ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Grado en Ingeniería del Software

Desarrollo de una aplicación en iOS para la Cámara de Comercio de Málaga

Development of an iOS application for the Malaga Chamber of Commerce

Realizado por
Francisco Andrade García
Tutorizado por
Eduardo Guzmán de los Riscos
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, febrero de 2015

Fecha defensa:
El Secretario del Tribunal

Resumen: Este trabajo consiste en el diseño y desarrollo de una primera versión de una aplicación móvil para dispositivos que ejecuten iOS, para la promoción de los servicios de la Cámara de Comercio de Málaga.
Se presentan las ofertas de los departamentos de Comercio Exterior y Turismo, Formación, ayudas y subvenciones gestionadas por el departamento de Servicios a Pymes y Autónomos, y las Licitaciones y ofertas de contratación.

La aplicación móvil obtiene los datos de las webs de la corporación en un objeto JSON, para lo que se ha desarrollado una pequeña aplicación en php. La aplicación móvil procesa este objeto JSON, y almacena la información en una base de datos SQLite, para presentarlos al usuario.

En la pantalla inicial se ofrece el acceso a las ofertas de servicios, y funcionalidad para contactar por correo electrónico o por teléfono.
La información de los servicios ofertados se ofrece en formato listado-detalle para los servicios, ofreciendo la solicitud de ampliación de información en un email formateado.

Con acceso desde la pantalla inicial, también se ofrecen servicios de geolocalización, mediante un sistema de rutas, en la que el usuario puede ver la localización de la Cámara de Comercio, calcular su ruta para llegar, y puede seguir esta ruta mediante una representación etapa por etapa, de manera gráfica, en el mapa, con indicaciones detalladas de camino, duración, etc.

Palabras claves:JSON, SQLite, iOS, MapKit, TableView, Xcode, Objective-C, Cámara de Comercio de Málaga

Abstract: This work involves the design and development of a first version of an mobile application for devices running iOS, to promote the services of the Malaga Chamber of Commerce. Offers departments of Foreign Trade and Tourism, Training, grants and loans by the Department of Services to SMEs and Self, and Tender Contracts are presented.

The mobile application gets data from the websites of the corporation in a JSON object, for which it has developed a small application in php. The mobile application processes this JSON object, stores the information in a SQLite database and present them to the user.

In the initial screen provides access to service offerings, and functionality to contact by email or by phone. The information offered services offered in-detail list for services format, featuring request for more information in a formatted email.

With access from the home screen, geolocation services are also offered through a routing system in which the user can see the location of the Chamber of Commerce, calculate your route to get there, and you can follow

this route by a representation stage by stage, graphically, on the map, with detailed information on the way, duration, etc.

Keywords: JSON, SQLite, iOS, MapKit, TableView, Xcode, Objective-C, Chamber of Commerce of Malaga

Índice de Contenidos

Índice de Figuras

Capítulo 1: Introducción

1.1 Introducción al TFG

El proyecto tiene como finalidad el estudio de las posibilidades de negocio y promoción que ofrece el intercambio de datos y la sinergia entre sitio Web y aplicación móvil, generando contenido informativo actualizado en dispositivos móviles, y posibilitando de esta manera alcanzar a un mayor número de clientes con la información actualizada en tiempo real.

Se desarrollará una aplicación que permita a los usuarios de dispositivos móviles con sistema iOS, acceder en cualquier momento a la información actualizada de los servicios y ofertas que presta la **Cámara de Comercio de Málaga.**

Paralelamente a este proyecto, la alumna Dª María Victoria Díaz Redondo realizará el desarrollo de esta aplicación móvil para dispositivos con sistemas Android en otro TFG.

El proyecto será desarrollado usando las plataformas de desarrollo de Apple, **Xcode**, **NetBeans**, de Oracle, y el gestor de base de datos **SQLite Database Browser.**

1.2 Objetivos

Con el diseño, implementación e implantación de esta herramienta, se pretende:

Perfeccionar la técnica de **toma de requisitos** de usuario, mediante la identificación y entrevista de los responsables de los servicios de la corporación, con el fin de estimar cuáles son susceptibles de ofrecerse mediante una solución de movilidad.

Una vez realizadas las entrevistas, se procederá a especificar el **ámbito del software** a desarrollar, con las funcionalidades y requerimientos necesarios. De acuerdo con esto se podrá realizar una mejor estimación de las fases temporales e hitos de desarrollo del proyecto.

Mediante el uso de **diagramas de clase** y **casos de uso,** se realizará el diseño del software a desarrollar.

Sin necesidad de especificar los servicios de la Cámara de Comercio que se tratarán en la aplicación, el desarrollo requerirá del diseño de una **funcionalidad web**, que se comunicará al menos con dos servidores de la corporación. Esta parte ha de obtener la información de dos bases de datos, y generar una salida de forma que pueda ser procesada y presentada por la aplicación móvil. Este servicio se desarrollará en PHP, y generará como salida un objeto **JSON** (*JavaScript Object Notation*).

La aplicación móvil procesará esta salida y almacenará los datos en una base de datos **SQLite** en el dispositivo móvil, con el fin de que esté disponible para una consulta sin conectividad (en cuyo caso se indicaría al usuario la fecha de actualización de la información).

A partir de la información almacenada en esta base de datos, la aplicación presentará a los usuarios la información categorizada de manera dinámica, a distintos niveles, según la información provista por el sitio Web.

La aplicación estará desarrollada para adaptarse a los requerimientos de los distintos dispositivos móviles de Apple, en sus distintos tamaños de pantalla y resolución, y las últimas versiones del sistema iOS.

Se utilizarán métodos de desarrollo en **cascada y prototipado**, en la fase de diseño y desarrollo inicial e implantación.

Como etapa final se hará una fase de implantación y pruebas entre los usuarios, en la que se podrá verificar el correcto funcionamiento de la aplicación.

Pedagógicamente, podemos considerar los siguientes objetivos del proyecto:

1) **Estudio de la Plataforma**. Se analizará el IDE Xcode, las librerías y nuevas funcionalidades para la nueva versión del sistema operativo iOS 8.2, y su compatibilidad con versiones anteriores, hasta la versión iOS 6.1

2) Estudio de mecanismos de **intercambio de datos**. Se estudiarán las tecnologías **PHP y JSON** para la actualización de contenidos desde el sitio web, en aplicaciones móviles.

3) Estudio de **Técnicas de Bases de datos**. Se diseñará y creará una base de datos **SQLite,** almacenada en el dispositivo móvil.

4) **Implementación de la aplicación móvil**. Aplicando técnicas y estándares estudiados anteriormente se desarrollará una aplicación móvil para mostrar la información de interés para el usuario en dispositivo móvil con sistema **iOS.**

1.3 Etapas de desarrollo

El proyecto se realizará en 4 etapas:

1) Toma de requisitos: 10 horas

2) Análisis, diseño, e implementación de la aplicación web: 24 horas

3) Análisis, diseño, e implementación de las aplicaciones móviles para los dispositivos: 223 horas.

4) Implantación en los dispositivos de la empresa, y pruebas: 40 horas

5) Subida a App Store: 3 horas.

1.4 Descripción de la solución propuesta

Para el cumplimiento de los objetivos del Trabajo Fin de Grado, se presenta una aplicación móvil que presenta información relevante desde los sitios web camaramalaga.com y formacioncamara.com, mostrándola de forma adecuada al dispositivo, y de manera funcional, a usuarios de dispositivos con sistemas Android.

La aplicación constará de dos partes:

- La primera parte consiste en una aplicación web que permite la extracción de datos de los distintos portales de la corporación donde se aloja. Esta parte ha de obtener la información de dos bases de datos, y codificar los resultados obtenidos en las consultas en un objeto JSON cuya estructura se definirá en la fase de estudio del proyecto. De este objeto JSON obtendrá la aplicación móvil los datos necesarios para presentar la información a los usuarios

- La segunda parte del proyecto es el diseño y desarrollo de la aplicación móvil. Esta aplicación almacenará los datos en una base de datos en el dispositivo, y la presentará al usuario, en una estructura que se definirá en el propio proyecto. En el caso de que el dispositivo se encuentre sin conexión, se mostrará la información de la base de datos descargada en la última visita con conexión.

Capítulo 2: Estado del arte

2.1 Evolución de las aplicaciones móviles.

Para poder situarnos históricamente en este contexto, recordaremos que el mercado español el boom de la telefonía móvil comenzó en los años 90 [1]:

- Inicialmente, en España, Telefónica sacó su marca Teleline. Fue la única del mercado hasta 1994, cuando se liberó el monopolio de Telefónica en el mercado.
- En 1995 nacen Movistar y Airtel.
- En 1996 las compañías regalaban los móviles, intentando de esta manera conseguir rápidamente una cuota mayoritaria en un mercado al que vaticinaban un crecimiento exponencial:

Figura 1- Regalos móviles

- En 1999 nace Amena, popularizando los precios (20 Ptas/min), y dirigida a un público más joven.

Las aplicaciones móviles tuvieron un crecimiento rápido cuando la tecnología WAP (Wireless Application Protocol) y la transmisión de datos empezaron a popularizarse.

Definido en 1998, WAP es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, que consiste en la especificación de un entorno de aplicación y de un conjunto de protocolos de comunicaciones para normalizar el modo en que los dispositivos inalámbricos, se pueden utilizar para acceder a correo electrónico, grupo de noticias y otros. [2]

La evolución de la capacidad de las comunicaciones ha marcado el ritmo de crecimiento de las aplicaciones móviles: al ser más rápidas, permiten aplicaciones más complejas, con más utilidad:

Año	Gen	Tecnología	Velocidad
1985	1	TACS (Total Access Communications System)	8kbps
1992	2	GSM (Global System for Mobile communications)	14 kbps
2000	2.5	GPRS (General Packet Radio Service)	56-115 kbps
2001	2.5	EDGE (Enhanced Data rates for GSM Evolution) o EGPRS (Enhanced GPRS)	384kbps
2001	3	UMTS (Universal Mobile Telecommunications System o UMTS)	2 Mbps
2008	3	HSPDA (High Speed Downlink Packet Access)	14 Mbps
2008	3	HSPDA+	84 Mbps
2010	4	LTE (Long Term Evolution)	100 Mbps

Figura 2- Evolución de las redes

A finales de los noventa, los terminales móviles eran ya de una segunda generación, y empezaban a ser considerados inteligentes (smartphones). Tenían incorporadas las primeras aplicaciones móviles, proporcionadas por los fabricantes, y ocasionalmente por los operadores, como eran el gestor de contactos (agenda).

Pero fue el nacimiento de las tiendas de aplicaciones lo que marcó el mercado de las aplicaciones móviles. Apple publicó su App Store año 2008 con apenas 500 aplicaciones. Android Market entró al negocio a los pocos meses, con un repositorio de 50 apps. La tercera fue BlackBerry App World y Ovi Store de Nokia en el 2009. Microsoft llegó mucho más tarde abriendo en el 2010 con Windows Phone Marketplace.

