UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"DESARROLLO DE UNA APP IOS EN LENGUAJE SWIFT CON CONEXIÓN A BASES DE DATOS EXTERNAS Y SCRAPPING"

TRABAJO FIN DE GRADO

Octubre -2015

AUTOR: Alberto Aznar de los Ríos

DIRECTOR/ES: César Fernández Peris

RESUMEN

El mercado de las aplicaciones móviles presenta en la actualidad un grandísimo potencial. Las empresas hacen uso de las aplicaciones para publicitar sus servicios y los usuarios obtienen información, juegan y se entretienen.

El desarrollo de aplicaciones móviles se perfila como una profesión de éxito y el futuro en este campo es muy extenso. La salida de nuevos lenguajes de desarrollo implica una serie de mejoras en el rendimiento que es interesante analizar y el uso de nuevas técnicas como web scrapping y conexión con bases de datos externas para adquirir contenidos es imprescindible para un buen desarrollo.

El objetivo de este TFG es la creación de una aplicación que resulte útil para el usuario y hacer uso de todas las fases de desarrollo, desde el prototipado hasta su publicación y marketing. Con ello, se analizarán e investigarán las ventajas del nuevo lenguaje Swift, por qué desarrollar en plataforma iOS antes que en Android y el uso de todas estas técnicas de obtención de datos.

La aplicación desarrollada para todo este análisis basa su contenido en la guía de televisión. El motivo es el aumento del número de canales de televisión, tanto de pago como TDT. Esto hace que los usuarios demanden una aplicación que filtre todo el contenido en base a los gustos del usuario y genere avisos o notificaciones de los contenidos que deseen ver.

PALABRAS CLAVE

Swift, iOS, Web Scrapping, APIs, Aplicación, Guía de televisión, Base de datos

INDICE DE CONTENIDOS

1. INTRODUCCIÓN

1.1 ANALISIS DEL ENTORNO Y MOTIVACIÓN

El mercado de las aplicaciones móviles presenta un grandísimo potencial y los anunciantes comienzan a ser conscientes de ello. La evolución de las aplicaciones para móviles se dio rápidamente gracias a las innovaciones en la tecnología WAP y transmisión de data (EDGE) seguido de un fuerte desarrollo del rendimiento y capacidad de memoria de los dispositivos móviles llegándose a llamar smartphones.

La incorporación al mercado de los smartphones y la creación de tablets, revolucionó el mundo de las aplicaciones móviles. Tanto Google con su sistema operativo Android, como Apple con iOS, marcaron distancia con sus competidores siendo líderes en el mercado de venta de aplicaciones para móviles. El surgimiento de la AppStore (para iOS) y Google Play (para Android) terminaron de impulsar el éxito de las apps y un cambio en la manera en que se distribuye y se comercializa todo este software.

Según un informe realizado por la prestigiosa plataforma impulsora de aplicaciones a través de internet “The App Date”, actualmente España es el primer país de Europa donde más ha irrumpido el mercado de smartphones. Un 66% de usuarios de móviles usan este tipo de dispositivos. En total son aproximadamente 23 millones de usuarios activos en España que se descargan una media de 3,8 millones de aplicaciones al día. Este es uno de los motivos más importantes para entrar en el mercado del desarrollo de aplicaciones móviles.

Además, las empresas tienden de manera natural a maximizar sus recursos y usar nuevas estrategias que le ayuden a impulsar su negocio. Ya no les basta con tener una página web para darse a conocer en internet, en la actualidad es necesario tener además una aplicación móvil de la empresa para difundir sus

servicios. Las aplicaciones para móviles pueden añadir una gran ventaja competitiva frente a otras empresas del mismo sector, por ello resulta interesante para mejorar la imagen de marca, para sincronizar y difundir contenidos de la aplicación a través de redes sociales y en consecuencia ampliar el número de clientes potenciales, ya que, ofrece un buen canal de comunicación entre el negocio y el cliente.

