda, en la que los datos no son borrados en la base de datos hasta que no se alcanza una cantidad de POIs máxima en la misma. Dicha cantidad puede ser definida por el desarrollador. Una vez que se ha alcanzado esta cantidad máxima, se han implementado dos políticas de borrado, una en la que se borran lo datos que hace más tiempo que se descargaron (FIFO) y otra en la que se borran los datos que hace más tiempo que no se usan (política LRU).

– Por último estaría en el **modo offline**. Este modo puede ser utilizado por petición del usuario pero también entra en funcionamiento automáticamente en caso de que no se detecte conexión 3G o WiFi. En este caso, la posición del dispositivo es tomada solamente con la ayuda de los satélites GPS, sin poder hacer uso del A-GPS (Assisted GPS) ya que como se ha dicho no hay conexión ni a 3G ni a WiFi. Esto tiene como consecuencia que la obtención de la posición será más lenta y, por otro lado, que los datos no son tomados de servidores externos sino de los datos que hubiera en la base de datos local, que se encuentren dentro del radio de búsqueda. Es por este motivo, por el que se ha señalado en apartados anteriores, la conveniencia de usar el modo mixto en el que no se borran los datos de la base de datos en cada petición para así disponer de una mayor información local y que esta modalidad disponga de una mayor variedad de datos.

G3) Categorización de los puntos de interés

Para darle mayor riqueza a la información de los puntos que se muestra a través de la realidad aumentada o de cualquiera de las otras formas posibles en las que puede ser integrado este framework, se ha decidido hacer una categorización de los puntos de interés en función de diferentes temáticas (en este caso orientadas al turismo).

Los servicios que se han seleccionado para añadir en este framework ya poseen un sistema de categorización de los puntos de interés y proporcionan esta información en cada petición, para cada uno de los puntos e incluso es posible realizar la petición de los puntos de una categoría concreta. El problema surge al tratarse de servicios heterogéneos, esto provoca que el sistema de categorización en cada servicio sea diferente y no coincidan las categorías existentes. Para solucionarlo se ha elaborado una categorización propia, por lo que cada vez que se reciben datos, estos son analizados y adaptados a una de las categorías propias existentes.

Para la realización de este mapeo de las categorías que proporcionan los diferentes servicios de datos, con la categorización propia creada, se ha implementado dentro de la base de datos local una nueva tabla en la que se introducen cada una de las correspondencias entre los dos sistemas de categorización. Por tanto, si un desarrollador desea introducir un nuevo servidor de datos lo que tendrá que hacer es introducir el nuevo mapeo con las nuevas categorías en la base de datos local. Así pues, el esquema de la base de datos de la aplicación, uniendo el almacenamiento de los puntos de interés y el mapeo de la categorización quedaría del siguiente modo (fig. 46):

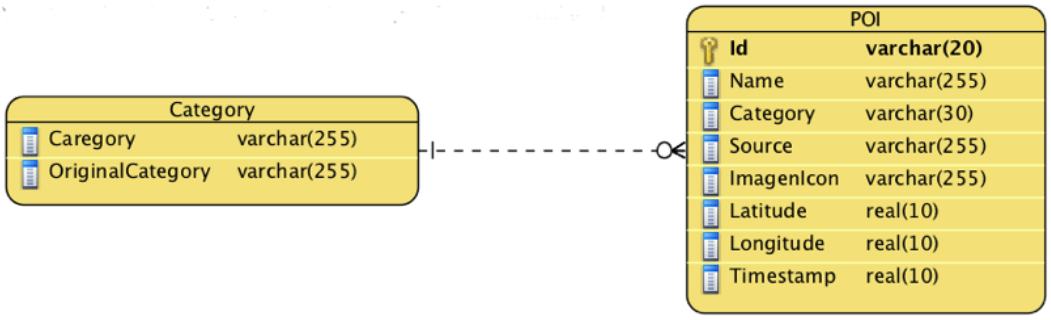


Figura 46. *Modelo E-R de la base de datos*

La categoría también es un atributo en la tabla de POIs con el fin de hacer más eficiente la consulta en la que se obtienen todos los puntos que hay almacenados, la cual es la consulta más repetida; a pesar de ser menos eficiente en cuanto al espacio, por la redundancia de datos. Esto se decidió ya que la base de datos no iba a ser muy grande y, por tanto, el espacio no era crítico.

La categorización propia que se ha utilizado sería la siguiente:

- Cultural:
  - Museum
  - Ethnographic
  - Historic
  - Archaeology
- Sport:
  - Facilities
- Nature:
  - Landscape
  - Protected
- Gastro:
  - Product
  - Producer
- Leisure:
  - Shop
    - Nightclub
- Service:
  - Health
    - Chemist
    - Centre
    - Hospital
    - Red cross
    - Other
- Public Safety
  - Civil guard
  - City police
  - Fire brigade
  - Other
- Transport
  - Taxi
  - Airport
  - Bus station
  - Train station
  - Post Office
  - Other
- Official
  - Tourist office
  - Cityhall
  - Other
- Misc
  - Cash
  - Bank
  - Petrol station
  - Carpark
  - Garage
  - Other
- Restaurant
  - Restaurant
  - Café
  - Bar
- Accommodation
  - Hotel
  - Apartment
  - Camping
  - Rural

G4) Diagrama de clases

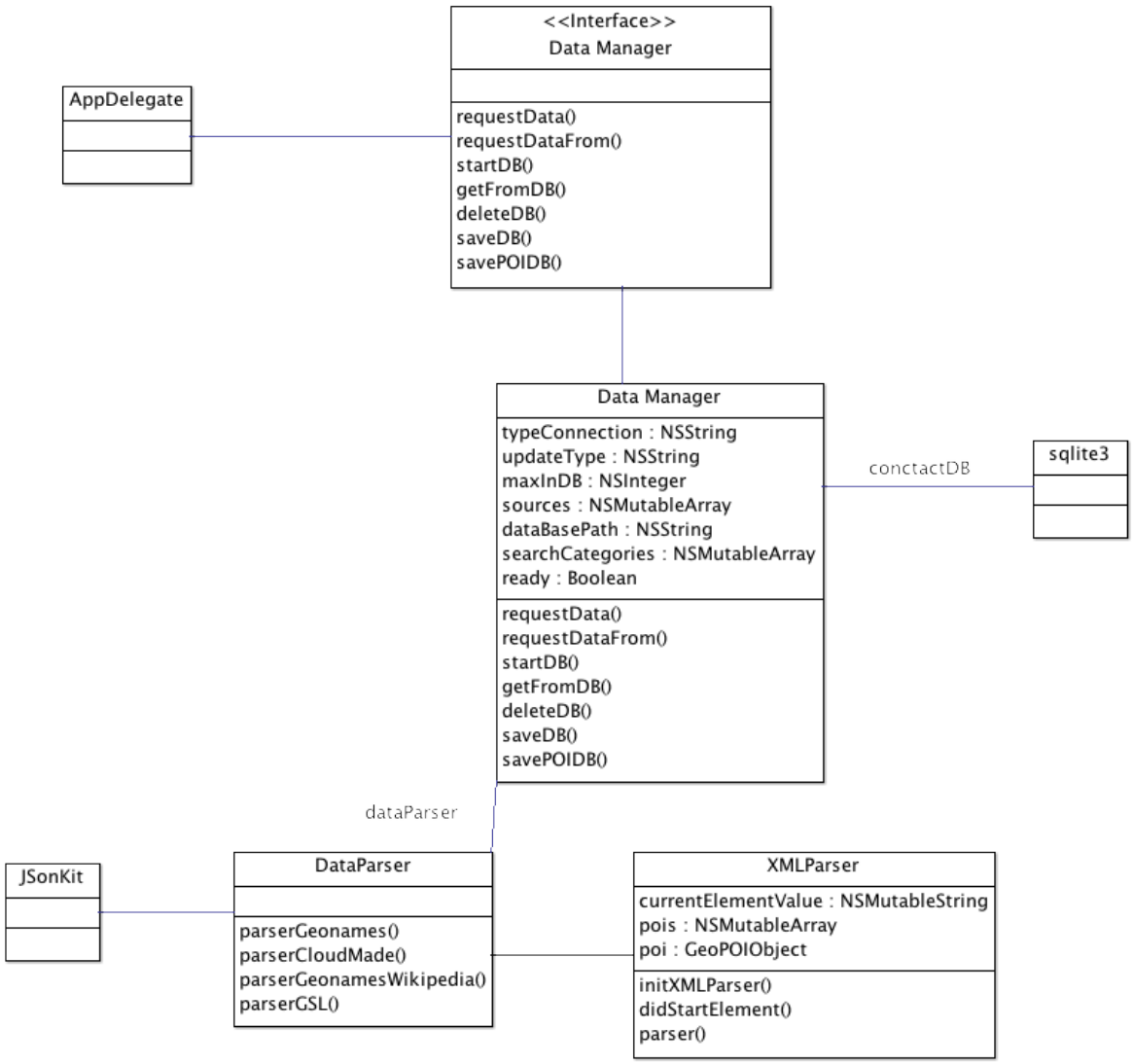


Figura 47. *Diagrama de clases del framework de gestión de datos*

El gráfico anterior (fig. 47) sería el diagrama de clases del framework de gestión de datos; como se puede ver se sigue el mismo patrón de diseño facade que en los demás casos. En este caso también se ha creado un protocolo que será el que lo comunique con el resto de la aplicación y de los frameworks con el nombre de **DataManagerProtocol**:

```
@protocol DataManagerProtocol

-(void) setTypeConnection:(NSString *)type;

-(NSMutableArray *) requestData: (CGFloat) lat Lon: (CGFloat) lon Alt: (CGFloat) alt radius:
(CGFloat) rad Elems: (NSInteger) elem;

-(NSMutableArray *) requestDataFrom: (NSString*) source Lat: (CGFloat) lat Lon: (CGFloat)
lon Alt: (CGFloat) alt radius: (CGFloat) rad Elems: (NSInteger) elem Lang: (NSString *) lang;

-(NSMutableArray *) getFromDB: (CGFloat) lat Lon: (CGFloat) lon Alt: (CGFloat) alt radius:
(CGFloat) rad;

-(void) startDB: (sqlite3 *) contact;

-(void) deleteDB;

-(void) saveDB: (NSMutableArray *) listPOIs;

-(void) savePoiDB: (GeoPOIObject *) poi;

@end
```

En él se pueden ver los mensajes principales:

setTypeConnection: Selecciona el tipo de conexión con el que los datos van a ser tomados como se ha explicado en el apartado de política de datos (Internet, mixto, offline).

requestData: Realiza la petición de datos desde una posición concreta y con un radio determinado a todos los servicios que estén activos en ese momento en el modo de conexión de internet y mixto.

requestDataFrom: Igual que el anterior pero se especifica al servidor que se quiere realizar la petición.

getFromDB: Realiza la petición en este caso a la base de datos local.

startDB y deleteDB: El primero de ellos inicia y crea (si no existe ya la base de datos local). El segundo borra todo el contenido que hubiera en la base de datos relativo a los puntos de interés.

saveDB y savePOIDB: El primero de ellos recibe como parámetro una lista de puntos de interés y los almacena en la base de datos y el segundo hace lo mismo pero de forma unitaria.

A continuación está la clase DataManager que implementa el protocolo que se acaba de explicar. Tiene diferentes atributos que permiten parametrizar diferentes comportamientos del framework, como son el tipo de política (**typeConnection**) de obtención de datos (internet, mixta u offline), la cantidad máxima de puntos almacenados en la base de datos local (**maxInDB**), la política de borrado (FIFO o LRU) cuando hay que borrar elementos que hay en la base de datos (**updateType**) ya que se ha alcanzado esta cantidad máxima (**maxInDB**), el listado de los servicios desde los que se quieren obtener los datos (**sources**) y el listado de las categorías sobre las que se quiere mostrar datos (**searchCategories**).

Todo lo referido a la conexión, la creación y la consulta de la base de datos se ha realizado por medio de la librería sqlite que aparece por defecto en el iOS SDK. Existen algunas otras posibilidades, como por ejemplo utilizar una capa adicional, existente en iOS implementada en el framework Core Data del propio SDK, que gestiona las conexiones a la base de datos. Sin embargo, debido a la sencillez de la base de datos necesaria para esta aplicación y cualquiera de este estilo que pueda hacer uso de este framework, se ha decidido no añadir una nueva capa

de complejidad a la arquitectura, ya que este framework de gestión perteneciente al iOS SDK, está pensado para bases de datos más grandes y complejas.

Por último estaría la clase DataParser que se encarga de traducir la información que llega de los servicios en forma de *JSON*, al modelo de datos de la aplicación. Para realizar este cometido, está el framework JSONKit que obtiene información de archivos en formato JSON.