ShoutEm ha publicado una infografía que muestra la evolución de las tiendas de aplicaciones [3]:

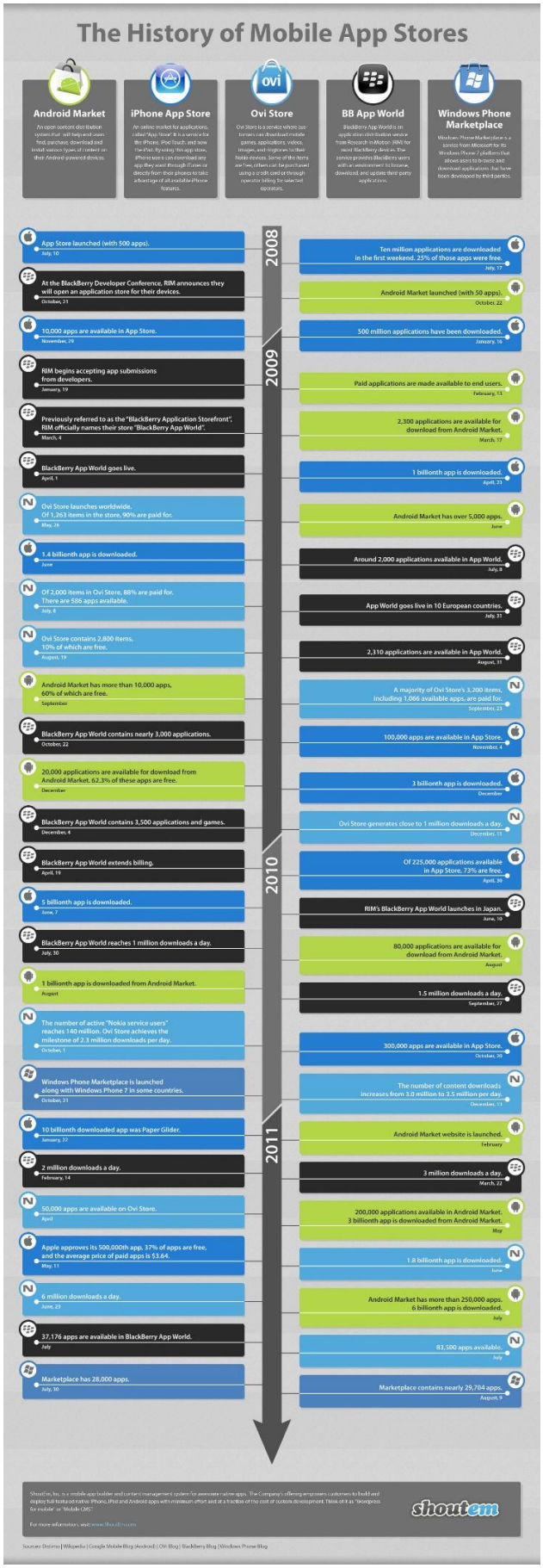


Figura 3- Evolución de las tiendas de aplicaciones

2.2 El mercado de aplicaciones de Apple: App Store

El *Software Development Kit* para iOS se anunció en el evento *iPhone Software Roadmap* del 6 de marzo de 2008. Este SDK permitía a los desarrolladores crear aplicaciones nativas para iPhone, iPod Touch, y iPad, usando el IDE Xcode.

Cuatro meses después, el 10 de julio del mismo año, mediante una actualización de iTunes, se abrió el App Store. Al día siguiente se lanzó al mercado el iPhone 3G, que venía de serie con iOS 2.01, que ya permitía el uso del App Store.

Las cifras de crecimiento del número de aplicaciones y descargas desde entonces han sido enormes:

Fecha	Apps Disponibles	Descargas hasta la fecha
11/07/2008	500	0
14/07/2008	800	10.000.000
09/09/2008	3.000	100.000.000
09/09/2014	1.300.000	75.000.000.000

Figura 4- Evolución del App Store

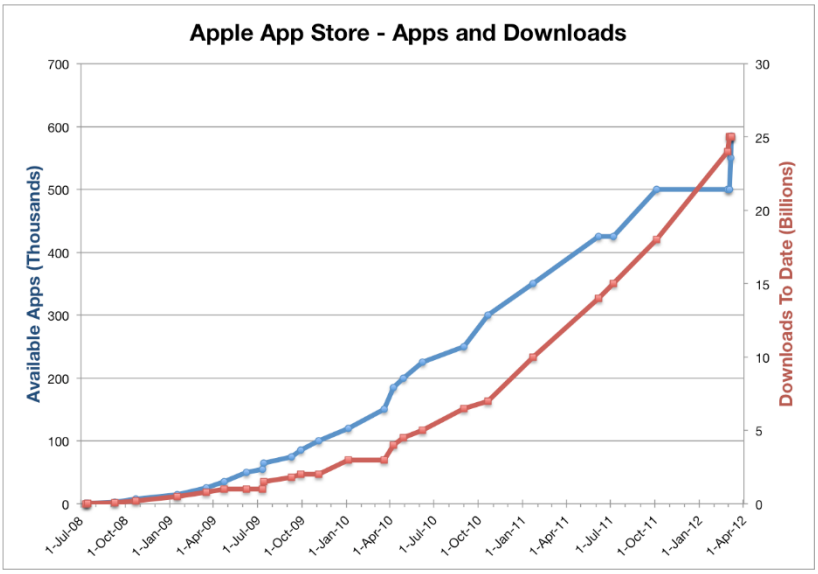


Figura 5- Estadísticas del App Store

El hecho de que las aplicaciones para iOS solo puedan distribuirse desde el App Store, permite que Apple controle la calidad de las aplicaciones, la conveniencia del contenido, e introducir una tasa, normalmente del 25%, del costo de la aplicación de forma directa.

2.3 Sistemas Operativos móviles

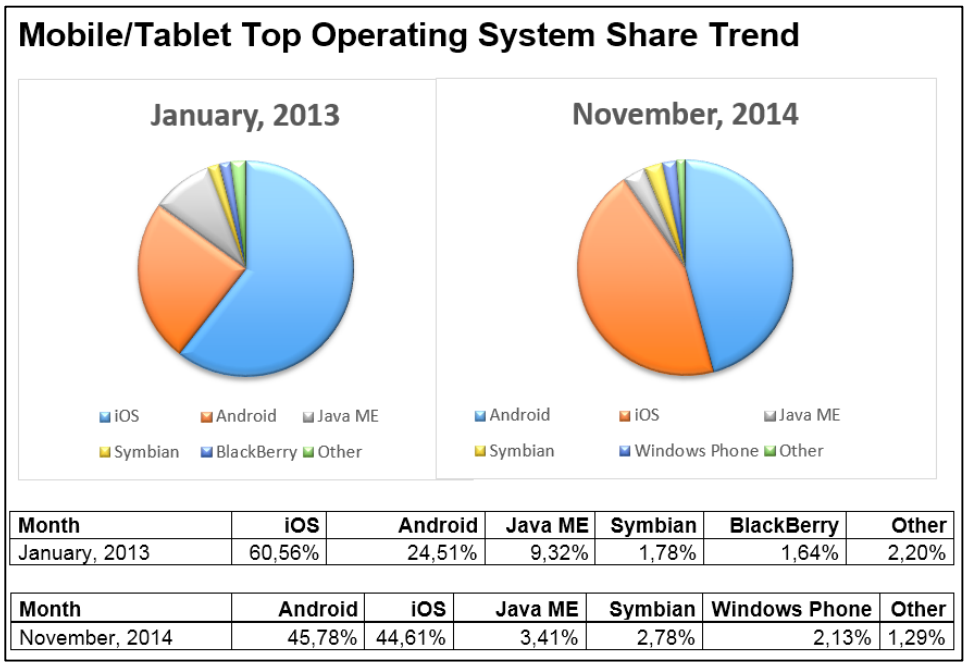
En los últimos años el crecimiento de las ventas de teléfonos móviles inteligentes o *smartphones* ha sido exponencial. Ya en 2011 logró superar las ventas de portátiles, PC's, tablets y netbooks llegando a los **500 millones de terminales**. [7]

El uso de teléfonos inteligentes alcanza ya **el 53,7% de la población española**, según el informe de la ONTSI "La Sociedad en Red". [8]

Por ello, las aplicaciones para dispositivos móviles se han posicionado como un mercado emergente con grandes posibilidades.

El mercado actual de las aplicaciones móviles está liderado por Apple y Google, con sus sistemas iOS y Android y con App Store y Play Store como plataformas de venta, respectivamente, según el número de aplicaciones y negocio generado.

La distribución en el mercado de dispositivos por SO móviles, desde enero de 2013, ha evolucionado de la siguiente forma: (http://www.netmarketshare.com/)



Month	iOS	Android	Java ME	Symbian	BlackBerry	Other
January, 2013	60,56%	24,51%	9,32%	1,78%	1,64%	2,20%

Month	Android	iOS	Java ME	Symbian	Windows Phone	Other
November, 2014	45,78%	44,61%	3,41%	2,78%	2,13%	1,29%

Figura 6- Evolución de los SO Móviles

2.3.1 Android

Android es el S.O. desarrollado por **Open Handset Alliance**, organización liderada por Google. La principal característica de Android es que **se desarrolla de forma abierta**, lo que permite a los desarrolladores crear librerías y programas generados en otros lenguajes y compilarlos en la arquitectura ARM de los terminales Android como si se tratasen de librerías nativas del sistema, permitiendo mejorar el sistema constantemente.

La oferta de teléfonos con Android es amplia y variada tanto en marcas como en precios. Se estima que hay unos **12.000 tipos de dispositivos distintos** ejecutando este SO, frente a los 4.000 que había en 2012:

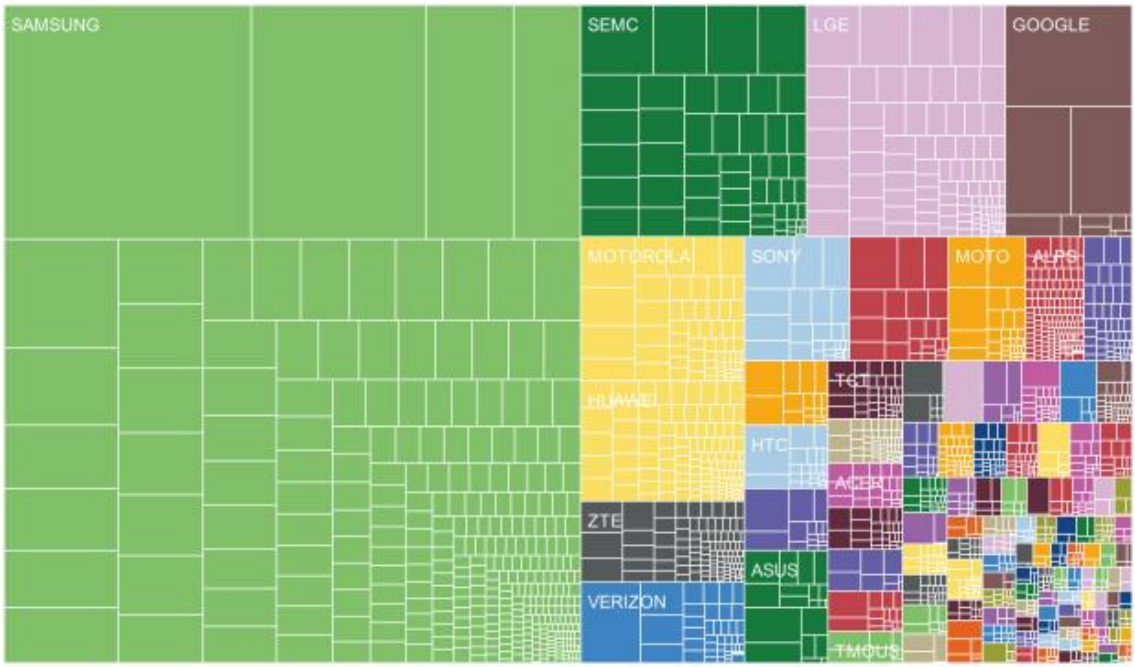


Figura 7- Dispositivos ejecutando Android

Además, esta gran diversidad a nivel hardware está generando también una gran fragmentación de versiones de Android, debido a la necesidad de hacer adaptaciones a cada dispositivo concreto. Esta situación se traduce en un incremento en la complejidad del desarrollo de aplicaciones, sobre todo a la hora de garantizar fiabilidad, soporte y experiencia de uso adecuada, características que son responsabilidad del desarrollador, debido a la inexistencia de revisiones y controles a la hora de publicar aplicaciones en la Play Store.