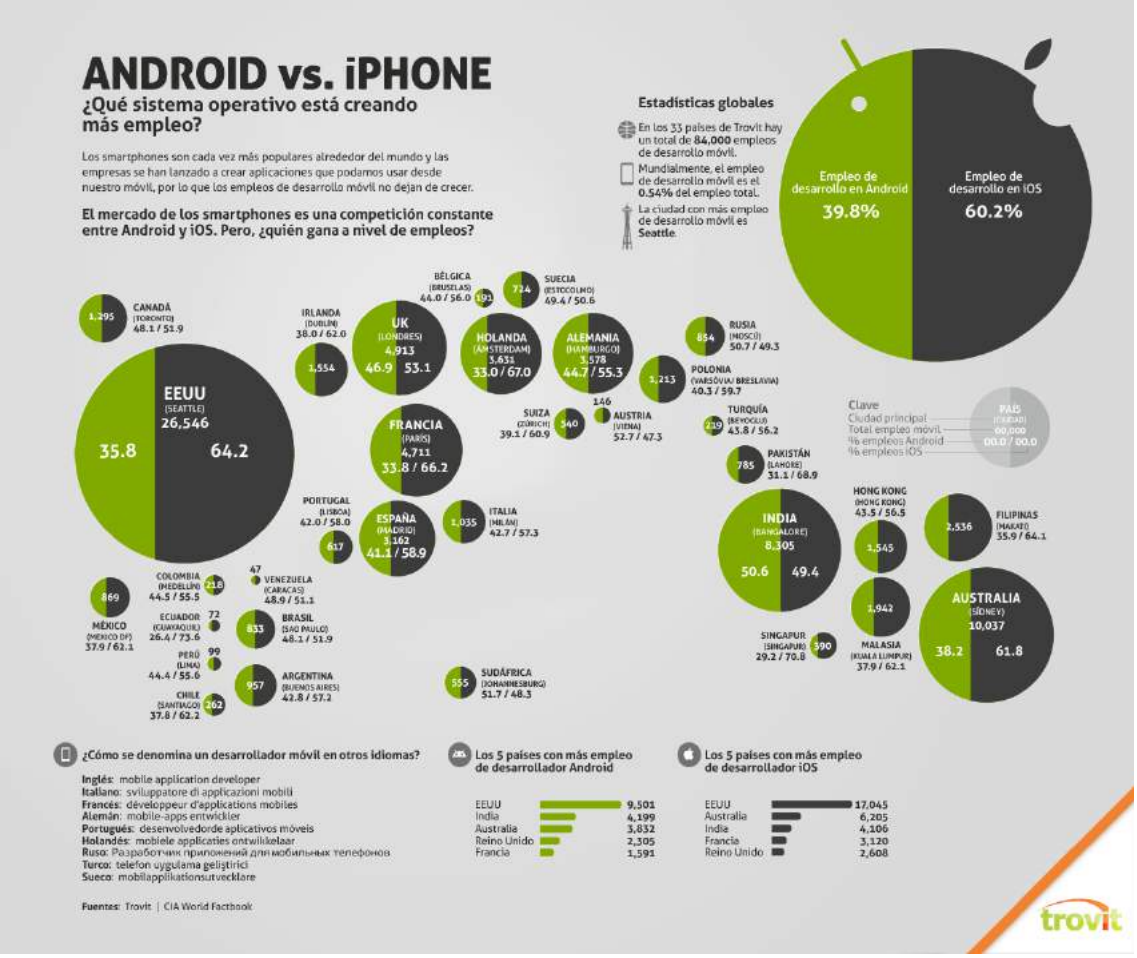
1.2 ¿PORQUÉ UNA APLICACIÓN MÓVIL COMO TFG?

Las aplicaciones móviles pueden incorporar tecnologías que no puede incorporar una página web como geo localización o realidad aumentada, tienen mayor capacidad, rendimiento y mejoran la navegación del usuario.

Además del enfoque a nivel empresarial, las ventajas a nivel usuario son infinitas dependiendo del tipo de aplicación ya que nos ayudan a resolver problemas de la vida diaria, proporcionan información interesante, nos pone en contacto con otros usuarios en tiempo real y todo se puede compartir a través de la red.

Todas estas ventajas en el uso de aplicaciones móviles para usuarios y empresas hacen que el mercado vaya en aumento.

El desarrollo de aplicaciones se perfila como una profesión de éxito. Según un informe proporcionado por Technet en 2012, en Estados Unidos se han generado medio millón de empleos desde 2007, y en el mismo periodo, se duplicaron las empresas dedicadas a esta actividad en España según TheAppDate. Por ello considero que realizar una aplicación móvil como TFG me prepara para un futuro profesional en un sector en auge.



Además, existe un motivación personal que hace superarme a mi mismo y aumentar mis conocimientos en el desarrollo de aplicaciones para dispositivos móviles, aprender nuevos lenguajes de programación e incrementar mi experiencia en la resolución de problemas.

1.3 PLATAFORMA ELEGIDA

Antes de comenzar este TFG es necesario analizar las diferentes alternativas de desarrollo y escoger la plataforma adecuada para programar la aplicación teniendo en cuenta las ventajas e inconvenientes de los diferentes lenguajes de programación, la posibilidad de comercialización y si la vamos a desarrollar de forma nativa, vía web o híbrida.

