iDashBoard, cuadro de mando operacional

Ángel Luis Peñalosa Roiz
Ingeniería Técnica en Telecomunicaciones, especialidad Telemática
Aplicaciones multimedia de nueva generación

Manel Llopart Vidal
David García Solórzano

08 de junio de 2016

FICHA DEL TRABAJO FINAL

Título del trabajo:	*iDashBoard, cuadro de mando operacional*
Nombre del autor:	*Ángel Luis Peñalosa Roiz*
Nombre del consultor/a:	*Manel Llopart Vidal*
Nombre del PRA:	*David García Solórzano*
Fecha de entrega (mm/aaaa):	06/2016
Titulación::	*Ingeniería Técnica en Telecomunicaciones, especialidad Telemática*
Área del Trabajo Final:	*Aplicaciones multimedia de nueva generación*
Idioma del trabajo:	*Castellano*
Palabras clave	*iDashBoard, Swift, iOS*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Este trabajo fin de carrera pretende desarrollar una aplicación para dispositivos iOS a través del entorno IDE, XCode, proporcionado por Apple. Será necesario implementar habilidades y conocimientos adquiridos durante la carrera de Ingeniería Técnica en Telecomunicaciones (Telemática). Uno de los puntos necesarios para abordar este trabajo será sumergirse en la programación orientada a objetos a través del nuevo lenguaje de programación Swift.

Bajo este contexto se quiere desarrollar una herramienta de control y gestión bajo una interfaz gráfica de usuario, esta herramienta se define habitualmente como cuadro de mando, en inglés DashBoard. La función de esta herramienta es informar en tiempo real los datos específicos del departamento de una empresa, en este caso se ha elegido simular los datos de una empresa de venta con diferentes productos. El departamento de ventas será el encargado para la captación de clientes elaborando campañas de ventas, ofertas… etc.

Estos datos serán leídos por la aplicación a través de un fichero genérico en formato XML (eXtensible Markup Language), el cual será definido previamente. Esto ofrece mucha flexibilidad, permitiendo adaptarla con facilidad a cualquier necesidad y cliente

al estar desacoplada del modelo de datos de origen.

La aplicación cargará el fichero XML y parseará los datos en una serie de objetos que se definirán y agruparán en arrays de datos, estos objetos serán pequeñas estructuras creadas bajo el nuevo lenguaje de programación Swift de Apple. La capa de diseño de la interfaz gráfica será realizada bajo el programa de diseño vectorial Sketch y mostrará un skin agradable para la muestra de datos.

El presente proyecto definido como cuadro de mando o dashboard (más conocido dentro del mundo del software), es una interfaz gráfica de usuario donde el usuario puede disponer de cierta información del sistema y poder analizarla de manera relevante.

Los tiempos en un proyecto son muy limitados por lo que es necesaria una adecuada organización para cumplir las fechas de entregas, implementar todas las funcionalidades y construir una interfaz amigable y entendible por el usuario.

Abstract (in English, 250 words or less):

This project will develop an application for iOS devices through the IDE, XCode environment provided by Apple. It will be necessary to implement skills and knowledge acquired during the career of Engineering in Telecommunications (Telematics). One of the points needed to perform this work will dive into object-oriented through the new Swift programming language.

In this context we want to develop a tool control and management under a graphical user interface, this tool is usually defined as DashBoard. The function of this tool is to report real-time data specific department of a company, in this case we have chosen to simulate the data of a company selling different products. The sales department will be responsible for customer acquisition developing sales campaigns, offers ... etc.

These data will be read by the application through a generic file in XML (eXtensible Markup Language), which will be previously defined. This offers a lot of flexibility, allowing adapt easily to any customer need and be uncoupled model source data.

The application will load the XML file and will parse the data into a series of objects to be defined and grouped in arrays of data, these objects will be small structures created

under the new Swift Apple language programming. Layer design of the graphical interface will be conducted under the program vector design Sketch and show a nice skin sample data.

This project defined as dashboard (better known in the world of software) is a graphical user interface where the user can have some system information and to analyze it in a relevant way.

The times when a project is very limited so that an appropriate organization to meet delivery dates, implement all the features and build an interface and understandable by the user is necessary

Agradecimientos

En estas líneas me gustaría hacer protagonistas a las personas que realmente han hecho que esto sea un sueño hecho realidad para mí, por su apoyo, por su comprensión y por su cariño.

Gracias María! que siempre me has apoyado y has sacrificado tantos días por mis estudios, incluso en estos últimos días con nuestra pequeña en tu barriguita. Mis herman@s, que siempre han estado tan orgullosos de mí y que me animaban a seguir para delante. A mis amigos que también han sufrido lo suyo y me han apoyado tanto. Y especialmente me gustaría agradecer a mis padres, los artífices de ser lo que soy a día de hoy. Papá lo conseguí!! seguro que estás con una sonrisa de las tuyas. Gracias por ser los mejores padres del mundo y por creer siempre en mí.

Gracias familia!! Gracias amigos!! Gracias!!

Índice

Tabla de contenido

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día las empresas necesitan de herramientas de gestión y control para el tratamiento de la información de su negocio, a veces esto se complica por la complejidad del software o la limitación del hardware frente a soluciones portables.

La evolución de los terminales móviles ha crecido considerablemente en los últimos años avanzando paralelamente junto a los sistemas de transmisión de datos y las conexiones inalámbricas. Han aparecido nuevos lenguajes de programación para los sistemas operativos de los dispositivos móviles, abriendo la posibilidad de desarrollar aplicaciones Ad hoc de una manera flexible y rápida.

Todo esto ha posibilitado que se desarrollen soluciones de software distintas para estos nuevos dispositivos ante el gran rendimiento que ofrecen a nivel de hardware.

Con este proyecto se pretende desarrollar una aplicación Ad hoc para la plataforma iOS de Apple que dé una solución de visibilidad a los datos del negocio que se requieran, siendo accesible desde cualquier lugar con conexión a internet y sin tener que estar dentro de la red local de la empresa.

1.2 Objetivos del Trabajo

- Crear una aplicación móvil que sirva como informe dinámico de resultados dado un fichero de datos genérico
- Planificación de los hitos y recursos
- Creación de una interfaz gráfica con Sketch
- Creación de la capa lógica con Swift
- Lectura del fichero de datos genérico
- Gestión del proyecto
- Escalabilidad para la aplicación
- Superar la asignatura

1.3 Enfoque y método seguido

Partimos de la elaboración de un nuevo producto basándonos en ciertas necesidades recogidas ante el auge de las nuevas tecnologías móviles. De esta manera se intenta elaborar un producto genérico que pueda integrarse de una manera sencilla al modelo de datos de cada compañía.

Se apuesta por el lenguaje de programación Swift en lugar de Objetive C, este último un lenguaje más antiguo y menos potente de los dos que nos ofrece Apple. Es cierto que al elegir Swift nos encontramos con menos documentación y recursos frente a Objetive C, un veterano dentro de la plataforma Apple, por lo que el nivel de investigación y esfuerzo de aprendizaje será mayor.

Será necesario partir de un recurso que nos facilite una fuente de datos que alimente la aplicación, para este caso y teniendo en cuenta que cada compañía tendrá un software de gestión de base de datos diferente, se opta por la exportación de los datos mediante un fichero genérico de datos estructurado que se subirá a la nube. De esta manera, al ser un fichero genérico, permite adaptarse con gran facilidad a cualquier necesidad y cliente.

En una futura versión se pretenderá ampliar la aplicación para permitir una configuración paramétrica, de tal manera que se puedan editar los campos, sus posiciones… etc.

1.4 Planificación del Trabajo

Se contará con un único recurso que adoptará diferentes roles en la ejecución del proyecto y al ser un proyecto académico el coste real es cero, por lo que resulta viable a nivel económico.

Sobre los recursos de software y hardware, Apple ofrece su IDE (Integrated Development Environment) Xcode de manera gratuita y para el hardware se dispone de un ordenador Apple requerido para los desarrollos por la propia marca, por lo que resulta viable a nivel técnico.

Sobre los plazos en la tareas realizadas se adjunta un diagrama de Gantt.

1.5 Breve sumario de productos obtenidos

Memoria del Proyecto
Fuentes de la aplicación
Aplicación compilada

1.6 Breve descripción de los otros capítulos de la memoria

Estado del arte
Se realiza una investigación documental que permite el estudio del conocimiento acumulado dentro de un área específica por lo que se ha trabajado en la línea del desarrollo de aplicaciones móviles.

Diseño
Señala con claridad la lógica que vincula las conclusiones de la etapa de análisis con los objetivos, resultados esperados y actividades del proyecto

Implementación
Se basa, esencialmente, en poner en práctica el diseño del proyecto y obtener el resultado del producto.

2. Estado del Arte

2.1 Introducción al desarrollo en dispositivos móviles

El teléfono inteligente, Smartphone en inglés, está considerado como la clave para el acceso a la red y se ha multiplicado su funcionalidad debido a las diversas evoluciones que ha tenido en muchos de sus componentes. Actualmente integran cámara de fotos y video, GPS, entre otros usos.

La penetración de los Smartphone es impactante a nivel mundial y a muy corto plazo toda la población mundial tendrá uno. A su vez tiene un efecto macroeconómico, se puede destacar como ejemplo la estimación sobre los nuevos iPhone, cuyo lanzamiento puede impulsar el PBI de los EE.UU. en un 0.5%.

Muchos medios demuestran que la adquisición de Smartphone y de las comunicaciones abarca el mayor crecimiento a nivel mundial.

Ilustración 1

Para 2016 se estima que haya más 2 mil millones de usuarios de Smartphone en el mundo.

En su momento, Steve Jobs calificó al iPad como un producto revolucionario y el mejor que jamás haya realizado. Otros de los dispositivos móviles con gran impacto son las tabletas. Son dispositivos que permiten realizar nuevas tareas y que redefinen la forma de hacer otras muchas.