2.3.2 iOS

iOS (iPhone OS), el sistema operativo desarrollado por Apple Inc. para iPhone, iPod Touch, iPad y Apple TV.

Es una evolución de Darwin BSD y por lo tanto un sistema operativo Unix. A diferencia de Android, iOS está desarrollado únicamente para unos pocos dispositivos diseñados por la propia Apple, constituyendo así un "ecosistema cerrado". [9]

Figura 8- iPhone 4-5-6-6+

De cara a los desarrolladores esta característica puede parecer una ventaja, ya que las herramientas de diseño de aplicaciones y frameworks proporcionadas por Apple están optimizadas para dicho hardware, simplificando así ciertas tareas de desarrollo, pero, por otro lado, dificulta, y en muchos casos impide, la modificación y sobrecarga de librerías y funciones, y el acceso a la gestión de ciertos recursos de los propios dispositivos. Además, las aplicaciones para iOS se distribuyen por medio de la App Store, pasando estrictos controles de calidad antes de ser publicadas, que incluyen la comprobación y/o veto de ciertas funcionalidades admitidas por otros publicadores, como Google.

2.3.3 Desarrollar para ambos lenguajes

Desarrollar una aplicación en Android e iOS, si se quieren realizar aplicaciones nativas, no usando plataformas como Titanium, etc., presenta grandes diferencias entre ambas.

Para empezar, aunque los lenguajes son orientados a objetos, las aplicaciones nativas en Android se desarrollan en **Java**, mientras que en iOS se hace en **Objective-C**, y/o (recientemente) **Swift.**

2.3.3.1 El proyecto.

En **Xcode** (IDE de Apple para iOS) es el desarrollador el que decide la estructura lógica del proyecto. Se pueden crear carpetas, y estructurarlo como uno quiera para que la organización sea la deseada. Esto podría ser un problema cuando se trabaja en equipo.

En Android, la estructura y la nomenclatura son más rígidas, lo que, si bien resta cierta flexibilidad, hace que la estructura sea más estándar.