Por ello, para escoger una plataforma se revisarán estos factores y así evitaremos cambiar de idea una vez iniciado el desarrollo, ya que supondría un retraso importante en los plazos de finalización del trabajo.

1.3.1 TIPO DE APLICACIÓN

Dependiendo del tipo de desarrollo que queramos realizar podemos diferenciar varios métodos. Podemos realizar la aplicación programando en el lenguaje del propio sistema operativo, también podemos desarrollarla a través de la web o una opción combinada entre estas dos. A continuación vemos los detalles de cada una de estas opciones.

1.3.1.1 APLICACIÓN NATIVA

Son aplicaciones desarrolladas en el lenguaje del sistema operativo o plataforma que estamos utilizando. De forma que, si queremos desarrollar una aplicación para Android, esta no estará disponible para iOS, Windows Mobile y resto de plataformas.

Para que esté disponible en el resto de plataformas deberemos volver a desarrollar la misma aplicación como versiones distintas, en el lenguaje determinado para cada plataforma.

Este trabajo lleva más tiempo, con lo cual el coste de desarrollo es mucho mayor y conlleva por lo tanto un mayor mantenimiento por el número de versiones realizadas, pero a la larga, es más estable ya que se utilizan los propios recursos del sistema operativo como la CPU, memoria, consumo de batería... Además, también se pueden utilizar todas las funcionalidades del teléfono como la cámara, GPS, contactos, acelerómetro…

Otra ventaja importante es que las aplicaciones desarrolladas mediante este método pueden guardar o cachear información en el dispositivo mediante el almacenamiento local, aportando seguridad y posibilidad de acceso sin disponer de conexión a internet.

Para la descarga de este tipo de aplicaciones es necesario pasar por la tienda específica de cada dispositivo. Con lo cual, en ciertas plataformas, pasaremos por un filtro de revisión y esto conllevará un tiempo para que acepten y se publiquen actualizaciones.

Ésta será la mejor opción a escoger entre el resto por sus características ya que en principio desarrollaremos la aplicación para un solo sistema operativo, con lo que necesitaremos realizar una única versión.

1.3.1.2 APLICACIÓN WEB

En este caso la aplicación se desarrolla como si fuera una página web a la que accedemos a través del móvil mediante una conexión a internet. Esta página web deberá estar optimizada para diferentes tamaños para los distintos dispositivos del mercado.

Realmente no es una aplicación móvil, es más una aplicación web que puede ser vista desde un móvil. Esto conlleva una serie de ventajas y en algunos casos puede ser una buena opción en función de los requisitos que se necesiten. Éstas están desarrolladas en HTML, CSS y JAVASCRIPT.

Requieren una conexión a internet y aunque pueden funcionar de forma local cacheando la información en el móvil, están pensadas para funcionar con internet ya que la información se actualiza a menudo. Ésta puede ser una gran desventaja.

A cambio, podemos tener una aplicación totalmente actualizada en el momento que queramos, sin revisiones de ningún tipo ya que la gestionamos a través de la web.

En este tipo de desarrollo no podremos usar de forma efectiva el rendimiento del teléfono y todas sus funcionalidades que he detallado anteriormente.

1.3.1.3 APLICACIÓN HÍBRIDA

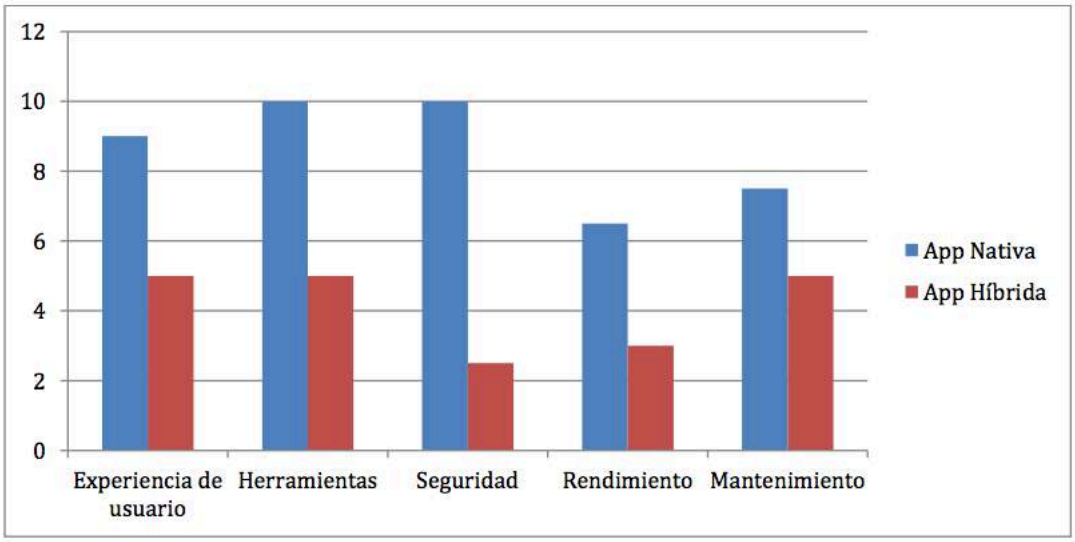
En una aplicación híbrida, se mezclan las características que nos proporciona una aplicación nativa y una aplicación web móvil. Este tipo de aplicaciones usa la tecnología web para el desarrollo de la interfaz y la tecnología móvil para el uso de todas las funcionalidades del sistema operativo.