La elección de este framework para serialización de los datos en *JSON* y no utilizar el que viene por defecto con iOS en sus últimas versiones se debe a que existen varios estudios de análisis de tiempos en los que se demuestra que se trata del framework más rápido para este cometido además de por su fácil manejo y a que, como se ha dicho, el otro framework cuyos tiempos son similares sería el que viene por defecto en iOS pero tiene el problema de que solo aparece en las últimas versiones de éste. Con todo ello se consigue que el framework de gestión de datos sea también aprovechable en versiones más antiguas de iOS. Algunos ejemplos comparativos con otros frameworks existentes en Objective-C con la misma función, se pueden ver en el siguiente gráfico[11] (fig. 48):

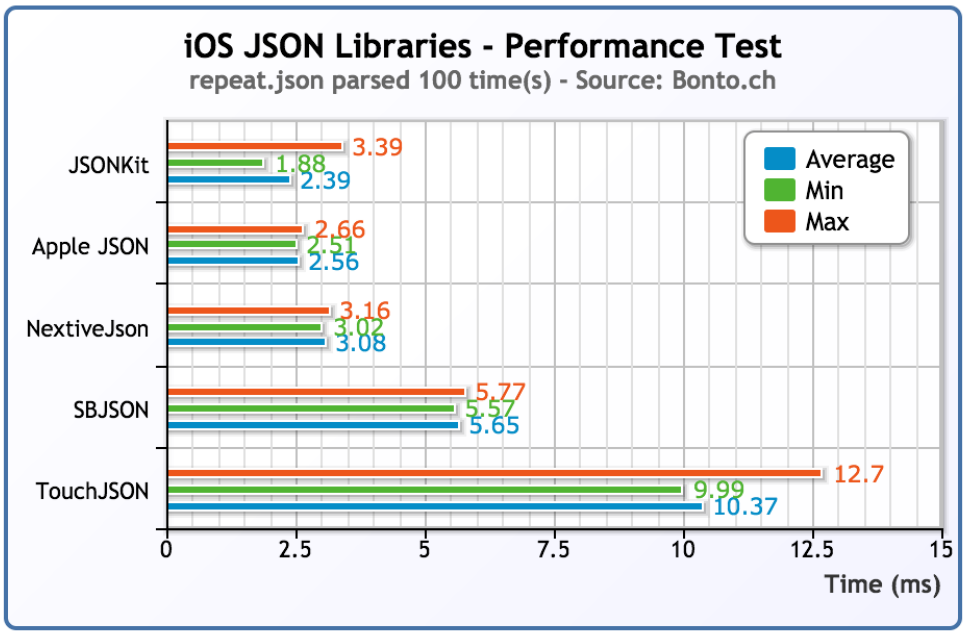


Figura 48. *Análisis de tiempos de diferentes framework de ficheros JSON*

O en el siguiente (fig. 49), realizado por los propios desarrolladores de JSONKit[12].

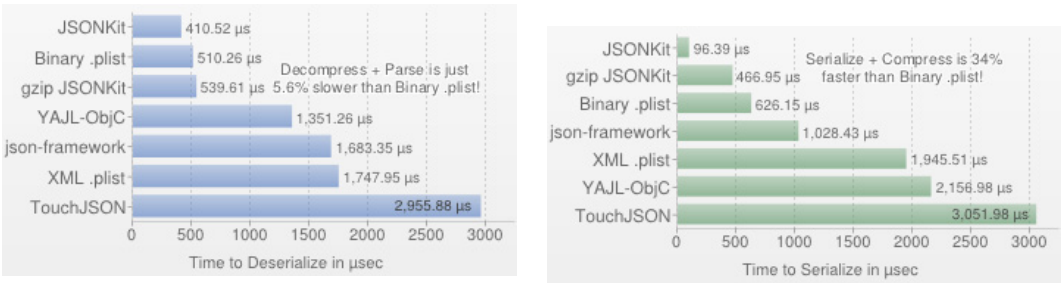


Figura 49. *Análisis de tiempos de diferentes framework de ficheros JSON*

11. http://www.bonto.ch/blog/2011/12/08/json-libraries-for-ios-comparison-updated/
12. https://github.com/johnezang/JSONKit

G5) Introducción de un nuevo servicio de datos

Este framework, como se ha dicho anteriormente, se ha implementado con tres servicios diferentes Geonames, CloudMade y GSL, pero la idea es que añadir un nuevo servicio no tuviera ningún problema y se pudiera realizar de una manera más o menos directa. Por ello, se van a explicar los diferentes pasos que habría que seguir para llevarlo a cabo.

En primer lugar, lo que habría que hacer sería implementar un nuevo mensaje para el parseo de los datos del nuevo servicio en la clase DataParser. Este proceso es muy dependiente del servicio que se esté parseando, ya que al ser servicios heterogéneos, la estructura en la que son recibidos los datos es muy distinta de un servicio a otro e incluso es posible que el nuevo servicio que se quiera añadir no transfiera los datos en formato *JSON* sino que use otro distinto como XML. Por todo ello no se ha podido realizar un parser genérico para todos los servicios.

El otro aspecto que habría que realizar sería la introducción del mapeo de las categorías del nuevo servicio en la base de datos local con el fin de poder categorizar los puntos que provengan de este nuevo punto. Para ello, lo único que habría que hacer es introducir las consultas SQL en el mensaje de la clase DataManager, creando para tal efecto **createCategoryQueries**.

El último detalle importante en la clase DataManager sería añadir en el método **requestDataFrom** la url del nuevo servicio para poder realizar peticiones.

H) Diseño y documentación framework de mapas

H1) Esquema general

En este framework se ha implementado lo relativo a la representación de los diferentes puntos de interés que han sido obtenidos por medio del framework de gestión de datos, sobre un mapa centrado sobre la posición actual del dispositivo.

En este framework no se hace un uso tan intensivo de los sensores del dispositivo como en el caso del framework de realidad aumentada, ya que en éste, solo es necesario el GPS para conocer la localización del dispositivo y poder centrar el mapa en dicho punto.

En un principio se han hecho pruebas con el framework Mapkit que está por defecto en iOS, que hasta la versión 5 utilizaba los mapas de Google, pero de la 5 en adelante utiliza su propio servicio de mapas. Una vez realizadas estas pruebas se ha decidido que era mejor la utilización del visor de mapas de *GeoSpatiumLab*, ya que ofrece algunas funcionalidades adicionales, como el acceso a diferentes servicios de mapas tileados. Pero la idea es que el cambio de un visor de mapas a otro no sea complicado y sean fácilmente intercambiables, para ello se ha creado una interfaz con la idea de las funciones básicas que el servicio en cuestión debería proporcionar.

Las funcionalidades principales serían:

- Representación de los puntos de interés sobre el mapa.
  - Representación de los puntos de interés que han sido descargados por el framework de gestión de datos.
  - Representación de la localización actual del dispositivo.
- Control de zoom y navegación por el mapa.
- Interacción con los objetos representados, captando el evento de tocado para mostrar la información que se dispone del mismo.
- Control del tipo de mapas sobre el que mostrar la información (normal, satélite e híbrida).
- Acceso a diferentes servicios de mapas como son Google Maps, Apple Maps, IDEZar, Cartociudad o PNOA.

H2) Diagrama de clases

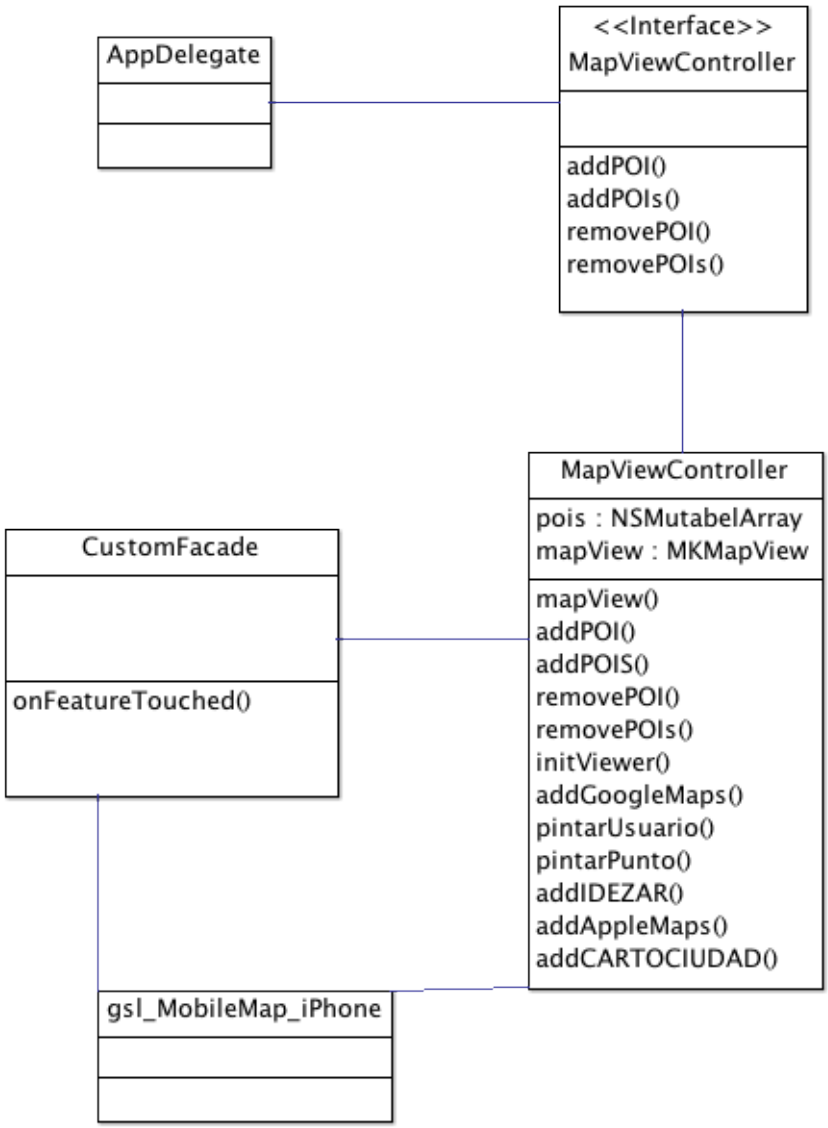


Figura 50. *Diagrama de clases framework de mapas*

Estas son las clases principales (fig. 50), en el diagrama no se han añadido los métodos por defecto utilizados para el control de un ViewController.

En primer lugar hay que comentar que el patrón de diseño seguido es el mismo que en el módulo de realidad aumentada.

Se ha definido un protocolo para indicar los métodos principales que el visor de mapas nos debería proporcionar:

```
@protocol MapViewControllerProtocol
//Añadir POIs obtenidos de las fuentes de datos
– (void)addPOI:(GeoPOIObject *)coordinate;
– (void)addPOIs:(NSArray *)newCoordinates;
//Borrado de datos
– (void)removePOI:(GeoPOIObject *)coordinate;
– (void)removePOIs:(NSArray *)coordinates;
– (void)setMapType:(int)mapType;
@end
```

Los métodos son básicamente los necesarios para añadir y borrar puntos de interés del mapa, pudiéndose añadir individualmente o en conjunto. El otro método se utilizaría para cambiar el tipo de mapa sobre el que se muestran los puntos (normal, satélite o híbrido).

MapViewController: Se trata de la clase principal en la que se implementan los métodos del protocolo y se inicializa la vista del mapa, en función del servicio de mapas elegido, de representar los diferentes POIs que rodean al usuario, así como al propio usuario. Una utilidad interesante que se ha implementado ha sido la del "usuario orientado"; esto consiste en que el usuario es representado por medio de una flecha, la cual se va orientando en tiempo real del mismo modo que el dispositivo.

CustomFacade: Se encarga de controlar los eventos de pulsado sobre las features del mapa. Como en el caso de los puntos, en el caso del framework de realidad aumentada, estos eventos son enviados hacia el controlador general de la aplicación

Gsl_MobileMap_iPhone: Se trata del framework, desarrollado por *GeoSpatiumLab*, utilizado como motor de renderizado de los mapas y de las features que representan los puntos de interés.

I) Diseño aplicación realidad aumentada

I1) Esquema general

Para el desarrollo de la aplicación de realidad aumentada que se desea implementar, se va a realizar una división en tres módulos bien diferenciados que se encargan de la realidad aumentada (AR), de la obtención de POIs (gestión de las fuentes de datos) y la visualización en mapas.

Todos estos módulos tienen una interfaz externa que hace que no existan dependencias entre ellos y puedan ser utilizados por separado o conjuntamente, según se desee, en futuras implementaciones.

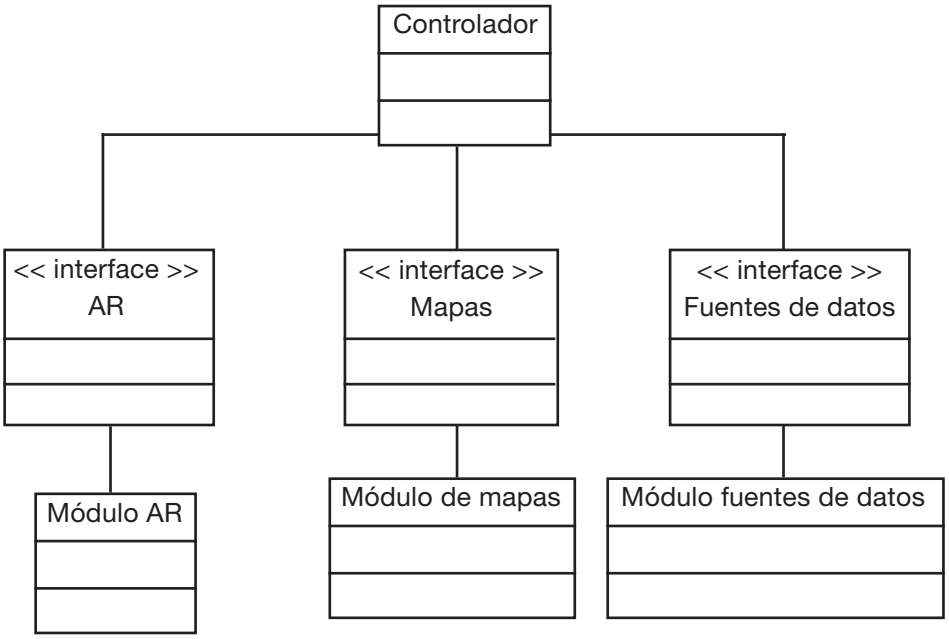


Figura 51. *Esquema conceptual de la utilización de los diferentes módulos implementados*

El diagrama anteriormente explicado (fig. 51) es un esquema conceptual de la arquitectura completa explicada, solo que en este caso está también el controlador que sería el encargado de crear instancias de cada uno de los módulos necesarios por parte de la aplicación que se esté implementando (en el caso implementado serían estos tres). Esto se debe a que la programación iOS se fundamenta en el modelo o patrón MVC (Modelo-Vista-Controlador) por lo que si algún elemento externo quiere comunicarse con alguno de los módulos lo hará por medio del paso de mensajes a través del controlador.