2.3.3.2 Gestión de memoria

En ambas plataformas se realiza, en gran medida, de forma automática. En iOS 5 se introdujo el **ARC** (*Automatic Reference Counting*) que realiza todas las operativas de *RETAIN* y *RELEASE* sin que tengamos que preocuparnos por ello. Esto facilita la tarea al desarrollador, aunque no deja de evita que esta gestión siga siendo crítica. De hecho, en todas las clases creadas con XCode aparece por defecto el método de la Figura 9, donde el desarrollador puede

```
- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

Figura 9- Gestión de memoria

particularizar la gestión en caso de que los recursos disponibles en el dispositivo se encuentren en una situación crítica.

Por defecto, Android utiliza el **Garbage Collector** de Java para liberar la memoria de los objetos sin referencias, que no vayan a usarse.

2.3.3.3 Interfaz gráfica

Aquí es donde viene una de las grandes diferencias, y es que las soluciones propuestas para la creación de una interfaz gráfica son muy distintas.

En Android las interfaces se construyen mediante ficheros XML legibles, mientras que Xcode facilita, con un sistema gráfico de creación de interfaces, el diseño de pantallas.

2.3.3.4 Activity y ViewController

En el caso de iOS, el controlador **UIViewController** es un componente del núcleo de la aplicación, con el cual podemos gestionar los ciclos de vida de los eventos, subvistas... Es el controlador dentro del modelo vista-controlador.

En Android, el equivalente es la clase **Activity**, que representa una pantalla en un dispositivo, aunque no son exactamente iguales. De hecho, las principales diferencias residen en la forma de gestionar los ciclos de vida de los eventos.

2.3.3.5 Delegate y Adapter

El **patrón de delegación**, del que se habla en el punto *4.4.1.5 Delegates y AppDelegate,* es muy utilizado en iOS mediante el uso de protocolos delegados. Sin embargo, en Android este patrón es representado con un **adaptador**.

2.3.3.6 El botón "Volver" de Android

Este "pequeño" detalle condiciona la funcionalidad y el diseño de las pantallas en iOS, ya que, al carecer de este botón del SO, los programadores han de incluirlo en el diseño de las pantallas, con la funcionalidad deseada.

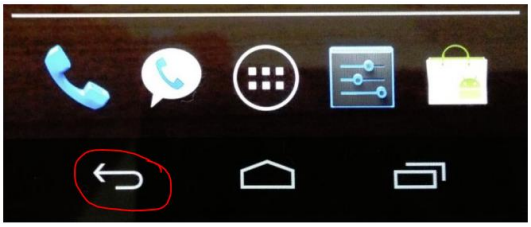


Figura 10- Volver

Para hacer esto, iOS proporciona ayudas mediante los *Navigation Controller*, que sirve para controlar la navegación por las vistas o escenas.

2.3.3.7 Simulador o emulador

En este sentido también encontramos diferencias bastante grandes. iOS utiliza un **simulador** que es bastante más rápido que el **emulador** de Android. Sin embargo, este último es más fiel a la situación que encontraremos en un dispositivo, ya que es una máquina virtual con una CPU virtualizada, y la cantidad de recursos de que dispone es limitada y más fiel.

2.4 Cámara de Comercio de Málaga

La Cámara es una Corporación de Derecho Público que se configura legalmente como órgano consultivo y de colaboración con las Administraciones Públicas y que tiene como finalidad la representación, promoción y defensa de los intereses generales del comercio, la industria y la navegación, así como la prestación de servicios a empresas que ejerzan las mencionadas actividades.

En España hay 88 cámaras de comercio, ya que en algunas provincias hay más de una (en Cádiz hay tres: la de Cádiz, la de Jerez, y la de Campo de Gibraltar), pero cada una de ella funciona de manera independiente en una ámbito territorial, donde las demás cámaras no tienen competencias.

En provincia de Málaga solo existe una cámara, la Cámara Oficial de Comercio, Industria, y Navegación de Málaga:

Existe, en la mayoría de las comunidades autónomas, un Consejo Regional de Cámaras. En Sevilla está el Consejo Andaluz de Cámaras.

A modo de coordinadora entre todas ellas, está la Cámara de España, que actúa como intermediaria entre las Cámaras de Comercio y el resto de administraciones nacionales e internacionales, sin perjuicio de la autonomía funcional de cada cámara, y delas relaciones entre estas y las administraciones locales.

En la actualidad mi puesto en la Cámara de Comercio es el de Responsable del Área de Nuevas Tecnologías, que incluye la coordinación tanto de la parte de administración de sistemas, desarrollo propio de software (que incluye el desarrollo web) y el asesoramiento técnico y participación en el diseño y desarrollo de los programas de Innovación.

Desde esta área planteamos la necesidad de ofrecer nuestro servicio e información a través de una aplicación móvil, cuya primera versión es la presentada en este trabajo.

En versiones posteriores se dotará de más funcionalidades, no solo informativa, de manera que permita a los usuarios, donde sea posible, realizar trámites completos desde ella.

Capítulo 3: Sistema operativo móvil: iOS

iOS (por sus siglas en inglés **iPhone/iPod/iPad Operating System**) es un sistema operativo móvil de la empresa Apple. Se desarrolló originalmente para el iPhone, y después Apple lo instaló en otros dispositivos como el iPod Touch, iPad y el Apple TV. Apple no permite la instalación de iOS en hardware de terceros.

En octubre-noviembre de 2014, más del 52% de los dispositivos iOS (iPad, iPod y iPhone) poseen iOS 7.3 o posteriores versiones, hasta la actual 8.2.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten en deslizadores, interruptores y botones. Los gestos reconocidos son, por ejemplo, un toque o multitoque, deslizar uno o varios dedos simultáneamente, pellizcar la pantalla, etc.

3.1 Evolución del iOS

La primera versión de iOS se lanzó al mercado el **29 de junio de 2007**, y durante su existencia se introdujeron muchos de los cambios y funcionalidades de gestos que siguen en activo hoy día, como las características multitouch.

La segunda versión salió al mercado con los teléfonos 3G, y con ella nació la App Store, donde se dio la posibilidad de descargar aplicaciones de terceros. Se lanzó en julio de 2008.

iOS3 e iOS4, con lanzamientos respectivos en junio de 2009 y junio de 2010, no supusieron ningún avance técnico diferencial.

iOS5, en junio de 2011, fue un cambio muy importante que introdujo el concepto actual del sistema, incluyendo los servicios en la nube.

iOS6, lanzado en junio de 2012, incluía una nueva aplicación de mapas, mejoras de integración con las redes sociales, detección de voz

iOS7, de septiembre de 2013, y la versión con la que comenzó este proyecto de fin de grado, supuso un cambio radical en el diseño y con funcionalidades como Airdrop (para conectividad inalámbrica).

iOS8, la versión actual, en la 8.2, lanzada en junio de este año, ha supuesto otro cambio radical de imagen, aunque su mayor cambio se ha llevado a cabo en pequeñas mejoras y funcionalidades, como en la gestión de mapas, la introducción del lector de huellas, etc.

3.2 La plataforma de desarrollo Xcode

Xcode es el IDE creado por Apple. Es gratuito para usuarios de Mac Os X y no tiene versiones para ningún otro sistema operativo.

Pueden realizarse programas en C, C++, Objective-C, Objective-C++, Java, y, en su última versión, se ha incluido el compilador para un lenguaje específicamente desarrollado para iOS y OS X, llamado Swift.

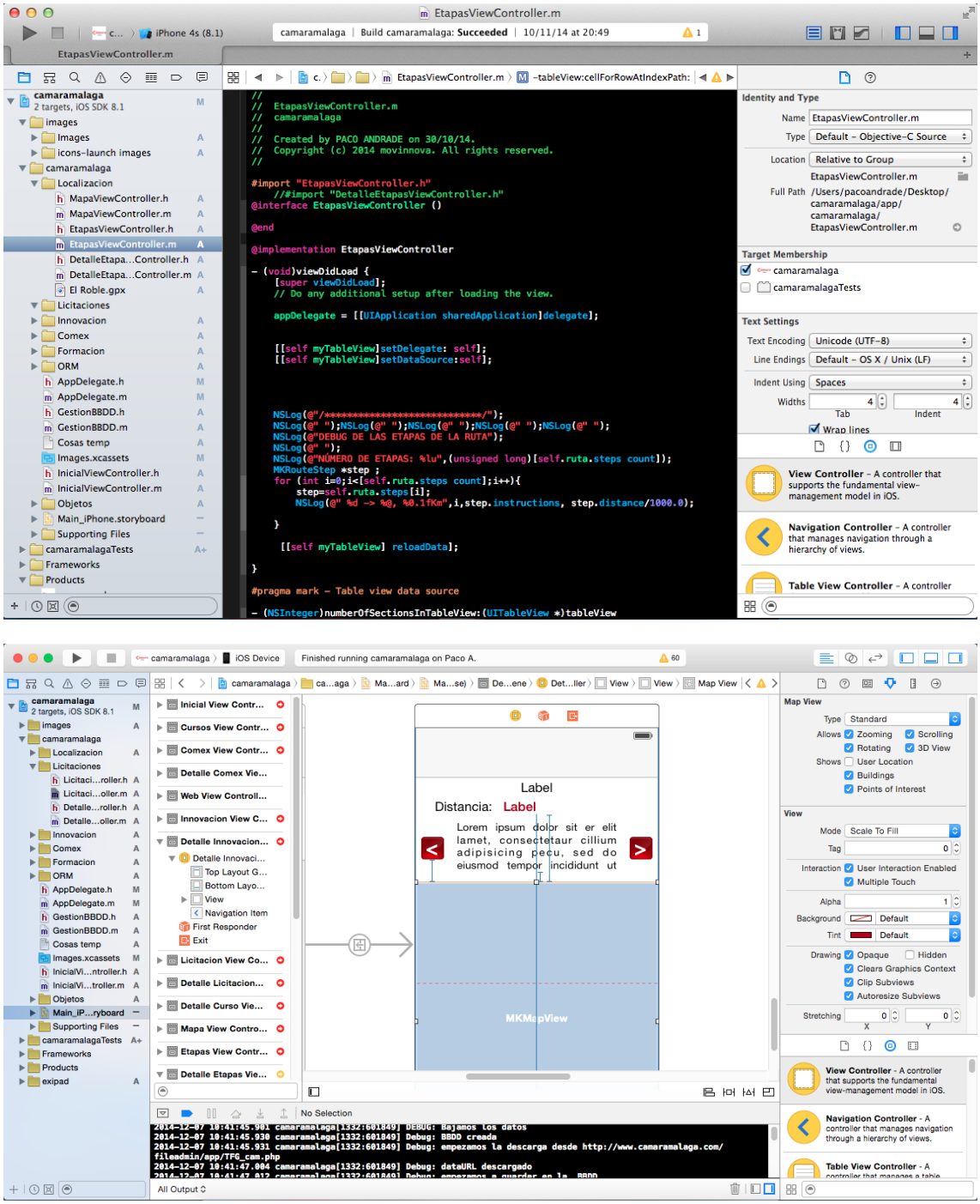


Figura 11- Xcode

Capítulo 4: Desarrollo de la aplicación

4.1 Análisis de la aplicación

Durante el proceso de diseño de la aplicación, nos hemos reunido con los jefes de los departamentos de la Cámara para obtener así los requisitos funcionales.

Basados en la información que éstos consideran más relevante, y en el análisis de las estadísticas de acceso de las webs cuyos datos se presentan en la aplicación, se establecieron estos requisitos funcionales.

Posteriormente, estudiamos y decidimos las modificaciones a realizar en el sistema actual, tanto para proporcionar a la aplicación móvil la información deseada, como para facilitar la gestión de los usuarios finales de la Cámara.

4.1.1 Funcionalidades

Para esta primera versión de la aplicación móvil de la Cámara de Comercio de Málaga, se ha decidido mostrar la información de los principales servicios que presta:

- Departamento de Comercio Exterior y Turismo. Se muestran las misiones comerciales, ferias internacionales, y eventos comerciales para externalizar los negocios.

- Departamento de formación. Se muestra el listado de cursos actuales y de próxima celebración.

- Departamento de Servicio a PYMES y autónomos. Se muestran los programas de ayuda y subvenciones activos.

- Se muestran las licitaciones y concursos públicos del año en curso.

De toda esta información, se muestran los listados separados por secciones, y accediendo a cada una de ella se puede ver la información detallada de cada actividad, curso, concurso…, solicitar información al respecto, y descargarse la a documentación donde sea posible.

Desde la escena inicial de la aplicación, se muestra, además del acceso a estos listados, tres botones que permiten llamar a la Cámara por teléfono, enviar un correo electrónico, o consultar la localización de la misma.

Desde la opción de localización, un usuario puede solicitar a la aplicación que le calcule la ruta para llegar a ella, y ver esta ruta segmentada, pudiendo consultar información detallada de cada etapa.

4.1.2 Casos de uso

Tras el análisis de las funcionalidades, se ha hecho un diagrama de casos de uso que indica la funcionalidad de la aplicación:

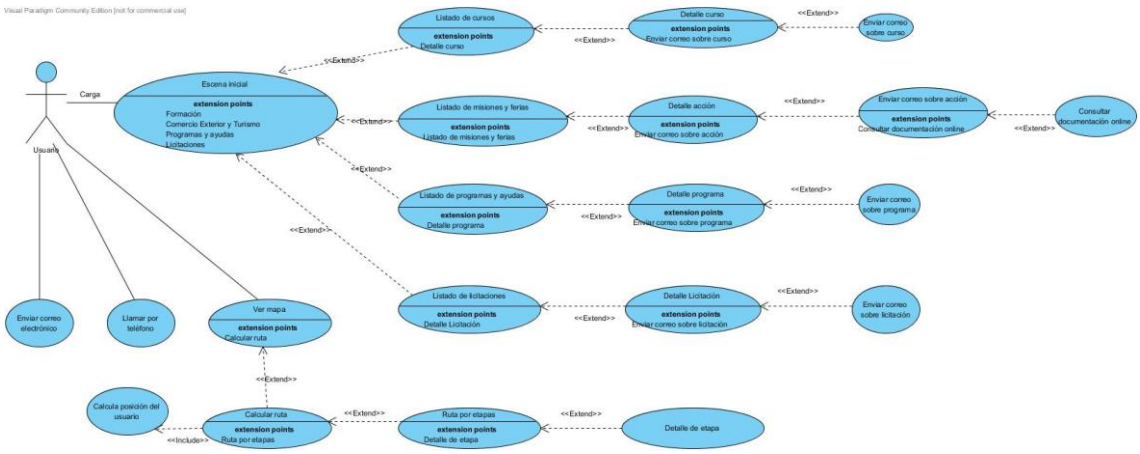


Figura 12- Casos de uso

4.1.3 Diagramas de secuencia.

Para las acciones más complejas, como las interacciones del usuario con el sistema operativo y el mapkit, se ha realizado el diagrama de secuencia:

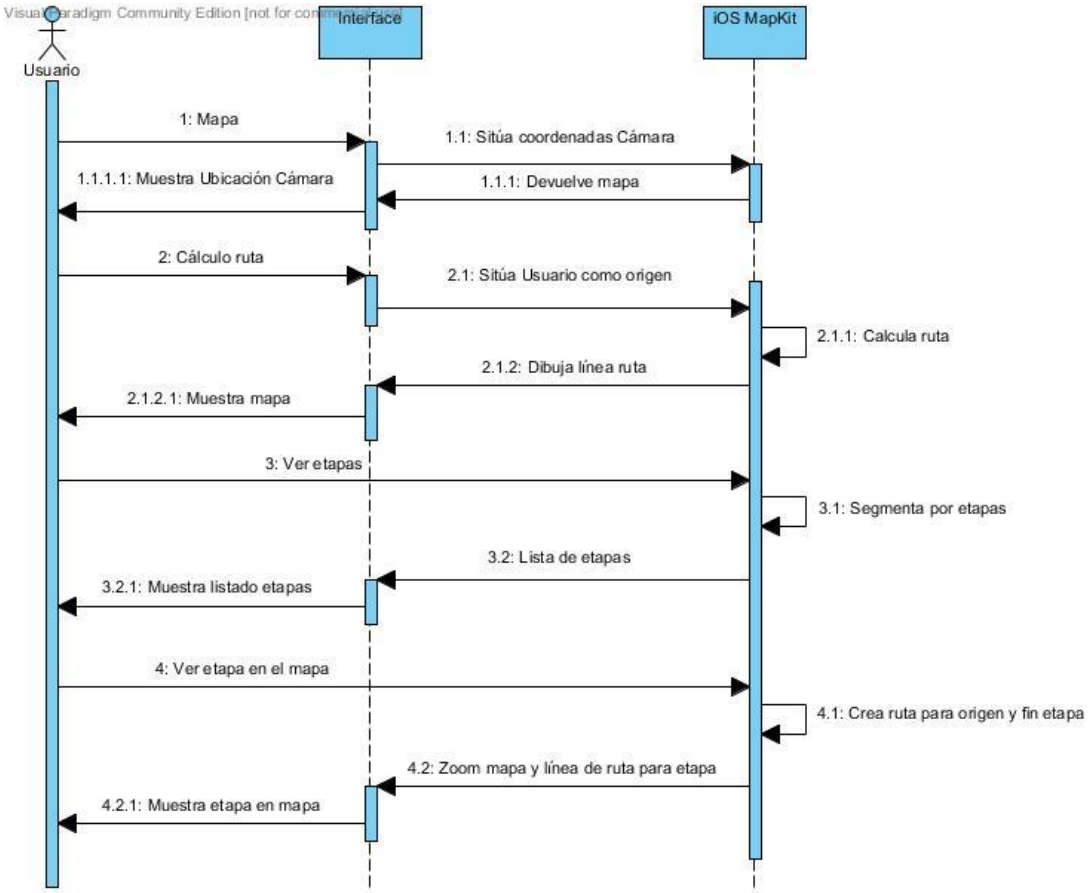


Figura 13- Diagrama de secuencia

4.1.4 Metodología de diseño (Espiral, cascada y prototipado)

Se ha optado por una metodología en espiral, y basada en prototipos, para que la aplicación final vaya evolucionando a lo largo del tiempo.

Se hará utilizando la base y siguiendo las directrices de este primer modelo, desarrollado con un modelo en cascada.

El prototipo resultante de este trabajo, que será publicado en el App Store, servirá de representación y permitirá a la corporación a supervisar y guiar el diseño y funcionalidades principales, para el próximo ciclo, además de permitir hacerse una idea concreta de cómo va a resultar la aplicación final.

4.2 Desarrollo web.

Los datos requeridos en la aplicación se encuentran en dos servidores, bajo los dominios www.camaramalaga.com y www.formacioncamara.com, diseñados con los gestores de contenido Typo3 y Joomla, respectivamente.

Parte de la información estaba gestionada como contenido estático en HTML. Para poder acceder a él, se han desarrollado aplicaciones integradas en ellos que incluían la creación de las tablas de la base de datos, de las páginas de acceso a datos, y en algunos casos, las de gestión de este contenido. Estas aplicaciones se han desarrollado en php.

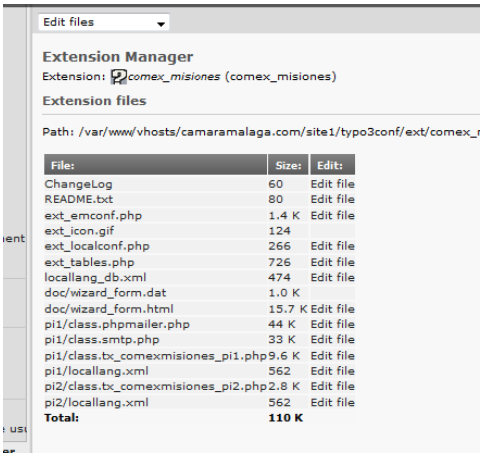


Figura 14- Typo3 Extension

Una vez realizada esta parte, se ha creado una pequeña aplicación online que obtiene los datos de las dos plataformas, y los transforma en un único objeto JSON, que será el accedido por el dispositivo móvil.

Figura 15- Cadena JSON

4.3 Desarrollo de herramientas locales

No siendo estrictamente necesario para la aplicación móvil, se han desarrollado herramientas locales para la gestión del nuevo contenido web dinámico.

Por requisitos de organización, como motivos de licencias, formación de los usuarios, y funcionalidades, tuvimos que utilizar el gestor de bases de datos Microsoft Access para desarrollar las interfaces de los usuarios administrativos locales que se encargan de gestionar y actualizar este contenido.

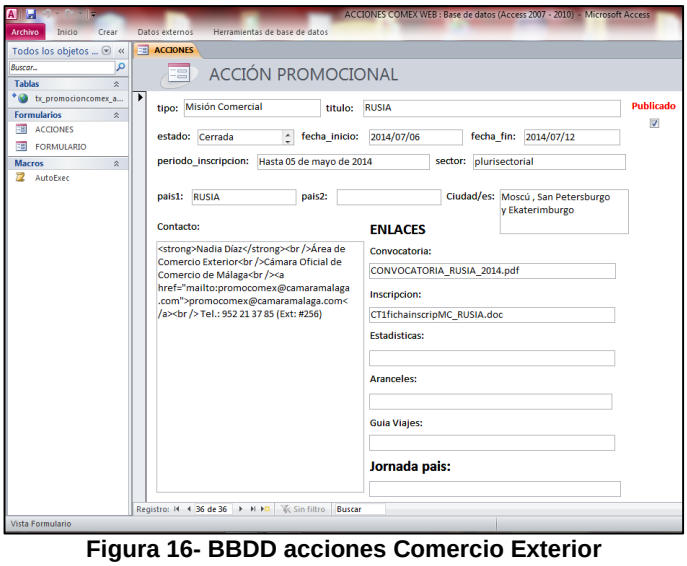


Figura 16- BBDD acciones Comercio Exterior

La conexión de estos interfaces locales con la web se realizó usando el conector ODBC de Sun para MySQL, que es un driver estándar para conectar con estas bases de datos desde plataformas Windows, Linux, Mac OS X, y Unix.

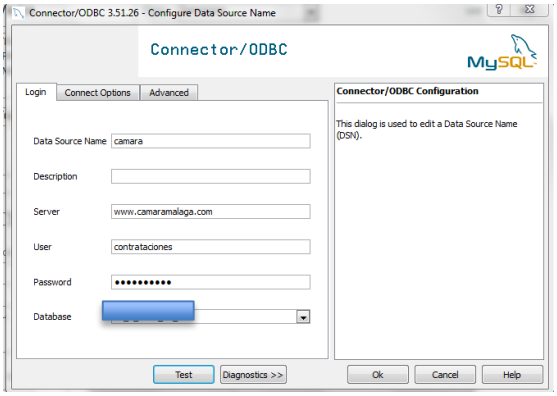
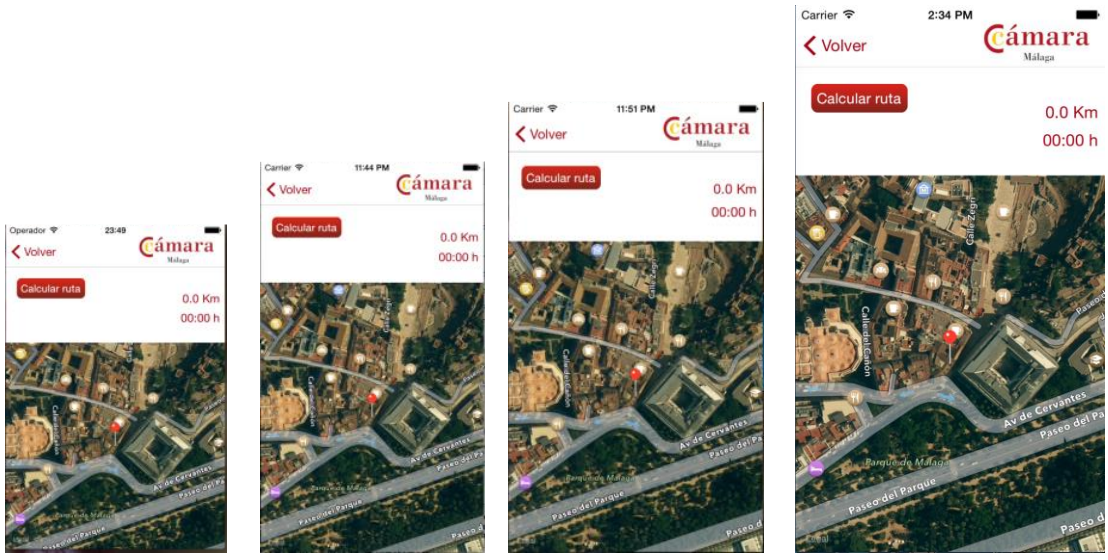


Figura 17- MySQL ODBC Connector

Para poder realizar esto, tuvimos que abrir puertos en el firewall de la sede central de la Cámara y en el firewall del servidor web, alojado en un ISP. Creamos una red privada virtual entre los dos firewall CISCO para mantener los requisitos de seguridad.

4.4 Desarrollo de la aplicación móvil

Como se ha mencionado antes, se ha utilizado el IDE para aplicaciones nativas de Apple, XCode. Esto presenta la ventaja de poder probar la aplicación en un simulador para los distintos dispositivos sin necesidad de tener uno físicamente:



iPhone 4-4s iPhone 5-5c-5s iPhone 6 iPhone 6+

Figura 18- Simulación en distintos iPhone

En este cuadro podemos apreciar las diferencias entre las pantallas de los modelos en los que puede ejecutarse la aplicación:

Modelo	Ancho	Alto	Diagonal	Resolución	PPP
4 - 4S	51.60mm	76.65mm	89mm (3.5”)	640x960	326
5 - 5C - 5S	51.60mm	90.25mm	102mm (4.0”)	640x1136	326
6	58.00mm	104.00mm	119mm (4.7")	750x1334	326
6+	68.00mm	122.00mm	140mm (5.5")	1080x1920	401

Figura 19- Tamaños de pantalla iPhone

El hecho de poder testar la aplicación completa en el simulador para los distintos tamaños de pantallas, es una ventaja muy grande que nos garantiza que el diseño de las interfaces sea el correcto para visionarse en ellas.

En realidad, el simulador lo que hace es escalar la pantalla del dispositivo a la resolución de la física usada:

Modelo	Puntos	Pixels	Escala	Pixels físicos	PPI físicos
3GS	480x320	480x320	1x	480x320	163
4 - 4S	480x320	960x640	2x	960x640	326
5 - 5C - 5S	568x320	1136x640	2x	1136x640	326
6	667x375	1334x750	2x	1334x750	326
6+	736x414	2280x1242	3x	1920x1080	401

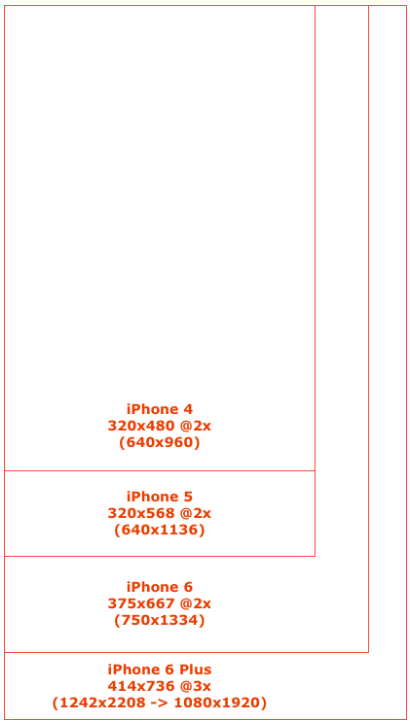


Figura 20- Escalado de pantallas

Por otro lado, como los recursos de memoria y CPU no son fieles a los disponibles en los dispositivos, en aplicaciones con un consumo crítico de recursos, la simulación, en cuanto a rendimiento, deja que desear.

La aplicación se ha desarrollado en Objective-c, usando una base de datos SQLite.

4.4.1 Estructura y componentes de la aplicación móvil

En una aplicación para dispositivos móviles iOS, desarrollada con el IDE nativo Xcode, son varias las partes que requieren de un tratamiento distinto e integrarse para el correcto funcionamiento de la aplicación.

4.4.1.1 Custom Managed Object Class: Datos de la aplicación.

Todo el funcionamiento de la aplicación iOS se basa en el manejo de clases. Las pantallas, los objetos en ellas (imágenes, botones, campos de texto), los componentes, etc., se controlan mediante clases de Objective-C, en este caso.

Para facilitar la implementación y el código de la aplicación, he creado entidades denominadas *Custom Managed Object Class*, que son clases sin métodos, propias de iOS, solo con las propiedades de los objetos de cada clase, correspondientes estas con las claves obtenidas al parsear el objeto JSON.

Las entidades de este tipo creadas para este proyecto son:

- Curso
- Mision
- Licitacion
- Programa