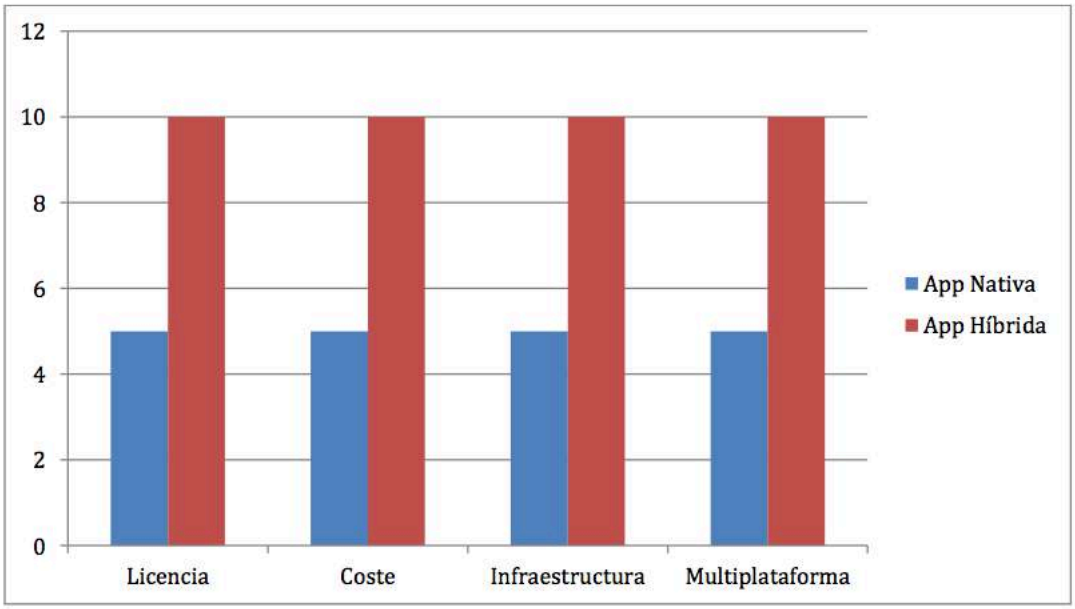
Podemos destacar grandes ejemplos que usan este sistema como método de desarrollo como Facebook, Instagram o LinkedIn. Como podemos observar, este tipo de aplicaciones se siguen basando mucho en la conexión a internet, ya que se sigue requiriendo conectarse a la aplicación web para obtener información.

Es una opción interesante para crear una aplicación que soporte varias plataformas, usando las características propias del sistema operativo y mejorando el rendimiento de la aplicación, ya que estamos usando los frameworks que dispone el sistema operativo.

Aún con todo esto, no podemos alcanzar el rendimiento máximo que consigue una aplicación nativa ya que seguimos dependiendo de una aplicación web para funcionar.

A continuación se muestra un gráfico comparativo de todas estas opciones teniendo en cuenta los principales factores.





1.3.2 ELECCIÓN DEL SISTEMA OPERATIVO

En la elección del sistema operativo es necesario tener varios factores en cuenta así como hacer un estudio de mercado para después ver las ventajas e inconvenientes del desarrollo en cada uno de los sistemas operativos.

Para este estudio he decidido omitir los sistemas operativos para Symbian, Palm y Blackberry, ya que frente a sus competidores, tanto su uso como la cuota de mercado, es realmente inferior en la actualidad.

1.3.2.1 ESTUDIO DE MERCADO

En la actualidad, abundan multitud de informes estadísticos de ventas y preferencias de los usuarios en el uso de unos dispositivos u otros de diferentes sistemas operativos y fabricantes del mercado móvil global.