Para analizar y poder mostrar gráficamente todos los componentes hardware que intervienen en la aplicación de realidad aumentada se ha creado un diagrama de despliegue (fig. 52) con los diferentes elementos que intervienen:

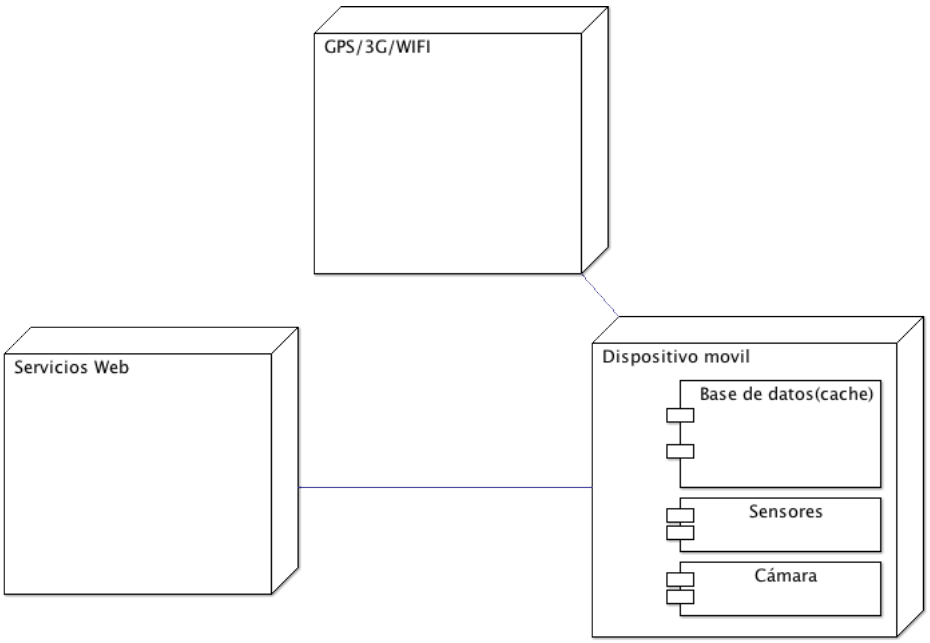


Figura 52. *Diagrama de despliegue de los diferentes componentes de la aplicación*

De los **Servicios Web** se obtienen los puntos de interés que rodean al usuario.

Por otro lado, estaría el sistema **GPS** del que se obtiene la localización geográfica del usuario, haciendo también uso del **3G** o el **Wifi** para obtenerla, cuando estén disponibles estos servicios, y tomando la que ofrezca mayor precisión en cada momento.

Por último, en el **dispositivo móvil** estarían una **base de datos** a modo de cache para reducir el tráfico de datos reduciendo el número de peticiones a los servicios web para obtener información y también permitir el funcionamiento offline cuando no esté disponible la red Wifi o 3G. También estarían los **sensores** como son: el acelerómetro, el giroscopio y la brújula que proporcionan la inclinación horizontal y vertical, así como la orientación del dispositivo; y la cámara.

I2) Esquema de navegabilidad

Como resultado del análisis de los frameworks analizados tanto para iOS como para iPhone, se ha observado que todos ellos siguen un esquema de navegabilidad muy similar (fig. 53), por lo que el esquema de la aplicación sigue un patrón similar.

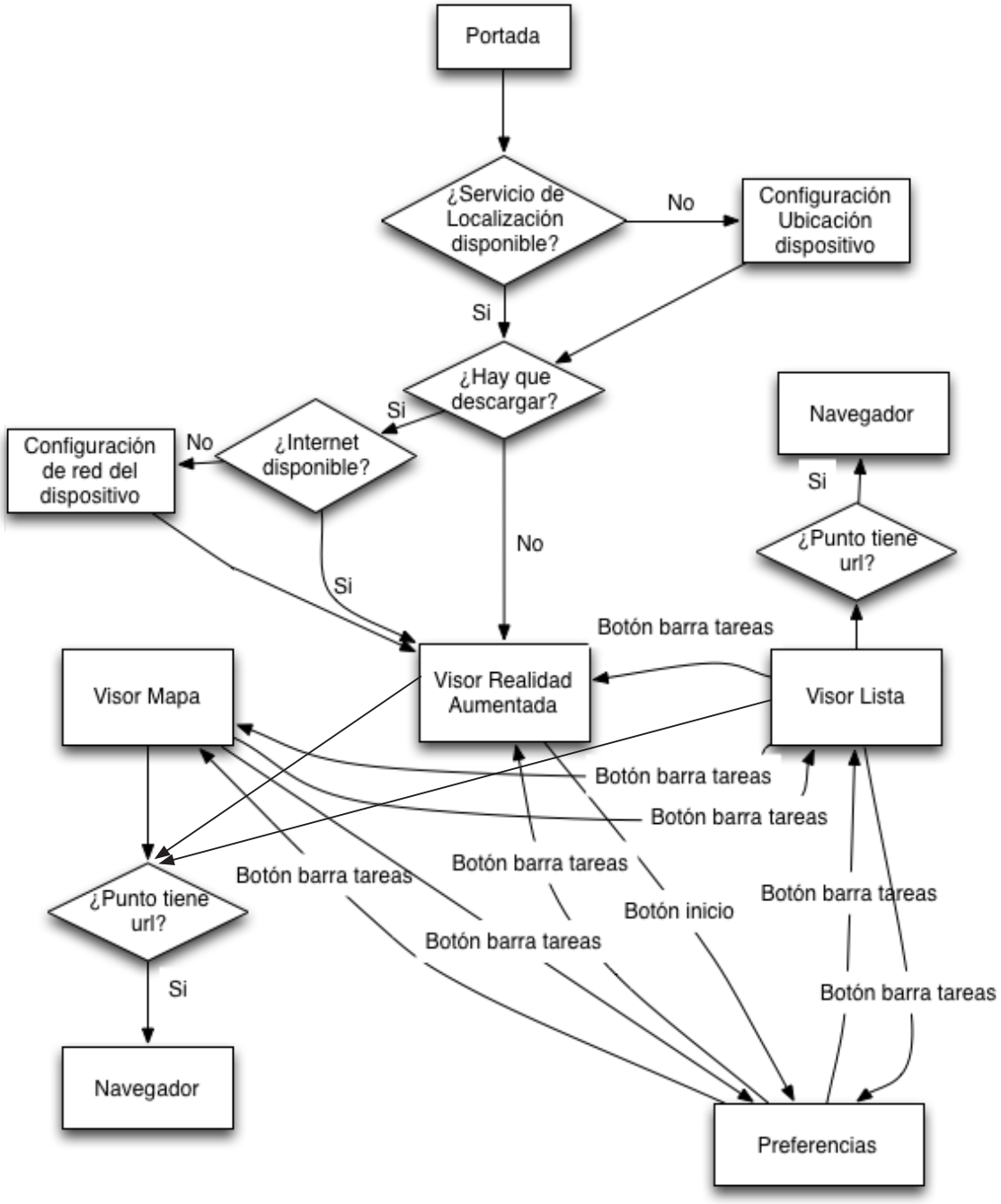


Figura 53. *Esquema de navegabilidad de la aplicación*

I3) Prototipado de pantallas

Un vez explicado el esquema de navegabilidad de la aplicación se va a mostrar el prototipado de cada una de las pantallas de ésta, que aparecían en el esquema y que correspondían con las cajas de éste.

Pantalla visor realidad aumentada

La primera pantalla al abrir la aplicación sería la siguiente (fig. 54):

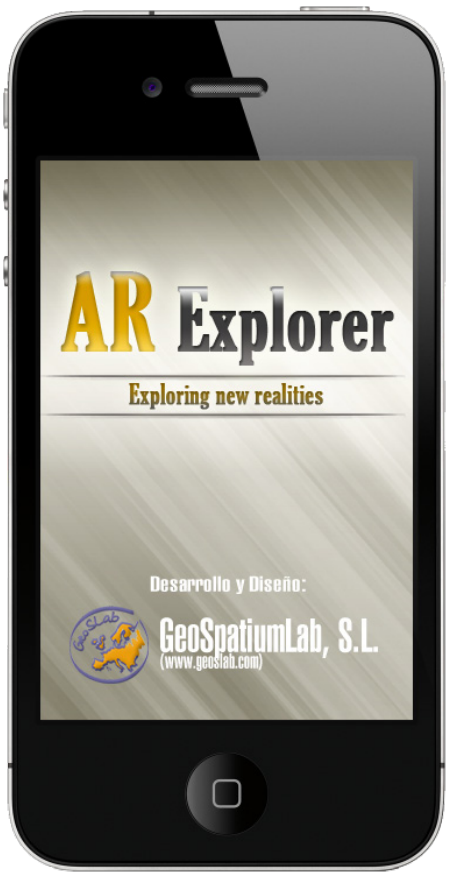


Figura 54. *Pantalla de inicio de la aplicación*

Una vez hecha toda la carga de datos y del visor de AR, aparecería la ventana principal de ésta (fig. 55):

Figura 55. *Pantalla principal AR*

A partir de esta pantalla se podría pasar a cualquiera de las pantallas de la aplicación gracias a la barra de menú inferior. Esta barra está presente en todas las pantallas, ya que se trata de un elemento típico de las aplicaciones en el sistema operativo iOS y por lo que los usuarios ya están acostumbrados a su manejo, lo cual hace que la interfaz sea mucho más sencilla y fluida de manejar. Así pues, la transición entre pantallas es realizada de una manera mucho más intuitiva ya que el modo de cambio es común en todos los casos.

Un aspecto importante sería la figura del radar que aparece en la esquina superior izquierda, se puede configurar de dos modos distintos: una en la que el radar está fijo y otra en la que el radar es móvil. Esta funcionalidad se explicará más detalladamente en la pantalla de preferencias.

El modo de interactuar con esta pantalla sería el siguiente: en primer lugar, en ésta se puede modificar directamente el radio de búsqueda, sin necesidad de ir a la pantalla de propiedades, pulsando el botón “Radius”, con el que aparecería la siguiente pantalla (fig. 56):

Figura 56. *Pantalla con selección de radio*

Como se puede ver sería idéntica a la anterior, solo que aparece una barra de escalado en la que se puede modificar dicho radio, siendo éste como máximo de 20 kilómetros.

Por otro lado, también se puede interactuar con los puntos que aparecen en la pantalla para obtener más información de ellos. En función de la información que haya almacenada o haya proporcionado el servidor, se pueden distinguir dos comportamientos distintos como se pueden ver en las siguientes dos imágenes (figs. 57 y 58). Esto se debe a que hay servicios de los que se utilizan en la aplicación, que proporcionan una url con una página con información del punto (en cuyo caso se muestra dicha url en el navegador a través de un botón) y en otros dicha url no está disponible, por tanto hay que mostrar la información que se disponga.

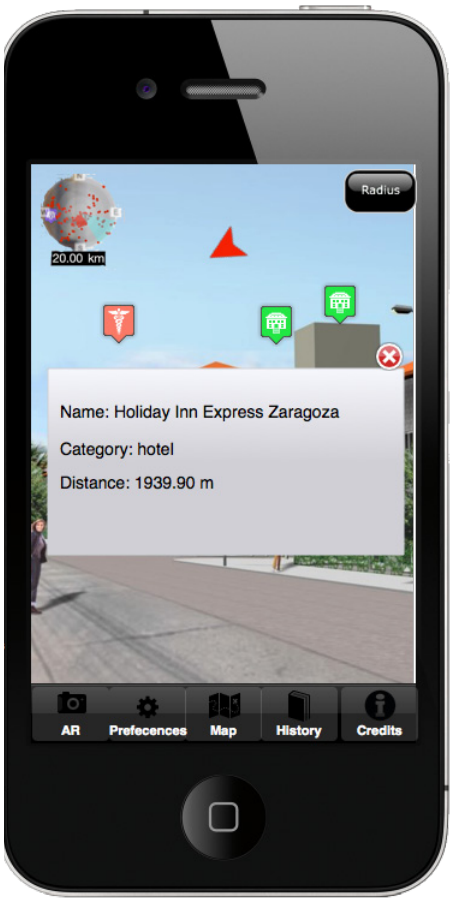


Figura 57. *Información de punto sin url*

Figura 58. *Información de punto con url*

En esta pantalla aparece un flecha roja con la que se fija el punto. Su utilidad radica en la necesidad de indicar la orientación del punto que se encuentra abierto cuando éste no está apareciendo en el visor de realidad aumentada.