```
//
//  Curso.h
//  camaramalaga
//
//  Created by PACO ANDRADE on 16/05/14.
//  Copyright (c) 2014 movinnova. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface Curso : NSObject



@property int uid;
@property (nonatomic, copy) NSString * nombre;
@property (nonatomic, copy) NSString * tipo;
@property (nonatomic, copy) NSString * modalidad ;
@property (nonatomic, copy) NSString * fecha_inicio ;
@property (nonatomic, copy) NSString * fecha_fin ;
@property  int  num_horas;
@property  int  num_plazas ;
@property (nonatomic, copy) NSString * importe;
@property (nonatomic, copy) NSString * estado ;
@property (nonatomic, copy) NSString * referencia ;

@end
```

```
//
//  Mision.h
//  camaramalaga
//
//  Created by PACO ANDRADE on 16/05/14.
//  Copyright (c) 2014 movinnova. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface Mision : NSObject

@property int uid;
@property (nonatomic, retain) NSString * tipo ;
@property (nonatomic, retain) NSString * titulo ;
@property (nonatomic, retain) NSString * estado ;
@property (nonatomic, retain) NSString * fecha_inicio;
@property (nonatomic, retain) NSString * fecha_fin ;
@property (nonatomic, retain) NSString * pais1 ;
@property (nonatomic, retain) NSString * ciudad ;
@property (nonatomic, retain) NSString * sector ;
@property (nonatomic, retain) NSString * enlace_inscripcion;
@property (nonatomic, retain) NSString * enlace_convocatoria ;
@property (nonatomic, retain) NSString * enlace_imagen ;
@property (nonatomic, retain) NSString * enlace_ficha_pais1 ;
@property (nonatomic, retain) NSString * pais2 ;
@property (nonatomic, retain) NSString * enlace_ficha_pais2 ;
@property (nonatomic, retain) NSString * periodo_inscripcion ;
@property (nonatomic, retain) NSString * contacto ;
@property (nonatomic, retain) NSString * guia_viajes ;
@property (nonatomic, retain) NSString * jornada_pais ;
@property (nonatomic, retain) NSString * aranceles ;