Un informe publicado por Strategy Analytics comenta la distribución de los sistemas operativos preferidos por los usuarios.

Este informe realizado en 2014 revela una posición bastante favorecedora para Google con su sistema Android. Según el estudio, el sistema Android ocupa más de tres cuartos del mercado de movilidad mundial con un porcentaje del 85%, dejando atrás a una diferencia realmente importante a sus principales competidores, Apple, Microsoft y BlackBerry.

Los analistas indican, según estos resultados, un crecimiento del 27% de las ventas mundiales de dispositivos móviles en el segundo trimestre de 2014 con una subida de 295.2 millones de unidades vendidas a 233 millones en el mismo periodo de 2013. Aún así es dificil superar el crecimiento de un 49% del sector del año anterior. Los analistas destacan también en este informe las variaciones en función de la región geográfica: África y Asia se caracterizan por una explosión en la demanda, mientras que EEUU y Europa se encuentran en un proceso de maduración

Global Smartphone Operating System Shipments (Millions of Units)	Q2 '13	Q2 '14
Android	186.8	249.6
Apple iOS	31.2	35.2
Microsoft	8.9	8.0
BlackBerry	5.7	1.9
Others	0.5	0.5
Total	233.0	295.2

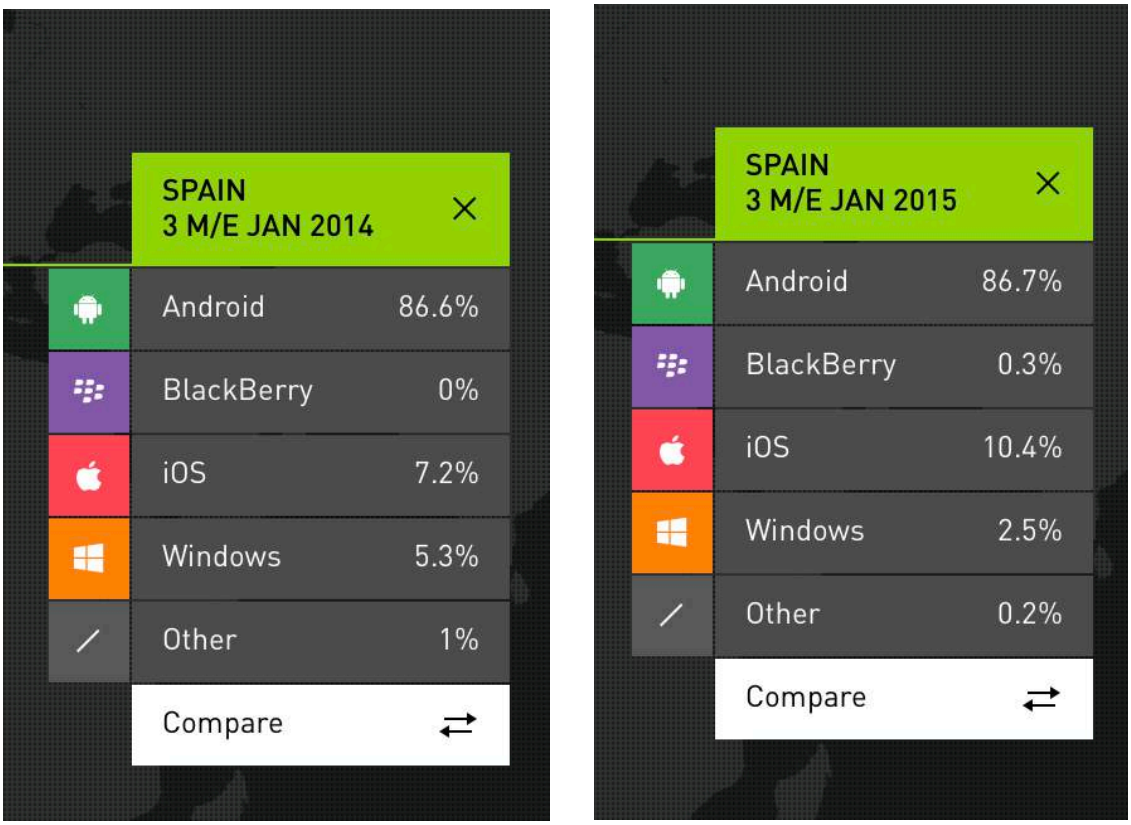
Global Smartphone Operating System Marketshare %	Q2 '13	Q2 '14
Android	80.2%	84.6%
Apple iOS	13.4%	11.9%
Microsoft	3.8%	2.7%
BlackBerry	2.4%	0.6%
Others	0.2%	0.2%
Total	100.0%	100.0%

Total Growth Year-over-Year %	48.9%	26.7%

Source: Strategy Analytics

En segunda mejor posición queda Apple aunque también ve un pequeño retroceso en la cuota de mercado debido a la presencia de smartphones low cost en el mercado. Apple, a diferencia de Google, no pretende competir en precios con el resto de sus competidores, a cambio dedica sus fuerzas a desarrollar nuevas funciones, tecnologías para diferenciarse del sector y penetrar en otros mercados, como los relojes inteligentes (iWatch) o más a largo plazo, en el mundo del automóvil.