Ilustración 2

El interés por las tabletas es impresionante, estimando más de 300 millones de dispositivos en el mercado y que en 2015 las tabletas superen en número a los ordenadores.

Estamos ante una evolución tecnológica sin precedentes provocando un crecimiento exponencial de las aplicaciones desarrolladas específicamente para estos nuevos sectores.

El desarrollo de estas aplicaciones supone un aprendizaje nuevo para cualquier programador debido a las capacidades técnicas que requiere y que suponen un cambio con respecto a los modelos existentes de otras aplicaciones.

Varios factores a tener en cuenta a la hora de desarrollar aplicaciones para dispositivos móviles.

2.1.1 Procesamiento/Memoria

Se debe cuidar las limitaciones de los dispositivos ya que no se dispone de los mismos recursos que tendríamos con un ordenador, para ello hay que tener bien estudiadas las capacidades técnicas de los dispositivos. Se deben desarrollar muy bien los algoritmos de procesamiento para optimizar y hacer uso de los recursos de una manera óptima y estable.

2.1.2 Interfaces

Los tamaños de estos dispositivos es variado y por lo general difiere mucho con las versiones de escritorio para ordenadores. Por ello, hay que tenerlo en cuenta a la hora de desarrollar la interfaz de usuario de la aplicación pues no sólo no son compatibles por tamaño, también por tener funcionalidades diferentes. Es necesario centrarse en la sencillez del diseño e implementarlo de la manera más intuitiva posible.

2.1.3 Almacenamiento

Un punto aún por pulir en los dispositivos móviles es su limitación de almacenamiento, aunque ya hay avances importantes que posibilitan un mayor almacenamiento elevando su coste. Es necesario cuidar en los desarrollos la información a almacenar en el dispositivo así como buscar alternativas, por ejemplo la nube, para no elevar mucho el espacio requerido de almacenamiento.

Las aplicaciones móviles son pequeños programas que muestran cierto contenido y permiten realizar diferentes funcionalidades, existen varias plataformas que facilitan descargar e instalar en los dispositivos inteligentes, por ejemplo App Store y Google Play.

El término apps se ha convertido mundialmente en un término común, miles de millones de aplicaciones se descargan desde las tiendas virtuales o provocando que se desarrollen cientos de miles de apps. Apple tiene más de 1.200.000 de apps y Android roza los 1.500.000 de apps, esto recuerda a la fiebre del oro.

La experiencia móvil se ha convertido en algo muy importante para los usuarios, los desarrollos de programas móviles crean una experiencia a la medida de los clientes, con creación de aplicaciones fantásticas, productivas e impactantes.
Pueden dirigirse hacia las necesidades de clientes, representantes, directores, etc.

2.2 Sistema Operativo iOS

Es el sistema operativo para todos los dispositivos móviles de Apple, posee una interacción con el sistema operativo que incluye gestos como deslizar, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz.

El kernel de iOS nace de OS X, por lo que es un sistema operativo tipo Unix, cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de “Servicios Presenciales”, la capa de “Medios” y la capa de “Cocoa Touch”.

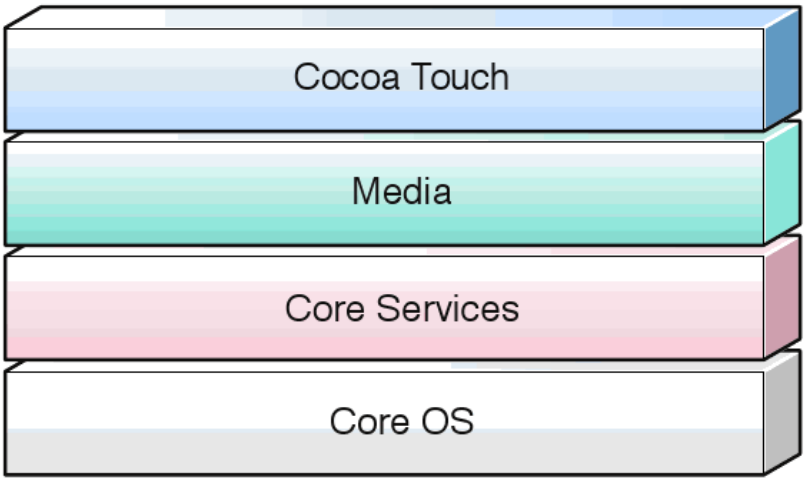

Ilustración 3

Apple se caracteriza por diseñar y crear sus propios equipos tanto nivel de hardware como de software, evitando problemas de compatibilidad y aumentando bastante el rendimiento de sus equipos con respecto a otros fabricantes.

Con el tiempo decidieron permitir aplicaciones por parte de terceros controlando estas mediante supervisión de ellos y licencias para el desarrollo, todo esto a partir de una SDK muy detallada. Esto ha permitido el aumento e interés por los desarrolladores que han ido implementando aplicaciones hasta llegar a números verdaderamente impensable años atrás.

Hoy en día han cambiado la filosofía de uso en la tecnología siendo una de sus

prioridades la experiencia del usuario puliendo al máximo los detalles para hacer todo de una forma más sencilla y simple al usuario.

Como es de notar iOS será el sistema elegido para el desarrollo de este trabajo, centrándonos en uno de los productos con más éxito de Apple en dispositivos móviles, el iPhone.

2.3 Arquitectura

Como ya se ha hablado anteriormente cuenta con una estructura de capas, la capa del “Núcleo” abarca los controladores y las interfaces de bajo nivel que usan todas las aplicaciones, incluyen características como ficheros del sistema, seguridad, gestión de memoria, drivers del dispositivo, etc. Estas interfaces se basan en C e incluyen tecnologías como Core Foundation, acceso a hilos POSIX, Cálculos matemáticos, Red (sockets BSD API) entre otros.

La capa de servicios, llamada “Servicios Presenciales” contiene los servicios fundamentales del sistema operativo que pueden ser usadas por todas las aplicaciones creadas por terceros, permitiendo al usuario acceder a todos los servicios básicos.

En la capa de “Media” se provee los servicios gráficos y multimedia de la capa superior, se dan las tecnologías basadas en E/S, Core Text, Quartz, GLKit y OpenGL, que son utilizadas para dar soporte al audio, video y dibujos 2D y 3D así como el motor de animación Core Animation.

La última capa, “Cocoa Touch”, es la más importante para el desarrollo de aplicaciones iOS ya que posee una API que proporciona un conjunto de Frameworks, está formada por dos Framework fundamentales:

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

Las capas más altas actúan como intermediarios entre el hardware y las

aplicaciones que se muestran en pantalla. Las capas superiores proporcionan una abstracción orientada a objetos haciendo que sea más fácil entender su arquitectura. Adicionalmente, cada capa está compuesta por un conjunto de frameworks.

Ilustración 4

2.4 Tipología de aplicaciones

Actualmente existen tres tipos de aplicaciones a la hora de desarrollar para iOS.

2.4.1 App Nativa

Es la aplicación que se desarrolla de forma específica para un determinado sistema operativo, llamado Software Development Kit o SDK. Además no requieren Internet para funcionar, por lo que ofrecen una uso más fluido y están realmente integradas al teléfono, lo cual les permite utilizar todas las características de hardware, como la cámara y los sensores (GPS, acelerómetro, giróscopo, entre otros).

A nivel de diseño, tienen una interfaz basada en las directrices del sistema operativo, logrando mayor consistencia y coherencia con el resto de aplicaciones y con el propio SO. Esto favorece la usabilidad y beneficia directamente al usuario que encuentra interfaces familiares.

2.4.2 Web App

Es la aplicación que se desarrollada con lenguajes muy conocidos por los programadores, como es el HTML, Javascript y CSS. Muestran algunos inconvenientes y restricciones en factores importantes como la gestión de memoria y no permiten aprovechar al máximo el potencial de los diferentes componentes de hardware del dispositivo.
Las aplicaciones web suelen tener una interfaz independiente del SO y más genérica, por lo que el usuario tendrá una experiencia con los elementos de navegación menor que en el caso de las nativas.

2.4.2 Web App nativa

Es una aplicación híbrida, una combinación de las dos anteriores, se podría decir que recoge lo mejor de cada una de ellas. Una vez que la aplicación está terminada, se compila o empaqueta de forma tal, que el resultado final es como si se tratara de una aplicación nativa.
Tienen un diseño visual que no suele identificarse con el del sistema operativo. En cambio, hay formas de usar controles y botones nativos para aproximarse más a la estética propia.
Existen varias herramientas para desarrollar este tipo de aplicaciones como Apache Cordova, que es una de las más populares, e Icenium.

Entonces… ¿Cuál usar?

Este proyecto se basará en las aplicaciones nativas ya que ofrecen una mejor experiencia de uso y sobre todo, rendimiento. Algunas apps como Facebook o LinkedIn, antes eran hibridas pero con el tiempo han pasado a ser nativas por ese motivo.

2.5 Herramienta SDK

Ilustración 5

El SDK es un conjunto de herramientas preparado por Apple que se distribuye gratuitamente con lo necesario para que cualquier programador pueda desarrollar aplicaciones en iOS.

En ella se dispone de las interfaces, recursos y herramientas para desarrollar una app desde cualquier ordenador Mac.
Entre los recursos que podemos encontrar en un SDK se puede destacar:

2.5.1 La interfaz de programación de aplicaciones (API).

Es una abstracción del funcionamiento del entorno en el que se va a trabajar, formado por un conjunto de funciones, estructuras de datos, clases y variables que nos permiten manipular el mecanismo del sistema sin conocerlo internamente.

2.5.2 El entorno de desarrollo integrado (IDE).

Es el editor que nos ayuda a escribir fácilmente el código fuente del programa, generalmente brinda una interfaz amigable. Sus funciones principales son:

i. Debugger. Es la función que nos permite testear el programa en cada paso de su ejecución.
ii. Compilador. Se encarga de traducir el código fuente a lenguaje máquina, obteniendo el programa ejecutable.
iii. Código de ejemplo y otra documentación. Ejemplos para empezar a desarrollar aplicaciones.
iv. Emulador del entorno. Ejecuta de manera virtual una simulación de cómo lo vería el usuario final en su dispositivo.