@end
```

Figura 21- Custom Managed Object Class

Estas mismas clases son las que se han creado como tablas en la base de datos de SQLite, gestionándose cada registro como un objeto:

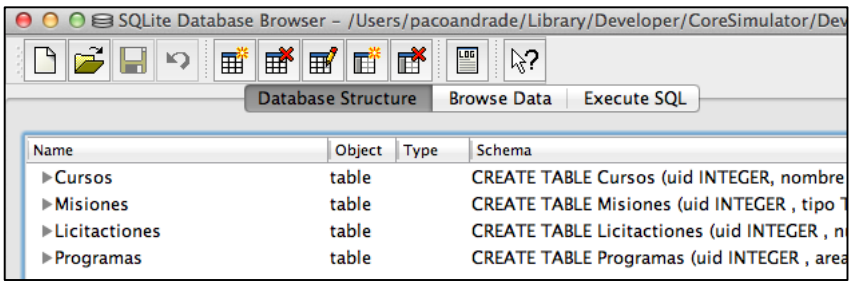


Figura 22- SQLite Database Browser

4.4.1.2 Frameworks

Los frameworks o librerías usados en este proyecto son todos estándares de iOS, y aquellos que se muestran en la Figura 23. Entre estos están las librerías para el acceso al teléfono, manejo de mapas y datos SQLite, gestión gráfica…

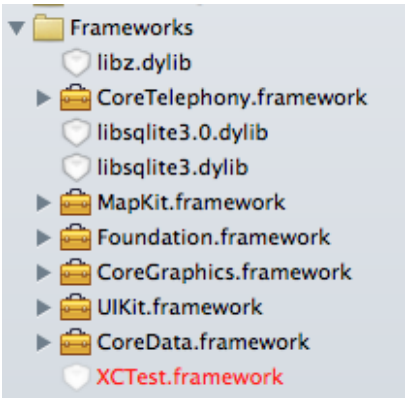


Figura 23- Frameworks

4.4.1.3 Storyboard

Un fichero *storyboard* va a contener un conjunto de escenas y de transiciones entre cada dos. Son ficheros del ***Interface Builder*** en los que pueden construirse visualmente la base de la interface de usuario. También en ellos se indican las transiciones entre los distintos elementos o vistas de la aplicación.

Se construyen en el ***lienzo o canvas del editor***, y son muy útiles puesto que ofrecen una idea bastante aproximada del funcionamiento de la

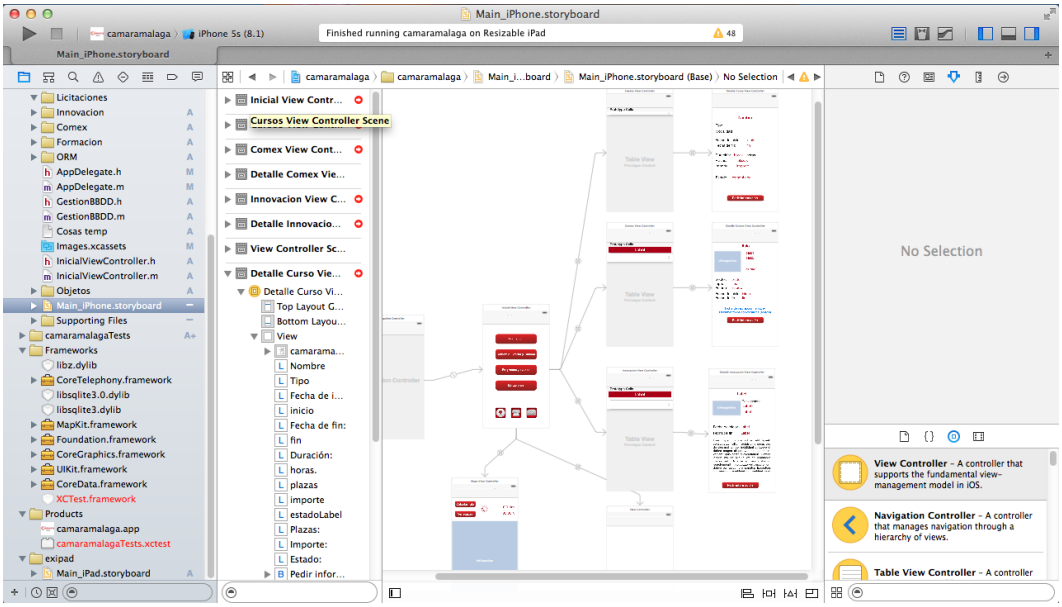


Figura 24- Canvas de Xcode

aplicación, en cuanto a su navegación se refiere.

El uso de *storyboard* facilita el diseño de ciertos componentes, pero cuando se requiere personalizar controles o acciones, esto ha de hacerse en el código de la clase asociada con la *View* correspondiente. También dificulta el trabajo colaborativo, ya que solo un desarrollador podría modificar el fichero cada vez.

4.4.1.4 Archivos Xib

Estos archivos contienen una descripción *XML* de los elementos y relaciones de la interface de usuario. Contienen una descripción de la interfaz, y se asocian a clases para darles funcionalidad. Se usa muy a menudo para crear *Vistas o Views* personalizadas, diferentes de las estándares de Xcode, o mezclando instancias de ellas.

Para este proyecto, los objetos construidos se han limitado a las celdas que presentan los listados en las tablas:

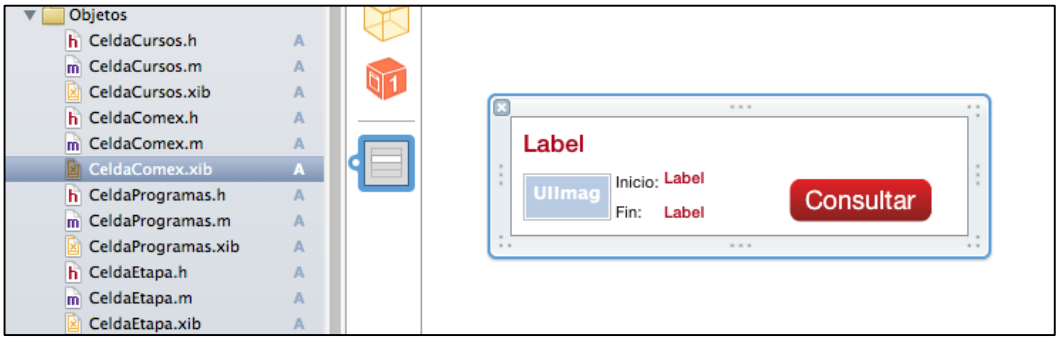


Figura 25- Archivos XIB

4.4.1.5 Delegates y AppDelegate

El concepto de ***Delegado* o *Delegate*** es quizás uno de los conceptos más importantes que hay que tener claro cuando se programa en Objective-C, ya que de ello depende el funcionamiento integrado de la aplicación.

Un *delegado* es un objeto que obtiene notificaciones cuando el objeto al que esté conectado recibe un evento destinado al primero o cambia de estado. Por ejemplo, cuando una V*ista o View* va a presentar una tabla (*TableView*), esta se convierte en su *delegado*:

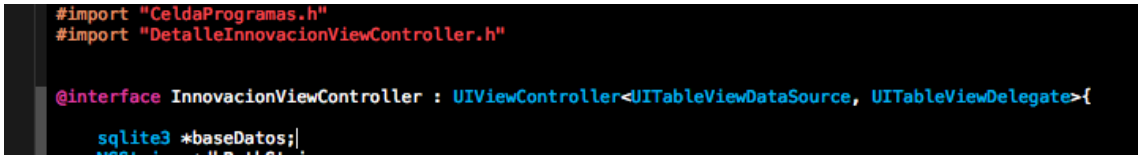


Figura 26- TableViewDelegate

Así, el *Delegado de la Aplicación o* ***AppDelegate*** es un objeto que recibe notificaciones cuando el objeto *UIApplication* alcanza ciertos estados.

> La clase *UIApplication* provee "un punto de control centralizado" para las aplicaciones corriendo en iOS. Cada aplicación tiene que tener exactamente una instancia de *UIApplication* (o una subclase de esta). Cuando una aplicación se lanza, se llama a la función *UIApplicationMain*, que crea un objeto *UIApplication* único.

El *AppDelegate* gestiona algunos estados especiales del objeto *UIApplication*, entre los que podemos destacar:

- *applicationDidFinishLaunching*: - Cuando este estado se alcanza, se llama a la función *application:didFinishLaunchingWithOptions* de la clase *AppDelegate*. Todas las opciones de configuración personalizada y de diseño que afecten a toda la aplicación suelen implementarse en esta función.

- *applicationWillTerminate*: Este estado se alcanza cuando la aplicación va a terminar, invocando a la función del mismo nombre del *AppDelegate*, y es donde se elimina de memoria la aplicación. Suele usarse para borrar los archivos temporales, imágenes, etc. creadas por la aplicación, y que no se van a necesitar en otras ejecuciones, o que el desarrollador considere necesario.

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{

      [[UINavigationBar appearance] setBackgroundImage:[UIImage imageNamed:@"transparente.png"] forBarMetrics:UIBarMetricsDefault];


    NSArray *path= NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    _docPath=[path objectAtIndex:0];
    _dbPathString = [_docPath stringByAppendingPathComponent:@"camara.db"];

    NSLog(@"%@",_dbPathString);

    GestionBBDD * ges =  [GestionBBDD new];

    [ges vaciaDatos];
    [ges cargaJSOn];


    return YES;
}
```

Figura 27- application:didFinishLaunchingWithOptions

En este código se coloca como fondo de la barra de navegación una imagen *transparente.png,* que se usará en todas las barras de navegación de la aplicación, y que será la que nos permita mostrar el logo de la Cámara en la esquina superior derecha de los interfaces de la aplicación.

Como variables de la clase *AppDelegate* se crean el path y la cadena que contendrá el acceso completo a la base de datos *cámara.db*. A lo largo de la aplicación estas variables estarán disponibles para todas las clases.

Puede verse también que se llama, al iniciar la aplicación, a las funciones que vacían la base de datos tras comprobar que hay conectividad a los datos actualizados, y que se cargan tras eso los nuevos datos.

4.4.2 Programación

La implementación completa de la aplicación conlleva la interacción de tecnologías Web, herramientas de escritorio para los usuarios y la aplicación móvil nativa.

4.4.2.1 Proceso previo a la carga

Durante la pantalla de carga, la aplicación accede a la dirección web de donde obtiene el objeto JSON. Una vez obtenida, se crea la base de datos, y las tablas correspondientes. A continuación se parsea la cadena, con el fin de almacenar los datos.

Durante este proceso se transforman las fechas del objeto JSON a tipo Date, y también se descargan las imágenes que son referidas en los datos descargados (banderas de países, logos de programas de ayuda, etc.). Estas imágenes se guardan en la carpeta de documentos de aplicación, tal y como muestra la Figura 28.

```
enlaceLogo=[programa objectForKey:@"logo"];
logo=[enlaceLogo lastPathComponent];
fotoData = [NSData dataWithContentsOfURL:[NSURL URLWithString:enlaceLogo]];
nuevoPrograma.logo=[NSString stringWithFormat:@"%@/%@",appDelegate.docPath,logo];
[fotoData writeToFile:nuevoPrograma.logo atomically:YES];
```

Figura 28- Descarga de imágenes

Se ha optado por esta forma de proceder con la carga de las imágenes por la gestión de tablas y pantalla que realiza iOS cuando una aplicación muestra una tabla que tenga en sus celdas imágenes. En la Figura 29 se muestra un conjunto de celdas personalizadas creadas durante el desarrollo de este proyecto:

Figura 29- Listado de acciones comerciales

iOS solo mantiene en memoria los datos e imágenes que permanecen en pantalla durante los desplazamientos, lo que implica que cada vez que uno se mueva por la tabla, se descargan y cargan de nuevo las imágenes ya antes mostradas. Esto produce un efecto visual muy molesto, dando la sensación de que la aplicación “se atasca” al deslizarse por los datos. También ocurre esto en los cambios de pantalla.