Si centramos en la situación de venta de dispositivos en España, podemos consultar estudios realizados por la consultora Kantar Worldpanel (http://www.kantarworldpanel.com/global/smartphone-os-market-share/) donde muestran un panel estadístico para diferentes países y la cuota de mercado de cada país.



Si comparamos Enero de 2014 y Enero de 2015 en España, podemos observar que las ventas de dispositivos Apple han aumentado ligeramente frente a sus otros competidores que se quedan más o menos en la misma posición.

El sistema operativo Android pasa de un 86.6% de cuota de mercado frente a un 86.7% en Enero de este año. En cambio, iOS pasa de 7.2% a un 10.4% una mejora realmente importante en la venta de smarphones gracias en una parte a la entrada en el mercado de phablets (el camino medio entre un smartphone y una Tablet, o smartphone de gran tamaño) con su iPhone 6 y iPhone 6 Plus.

A la vez que estos dos competidores se reparten el mercado, otros competidores como Windows caen considerablemente de un 5.3% a un 2.5%, poco más de la mitad de sus ventas comparando con el mismo mes el año anterior.

Otro informe realizado por la consultora Kantar Worldpanel detalla como la reciente salida al mercado del iPhone 6 ha supuesto una revolución en los meses entre Diciembre 2014 y Febrero 2015, siendo el móvil más vendido en España. En total, la venta de smartphones representaron el 82.5% de todos los teléfonos móviles vendidos en España frente al 73% del mismo periodo de 2014.

En cuanto a otros países, desde la salida del iPhone 6 y iPhone 6 Plus ha habido un gran entusiasmo por los productos Apple en China. De esta forma, iOS se hace con el 27.6% de la cuota de mercado, 9.7 puntos más que el mismo periodo del año anterior.

En Estados Unidos, la cuota de mercado para Apple se ve reducida pero manteniendo la posición frente a sus competidores. Exactamente de un 39.3% a un 38.8%, aun así, el iPhone 6 sigue siendo el teléfono más vendido en este país por diferentes motivos como el tamaño de la pantalla (45% de los encuestados), el LTE (44%) y la fiabilidad y durabilidad del dispositivo (43%).

En Europa, la cuota de mercado para Android se ha visto reducida en este periodo de Diciembre 2014 a Febrero 2015 en 2.9 puntos porcentuales quedándose en un 67.6% del total, mientras que iOS ha incrementado en la misma medida.

1.3.2.2 WINDOWS PHONE

Windows Phone es el nuevo sistema operativo para dispositivos móviles que ha desarrollado Microsoft. Tras el lanzamiento en 2010 se distanció totalmente de su versión anterior Windows Mobile. Ahora este sistema operativo está tomando fuerza desde que BlackBerry empieza a retroceder en ventas y del acuerdo que tomaron Microsoft y Nokia con el objetivo de reinventarse, ha llevado a esta plataforma a situarse en tercera posición en número de dispositivos más vendidos.

Para el desarrollo de esta plataforma se usan los lenguajes de programación C# y Visual Basic .NET

Esta opción de desarrollo la he descartado directamente ya que sus otros dos competidores aún así están distanciados en cuota de mercado de este sistema operativo. Por lo tanto, el desarrollo sigue resultando más rentable para las otras dos plataformas.

1.3.2.3 ANDROID

Android es el sistema operativo que más se usa en la actualidad con una cuota de mercado que supera en más del doble en sus competidores más cercanos como Windows o iOS

Este sistema de la compañía Google es abierto y disponible para cualquier fabricante que decida incluirlo en sus dispositivos. También se añade la opción de incorporar una capa propia desarrollada por el fabricante sobre la capa original, para dotar de una cierta experiencia de usuario que no tenga que ver con la proporcionada por Google.

Esta libertad de desarrollo se contrapone con la estabilidad o robustez que suponen otras plataformas como iOS, que incrementan el rendimiento de la aplicación. Este punto se hace más notorio cuanto más compleja es la aplicación a desarrollar, o más recursos necesitamos.