Una de las herramientas más importante y que integra gran parte del SDK de iOS se denomina Xcode.

Xcode es el nombre que recibe el paquete de software diseñado por Apple que ofrece las herramientas necesarias para programar tanto en iOS como en OS X. Las herramientas más interesantes se definirán a continuación.

2.5.2.1 Xcode IDE

Es el editor de código que se encuentra muy integrado con los frameworks Cocoa y Cocoa Touch y ofrece un entorno de desarrollo fácil de usar y bastante productivo. De hecho es la herramienta que usó Apple para crear los sistemas operativos OS X y iOS.

Integra todas las herramientas que el desarrollador necesita en una única que permite suaves transiciones entre el editor de código, el depurador y el diseño de interfaces en la misma ventana.

Ilustración 6

Posee de una validador de código que simplifica y ayuda a la detección de errores de código inmediatamente mostrando el detalle del error, se llama Live Issues.

Posibilita la ejecución del código de la aplicación implementando el contenido en el dispositivo virtual y poder depurar el código creado. También contiene inspectores de variables en tiempo real de ejecución lo que facilita seguir desde código la depuración a la vez que lo visualizas en el simulador.

Para la compilación utiliza el compilador GNU Compiler Collection, un conjunto de compiladores creados por el proyecto GNU, es de software libre y se distribuye bajo la licencia general publica GPL.

2.5.2.2 Interface Builder

Nos permite la creación de interfaces de usuario de manera fácil y rápida, facilita el ahorro de código necesario para crear, configurar y posicionar los objetos de nuestra interfaces. Su función de editor visual que nos muestra ofrece obtener en tiempo real una visión exacta del aspecto que tendrá la interfaz. Los objetos pueden añadirse arrastrando y soltando componentes preconfigurados como switches, buttons, text fields, etc.

2.5.2.3 Instruments

Las aplicaciones de Apple se caracterizan por ofrecer una gran experiencia de usuario por lo que esto no implica sólo diseño, requieren que sean rápidas, aportando fluidez y velocidad.
Para ello Xcode facilita esta herramienta capaz de localizar los embudos de rendimiento en aplicaciones iOS.

Recopila en tiempo real datos, memoria, actividad de red o uso de CPU del dispositivo, incluso de forma remota, y posteriormente los muestra gráficamente en un timeline, lo cual facilita la detección de problemas de código.

2.5.2.4 Simulator

El simulador iOS ejecuta las aplicación desarrollada como si corriera en un dispositivo físico, nos muestra la forma real en la que la aplicación aparecerá y permite comprobar que todo funciona y es lo que el desarrollador espera.

2.6 Lenguajes de programación iOS

Actualmente existen dos lenguajes de programación para desarrollar en iOS, estos dos lenguajes coexisten conjuntamente sin problemas dentro de una misma aplicación, sus nombres son Objective-C y Swift.

2.6.1 Objective-C

Fue el primer lenguaje en desarrollo para iOS siendo una extensión del lenguaje C pero añadiendo una programación orientada a objetos. Aparentemente más sencilla e intuitiva que la de su predecesor nos muestra un código más fácil de entender a simple vista. Consiste en una capa muy fina por encima de C por lo que es totalmente compatible con código escrito en C.

Objective-C requiere que la interfaz e implementación de una clase estén en bloques de código separados. Por convención, la interfaz es puesta en un archivo cabecera y la implementación en un archivo de código. Los archivos cabecera, que normalmente poseen el sufijo .h, son similares a los archivos cabeceras de C. Los archivos de implementación (método), que normalmente poseen el sufijo .m, pueden ser muy similares a los archivos de código de C.

2.6.2 Swift

Es la nueva herramienta presentada por Apple, un nuevo lenguaje de programación que resalta por su sencillez y fluidez para crear códigos robustos. Se basa en el paradigma de la programación orientado a protocolo, lo que significa que prefiere el uso de protocolos, mejor conocidos como interfaces, en lugar de clases, logrando un código más flexible y modular.

Recoge las mejores características de C y Objective-C como el control de flujo, tipos, operadores, etc. e incluye algunas características enfocadas a la programación orientada a objetos como las clases y los protocolos así como hace del código algo más expresivo:

i. Múltiples valores de retorno.
ii. Iteración rápida sobre una colección.
iii. Estructuras que soportan métodos, extensiones y protocolos.
iv. Patrones de programación como mapas o filtros.
v. Etc.

Ilustración 7

2.7 Estudio de Mercado

Para asegurar el éxito de un producto es necesario conocer qué soluciones pueden existir dentro del desarrollo de aplicaciones móviles y la competencia existente antes de lanzar nuestra aplicación.

Sobre la idea expuesta para la aplicación móvil de este TFC analizaremos cómo está el mercado: cuales son nuestros competidores, los precios de las apps similares a las nuestras, en qué categoría se venden,… etc.

Hacer un estudio de mercado en el mundo de las apps es relativamente fácil en comparación con otros productos. Toda la información que necesitamos está recogida en unos puntos de venta que son, además, cerrados: la App Store.

Dentro de la App Store encontramos pocas apps referente al control del negocio empresarial adaptándose a los recursos de cada empresa, nos centraremos en aplicaciones de gestión de datos importados mediante algún medio, siendo totalmente adaptable para cualquier negocio de una empresa.

Encontramos una app interesante que enfoca el tratamiento de datos mediante

ficheros Excel disponibles desde la nube, estos datos importados y mostrados mediante un cuadro de mando. Esta app se asemeja a lo propuesto en este TFC solamente con la diferencia que la extracción y carga de datos se realizará mediante ficheros de lenguaje estructurado XML.

La app se denomina “andara pro: Balances Scorecard & Cuadros de Mando” y está englobada dentro de la sección “Economía y Empresa” y “Business & Strategy” de la app store.

Detalla las siguientes características:

“El control de su negocio al alcance de su mano, diseñe su propio Cuadro de Mando o Balanced Scorecard y controle eficientemente su grado de éxito mes a mes a través de la medición de sus indicadores mas importantes.

andara sincroniza sus datos de Excel con la app a través de su cuenta en la nube con box.com

Una herramienta antes exclusiva de las grandes multinacionales, ahora en el Ipad con un diseño fácil e intuitivo.

“El 75 % de las empresas del Ranking Fortune 500 lo tienen implantado, siga sus pasos”

Características versión PRO:

- Herramientas Colaborativas (añada comentarios y comparta documentos con sus colaboradores)

- Siempre sincronizado con Excel (1 mes gratis)

- Comparta sus Cuadros de Mando

- Reportes e Informes, envielos por mail a sus equipo.

- SSL Conexión Segura

- Modo de acceso Offline a sus datos

- Retos entre empleados para mejorar la productividad

andara es mas que una app, es una nueva experiencia en Business Intelligence y Gestión del Rendimiento en su compañía.

Sin costes de implementación a través del Marketplace de Plantillas, podrá seleccionar por Industria o por Rol plantillas predefinidas con todos los objetivos e indicadores mas relevantes para su compañía.

No necesitara licencias extras, solamente descargue andara desde la Appstore

y empiece en minutos a tomar las mejores decisiones y de la forma mas ágil.”

Se adjunta enlace de la app store para su consulta.

https://itunes.apple.com/es/app/andara-pro-balanced-scorecard/id532111723?mt=8

Ilustración 8

Es interesante como ofrece compras dentro de la app y dispone de diferentes plantillas para el modelado que necesite, enmascarando la aplicación en un rol especifico dentro del sector que requiera el usuario.

Este punto es bastante interesante a tener en cuenta para posibles mejoras de adaptabilidad a diferentes negocios dentro de la app propuesta.

Por otro lado implementa métodos de seguridad SSL permitiendo una conexión segura para los datos de la empresa, también bastante interesante para los datos que contengan información sensible.

Otra app encontrada interesante es “Bsc-i: Cuadro de Mando Integral – Balanced Scorecard”, basada también en un cuadro de mando pero más enfocada a Bussines Intelligence y generación de informes. Tambien se encuentra englobada dentro de la sección “Economía y Empresa” y “Business & Strategy” de la app store.

https://itunes.apple.com/es/app/bsc-i-cuadro-mando-integral/id414209134?mt=8

Ilustración 9

3. Diseño

El diseño de la aplicación se ha realizado utilizando la aplicación de diseño vectorial Sketch, siguiendo las normas de diseño del sistema operativo iOS, especificadas por parte de Apple en sus “Human Interface Guidelines” (https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html?utm_source=twitterfeed&utm_medium=twitter)

3.1 Tecnología utilizada

El proyecto se ha realizado utilizando el IDE de desarrollo proporcionado por Apple “Xcode”, el cual incluye todas las APIs y Frameworks necesarios para crear aplicaciones para la plataforma iOS.

Este IDE sólo corre bajo plataforma OSX, habiendo sido utilizada en este caso la última versión vigente: OSX “El Capitán” (10.11).

Además, se ha recurrido al uso de frameworks de terceros, mediante el uso de CocoaPods (https://cocoapods.org).

Los frameworks utilizados han sido:

i. ChameleonFramework: para la utilización de colores en gradiente predefinidos.(https://github.com/ViccAlexander/Chameleon)

ii. Spring: para la realización de animaciones sobre diferentes elementos de la interfaz. (https://github.com/MengTo/Spring)

3.2 Estructura

Respecto a la alimentación de datos para la aplicación se contempla el siguiente diagrama de bloques para definir su flujo.