Al pulsar el botón “Close” se vuelve de nuevo a la pantalla inicial y desaparece la flecha de orientación.

Pantalla de preferencias

Como se ha comentado en el apartado anterior, si se pulsaba el botón “Preferences” en la barra de menú, se accedía a la pantalla de preferencias que se encuentra integrada dentro del menú de la aplicación como se puede ver en la siguiente imagen (fig. 59):

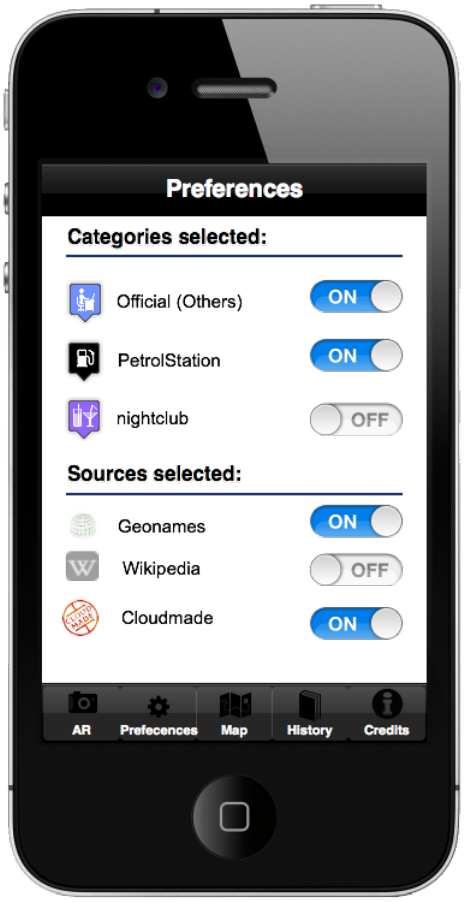


Figura 59. *Pantalla de preferencias*

En esta pantalla se encuentra ubicada toda la configuración de la aplicación. En ella se pueden seleccionar las categorías con las que se va hacer las búsquedas de los puntos de interés, activando solo las que le interese al usuario. También se pueden activar los servicios a los que se quiere acceder, ya que en esta aplicación se han incluido tres diferentes: Cloudmade y dos de Geonames distintos, uno de ellos con enlaces a artículos de Wikipedia georreferenciados.

Además de esto también se pueden configurar otros aspectos de funcionamiento interno de la aplicación como son: el modo de petición de datos que se implementó en el framework de gestión de datos, es decir, el modo internet, en el que los datos son siempre pedidos on-line a los diferentes servicios; el modo BBDD, que sería equivalente al modo off-line cuando no hay red disponible, en la que los puntos son siempre pedidos a la base de datos, y, por último, el modo mixto que sería una combinación de los otros dos anteriores.

Por último estaría la configuración del modo del radar del visor de realidad aumentada. Por un lado existe el modo fijo, en el que el radar no tiene movilidad, solo cambia la flecha de orientación azul del radar y por otro el modo móvil en el que esta flecha está fija y es el radar y los puntos los que se mueven. Así como la selección de servicio de mapas, pudiendo elegir entre el servicio por defecto (google, si el móvil es inferior a iOS 5 y Apple, si es posterior), idezar y cartociudad.

Pantalla visor de mapa

En esta pantalla se implementa un visor de mapas en el que se superpone una capa con los puntos de interés correctamente posicionados (figs. 60 y 61):

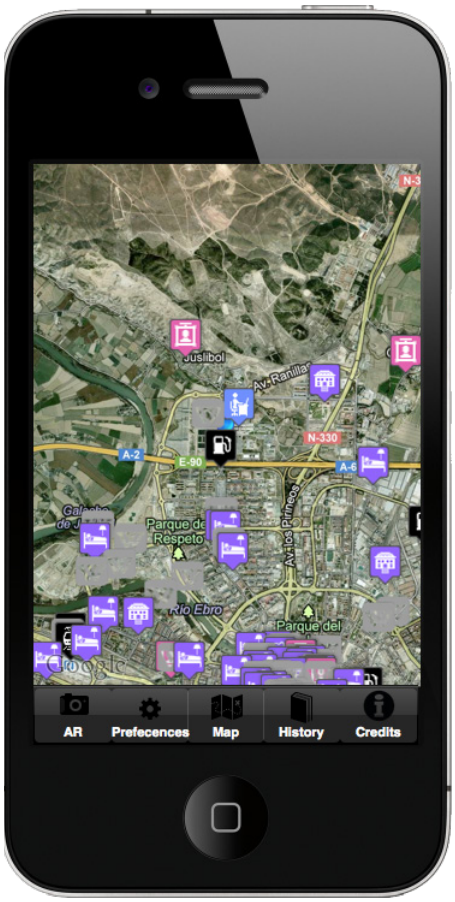


Figura 60. *Visor de mapas*

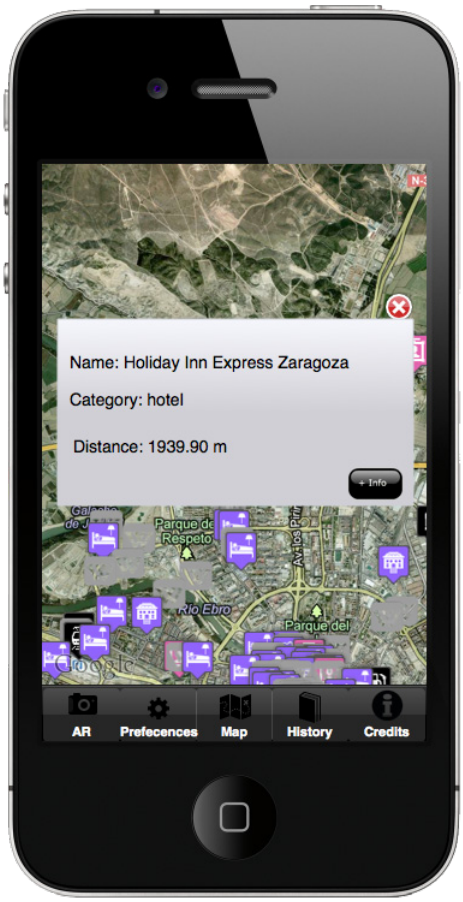


Figura 61. *Visor de mapas con información de POI abierto*

Como sucede en el visor de realidad aumentada, en este caso también se puede interactuar con los puntos de interés. Al pulsar sobre uno de ellos surge un pop-up con su información y en el caso de disponer de url, aparecerá un botón, que abre el navegador del dispositivo con la página referenciada en dicha url.

Pantalla visor lista

Esta última pantalla (fig. 62) sería una simplificación de las anteriores en las que se muestra un listado, ordenados por distancia desde el punto en el que se encuentre el usuario, de los puntos de interés:

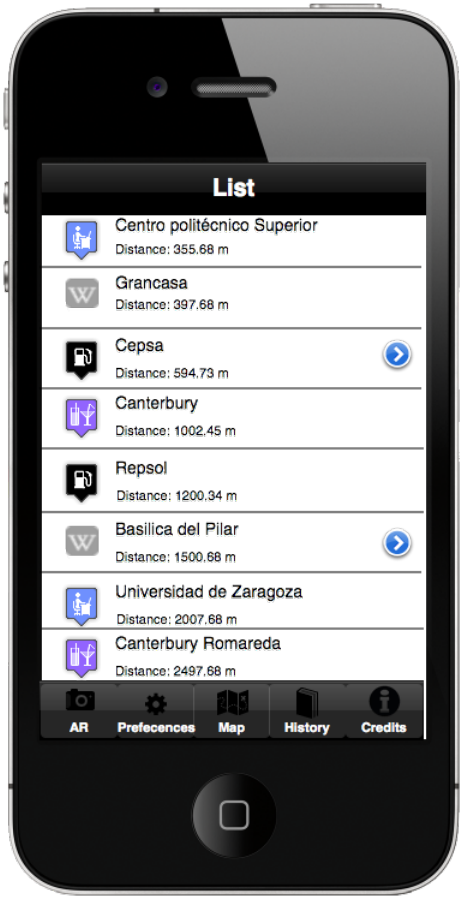


Figura 62. *Pantalla de listado de POIs*

En este caso la interacción con los puntos es muy simple, la información ya aparece en el propio listado y solo en el caso de que se disponga una url, aparece un botón para abrir en el navegador del dispositivo, la información asociada a dicho punto.

Por último estaría la pantalla de créditos de la aplicación (fig. 63) con el logotipo de la misma y con el de la empresa en la que se ha desarrollado, *GeoSpatiumLab, SL.*

Figura 63. *Pantalla de créditos de la aplicación*

Para finalizar si el usuario quisiera volver al visor de realidad aumentada, tan solo tendría que pulsar sobre el botón “AR” del menú principal.

Es importante comentar que toda la aplicación funcionaría en posición vertical del dispositivo, como se ha visto en las imágenes anteriores, y en la disposición horizontal del mismo.

Otros prototipados descartados

A continuación se van a presentar algunos prototipados descartados, y el motivo por el que no se optó por este modelo.

PROTOTIPO 1 (fig. 64):

Este modelo es similar al anterior, solo que la salida de la página de realidad aumentada se hace a través de un menú en la parte superior derecha, como se ve en la imagen.

Figura 64. *Resumen de diferencias del prototipado 1*

Este prototipo fue descartado debido a que la navegación no estaba unificada entre todas las pantallas, lo cual puede despistar al usuario y que sea menos ágil el manejo y la navegación de la aplicación.

PROTOTIPO 2 (fig. 65):

En este caso, el menú que en la ocasión anterior se usaba solo en la realidad aumentada, se extendió a todas las pantallas con el fin de solucionar el problema que tenía el prototipo anterior. A pesar de solucionar el problema, tiene el inconveniente de que son necesarios dos clicks para cambiar de pantalla en lugar de uno como en el caso del modelo elegido. Además no es un tipo de menú muy comúnmente usado de iOS, por lo que el usuario se puede mostrar confuso sobre la navegación al comienzo. A esto se añade el problema en el caso de las pantallas con forma de listado, ya que no se acopla perfectamente a la pantalla, debido a que al abrir el menú se oculta algo de información del listado de la pantalla.

Figura 65. *Resumen de diferencias del prototipado 2*

I4) Modelo estático-Diagrama de clases

Al comienzo de este apartado se mostró un esquema conceptual de lo que sería la arquitectura general de cualquiera de las aplicaciones que hicieran uso de los frameworks que se han implementado en este proyecto. Se va a mostrar el diagrama de clases de la aplicación (fig. 66) que se ha desarrollado en el que se pueden distinguir claramente como interactúan los diferentes componentes:

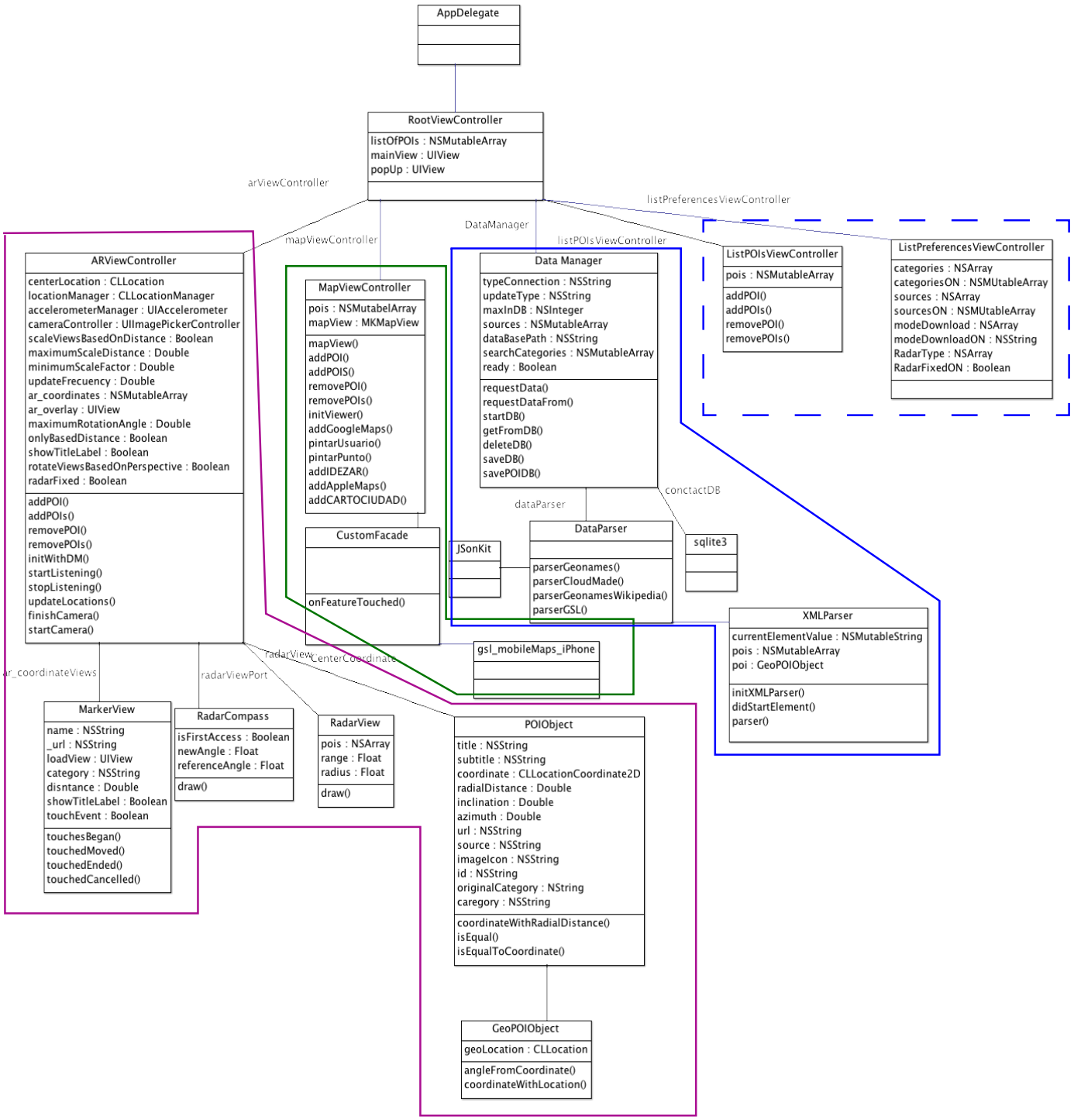


Figura 66. *Diagrama de clases simplificado de la aplicación ARExplorer*

Como se puede ver en la imagen se trata de un esquema simplificado de lo que sería la aplicación con todos sus módulos. En este diagrama se puede ver claramente como se ha utilizado el patrón de diseño fachada o facade en el que existen los diferentes módulos que son controlados por una única clase controladora o fachada llamada "RootViewController".