Dado que el número de imágenes, y sobre todo su tamaño, es muy pequeño, esto no supone un problema ni en el tiempo de carga de la aplicación, que es cuando se descargan las imágenes, ni en el tamaño de memoria ni usos de datos del terminal (Figura 30):

```
2014-12-07 10:41:45.931 camaramalaga[1332:601849] Debug: empezamos la descarga desde http://www.camaramalaga.com/
fileadmin/app/TFG_cam.php
2014-12-07 10:41:47.004 camaramalaga[1332:601849] Debug: dataURL descargado
2014-12-07 10:41:47.012 camaramalaga[1332:601849] Debug: empezamos a guardar en la  BBDD
2014-12-07 10:41:47.100 camaramalaga[1332:601849] DEBUG: Teóricamente se han insertado 12 cursos
2014-12-07 10:41:49.313 camaramalaga[1332:601849] DEBUG: Teóricamente se han insertado 12 misiones
2014-12-07 10:41:49.395 camaramalaga[1332:601849] DEBUG: Teóricamente se han insertado 11 licitaciones
2014-12-07 10:41:50.189 camaramalaga[1332:601849] DEBUG: Teóricamente se han insertado 5 programas
2014-12-07 10:41:50.190 camaramalaga[1332:601849] DEBUG: Cerramos la base de datos
```

Figura 30- Tiempo de carga

4.4.2.2 Implementación de la navegación y escenas

Como se ha mencionado anteriormente, en la aplicación se ha creado un archivo *.storyboard* para el diseño funcional y visual básico de la aplicación. También se ha definido en él el tipo de transiciones posibles entre las escenas (Figura 31):

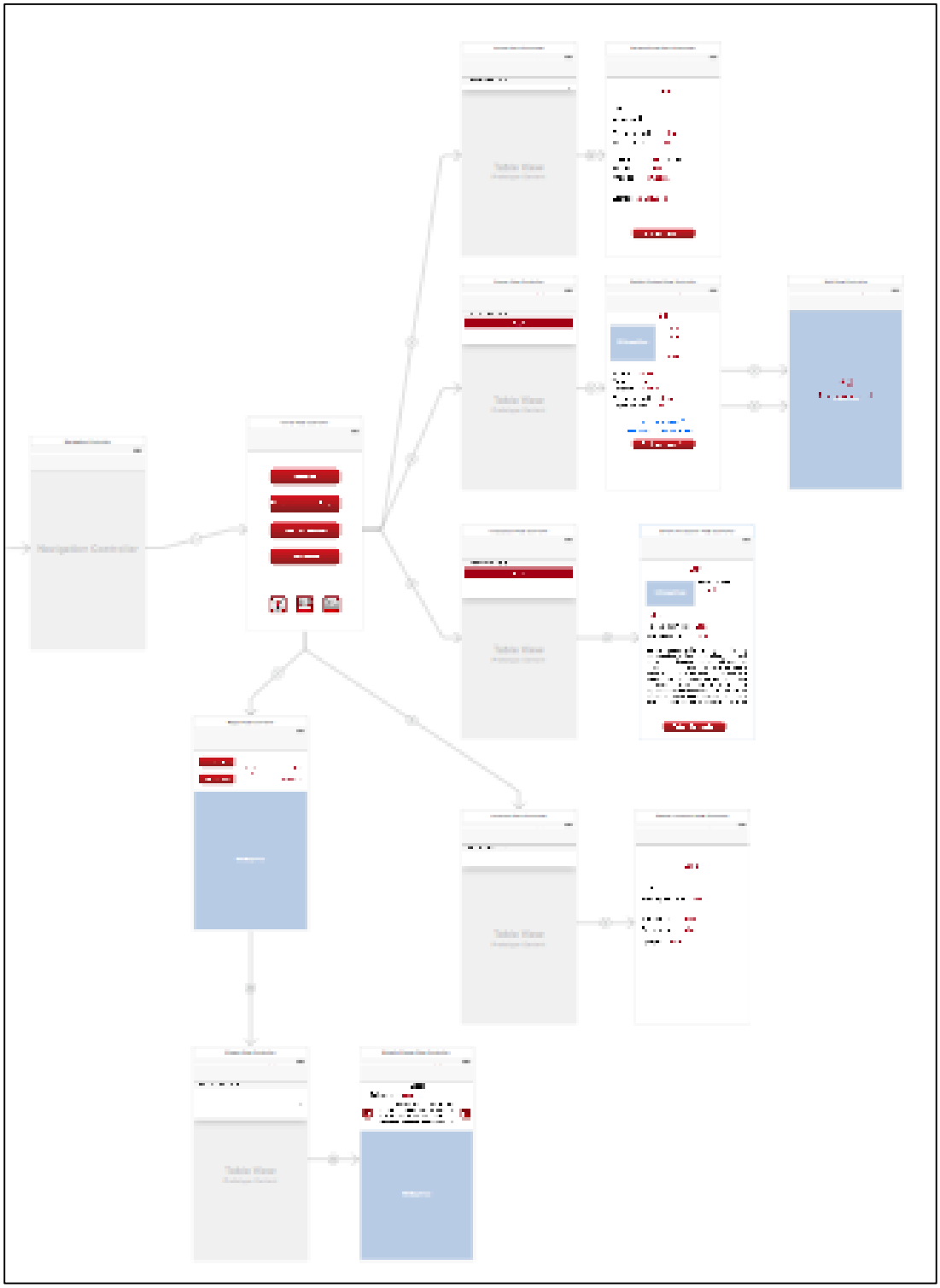


Figura 31- Storyborad

Se han creado las clases Objective-C para dar funcionalidad a los objetos y escenas. En estas clases se referencian controles (botones, etiquetas…) para darles diseño y funcionalidad, tratándolas como propiedades de las clases.

```
//
//  DetalleEtapasViewController.h
//  camaramalaga
//
//  Created by PACO ANDRADE on 08/11/14.
//  Copyright (c) 2014 movinnova. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>
#import "AnotacionMapa.h"

#define IS_OS_8_OR_LATER ([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0)

@interface DetalleEtapasViewController : UIViewController<MKMapViewDelegate>

{

}
@property (weak, nonatomic) IBOutlet UITextView *instructionsTextView;
@property (weak, nonatomic) IBOutlet UILabel *distanceLabel;
@property (weak, nonatomic) IBOutlet MKMapView *mapView;
@property (weak, nonatomic) IBOutlet UILabel *labelTitulo;

@property (strong, nonatomic) MKRouteStep *etapa;
@property (assign, nonatomic) NSUInteger numPaso;
@property MKRoute * ruta;

@property(nonatomic, retain) CLLocationManager *locationManager;


-(IBAction)verAnterior: (id)sender;
@property (weak, nonatomic) IBOutlet UIButton *botonAnterior;

-(IBAction)verPosterior: (id)sender;
@property (weak, nonatomic) IBOutlet UIButton *botonPosterior;

@end
```

Figura 32- DetalleEtapasViewController.h

Para poder modificar el diseño estándar proporcionado por Apple en su IDE, las tablas se han insertado dentro de otras *Views*, siendo sus delegadas. De esta forma podemos añadir imágenes y títulos, y crear secciones, manteniendo la funcionalidad de las tablas. El diseño de Apple solo permite un listado de los elementos de la tabla, por lo que su apariencia no era la deseada.

A estas tablas, en tiempo de ejecución, se le asignan como celdas las creadas en los archivos *.xib,* mencionadas en el punto 4.4.1.4.

```
-(UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *simpleTableIdentifier = @"SimpleTableCell";
        //creamos una celda, pero de la clase personalizada nuestra
    CeldaLicitaciones *cell = (CeldaLicitaciones *)[tableView dequeueReusableCellWithIdentifier:simpleTableIdentifier];
    if (cell == nil)
    {
        NSArray *nib = [[NSBundle mainBundle] loadNibNamed:@"CeldaLicitaciones" owner:self options:nil];
        cell = [nib objectAtIndex:0];
    }
    Licitacion *lici;
    lici=[listadoLicitaciones objectAtIndex:indexPath.row];
    cell.labelNombre.text=lici.contratacion;
    cell.labelEstado.text=lici.estado;
    cell.labelTipo.text=lici.tipo;

    [cell.botonConsultar addTarget:self action:@selector(detalle:) forControlEvents:UIControlEventTouchUpInside];

    return cell;
}
```

Figura 33- Carga de celdas personalizadas

Para llevar a cabo el traspaso de información de una escena a otra, se crean instancias de la clase personalizada a la que pertenece la de destino, de manera que se puede asignar valores a sus propiedades:

```
if ([segue.identifier isEqualToString:@"verEtapa"])
  {
    DetalleEtapasViewController *controlador = segue.destinationViewController;
    [controlador setEtapa:etapaSeleccionada];
    [controlador setRuta:self.ruta];
    [controlador setNumPaso:numPaso];

  }
```

Figura 34- Transición entre escenas: segue

4.4.2.3 Gestión de mapas: MapKit de Apple.

Antes de iOS 5.1 el ***MapKit* framework** de Apple usaba el Servicio de mapas de Google (*Google Mobile Maps* (*GMM*)) para su funcionamiento.

Ya en iOS 6 empezó el *MapKit* a presentar más funcionalidades, y unas aproximaciones y cálculos mejores.

Fue con la llegada de iOS 7 con la que se añadieron funcionalidades más importantes a este *framework,* de las que se han usado las siguientes:

- ***MKDirections***, que nos permitirá obtener y pintar sobre el mapa rutas entre distintos puntos.
- ***MKOverlayRenderer***, que nos permite gestionar las capas sobre el mapa. Se usa para pintar las líneas de ruta, y puede permitirnos superponer una sobre otras, darles transparencia, etc.
- ***MKPolyline****,* para pintar una línea en el mapa.
- ***transportType****,* que permite especificar el método de transporte. (*MKDirectionsTransportTypeAutomobile*)
- Distancia y tiempo estimado (***distance*** y ***expectedTravelTime***).
- ***MKRouteStep****,* que nos permite gestionar cada etapa de la ruta.

```
- (MKOverlayRenderer *)mapView:(MKMapView *)mapView rendererForOverlay:(id<MKOverlay>)overlay
{
    MKPolylineRenderer *renderer = [[MKPolylineRenderer alloc] initWithPolyline:overlay];
    renderer.strokeColor = [UIColor redColor];
    renderer.lineWidth = 3.0;
    return  renderer;
}
```

Figura 35- MKOverlayRenderer

La primera vez que la aplicación se ejecuta en un dispositivo, al entrar en la escena del mapa, muestra la localización de la sede central de la Cámara de Comercio, y le ofrece la opción de calcular la ruta.