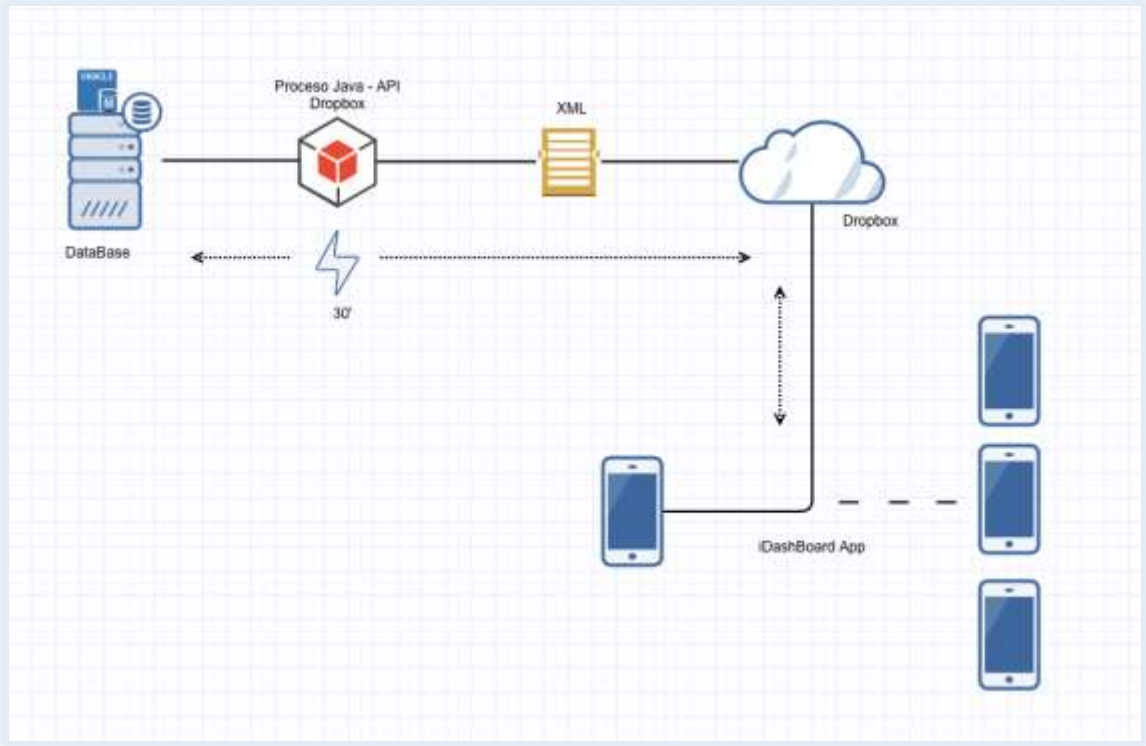

Ilustración 10

Para la obtención de datos en el cuadro de mando se realizará una extracción sobre una base de datos Oracle 11g mediante un proceso Java ejecutado mediante una tarea batch programada cada 30 minutos, este proceso elaborará un fichero XML (eXtensible Markup Language) que permitirá definir y estructurar un documento con los datos necesarios de la empresa para mostrar en la aplicación.

Este proceso depositará en la nube (Dropbox) el fichero para ser recuperado en cada ejecución o refresco por la aplicación, de tal manera que realizará un parseo de este fichero para la obtención de datos.

Será necesario conexión de datos en el dispositivo para que la aplicación pueda acceder a internet y tomar el fichero para su carga.

Previamente se ha definido obtener la información de los datos más sensibles y necesarios que se quieran mostrar, para ello tendremos como piloto una empresa que realiza ventas telefónicas y cuyo negocio vendrá definido por las

campañas de ofertas que ofrece al mercado y las agencias que operan de manera externa para la ejecución de sus ventas.

Esta empresa solicita un iDashBoard adaptado a sus necesidades, por lo que se realizará un enfoque personalizado de la aplicación utilizando la estructura de la aplicación.

De esta manera podemos ajustar el proyecto a un cliente real con unas necesidades reales a día de hoy, utilizando además campos y valores acordes al negocio enfocado.

A continuación se muestra el diagrama UML del flujo funcional de la aplicación, se puede identificar la relaciones entre las diferentes vistas y sus objetos.

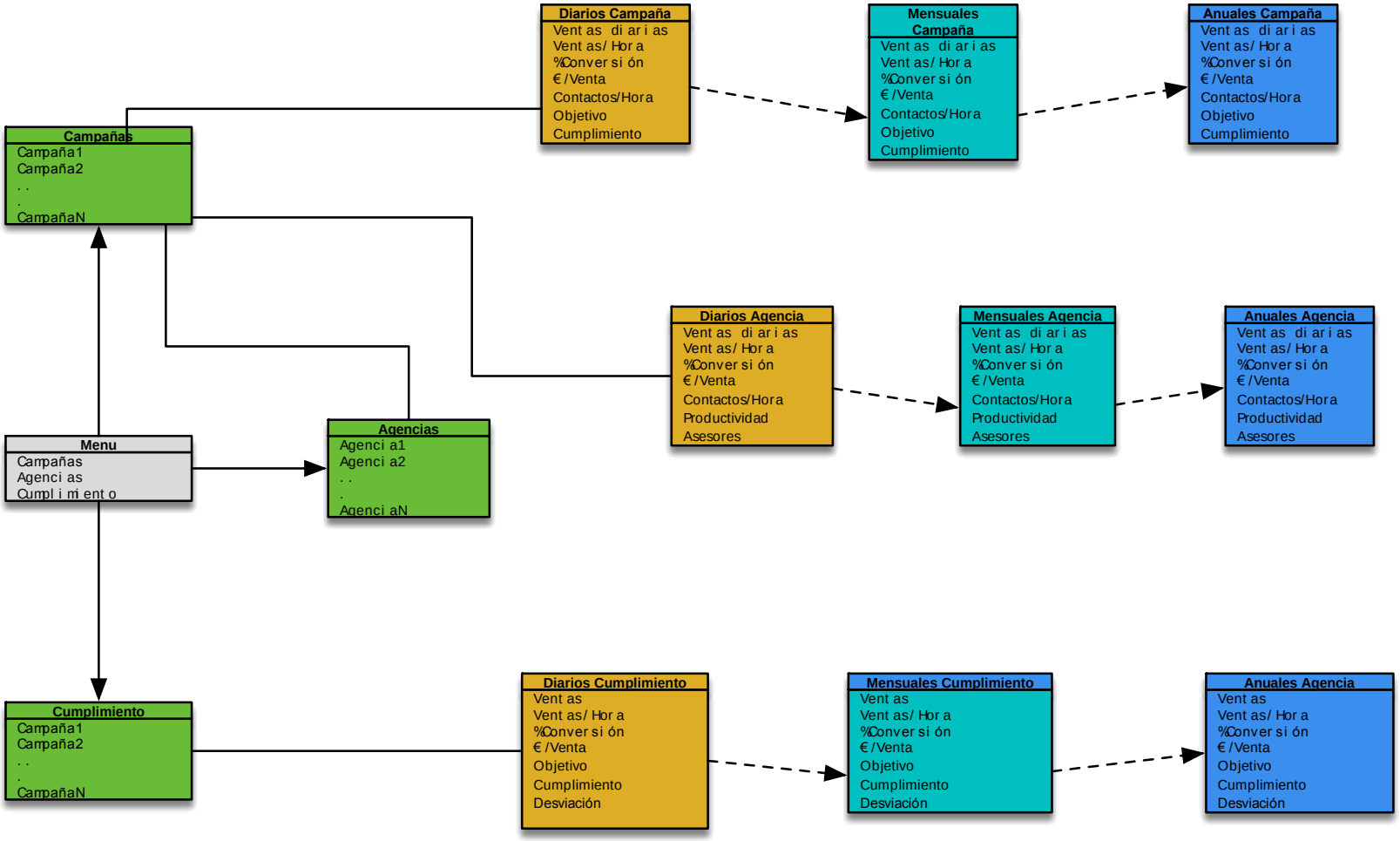
Diarios Campaña
Ventas diarias
Ventas/Hora
%Conversión
€/Venta
Contactos/Hora
Objetivo
Cumplimiento
Mensuales Campaña
Ventas diarias
Ventas/Hora
%Conversión
€/Venta
Contactos/Hora
Objetivo
Cumplimiento
Anuales Campaña
Ventas diarias
Ventas/Hora
%Conversión
€/Venta
Contactos/Hora
Objetivo
Cumplimiento
Campañas
Campaña1
Campaña2
. .
.
CampañaN
Diarios Agencia
Ventas diarias
Ventas/Hora
%Conversión
€/Venta
Contactos/Hora
Productividad
Asesores
Mensuales Agencia
Ventas diarias
Ventas/Hora
%Conversión
€/Venta
Contactos/Hora
Productividad
Asesores
Anuales Agencia
Ventas diarias
Ventas/Hora
%Conversión
€/Venta
Contactos/Hora
Productividad
Asesores
Menu
Campañas
Agencias
Cumplimiento
Agencias
Agencia1
Agencia2
. .
.
AgenciaN
Cumplimiento
Campaña1
Campaña2
. .
.
CampañaN
Diarios Cumplimiento
Ventas
Ventas/Hora
%Conversión
€/Venta
Objetivo
Cumplimiento
Desviación
Mensuales Cumplimiento
Ventas
Ventas/Hora
%Conversión
€/Venta
Objetivo
Cumplimiento
Desviación
Anuales Agencia
Ventas
Ventas/Hora
%Conversión
€/Venta
Objetivo
Cumplimiento
Desviación

Ilustración 11

3.3 Navegación

Definiendo el diseño de la aplicación cuenta con una primera vista “corporativa” en la que aparece el logotipo, desde el cual se puede acceder al contenido de la app, y la cual se muestra siempre que se salga de la aplicación y vuelva a ejecutarse.

Después contamos con un menú dividido en 3 opciones:

i. Campañas: listado de las diferentes campañas

ii. Agencias: listado de agencias y sus campañas asociadas

iii. Cumplimiento: listado de las diferentes campañas.