Por otro lado también se puede observar como se ha utilizado una estructura MVC (modelo-vista-controlador), ya que esto está definido por el propio sistema iOS como se comentó en otros apartados. Por un lado estaría el controlador, que como se ha comentado sería la clase “RootViewController”. Después estarían las vistas que serían el framework de AR delimitado en rojo, el de mapas (en verde) y que toma como base el motor de renderizado de mapas en iOS, desarrollado por GSL y las vistas correspondientes a las demás pantallas (listado de puntos y preferencias). Por último estaría el modelo, que sería el framework de gestión de datos señalado en azul oscuro.

J) Análisis de prestaciones de la gestión de los datos en el módulo de realidad aumentada

En este apartado se ha llevado a cabo un estudio para tratar de ver cual sería la mejor decisión de la política de gestión de datos procedentes de uno o varios servicios diferentes. Para la realización de estas pruebas se han elegido dos servicios muy utilizados, como son: GeoNames y CloudMade y que, analizando las diferentes posibilidades, se vio que eran los únicos que ofrecían lo servicios que se demandan. Pese a eso, la medición de tiempos en este análisis se ha realizado mayoritariamente con el segundo, debido a que ofrecía una mayor cantidad de información y a que era indiferente realizarlo con cualquiera de los dos en lo referido a los tiempos de descarga ya que estos eran muy similares en ambos casos para un mismo volumen de datos.

Como primera aproximación en la política de gestión de datos provenientes de alguno de los servicios REST de los que se dispone, se ha comenzado con la más sencilla posible. Es decir, si simplemente se dispone de una cache en memoria en la que los datos son pedidos al comienzo de la aplicación y cuando es detectado algún cambio en los parámetros de la petición (radio o coordenadas) se vuelve a realizar una nueva petición borrando todos lo datos antiguos. Se ha conseguido realizar peticiones de hasta 250 puntos diferentes con el servicio CloudMade, lo cual se ha considerado un valor suficientemente alto. Esto es debido a que en estos casos en los que además había que mostrar un número muy elevado de puntos por pantalla simultáneamente, el proceso de renderizado de la imagen ya no era tan fluido como en un caso normal, además de no tener mucho sentido, ya que al estar trabajando con dispositivos móviles (iPhone e iPad) la información no se podía mostrar de un modo claro.

Pese al problema que acabamos de comentar de la visibilidad de los puntos de interés, estas pruebas no carecen de sentido, ya que se pretendía llevar al límite en este contexto al servicio de gestión de datos.

A continuación se muestran los datos obtenidos de diferentes ratios obtenidos en el servicio CloudMade para diferentes volúmenes de datos; en caso de Geonames las conclusiones que se obtienen son las mismas solo que para volúmenes de datos inferiores, por lo que se ha considerado más oportuno realizar el análisis con el primero.

En primer lugar se presentan los tiempos medios de descarga para diferentes cantidades de datos (fig. 67):

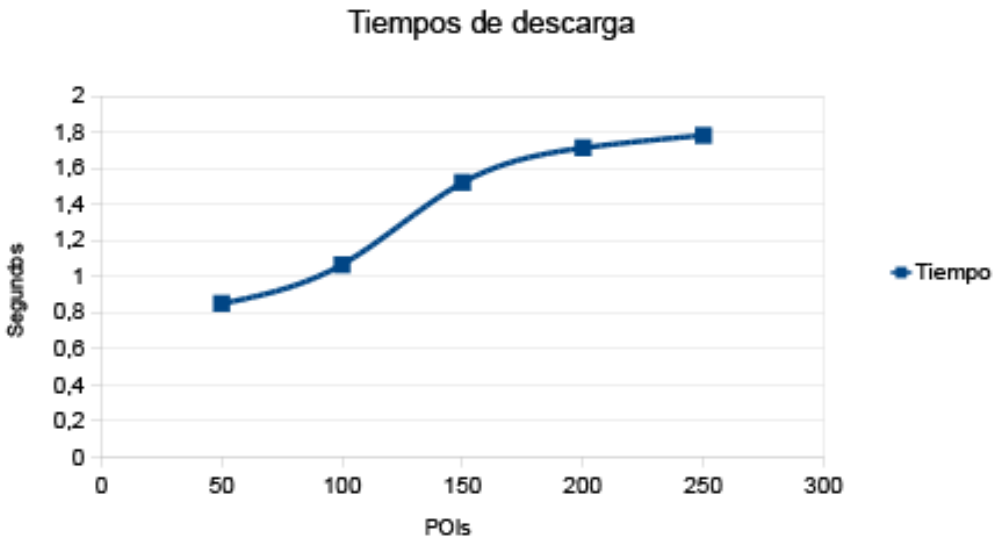


Figura 67. *Tiempo medio de descarga desde servidores externos*

Ahora se van a mostrar los costes unitarios de descarga en función de la cantidad de datos pedidos (fig. 68):

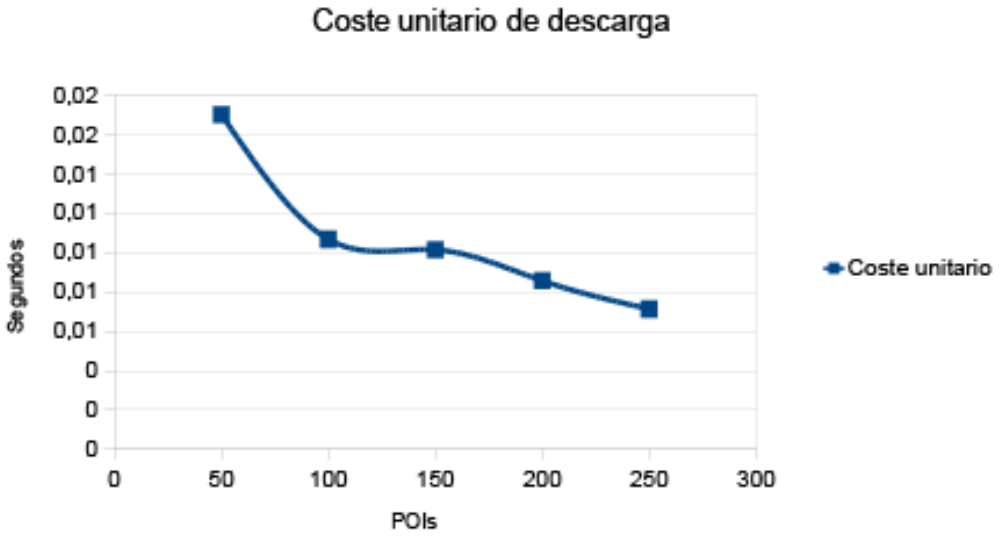


Figura 68. *Coste unitario de descarga desde servidores externos*

A la vista de los datos se puede observar que a mayor cantidad de datos el tiempo de descarga puede que sea algo mayor pero los costes unitarios se reducen, lo cual, debe ser algo a tener en cuenta a la hora de establecer una política de descarga u otra.

Éstos, eran los tiempos únicamente de descarga, ahora se van a analizar los de descarga y de carga en la cache de los datos:

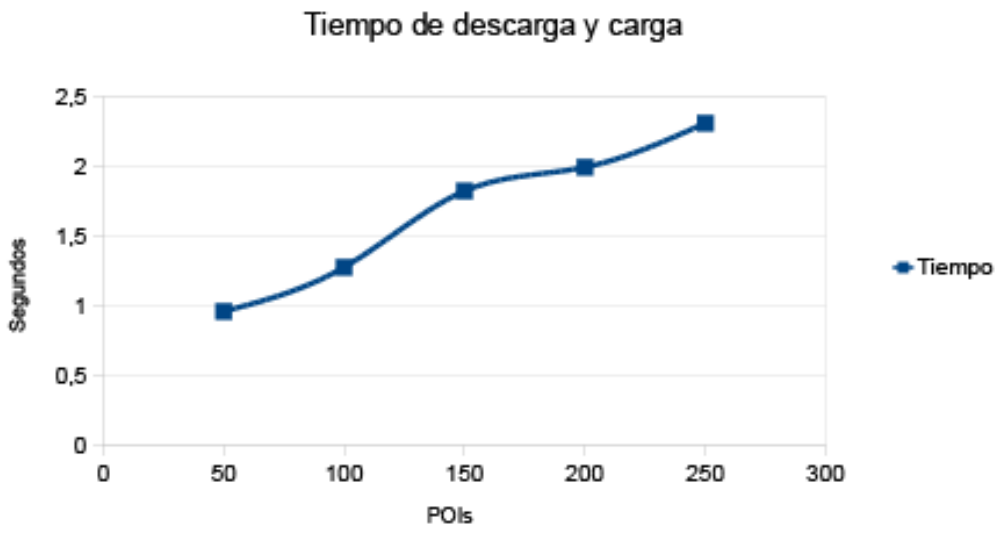


Figura 69. *Tiempo medio de descarga desde servidores externos y carga en cache local*

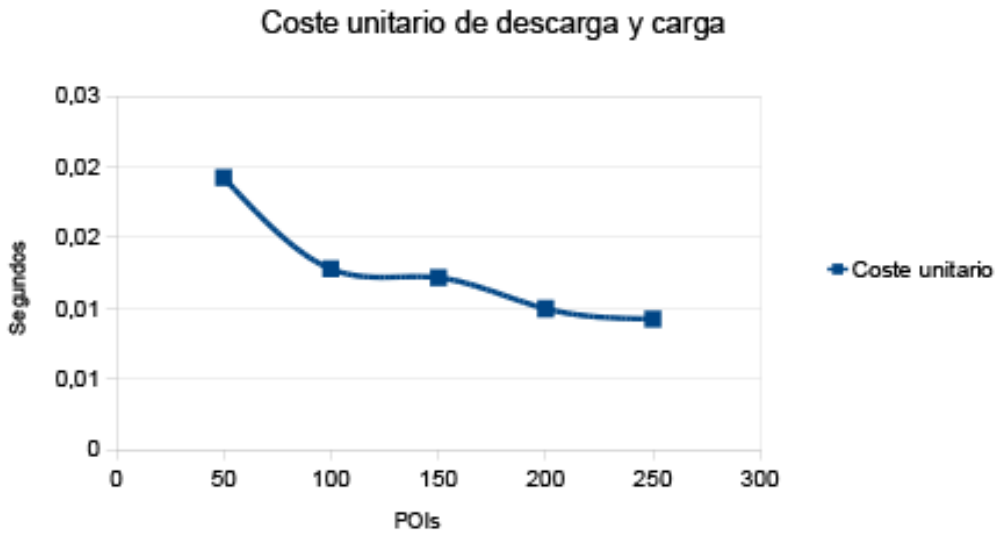


Figura 70. *Coste unitario de descarga desde servidores externos y carga en cache local*

En las dos gráficas anteriores se puede observar que tienen la misma forma que en la de los tiempos de descarga (figs. 69 y 70), de esto se deduce que los costes unitarios de carga en la cache se comportan de una manera aproximadamente lineal con respecto a la cantidad de datos a introducir, como se puede ver en las siguientes gráficas (figs. 71 y 72):