VENTAJAS

- Java: El lenguaje es mundialmente conocido y existen recursos suficientes para aprender el lenguaje y desarrollar la aplicación sin problemas.

- Software libre: Android es completamente personalizable. Cualquier desarrollador puede realizar modificaciones en las partes internas de un programa a diferencia de productos cerrados que distribuyen el código bloqueado. Gracias a la licencia Apache, se puede compartir, utilizar código, realizar versiones a nuevos sistemas… Se pueden crear nuevas capas sobre la original de Android.

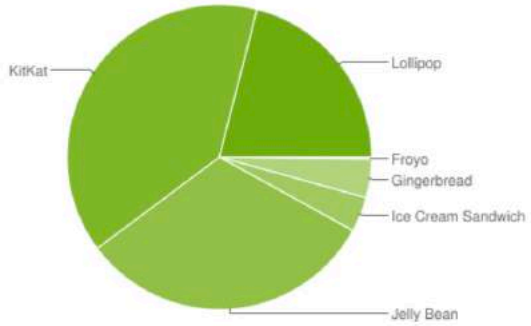
- Accesibilidad: La participación de terceros desarrolladores es aún más accesible en este sistema operativo. Las APIs de Android permiten crear y desarrollar facilitando el trabajo de forma considerable.

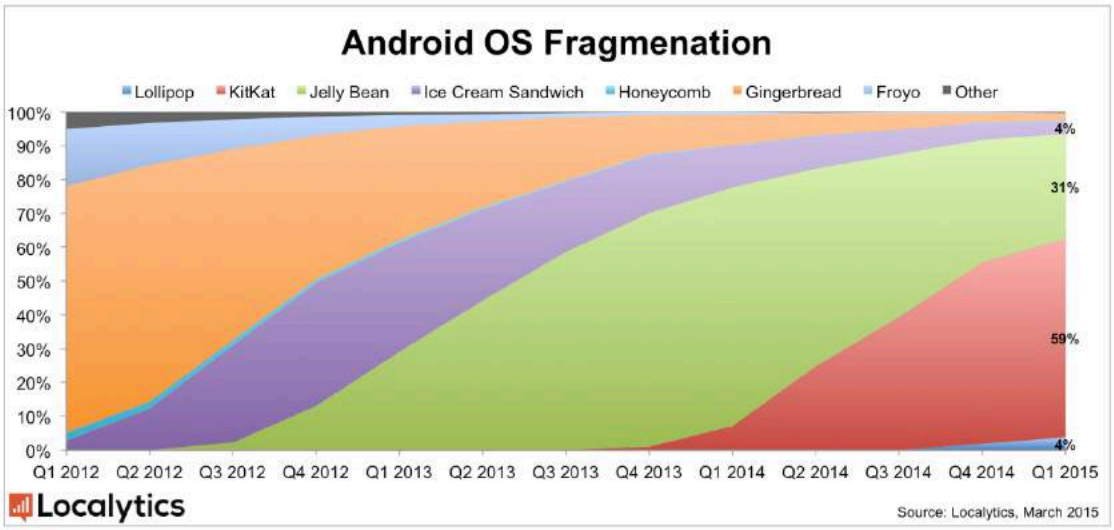
- Comunidad amplia: Existen una gran comunidad de desarrolladores Android creando aplicaciones y frameworks para extender las funcionalidades del sistema operativo.

INCONVENIENTES

- Fragmentación en sus versiones: Se critica mucho la fragmentación que sufren los terminales Android al no disponer de actualizaciones constantes por los fabricantes. Con esto, nos podemos encontrar dispositivos Android con diferentes versiones en sus actualizaciones, no hay un modelo común en el cual se puedan actualizar todas las marcas con plataforma Android a la misma versión. Esto provoca una mayor dispersión en las instalaciones de versiones. Principalmente se debe a que Google deja en manos de los fabricantes la gestión de nuevas versiones y actualizaciones del sistema operativo.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	4.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	3.7%
4.1.x	Jelly Bean	16	12.1%
4.2.x		17	15.2%
4.3		18	4.5%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	15.9%
5.1		22	5.1%