Cuando el usuario pulsa el botón de cálculo de ruta, le informa del tiempo estimado y distancia para llegar al destino, y le ofrece la opción de ver la ruta por etapas:

Figura 36- Cálculo de ruta

En la pantalla del listado de etapas, la aplicación ofrece seleccionar una etapa para poder ver exactamente su duración, recorrido y distancia.

Figura 37- Ruta por etapas

En la vista de detalle de la etapa, puede ir avanzando o retrocediendo etapa por etapa.

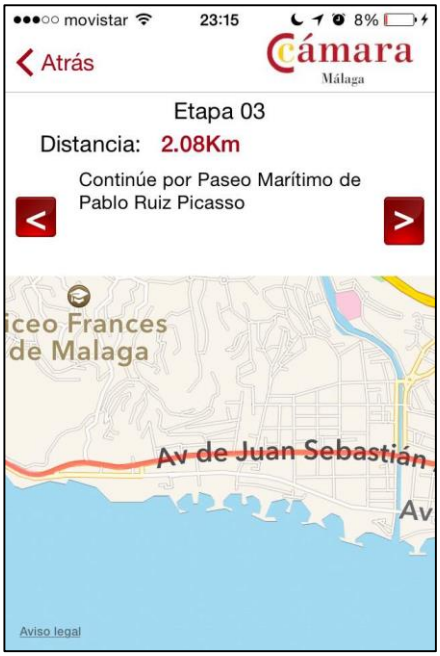


Figura 38- Etapa de la ruta

4.4.3 Dificultades en el desarrollo

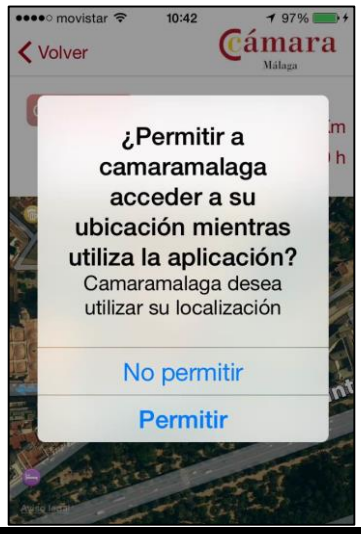
Durante el desarrollo de la aplicación, hablando solamente de la parte de la aplicación móvil, se han encontrado varias dificultades, sobre todo por la modificación de requisitos y funcionalidades, motivadas por el cambio de versiones de iOS (empecé con la 7, y estamos en la 8.2), por las salidas de las nuevas versiones (subversiones) del IDE, y por la aparición de los iPhone 6 y iPhone6+, que han introducido novedades en el sistema de diseño de interfaces de Apple.

Con la liberación de iOS8, iOS8.1 y ahora iOS8.2, Apple ha ido actualizando su SDK y Xcode, modificando algunos aspectos del diseño de la interfaz, sobre todo, en lo que a este programa se refiere, ocurriendo que algunos métodos y formas de ajustar la aplicación a los distintos dispositivos por el tamaño de sus pantallas y resoluciones, que antes se hacían, se han depreciado, y ya no funcionaban, con lo que he tenido que actualizarlos, de manera que mantengan, además, la compatibilidad hacia atrás en iOS.

Con la salida al mercado de los nuevos iPhone6 y iPhone6+, Apple ha introducido dos tamaños nuevos de pantalla, lo que me ha supuesto calcular de nuevo el ajuste de las pantallas y controles a estas, para que se mostrasen bien.

En la parte de gestión de mapas, las dificultades han sido, principalmente, estas:

En primer lugar, con la aparición de iOS8, las aplicaciones deben pedir permiso para calcular la posición del usuario:




```
#define IS_OS_8_OR_LATER ([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0)
```

```
self.locationManager = [[CLLocationManager alloc] init];
if(IS_OS_8_OR_LATER) {
    [self.locationManager requestWhenInUseAuthorization];
        // [self.locationManager requestAlwaysAuthorization];
}

[self.locationManager startUpdatingLocation];
```

Figura 39- Nuevos permisos iOS8

La complejidad ha venido motivada por un error en el diseño del framework de Apple. Cuando se calcula la ruta por etapas, hay algunas en las que no se indica movimiento, sino que solo se indica un giro. En estas etapas, la librería no asigna como origen y destino de la etapa las coordenadas de la posición actual, y el mapa aparece situado en el Océano Ártico:

Figura 40- Error del framework

Para solucionar esto, en este proyecto se desarrolló un método que, al mostrar una etapa de la ruta, comprueba que suponga un movimiento. Si es así, se procede tal y como se mencionó en la sección sobre gestión de mapas.

Si no tiene movimiento, lo que hace el procedimiento es calcular el punto donde se encuentra y centrar el mapa en él, en con un radio de 200 metros. Para esto necesitamos las coordenadas de dicho punto. El mapkit no las proporciona y para conseguirlas dibujamos (hemos de dibujarlo, si no, no se puede) el mapa de la etapa anterior y el polinomio de la ruta, que sí están almacenados. Una vez hecho esto, se utilizan las coordenadas del punto final de la línea en la pantalla (x,y), y se pregunta a qué coordenadas corresponde en el mapa, como si se estuviese tocando la pantalla. Aquí el mapkit nos devuelve el punto que necesitamos, y de esta forma se dibuja de nuevo el mapa actual con el punto centrado.

Si bien es cierto que este procedimiento no es óptimo, es transparente al usuario y no se aprecia ni parpadeo ni retraso. No se ha encontrado otra solución mejor para ello, hasta que Apple arregle el fallo del mapkit..

Resumiendo, en seudocódigo:

```
Si distancia=0 entonces
    Si paso=0 entonces
        centro_mapa=posición_usuario
    Sino
        dibujamos_paso_anterior
        calculamos coordenadas de pantalla (x,y) de fin del paso
        calculamos coordenas (lat, long) del mapa en ese punto de
    pantalla
        centramos el mapa en ese punto (radio=200m)
    Finsi
Finsi
```

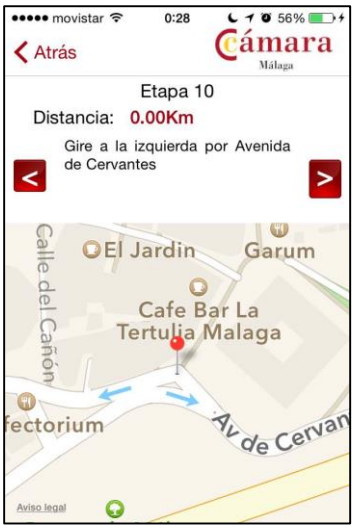


Figura 41- Corrección de la situación

4.5 Pruebas de la aplicación

Durante el transcurso de este trabajo y a la finalización del mismo, se han realizado pruebas para verificar el correcto funcionamiento de la aplicación desarrollada.

4.5.1 Pruebas de integración

Estas pruebas se han centrado en las comunicaciones y las conexiones entre los módulos de la aplicación.

Durante la ejecución de este trabajo se han actualizado las versiones de MS Access usadas para la gestión de los datos por parte de los usuarios responsables de su mantenimiento, las versiones de los gestores de contenido Joomla y Typo3, en los que se gestionan las bases de datos y contenidos mostrados en la aplicación, y también el sistema operativo móvil iOS se ha actualizado, a la versión 8, y posteriores subversiones.

Se ha verificado la correcta funcionalidad de las comunicaciones entre las diferentes versiones

4.5.2 Pruebas de compatibilidad

Se ha verificado el correcto funcionamiento y visualización del software desarrollado en los dispositivos reales iPhone 4S, 5, 5S, iPad 2, y en el simulador de los iPhone 6 y 6+ que proporciona Apple para tal fin., comprobándose el correcto visionado, como se mostró en la Figura 18.

Capítulo 5: Conclusiones y trabajos futuros

Una de las conclusiones más importantes consecuencia de la realización de este proyecto está relacionada con la funcionalidad en las aplicaciones móviles.

Al diseñar una aplicación móvil, hay que tener en cuenta el tamaño de pantalla menor en el que podrá mostrarse la información. Con independencia de que la aplicación deberá adaptarse a los distintos tamaños, proporciones y resoluciones que los dispositivos puedan tener, esta es la que nos delimitará o dictará la funcionalidad que le podremos dar a cada pantalla en nuestra aplicación. Hemos de ajustar la información y controles al menor tamaño de pantalla para que la usabilidad de la aplicación sea la correcta.

No podemos desarrollar una aplicación que muestre funcionalidades u opciones distintas según el tamaño de la pantalla del dispositivo en que se está ejecutando, ya que esto dificultaría exponencialmente la dificultad y coste de mantenimiento, además del de desarrollo.

En este sentido, en el desarrollo de esta versión inicial de la aplicación móvil de la Cámara de Comercio, las limitaciones han significado la imposibilidad de recrear los formularios de inscripción existentes, ya que la cantidad de información que se pide en unos casos, la necesidad de adjuntar documentación en otros, y ciertos requisitos legales, como la necesidad de firmar electrónicamente alguna entrega, no han sido solventadas, en gran parte porque el proceso de inscripción no es modificable al venir especificado por la Cámara de España o protocolos de la UE.

Pretendemos, en las próximas versiones de las aplicaciones, tanto Android como iOS, añadir funcionalidades como:

- Servidor de notificaciones push, para indicar plazos, nuevas ofertas, etc.
- Sistema de reservas de plaza.
- Concertación de citas con los técnicos.

Todas estas funcionalidades requieren un desarrollo tanto web como de las aplicaciones móviles.

Bibliografía

Libros:

- **iOS 6 By Tutorials, By the raywenderlich.com Tutorial Team.** *Adam Burkepile, Charlie Fulton, Matt Galloway, Jacob Gundersen, Kauserali Hafizji, Matthijs Hollemans, Felipe Laso Marsetti, Marin Todorov, Brandon Trebitowski, Ray Wenderlich.* Copyright © 2012 Razeware LLC.
- **Mobile Design and Development**; *Brian Fling*; O'REILLY; 2009
- **iOS App Programming Guide**, © 2013 Apple Inc.

Webs de documentación:

- https://developer.apple.com

Webs de consulta:

- http://stackoverflow.com/
- http://www.cocoalab.com/
- http://www.raywenderlich.com/

Referencias bibliográficas

[1] Inicios de la telefonía móvil en España
http://www.xataka.com/historias-de-la-tecnologia/la-prehistoria-de-la-telefonia-movil-en-diez-anuncios-de-los-90

[2] Evolución de las redes móviles
http://es.wikipedia.org/wiki/Wireless_Application_Protocol

[3] historia de los mercados de aplicaciones móviles
http://blog.shoutem.com/2012/02/07/infographic-the-history-of-mobile-app-stores/

[4] Primeras aplicaciones móviles
http://applicantes.com/historia-primeras-apps-iphone-2007/

[5] El App Store
http://en.wikipedia.org/wiki/App_Store_(iOS)

[6] Estadísticas del App Store
"AppleAppStoreStatistics" by W3bbo (talk)

[7] El éxito de las plataformas de aplicaciones móviles
http://www.gradiant.org/actualidad/noticias/606-el-exito-de-las-plataformas-de-aplicaciones-moviles.html

[8] Observatorio Nacional de las Telecomunicaciones y la SI
http://www.ontsi.red.es/

[9] iOS
http://es.wikipedia.org/wiki/IOS