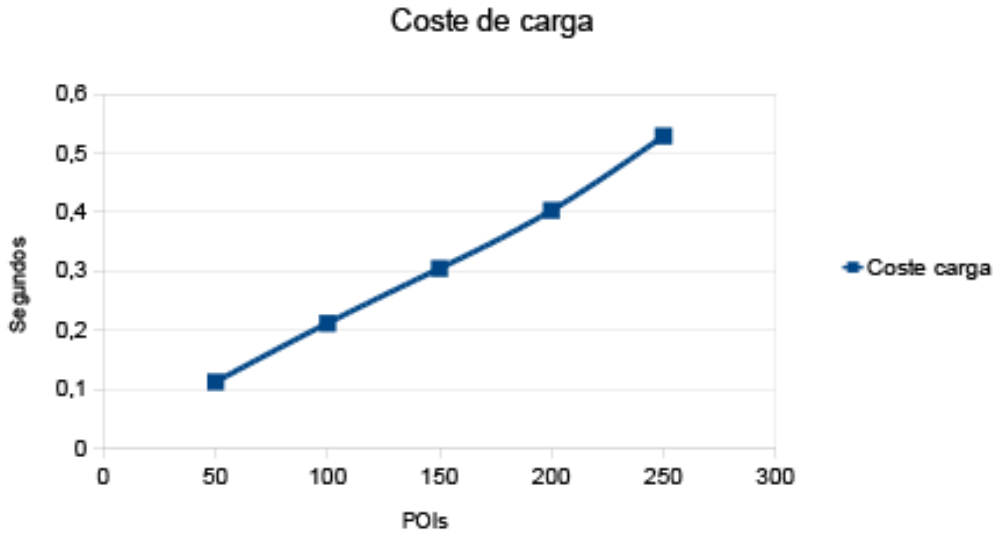


Figura 71. *Tiempo medio de carga en cache local*

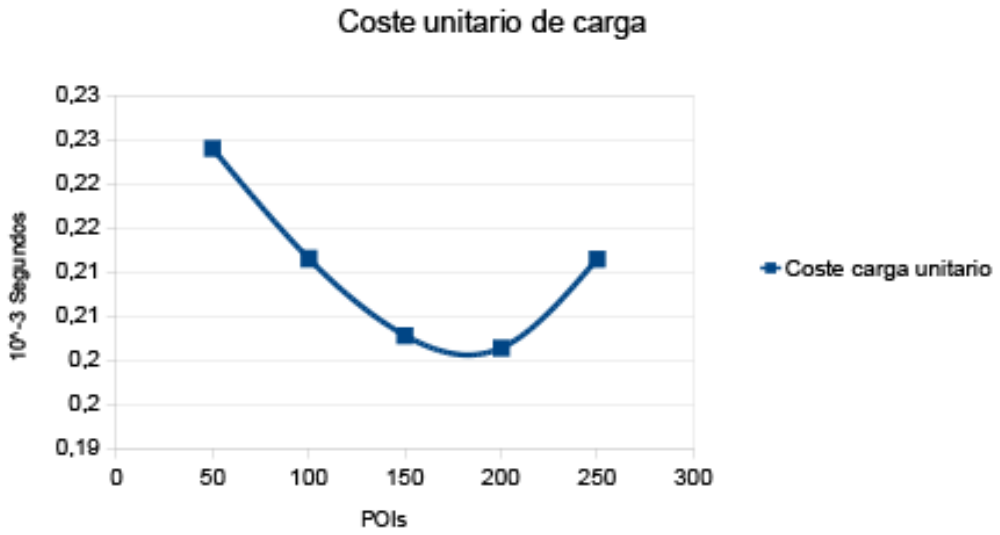


Figura 72. *Coste unitario de carga en cache local*

En esta última, a pesar de que parezca que no es un comportamiento constate, hay que constatar que las diferencias son inferiores al milisegundo, los costes unitarios si que se podrían considerar constantes, no obstante se ve que a partir de ese 200 puntos aumenta ligeramente.

Por último se van a mostrar los datos de los tiempos de todo el proceso del refrescado de la pantalla en el momento que se realiza una nueva petición (figs. 73 y 74):

Figura 73. *Tiempo medio de renderizado de POIs*

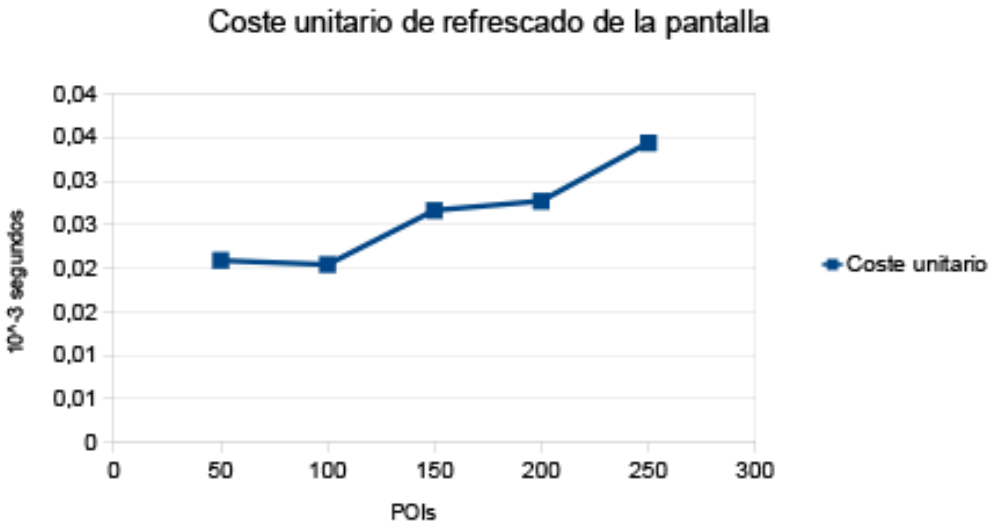


Figura 74. *Coste unitario de renderizado de POIs*

A la vista de las gráficas anteriores se puede observar que el coste unitario comienza a aumentar en el momento en el que el número de puntos empieza a ser muy grande, esto explicaría el comportamiento no tan fluido, que se explicaba al comienzo de este análisis.

A la vista de los resultados que se han observado en el análisis, se puede ver que el tiempo crítico que hay que tratar de reducir es el tiempo de descarga, el cual no es fijo sino que oscila en algunos entre valores de 0,5 segundos y casi 3 segundos para un mismo volumen de datos.

Observando esto, se puede decir que lo que habrá que intentar mejorar será la política de petición de datos, y habrá que tratar de reducir los tiempos de alguna manera reduciendo el número de peticiones o la cantidad de datos a pedir en cada petición, para lo que se ha pensado en aprovechar el sistema de base de datos que iba a ser necesario implementar para poder desarrollar el funcionamiento off-line de la aplicación, reduciendo así la cantidad de datos a pedir en cada petición intentando aprovechar los datos que ya estaban disponibles y no volviéndolos a pedir, siempre y cuando los servicios empleados nos permitan realizar este tipo de consultas.

J1) Implantación de la base de datos local

Como se ha comentado en el apartado anterior, con el fin de mejorar la política de petición de datos y así reducir el número de peticiones al servidor, se ha decidido implementar una base de datos local. Para justificar su implantación se han medido los tiempos de descarga de los datos de la base de datos y de carga en la cache en memoria en la que se leen los datos, teniendo como resultado (fig. 75):

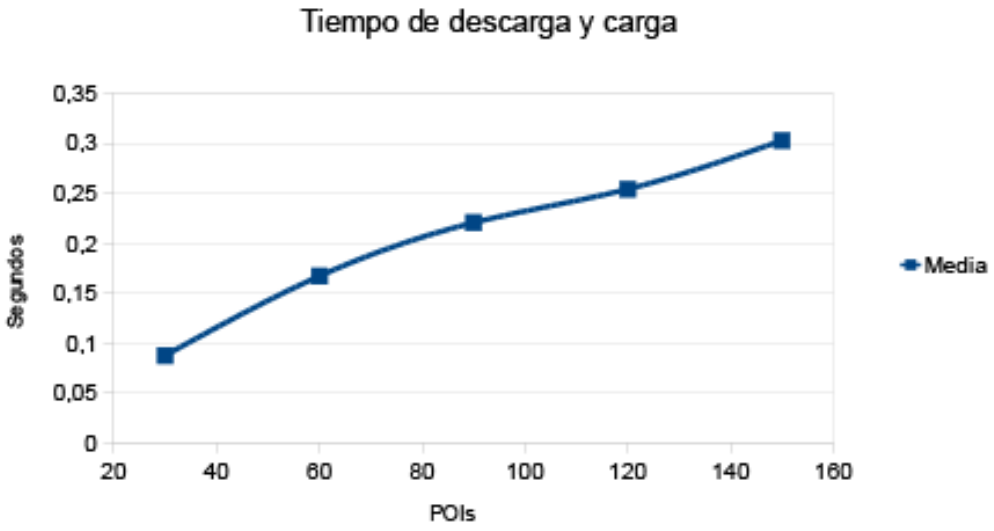


Figura 75. *Tiempos medios de obtención de POIs de la base de datos y carga en la cache local*

Como se puede ver en las gráficas, el tiempo hasta que los datos se encuentran disponibles en memoria listos para ser representados se reduce, además de reducir el número de descargas. Para verlo mejor se han superpuesto las dos gráficas (fig. 76):

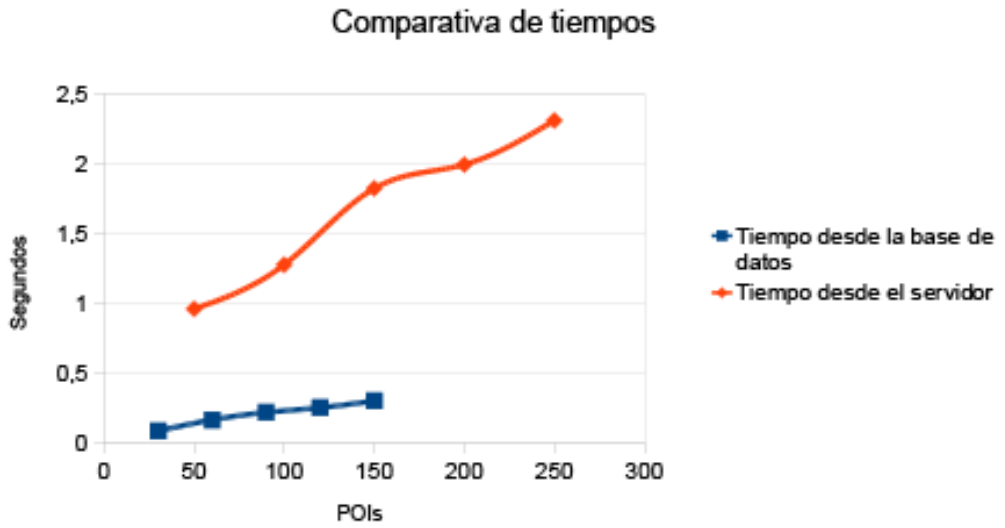


Figura 76. *Costes unitarios de obtención de POIs de la base de datos y carga en la cache local*

Como se puede ver, los tiempos se reducen notablemente además de disponer en este caso de la persistencia necesaria para implantación del modo offline de la aplicación. Por otro lado, los tiempos en el caso de que los datos de la base de datos no sean válidos y tenga que hacerse la petición al servidor aumentarán con respecto a la ausencia de base de datos. Para solucionar este problema habrá que elegir una buena política de petición de datos reduciendo el número de peticiones y evitar esas mayores esperas, así como reducir el consumo de datos.

Para ello se va a realizar un análisis de tiempo, del caso crítico en el que haya que eliminar los datos no válidos de la base de datos, hacer la petición al servidor y guardar los nuevos datos válidos en la base de datos (fig. 77).

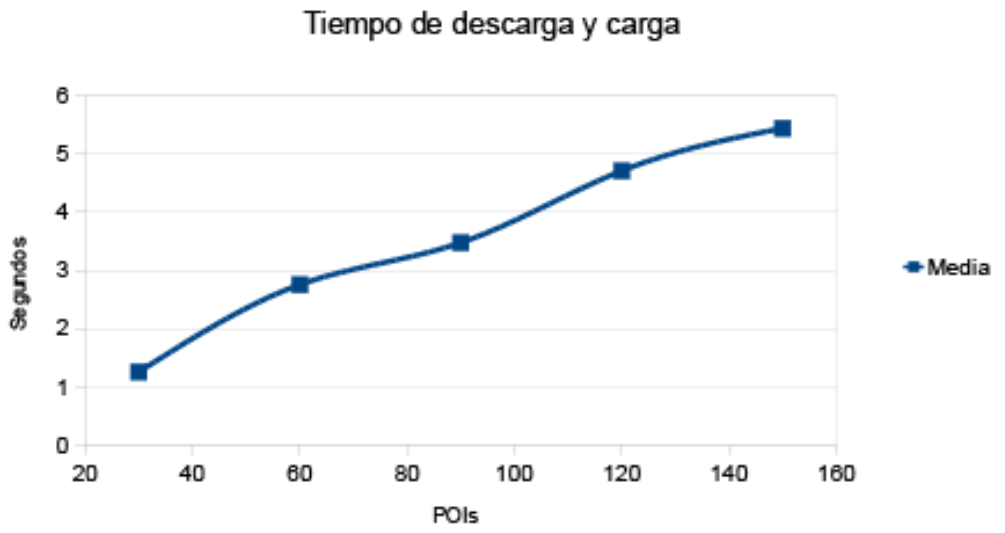


Figura 77. *Tiempos medios de obtención de POIs de servidores externos con borrado y recarga de datos en la base de datos local*

En la imagen se ve el caso crítico que se ha comentado antes; como se puede observar, los tiempos aumentan considerablemente, aunque también es cierto que este caso no se debería dar muchas veces si se elige una buena política de actualización de datos.