Ilustración 12

3.3.1 Campañas

Si seleccionamos Campañas en el menú de navegación veremos las diferentes campañas que existen.

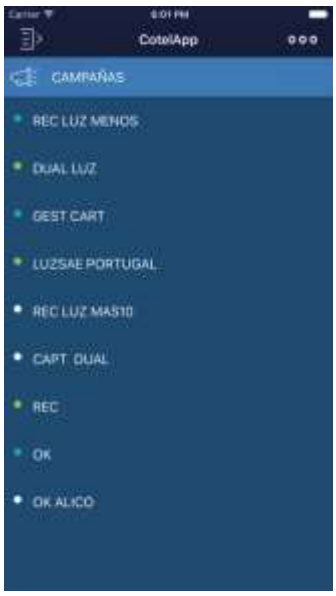

Ilustración 13

Una vez seleccionamos una campaña se puede ver de un primer vistazo una serie de parámetros tales como:

- ✓ Ventas diarias
- ✓ Ventas/Hora
- ✓ % Conversión
- ✓ € /Venta
- ✓ Contactos/Hora
- ✓ Objetivo
- ✓ Cumplimiento

Ilustración 14

3.3.2 Agencias

Si seleccionamos Agencias en el menú de navegación veremos las diferentes Agencias y todas las campañas asociadas a cada una.

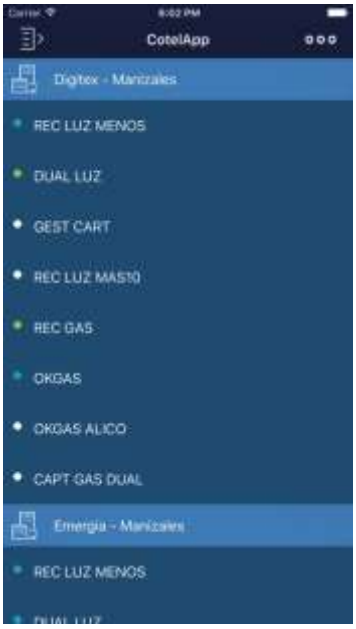

Ilustración 15

Al igual que en campañas se puede ver de un primer vistazo una serie de parámetros tales como:

- ✓ Ventas diarias
- ✓ Ventas/Hora
- ✓ % Conversión
- ✓ € /Venta
- ✓ Contactos/Hora
- ✓ Productividad
- ✓ Asesores

Ilustración 16

Se utiliza un punto de color para indicar a simple vista la clasificación de ese parámetro en función de su valor actual, es decir, el color nos indicará si cumple o no con los objetivos establecidos para ese campo.

3.3.3 Cumplimiento

Si seleccionamos Cumplimiento en el menú de navegación veremos los cumplimientos globales por cada campaña teniendo en cuenta todas las Agencias.

Ilustración 17

Esta vista varía respecto a Campañas y Agencias, informando una serie de campos relacionados con el cumplimiento:

- ✓ Ventas
- ✓ Ventas/Hora
- ✓ % Conversión
- ✓ € /Venta
- ✓ Objetivo
- ✓ Cumplimiento
- ✓ Desviación

Ilustración 18

En los 3 casos, deslizando la pantalla de izquierda a derecha, se puede pasar entre las vistas diaria, mensual y anual de esos datos.
Por último, se dispone de una vista informativa. En cualquier momento se puede volver al menú principal pulsando sobre el icono situado en la parte superior izquierda de la interfaz.

La carga de datos se ha realizado de manera paramétrica, es decir, que podemos incluir una campaña, relacionar más campañas a una agencia, incluir campañas…etc., con solo añadirla al fichero fuente XML. El parseo de datos trabaja con arrays que contendrán structs (parecido a objetos) o con arrays que contienen otros arrays de structs (matrices) lo que otorga a la aplicación una escalabilidad dinámica sin tener que modificar los fuentes de la aplicación.

Definición de los objetos de carga de datos en Swift, se han definido como ‘struct’, estructuras de datos definidas previamente.

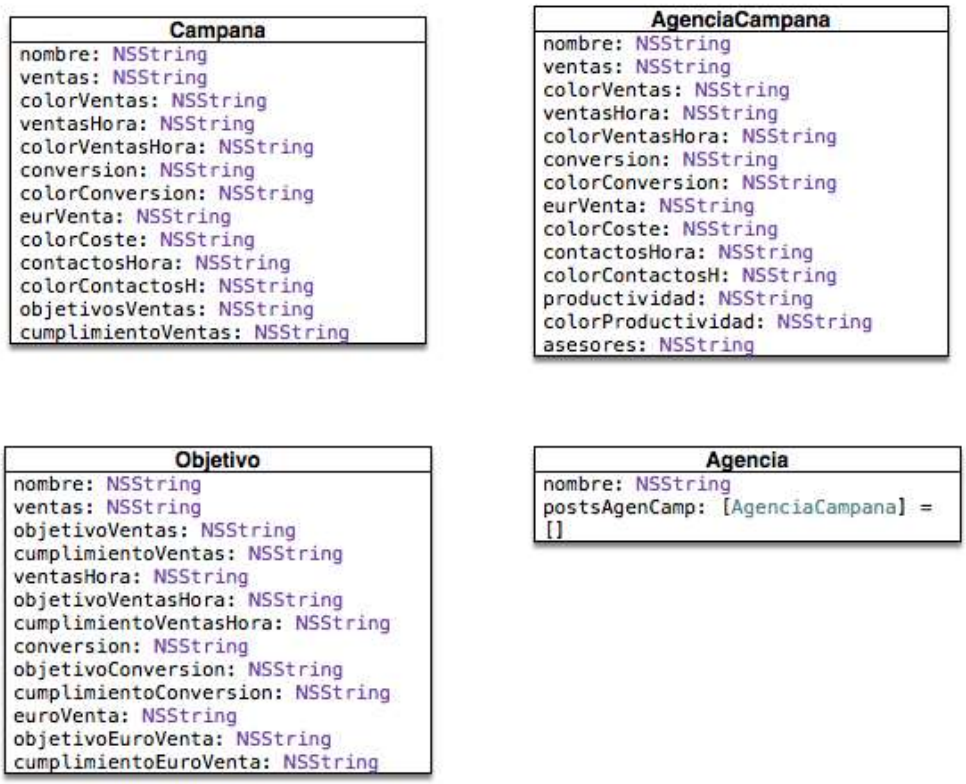

Ilustración 19

3.3.4 Acerca de la implementación

La totalidad de la aplicación se ha desarrollado utilizando el lenguaje de programación Swift, a excepción de la parte de obtención de datos, para la cual se ha utilizado XML.

La aplicación ha sido implementada para ejecutarse correctamente en la versión de iOS más reciente en la actualidad: iOS 9, presente en más del 80% de terminales iOS a nivel mundial.

4. Implementación

Se definen los principales objetos que controlan y configuran los principales rasgos de la aplicación, a continuación se detalla toda la estructura de cómo está implementado el proyecto.

4.1 AppDelegate.swift

En este archivo especificamos determinadas características que queremos estén presentes en el conjunto de la app. También es el lugar en el que podemos determinar ciertas acciones en momentos puntuales de la ejecución de la app como: entrada en reposo, vuelta de segundo plano…etc.
En este caso concreto, utilizamos el método “applicationDidBecomeActive” para conseguir mostrar el menú principal de la aplicación siempre que volvemos de segundo plano.
Dentro del método “didFinishLaunchingWithOptions” introducimos el código que configura el background degradado.

4.2 Visual Controllers (VC)

Estos ficheros determinan el comportamiento de las clases mediante las que configuramos la parte visual de la app. Todo componente tiene su parte en código relacionada con su parte visual.

4.2.1 MenuScreenViewController

Es la pantalla de menú principal, en la que seleccionamos el listado correspondiente que queremos visualizar: campañas, agencias o cumplimiento. Tanto “campanaView” como “agenciaView” y “cumplimientoView” son elementos que están definidos con esos nombres dentro del correspondiente Storyboard.

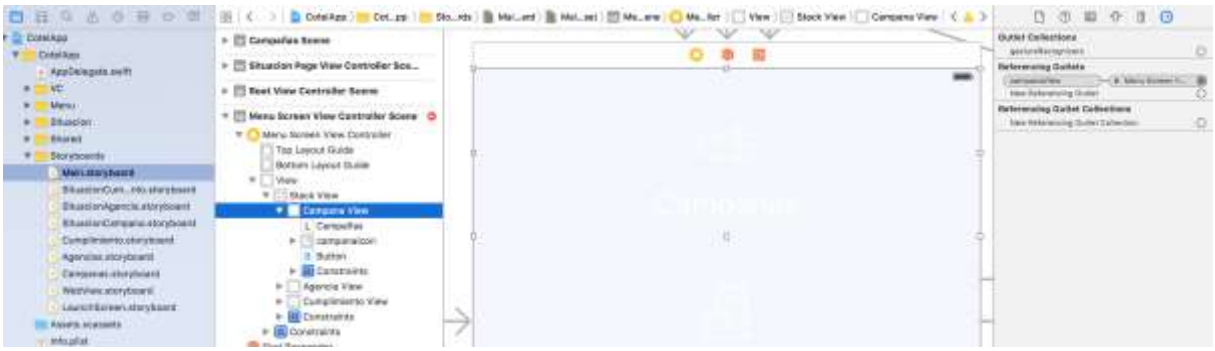

Ilustración 20

```
@IBAction func campanaButton(sender: AnyObject) {
campanaView.backgroundColor = UIColor.flatLimeColor()
}

@IBAction func agenciaButton(sender: AnyObject) {
agenciaView.backgroundColor = UIColor.flatLimeColor()
}

@IBAction func cumplimientoButton(sender: AnyObject) {
cumplimientoView.backgroundColor = UIColor.flatLimeColor()
}
```

4.3 Menu

4.3.1 MenuTableViewCell.swift

Cuando queremos utilizar tablas con un aspecto personalizado, que difiere del establecido por defecto en iOS, tenemos que crear en primer lugar una celda personalizada, creando una clase que herede de UITableViewCell, y determinando los elementos que queremos se muestren en las celdas de nuestra tabla. En este caso, el nombre del elemento y un círculo de color indicativo. Para el círculo, creamos un elemento de UIView, al que modificamos su propiedad cornerRadius.

Ilustración 21

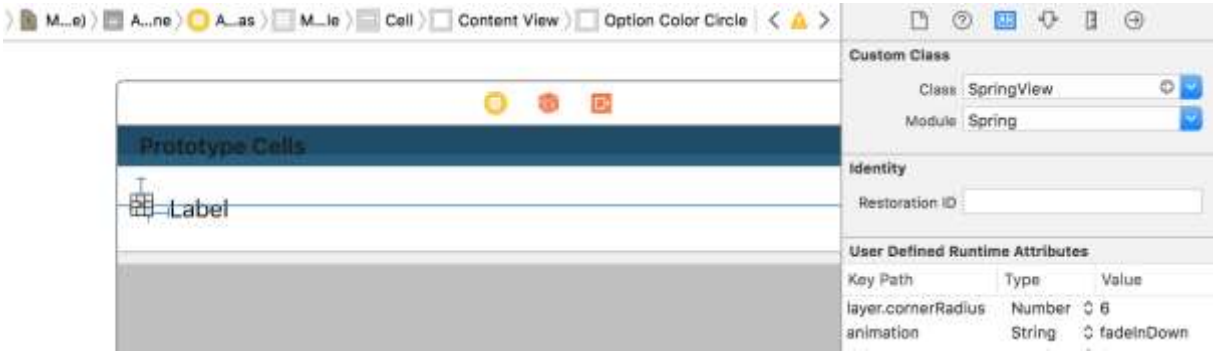

Ilustración 22

```
@IBOutlet var optionLabel: UILabel!
@IBOutlet weak var optionColorCircle: UIView!
```

4.4 Situación (ViewController y PageViewController)

Para la vista de interacción, en la que tenemos una paginación entre las vistas diaria, mensual y anual (cada una de tipo UIViewController), debemos implementar el elemento UIPageViewController, el cual controla esas 3 vistas.

Este es el método en el que configuramos los datos de cada una de las 3 vistas mediante el PageController.

4.5 Parsing con NSXMLPARSE

Para la lectura del fichero de datos XML se utilizará un parseador de Swift, NSXMLParser. Los datos con los que se alimenta la app se cargan desde un fichero XML actualizado reiteradamente en la nube Dropbox, para ello se define la url del fichero desde donde se realizará la lectura y posterior parseo. Una vez cargado en memoria, el parser recorre todos los elementos del XML hasta el final del documento. Para la recuperación de datos se ha elaborado un algoritmo de control de los elementos parseados y cargados sobre structs previamente definidos.

```
// Parseador

func beginParsing()
{
//   postsC = []
parser =
NSXMLParser(contentsOfURL:(NSURL(string:"https://dl.dropboxusercontent.co
m/u/12050436/appuoc.xml"))!)!
parser.delegate = self
parser.parse()

menuTable!.reloadData()
self.refreshTableData?.endRefreshing()
}
```

4.6 Refresco de Vistas

A continuación, en el método “viewDidLoad”, que es el primero que se ejecuta una vez se carga una vista determinada de la app, llamamos al método beginParsing(), para comenzar con el parseo de los datos.

Es en este método también, donde añadimos el código necesario para refrescar los datos de la tabla, realizando el gesto de deslizarla hacia abajo, lo que es conocido como “pullToRefresh”.

```
override func viewDidLoad() {
super.viewDidLoad()
self.beginParsing()

// Refrescar datos de la table principal, "deslizando" tabla hacia abajo
refreshTableData = UIRefreshControl()
refreshTableData.tintColor = UIColor.whiteColor()
refreshTableData.addTarget(self, action: "beginParsing", forControlEvents:
UIControlEvents.ValueChanged)
menuTable.addSubview(refreshTableData)

}
```

Lo que queda para mostrar los datos, es implementar los métodos necesarios de la clase UITableView, que son los siguientes:

numberOfSectionsInTableView

Indicamos número de secciones que va a tener nuestra tabla.

viewForHeaderInSection

Configuramos el título de una sección de la tabla.

heightForHeaderInSection

Determinar altura de las filas de la tabla.

numberOfRowsInSection

Obtenemos número de filas de nuestra tabla.

cellForRowAtIndexPath

Uno de los métodos más importantes, es aquí donde insertamos en cada fila de la tabla los datos obtenidos desde el servidor.

4.6.1 Paso de datos entre vistas

Un punto importante es el paso de información entre diferentes vistas, para ello tenemos que implementar el método “prepareForSegue”.

En primer lugar vemos que tenemos que dar en el Storyboard un nombre al Segue. Un segue es cada una de las flechas que unen las diferentes vistas en el Storyboard, y seleccionándolos, en el inspector de atributos podemos darle un nombre, de tal forma que podamos acceder a él mediante código.

```
if segue.identifier == "showAgenciasDetail"
```

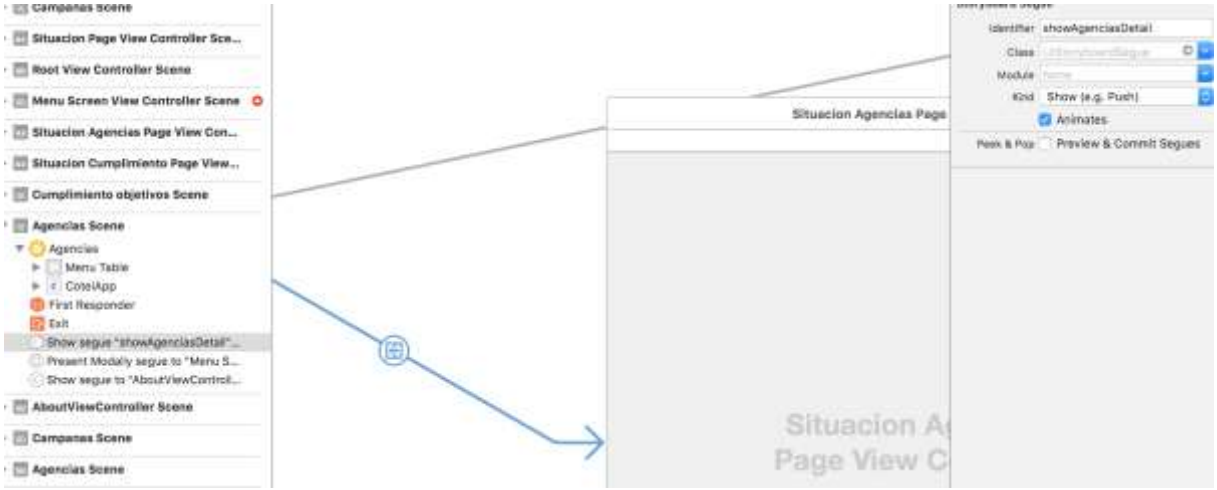

Ilustración 23

Después, ya podemos acceder a los elementos de esa vista a la que queremos llegar, utilizando el objeto destinationController y asignándole los valores correspondientes de la vista anterior.

4.7 Compartido (Shared)

Las clases, Agencia, Campaña, Objetivo y UIColorExtension, son utilizadas a lo largo de la app en distintas partes de la misma. Son clases que definen las propiedades que tendrán los objetos de esas clases.

En los tres primeros casos, indicamos las diferentes propiedades o datos que necesitamos mostrar de cada Campaña o Agencia, los cuales obtenemos desde el XML, mientras que UIColorExtension es una extensión de la clase UIColor (la cual define los colores en iOS), para poder indicar los valores de los

colores utilizados con valor hexadecimal (más común) y la extensión de encarga de “traducir” estos valores a tipo UIColor.

4.8 Storyboards

En los storyboards se define la interfaz visual de la aplicación, la parte correspondiente a la Vista dentro del patrón MVC utilizado a la hora de desarrollar apps para iOS.
Mediante una sencilla interfaz de arrestar y soltar, colocamos en la interfaz los diferentes elementos disponibles en el sistema: textos, imágenes, mapas, gestos…etc.

Desde aquí podemos crear las relaciones entre diferentes vistas y configurar algunas de las propiedades de los elementos de interfaz, así como darles un nombre mediante el cual poder acceder a los elementos desde el código de nuestra app, para configurar otras propiedades o para darles valores obtenidos o calculados desde otras fuentes.

Se puede tener toda la interfaz de una app dentro de un mismo storyboard, pero a medida que la cantidad de vistas va creciendo, como es el caso de esta App, es más conveniente dividir la interfaz en varios storyboards y acceder a ellos mediante referencias, para así conseguir tener todo más organizado y que el trabajo con los Storyboard no sea tan pesado, ya que a medida que crecen en tamaño, también consumen más recursos del sistema.