Los tiempos de las tres situaciones se podrían comparar en la siguiente gráfica (fig. 78):

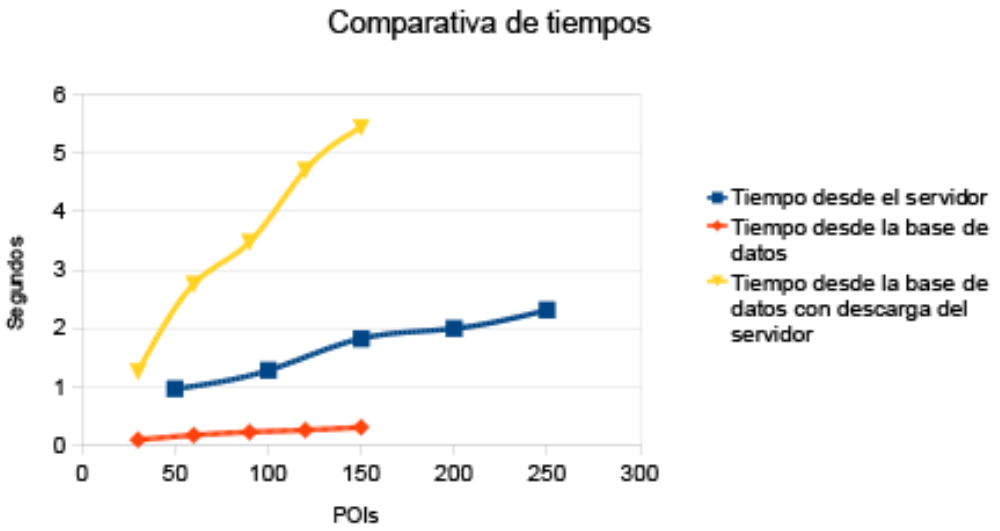


Figura 78. *Comparativa de tiempos entre los tres modelos analizados*

J2) Implantación de la base de datos local y multitarea

Para intentar mejorar los tiempos de la peor situación que se ha comentado en el apartado anterior se ha pensado paralelizar el proceso de guardado en la base de datos y de refrescado de la pantalla en cuanto se reciban los datos del servidor, por lo que los tiempos en el caso de que hubiera que hacer petición al servidor, serían los de la gráfica naranja y en el caso peor que hubiera que renovar los datos de la base de datos con nuevos del servidor tendría tiempos aproximados a la gráfica azul e incluso se podría llegar a mejorar.

Todo ello se ha implementado por medio del sistema de colas asíncronas existentes en iOS. Al existir concurrencia aparece un nuevo problema que es el acceso simultáneo a la base de datos que no está bien soportado por la base de datos implementada sqlite, pero tampoco está bien soportado por ninguno de los otros sistemas de persistencia de datos disponibles para iOS que se analizaron como CoreData que está implementado dentro del propio iOS SDK. Por lo que ha habido que implementar también un acceso en exclusión mutua a dicha base de datos mediante la directiva existente para tal uso @synchronized. En todo caso, este acceso concurrente es muy extraño, ya que solo se podría dar en caso de que el usuario estuviera cambiando de radio constantemente en intervalos inferiores al segundo, lo cual no es un uso normal de la aplicación. Con todo ello, si esta situación se diera lo único que se notaría es que en esa consulta en particular no se devolvían puntos. Para solucionar esto, lo que se ha hecho es que cuando se dé este caso, se haga una petición al servidor directa, sin necesidad de guardar en la base de datos, ya que los datos en ese momento ya se estaban guardando en segundo plano.

Con este nuevo modelo se ha realizado un nuevo estudio de los tiempos de descarga del servidor y de la carga en memoria de los datos necesarios para poder mostrar los nuevos datos (fig. 79):

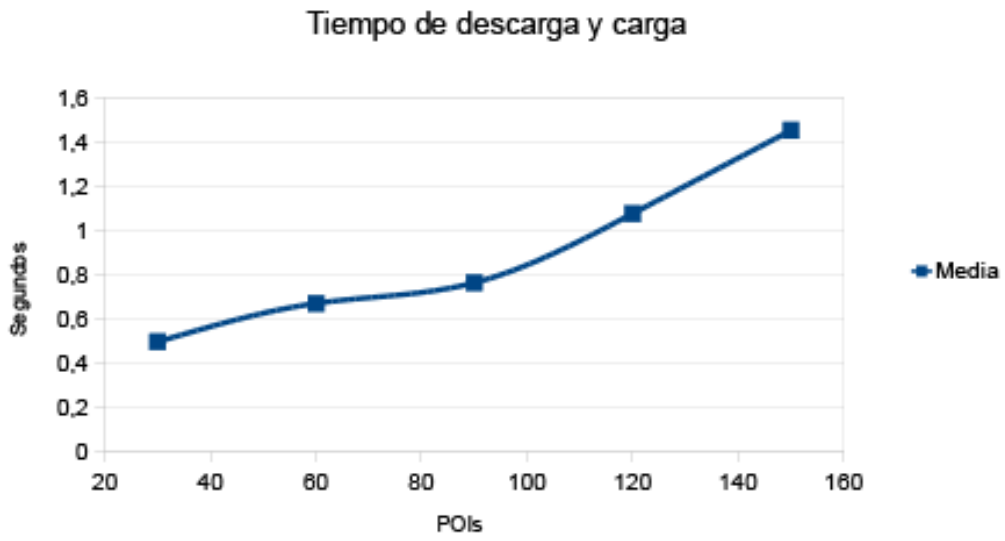


Figura 79. *Tiempos medios con la utilización de multitarea*

En la siguiente gráfica se puede ver claramente como los tiempos se reducen considerablemente (fig. 80):

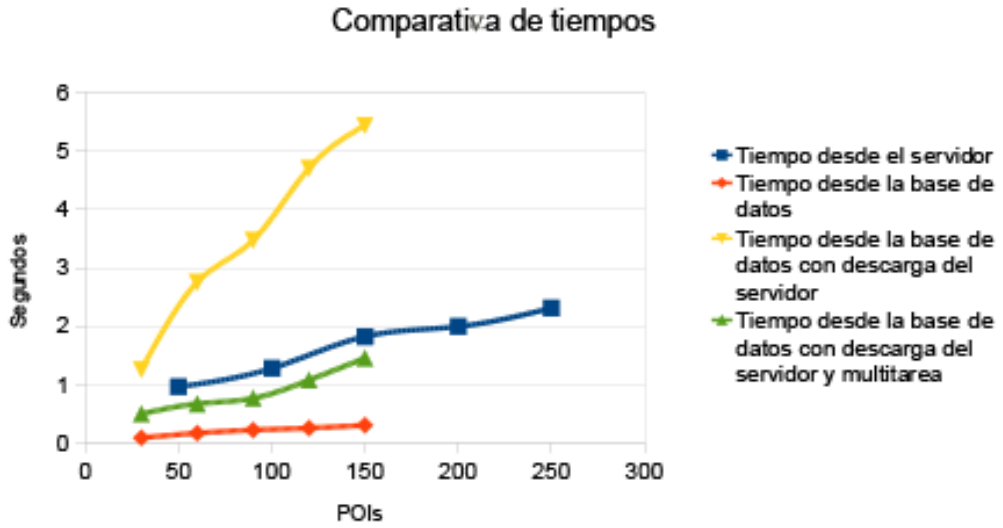


Figura 80. *Comparativa de tiempos entre los diferentes modelos de gestión de datos*

Como se puede ver en la gráfica, el mejor caso y además más frecuente sería la gráfica naranja y el caso crítico, que se explicaba antes y que era la de color amarillo, se ha reducido bastante siendo en este nuevo modelo la gráfica de color verde, en la que por ejemplo para 150 POIs el tiempo pasa de 5.4417 segundos a 1.4522, suponiendo una rebaja del 73.31%.

K) Pruebas de funcionalidad e integración

Se van a detallar todas las pruebas realizadas al sistema desarrollado. El objetivo de estas pruebas ha sido la de descubrir fallos de implementación y el aseguramiento del cumplimiento de los requisitos funcionales y no funcionales del sistema.

Así, pues, se ha elaborado una batería de pruebas que cubre la mayoría de los escenarios posibles en la utilización de la aplicación y de los frameworks. Además, después de cada una de las iteraciones, como se ha comentado en apartados anteriores, se han realizando pruebas unitarias que garantizaban el buen funcionamiento de las funcionalidades añadidas en cada iteración.

Las pruebas realizadas fueron:

K1) Pruebas generales o comunes

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
Nombre de la aplicación		Instalar la aplicación	ARViewer
Portada		Abrir la aplicación	Se muestra la portada con la resolución adecuada y se visualizan correctamente el texto
Landscape left	Cada una de las ventanas de la aplicación salvo la ventana de créditos	Giramos el móvil para que quede horizontalmente	Comprobamos que la aplicación gira en cada una de las pantallas, salvo la de créditos
Portrait	Cada una de las ventanas de la aplicación	Giramos el móvil para que quede verticalmente	Comprobamos que la aplicación gira en cada una de las pantallas
Landscape right	Cada una de las ventanas de la aplicación	Giramos el móvil para que quede verticalmente	Comprobamos que la aplicación no gira en cada una de las pantallas
Menú - Salir	Aplicación cargada	Pulsar el botón menú del dispositivo	Cierra la aplicación
Selector de categorías	Aplicación cargada en la pantalla de preferencias	Seleccionar categorías, en la pantalla de preferencias	Las categorías de los POIs representados en el modo AR, lista o mapa, se corresponden solo con las activadas en la pantalla de preferencias
Selector de fuentes de datos	Aplicación cargada en la pantalla de preferencias	Seleccionar fuentes de datos, en la pantalla de preferencias	Los POIs representados en el modo AR, lista o mapa, proceden de las fuentes de datos activados en la pantalla de preferencias
Tipos de descarga	Aplicación cargada en la pantalla de preferencias	Seleccionar tipo de descarga, en la pantalla de preferencias	Los POIs son obtenido de la base de datos o de los servicios, en función de la modalidad activada en la pantalla de preferencias
No hay conexión de datos y no hay POIs locales	No hay conexión de datos y se desinstala la aplicación para borrar los datos locales	Entra el modo offline	Aparece la pantalla de AR, con el visor de la cámara sin puntos, en el resto de visores, se muestran vacíos (mapa vacío o lista vacía)

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
No hay conexión de datos y la capa está almacenada localmente.	Cargas todas las capas y desactivas la conexión a internet	Entra el modo offline	Aparecen en la pantalla de AR los POIS que hubiera en la base de datos y que cumplieran las condiciones de posición y distancia, al igual que en el resto de visores
Pruebas idioma inglés	Poner el idioma del móvil en inglés u otro idioma que no sea español	Utilizar toda la funcionalidad de la aplicación	Comprobar que todas las cadenas en inglés son correctas y no se altera la estructura del gui
Pantalla de créditos: vertical	Cualquier pantalla de la aplicación	Pulsar botón de créditos	La pantalla de créditos aparece (centrada con espacios en negro si no cubre toda la pantalla)
Apertura inicial del visor de AR	Aplicación cerrada	Abrir la aplicación	Una vez cargada la aplicación, automáticamente se activa el visor de realidad aumentada
Apertura por menú del visor de AR	Aplicación en otra pantalla distinta a la de AR	Pulsar en el menú la opción de AR	Se carga el visor de AR
Apertura por menú del visor de mapa	Aplicación en otra pantalla distinta a la del mapa	Pulsar en el menú la opción de mapa	Se carga el visor de mapa
Apertura por menú del visor de lista de POIs	Aplicación en otra pantalla distinta a la del listado	Pulsar en el menú la opción de listado	Se caga la lista de POIs
Pantalla de créditos no gira a landscape	Aplicación en pantalla créditos y móvil en protrait	Girar la aplicación	La pantalla de créditos no gira
Pantalla de créditos no se abre en landscape	Aplicación en otra pantalla distinta a la de créditos y móvil en posición landscape	Pulsar botón del menú de créditos	La pantalla no cambia

K2) Pruebas del visor de AR

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
Los POIs se visualizan correctamente		Cargar visor AR	Al iniciarse el visor AR, los POIs empiezan a posicionar sobre el visor de la cámara
La solo se muestran los POIs visibles en la orientación del dispositivo		Orientar el dispositivo hacia una orientación de la que se conocen los POIs visibles	Solo se muestran los POIs con esa orientación y no otros
La posición de los POIs en la pantalla es la correcta en la pantalla, en función de la distancia		Orientar el dispositivo hacia una orientación en la que haya POIs a diferentes distancias	Los POIs más cercanos se muestran en la parte inferior de la pantalla y los alejados en la parte superior, de una manera gradual. El tamaño del POI también se reduce en función del aumento de la distancia con el fin de simular esta distancia.

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
Al pulsar un POI sea señalado		Pulsar un POI	El POIs pulsado, es remarcado con un contorno amarillo y aparece una flecha que señala al POI seleccionado, con el fin de indicar su orientación en el caso de que no aparezca en pantalla porque se gire el dispositivo
Al pulsar un POI se muestre su información en un popup		Pulsar un POI	Al pulsarse el POI, aparece sobreimpresionado un popup con la información del POI seleccionado
Se puede acceder a la web de un POI, si está disponible		Pulsar un POI con información web disponible	Al abrirse el popup, aparece un botón si el POI dispone de web y al pulsarlo se abre la aplicación del navegador del dispositivo
Se puede cerrar el popup de información	POI pulsado y popup abierto	Pulsar botón de cerrado	El popup dispone de un botón en la parte superior derecha, que el pulsarlo, cierra el popup, deselecciona el POIy la flecha de orientación desaparece
Se puede volver de la aplicación del navegador del dispositivo cuando se accede a la web de un POI	POI pulsado con web y navegador abierto	Pulsar dos veces seguidas el botón menú del dispositivo	Aparecen las aplicaciones activas del dispositivos, y al pulsar sobre ARViewer se vuelve al estado anterior de la aplicación
Los POIs son actualizados al cambiar el radio		Pulsar el botón de cambio de radio	Al cambiar de el radio, la posición y el tamaño de los POIs en pantalla se ve modificado y aparecen o desaparecen puntos en función de si se amplia o se reduce el radio respectivamente
Los POIs aparecen correctamente con subtítulo	Opción de subtítulos en pantalla de preferencias activa	Abrir visor de AR	Los POIs se representan en pantalla, con un icono de la categoría a la que corresponda el punto y bajo este aparece el nombre del punto
Los POIs aparecen correctamente sin subtítulo	Opción de subtítulos en pantalla de preferencias desactivada	Abrir visor de AR	Los POIs se representan en pantalla, con un icono de la categoría a la que corresponda el punto
Funcionamiento correcto del radar fijo	Opción de radar en pantalla de preferencias en modo fijo	Abrir visor de AR	En este modo los POIs aparecen fijos sobre el radar y es la brújula la que va girando en función de la orientación del dispositivo indicando los puntos que son visibles en cada momento
Funcionamiento correcto del radar móvil	Opción de radar en pantalla de preferencias en modo fijo	Abrir visor de AR	En este modo los POIs van girando en función de la orientación del dispositivo y la brújula permanece fija e indicando los puntos que son visibles en cada momento
Representación correcta de los POIs en el radar con la brújula del mismo		Abrir el visor de AR	Los POIs se dibujan con el radar y coinciden los puntos que están viendo en AR, con los que marca la brújula del radar, en cada momento en función de la orientación del dispositivo

K3) Pruebas del visor en listado

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
Se puede acceder a la web de un POI, si está disponible		Pulsar un POI con información web disponible	Al abrirse el popup, aparece un botón si el POI dispone de web y al pulsarlo se abre la aplicación del navegador del dispositivo
Los POIs se encuentran ordenados en función de la distancia al usuario		Listado de POIs abiertos	Los primeros POIs que aparecen en el listado son los más cercanos al usuario y se van posicionando en relación a la distancia al mismo

K4) Pruebas del visor de mapas

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
Visualizar con mapa IDEZar		Entrar en la aplicación	Se muestra el mapa con cartografía de IDEZar
Visualizar con mapa Google Maps		Cambiar la capa base del mapa a Google (o Apple en iOs 6)	Se muestra el mapa con cartografía de GoogleMaps centrado en el mismo punto en el que se encontraba el mapa
Zoom out		Pulsar con dos dedos y separarlos	El mapa se 'aleja' hasta el nivel de escala previo
Zoom in		Pulsar con dos dedos y juntarlos	El mapa se 'acerca' hasta el nivel de escala posterior
Panning		Pulsar y arrastrar	El mapa se mueve donde el usuario soltó
Max extent		Pulsar y arrastrar hasta alcanzar el extent máximo	No permite 'moverse' más allá
Max zoom out		Pulsar con dos dedos y separarlos hasta llegar al máximo	No permite 'alejarse' más
Max zoom in		Pulsar con dos dedos y juntarlos hasta llegar al máximo	No permite 'acercarse' más
Centrar en localización de usuario		Georreferenciarse	El mapa hace un panning automático hasta la posición del usuario

K5) Pruebas de conexión a los servicios

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
POIs – Actualización datos estáticos inicial correcta	No hay datos en la base de datos	Cargar POIs	(Aparece un diálogo de progreso con mensajes indicando el proceso en caso de no terminar la actualización en la portada). Los datos se descargan y procesan correctamente y se muestran en el modo de visión que se desee
Los POIs, son descargados correctamente de Geonames	No hay datos en la base de datos y selecciona en pantalla de preferencias el servicio Geonames	Cargar POIs	(Aparece un diálogo de progreso con mensajes indicando el proceso en caso de no terminar la actualización en la portada). Los datos se descargan y procesan correctamente y se muestran en el modo de visión que se desee POIs procedentes de Geonames
Los POIs, son descargados correctamente de Cloudmade	No hay datos en la base de datos y selecciona en pantalla de preferencias el servicio Cloudmade	Cargar POIs	(Aparece un diálogo de progreso con mensajes indicando el proceso en caso de no terminar la actualización en la portada) Los datos se descargan y procesan correctamente y se muestran en el modo de visión que se desee POIs procedentes de Cloudmade
Los POIs, son descargados correctamente de Geonames-Wikipedia	No hay datos en la base de datos y selecciona en pantalla de preferencias el servicio Geonames-Wikipedia	Cargar POIs	(Aparece un diálogo de progreso con mensajes indicando el proceso en caso de no terminar la actualización en la portada). Los datos se descargan y procesan correctamente y se muestran en el modo de visión que se desee POIs procedentes de Geonames-Wikipedia
Los POIs, son descargados correctamente del servicio propio de GSL	No hay datos en la base de datos y selecciona en pantalla de preferencias el servicio de GSL	Cargar POIs	(Aparece un diálogo de progreso con mensajes indicando el proceso en caso de no terminar la actualización en la portada). Los datos se descargan y procesan correctamente y se muestran en el modo de visión que se desee POIs procedentes del servicio propio de GSL
Se toman los datos de los que se disponga en la base de datos si no hay conexión a un servicio	No hay conexión con uno de los servicios	Cargar POIs	(Aparece un diálogo de progreso con mensajes indicando el proceso en caso de no terminar la actualización en la portada). Los datos se obtienen de la base de datos y de los otros servicios disponibles y procesan correctamente y se muestran en el modo de visión que se desee POIs procedentes del servicio propio de GSL

K6) Pruebas de rendimiento

DESCRIPCIÓN/ PROPÓSITO	PRECONDICIONES/ ACCIONES PREVIAS	ACCIONES DE EJECUCIÓN	RESULTADOS ESPERADOS
Medición de gasto de energía y de memoria	Medir con programa Instrument de Xcode	Realizar una ejecución completa de la aplicación	El consumo de memoria y cpu es razonable

K7) Resultados

Todas las pruebas expuestas anteriormente fueron realizadas en su conjunto al finalizar el desarrollo, siendo superadas con éxito cada una de ellas. No obstante, como se puede ver, estas pruebas están agrupadas en varios apartados, clasificados por las diferentes partes del proyecto. Así pues, también fueron realizadas cuando se finalizaba cada una de estas partes, no superándose algunas y teniéndose que solucionar estos problemas.

Los problemas más importantes, que fueron solventados con la realización de estas pruebas han sido:

– En las pruebas de rendimiento, en lo referido al gasto de memoria no era muy eficiente, provocando fallos por falta de memoria cuando la aplicación llevaba un tiempo utilizándose. Esto es debido a que la gestión automática de memoria en iOS se introdujo en su versión 5 y el desarrollo del proyecto se comenzó a realizar con la versión 4, por lo que la gestión de memoria era controlada explícitamente por el desarrollador. Esto se solucionó con una revisión de esta gestión y en la versión final han desaparecido los problemas.

– En la introducción del servicio de datos propio de GSL se observó que no proporcionaba todos los puntos que rodeaban al usuario. Esto se debía a que este servició también implementaba el cálculo de visibilidad y orientación de la petición y que para que funcionara correctamente había que desactivar esta funcionalidad en la consulta de los datos y por tanto la petición no se estaba realizando correctamente.

– En el caso de los mapas, se observó que no se soportaba los giros del dispositivo para el ajuste correcto en la pantalla a través del visor utilizado. Esto se ha solucionado borrando el mapa que se está observando y redibujándolo en la nueva posición cuando se produce un giro del dispositivo. Este refrescado de mapas es lo suficientemente rápido como para que el resultado sea aceptable e inapreciable.

Además de estos problemas han surgido algunos otros pero de menor envergadura, los cuales han sido solucionados de manera inmediata. El protocolo de actuación de la realización de estas pruebas ha sido el siguiente: en primer lugar se realizaba la prueba en concreto; si el resultado era positivo, esta se daba por superada; en caso contrario se trataba de solucionar y todas las pruebas volvían a ser realizadas, repitiendo esta iteración hasta que el resultado de la prueba fuera positivo.

Para la realización de estas pruebas se ha utilizado en algunas de ellas el simulador de iOS en XCode, tanto para iPhone como para iPad, ya que de este último no se disponía de dispositivo físico. Sin embargo, la mayoría de ellas hubo que realizarlas directamente en el dispositivo ya que eran necesarios algunos sensores los cuales no son simulables. Así pues, todas ellas han sido realizadas en un dispositivo físico con las siguientes características:

Dispositivo: Smartphone iPhone4 GSM
Sistema Operativo: iOS 6.0.1
Procesador: ARM Cortex A8 a 800 MHz
RAM: 512 MB DRAM
Resolución: 960×640 píxeles
Tamaño de pantalla: 3.5”

Bibliografía, Acrónimos e Índice de figuras

Bibliografía

Objetive- C, iOS SDK

[1] Mark Dalrymplo, Scott Knaster: LearnObjective-C on the Mac, Apress

[2] Dave Mark, Jack Nutting, Jeff LaMarche: Beginning iPhone 4 Development, Exploring the iOS SDK, Apress

[3] Dr. Rory Lewis: iPhone and iPad Apps for Absolute Beginners, Apress

[4] Stephen G. Kochan: Programming in Objective-C (5th Edition), Developer's Library

[5] Bert Altenberg, Alex Clarke and Philippe Mougin: Become an Xcoder, (2008) http://www.cocoalab.com/BecomeAnXcoder.pdf

[6] Ray Wenderlich Blog: http://www.raywenderlich.com

Teoría

[7] T. Vicenty: Direct and inverse solutions of geodesics on the ellipsoid with applications of nested equations. Survey Review April 1975. http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf

[8] National Geophysical Data Center (NGDC): The US/UK World Magnetic Model for 2010- http://www.ngdc.noaa.gov/geomag/WMM/data/WMM2010/WMM2010_Report.pdf

[9] National Geospatial Intelligence Agency (NGA) https://www1.nga.mil/Pages/default.aspx

Análisis estado del arte

[10] Mixare: http://www.mixare.org/

[11] Vuforia: http://www.qualcomm.com/solutions/augmented-reality

[12] Layar: http://www.layar.com/

[13] Wikitude: http://www.wikitude.com/

[14] NyARToolKit: http://nyatla.jp/nyartoolkit/wp/

[15] ARviewer: http://www.libregeosocial.org/node/24

[16] LookAR: http://www.lookar.net/

[17] Nokia City Lens: http://betalabs.nokia.com/trials/nokia-city-lens-for-windows-phone

Solución de problemas, programación, documentación de servicios

[18] iOS Developer Program: https://developer.apple.com/programs/ios/

[19] JSONKit: https://github.com/johnezang/JSONKit

[20] Geonames Documentation: http://www.geonames.org/export/web-services.html

[21] CloudMade Documentation: http://developers.cloudmade.com/projects/show/geocoding-http-api

[22] Open Street Maps: http://wiki.openstreetmap.org/wiki/Main_Page

[23] GitHub. Code Hoting: https://github.com

[24] StackOverFlow: http://stackoverflow.com [Resolución de problemas de programación]

Diseño y desarrollo

[25] Raquel Trillo Lado: Patrones de diseño: http://webdiis.unizar.es/~jmerse/IS-2/TeoriaPatronesV2_2.pdf

[26] Desarrollo iterativo e incremental: http://www.proyectosagiles.org/desarrollo-iterativo-incremental

Nota: Las direcciones de internet que aparecen en esta bibliografía están revisadas y accedidas a fecha 01-09-2013.

Acrónimos

ACID: Atomicity, Consistency, Isolation, and Durability
AJAX: Asynchronous Javascript and XML
API: Application Programming Interface
AR: Augmented Reality
A-GPS: Assisted Global Positioning System
CASE: Computer Aided Software Engineering
GeoJSON: Geographic JavaScript Object Notation
GIS*:* Geographic information system
GPS: Global Positioning System
GSL: GeoSpatiumLab, SL
GUI: Graphics Unit Interface
IAAA: Grupo de Sistemas de Información Avanzados
IDE: Integrated Development Environment
iOS: iPhone Operating System
I3A: Instituto Universitario de Investigación en Ingeniería de Aragón
JPEG: Joint Photographic Experts Group
JSON: JavaScript Object Notation
LRU: Least Recently Used
MVC: Modelo-Vista-Controlador
OGC: Open Geospatial Consortium
OS: Operating System
PDF: Portable Document Format
PFC: Proyecto Final de Carrera
PNG: Portable Network Graphics
POI: Point Of Interest
REST: Representational State Transfer
SDK: Software Development Kit
SQL: Structured Query Language
TIFF: Tagged Image File Format
UML*:* Unified Modeling Language
URL: Uniform Resource Locator
UTM*:* Universal Transverse Mercator
WGS -84: World Geodetic System 1984
WIFI: Wireless Fidelity
WMM: World Magnetic Model
W3C: World Wide Web Consortium
XML: eXtensible Markup Language
3D: Three Dimensional
3G: 3rd Generation

Índice de figuras