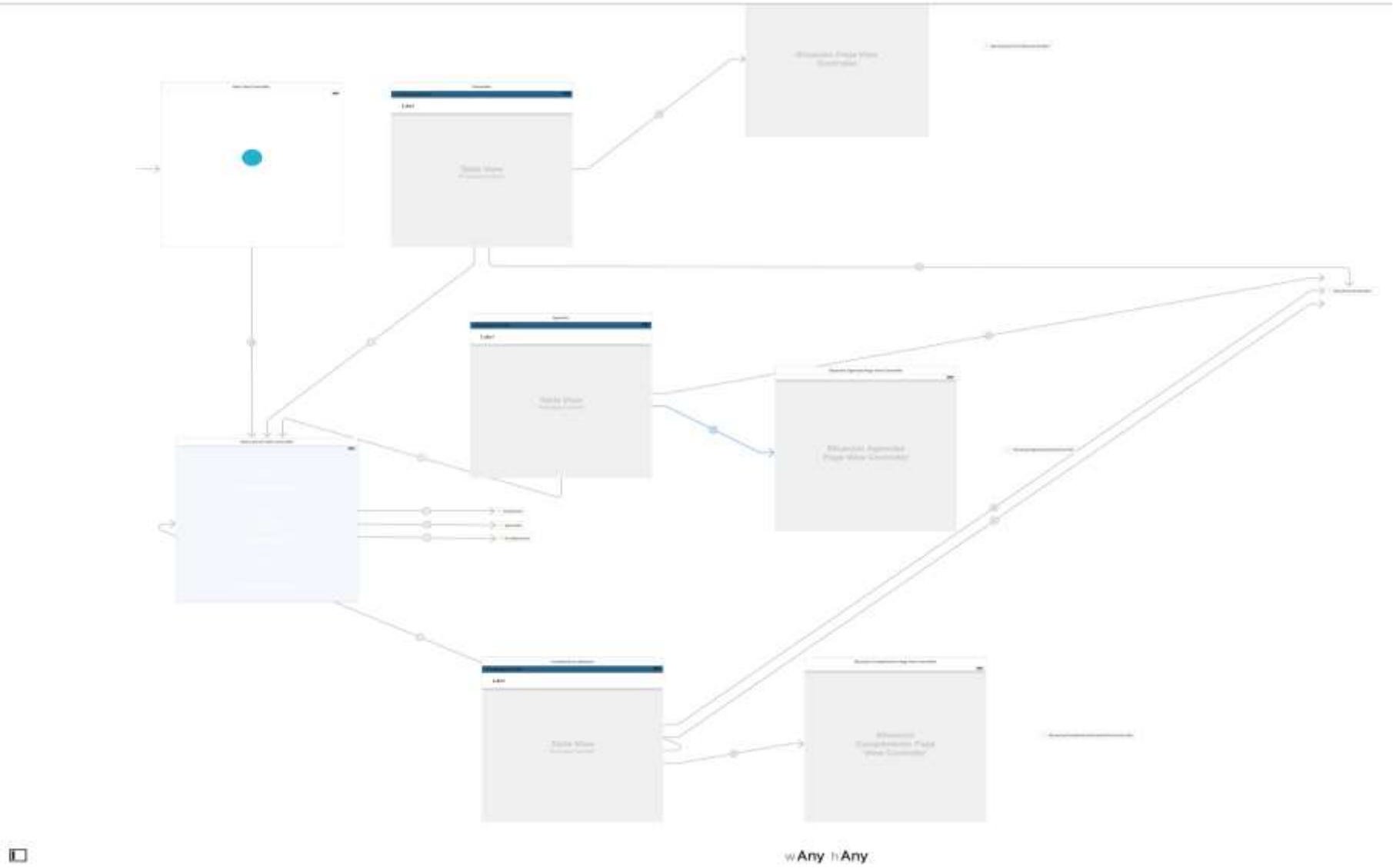

Ilustración 24

4.8.1 Auto Layout

Mediante Autolayout, un sistema de operaciones matemáticas creado por Apple para facilitar la adaptación de nuestras interfaces a diferentes tamaños de pantalla, indicamos a cada elemento, su tamaño, disposición en pantalla y distancia con los márgenes y otros elementos.

Ilustración 25

4.9 Assets

En esta carpeta incluimos todos los assets de la app. Esto incluye el icono de la aplicación, así como todos los iconos, logotipos o imágenes que vayamos a incluir. Hay que tener en cuenta que hay que proporcionar a Xcode las diferentes resoluciones necesarias para el conjunto de dispositivos iOS. Actualmente las resoluciones son @1x, @2x, @3x.

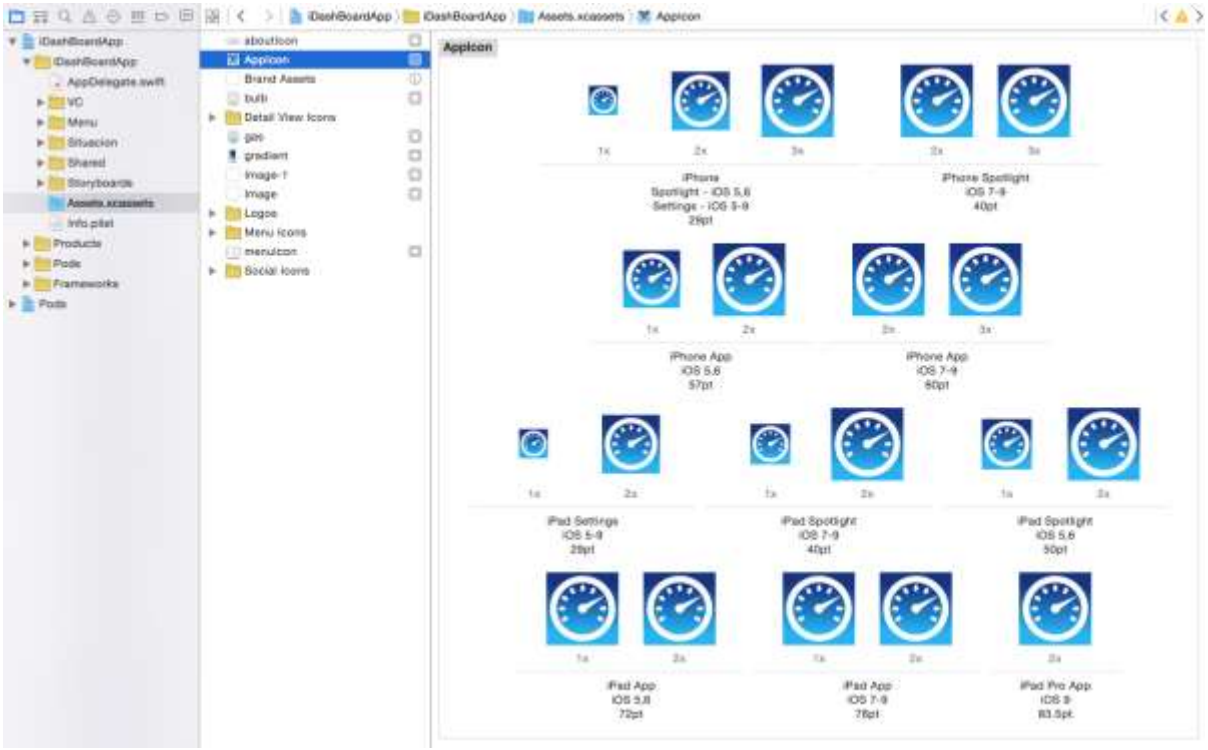

Ilustración 26

Estos assets los genera y proporciona el diseñador o equipo de diseño y se exportan en Xcode arrastrando y soltando.

4.10 Info.plist

Este fichero sirve para configurar partes importantes de la app, como orientaciones soportadas, nombre del paquete de la app, estilo de la barra de estado…etc.

Key	Type	Value
▼ Information Property List	Dictionary	(18 items)
Localization native development re...	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.2
Bundle creator OS Type code	String	????
Bundle version	String	6
Application requires iPhone enviro...	Boolean	YES
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
UIRequiresFullScreen	Boolean	YES
Status bar style	String	UIStatusBarStyleLightContent
▼ Supported interface orientations	Array	(1 item)
Item 0	String	Portrait (bottom home button)
▼ Supported interface orientations (i...	Array	(3 items)
Item 0	String	Portrait (bottom home button)
Item 1	String	Landscape (left home button)
Item 2	String	Landscape (right home button)
View controller-based status bar a...	Boolean	NO

Ilustración 27

4.11 Gradientes

Los colores con estilo degradado no están incluidos por defecto dentro del SDK de iOS, por lo que hay que crearlos insertando una subcapa de tipo CAGradientLayer de la siguiente forma:

```
let gradient:CAGradientLayer = CAGradientLayer()
gradient.frame = view.bounds
gradient.colors = [UIColor(red:0.02, green:0.08, blue:0.23,
alpha:1).CGColor,UIColor(red:0.38, green:0.82, blue:0.95,
alpha:1).CGColor]
self.view.layer.insertSublayer(gradient, atIndex: 0)
```

4.12 Pods

Para añadir determinadas características de la app, como las animaciones de los diferentes elementos de la interfaz, es necesario utilizar librerías de terceros. Una de las formas más populares es utilizar el instalador de paquetes Cocoapods. Para ello hay que añadirlo al proyecto, instalándolo desde la terminal siguiendo las instrucciones de la página web oficial:

https://cocoapods.org

Una vez instalado, hay que editar el fichero “Podfile”, para añadirle los pods que estemos utilizando en cada caso. Por ejemplo, para utilizar las animaciones hemos utilizado “Spring” (https://github.com/MengTo/Spring) y en la propia web ya nos indica el texto a introducir dentro del Podfile.

```
platform :ios, '8.0'
use_frameworks!

target 'iDashBoardApp' do
pod 'ChameleonFramework/Swift'
pod 'Spring', :git => 'https://github.com/MengTo/Spring.git', :branch =>
'swift2'
end
```

4.13 Animaciones

Para darle el comportamiento de animación a logotipos, iconos, círculos de color… hay que cambiar su clase dentro del inspector de atributos en el Storyboard correspondiente.

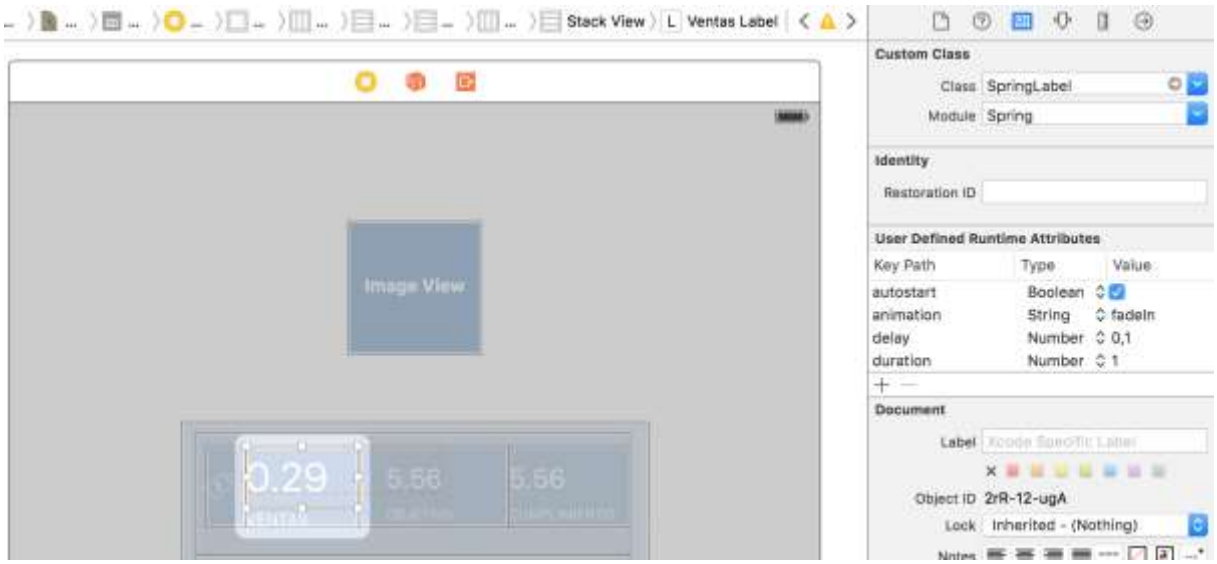

Ilustración 28

En este caso, para un texto, sustituimos el tipo UILabel por defecto, por SpringLabel, que es el tipo que define un texto que se puede animar, importado desde el pod Spring.

Después, configuramos a nuestro gusto las diferentes variables que conforman la animación.

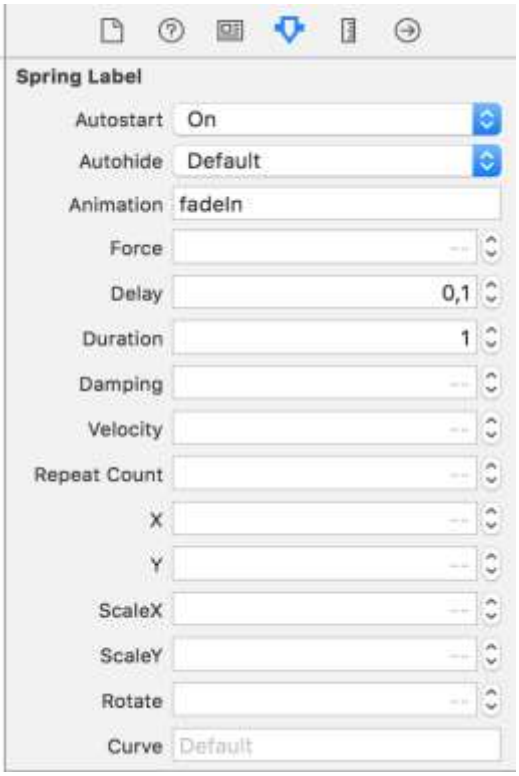

Ilustración 29

5. Conclusiones

El propósito de la aplicación es proporcionar al usuario toda la información que necesita de una forma rápida, sencilla e intuitiva. Una vez concluido el desarrollo de la aplicación podemos afirmar que los objetivos previstos inicialmente se han cumplido satisfactoriamente.

El desarrollo de este proyecto me ha permitido entender cuales son los procesos que hay que cumplimentar para el desarrollo de aplicaciones para dispositivos móviles.

Durante el desarrollo del proyecto me he visto forzado a aprender un lenguaje relativamente distinto al resto de los lenguajes que conocía, especialmente en la sintaxis. Swift resulta un lenguaje fácil de comprender y tiene una curva de aprendizaje corta.

El seguimiento de la planificación ha sido dificultoso pues compaginarlo con el trabajo y unas circunstancias personales especiales han hecho que no pueda haber llevado al 100% la planificación. La metodología ha sido correcta, pero como ya he comentado anteriormente he tenido algunas complicaciones.

La aplicación tiene muchas posibilidades de ampliación por lo que es escalable y habría que trabajar más en una interfaz de configuración para cada usuario, de tal manera que pueda personalizarla a sus necesidades.

6. Glosario

DashBoard: Cuadro de instrumentos que ayudan a un entendimiento visual rápido de los datos.

Xml: Lenguaje de marcas desarrollado por World Wide Web.

Xcode: Entorno de desarrollo integrado.

Swift: Lenguaje de programación multiparadigma creado por Apple enfocado en el desarrollo de aplicaciones para iOS y Mac OS X.

Objetive C: lenguaje de programación orientado a objetos creado como un superconjunto de C

Parser: Transforma una entrada de texto en una estructura de datos.

Java: lenguaje de programación de propósito general, concurrente, orientado a objetos.

Nube: En realidad la nube es una metáfora empleada para hacer referencia a servicios que se utilizan a través de Internet.

Array: Es un medio de guardar un conjunto de objetos, normalmente en forma de pila.

Struct: Estructura de datos creada en Swift.

Batch: La ejecución de una serie de programas en un computador sin la interacción humana.

Framework: Conjunto estandarizado de conceptos, prácticas y criterios.

API: Es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Dropbox: es un servicio de alojamiento de archivos multiplataforma en la nube.

Apache cordoba: es un marco de desarrollo móvil de código abierto.

Ad Hoc: solución específicamente elaborada para un problema o fin preciso

Sketch: es un programa de diseño pensado para diseñadores, que tiene algunas herramientas de diseño vectorial.

Debugger/Depurador: es un programa usado para probar y depurar (eliminar) los errores de otros programas.

Compilador: es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación.

7. Bibliografía

Nuevo Framework de Apple, página que explica e introduce al programador al mundo Swift mediante pequeños cursos y ejercicios. Tipo de datos, Iteraciones, bucles…
Web: http://www.aprendeswift.com

Tutoriales con ejemplos de Swift y comunidad de fuentes y foro.
Web: http://www.theappguruz.com

Interesantes tutoriales en español que nos familiariza con el IDE de Apple, XCode, y nos propone prácticas interesantes.
Web: http://www.manzanamagica.com/desarrollo/

Libro de lenguaje de programación Swift
Web:https://itunes.apple.com/book/swift-programming-language/id881256329?mt=11

Libro en español para Swift

Web:https://applecoding.com/libros/aprendiendo-swift-venta-ibookstore-ibook-libro

---

Cursos de Swift

Web: https://applecoding.com/category/cursos

Spring Animations , Liberia de efectos de animación

Web: https://designcode.io

Web: https://github.com

SpringApp Test, fuente de un ejemplo de animación Spring que ha servido de gran ayuda.

Web: https://github.com/MengTo/Spring

Apple Documentation.

Web: https://developer.apple.com/library/ios/navigation/

Stackoverflow, foro social que muestra como resolver bastantes problemas a nivel técnico.

Web: http://stackoverflow.com

8. Anexos

8.1 Anexo I

Fichero de pruebas utilizado en la nube Dropbox , appUoc.xml

https://dl.dropboxusercontent.com/u/12050436/appUoc.xml

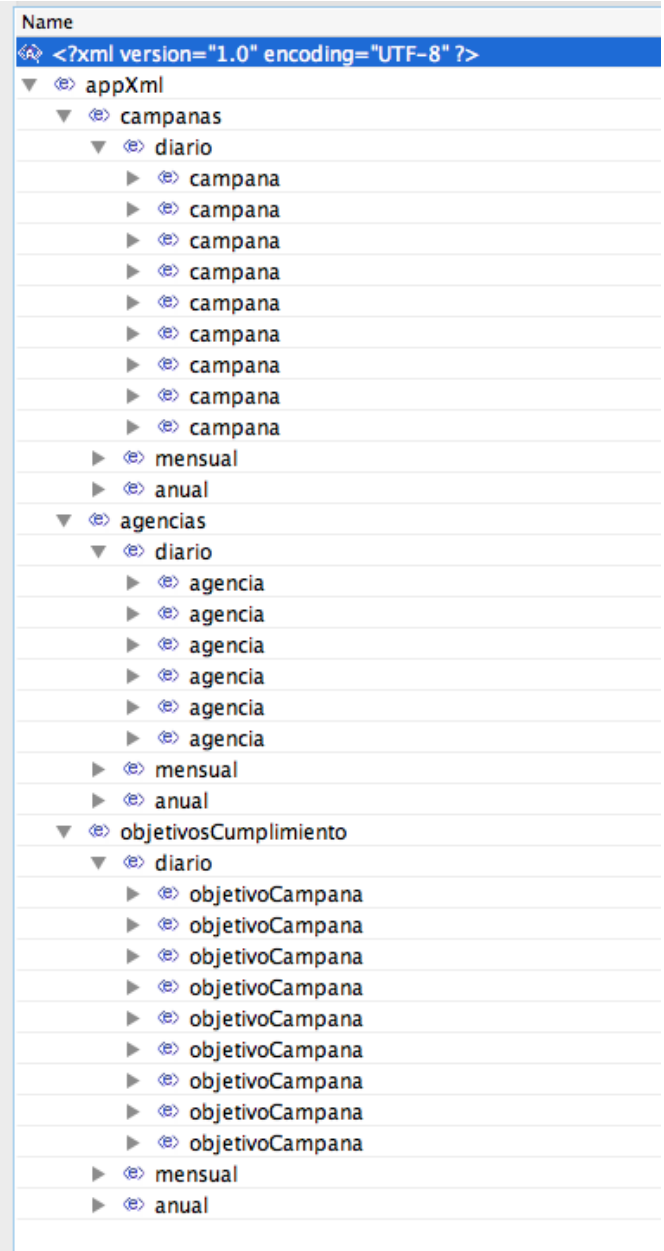

Ilustración 30