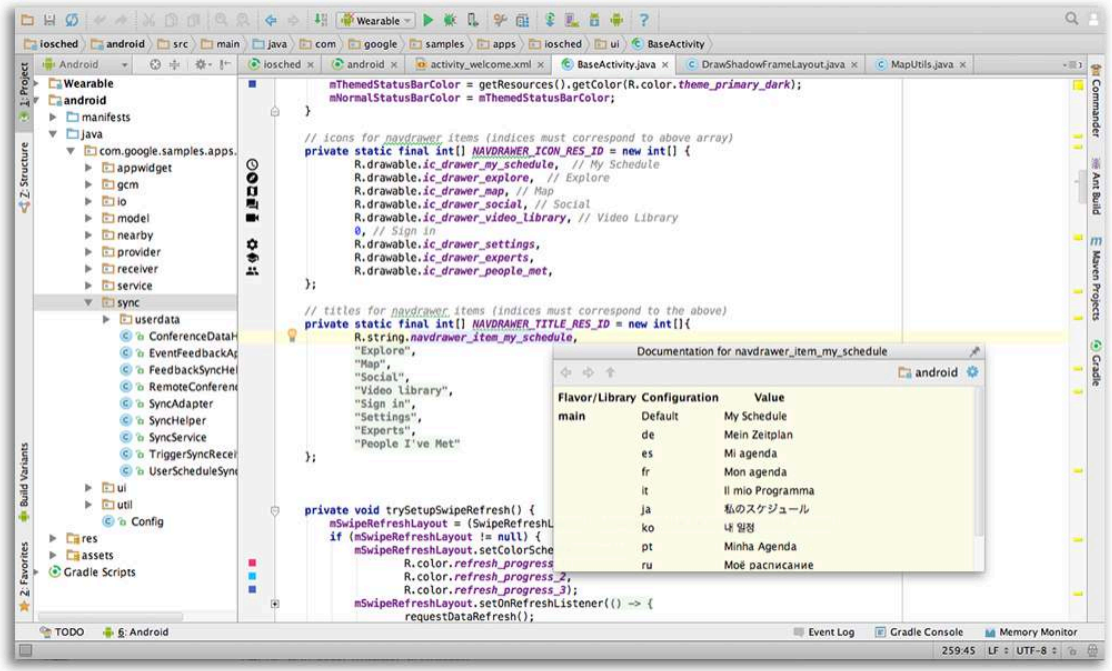
- Aplicaciones: Este es el punto más débil que tiene Android. Tanto las actualizaciones y las aplicaciones necesitan de un control mínimo, exigencias mínimas que no conviertan la bolsa de aplicaciones en un conjunto de aplicaciones “basura” donde las aplicaciones con cierto valor queden camufladas por otras sin sentido. Este es un gran punto en contra, ya que hace que sea realmente difícil destacar la aplicación desarrollada entre tanta aplicación sin valor. Al no existir ningún filtro, se

suben a la tienda multitud de aplicaciones mal implementadas, con errores, y otras que ponen en peligro nuestros datos, malware…

- Comunidad poco crítica: Muchos de los usuarios de la comunidad Android son poco críticos con las aplicaciones y disculpan fallos en el sistema, errores en las aplicaciones y por lo tanto, el filtro que debería haber por parte de los usuarios es mínimo.

ENTORNO DE DESARROLLO

- Android Studio: Android ofrece en la actualidad un nuevo entorno de desarrollo integrado (IDE) con el lanzamiento de una versión estable Android Studio para Windows y Mac. De esta forma intenta acercarse cada vez más al XCODE de Apple. Este software ofrece un potente entorno de desarrollo basado en IntelliJ IDEA de la compañía JetBrains, reemplazando totalmente al antiguo SDK compuesto por Eclipse y el plugin ADT.



1.3.2.4 IOS

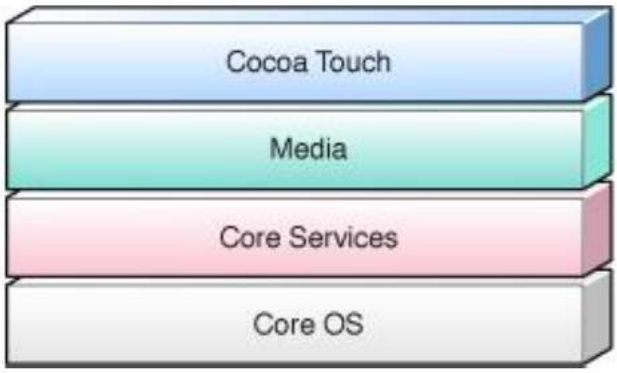
Éste es un sistema operativo que la reconocida empresa Apple incorpora en sus dispositivos a partir de 2007. Aunque iOS era desarrollado simplemente para iPhone, poco a poco se ha ido incorporando a otros dispositivos dentro de la propia empresa como iPod, iPad y AppleTV.

A diferencia de Android, Apple no permite la instalación del sistema operativo en hardware de terceros lo que permite cierta exclusividad y permite desarrollar una plataforma que encaje a la perfección con todos sus productos. Al ser un sistema operativo propio para su gama de productos hace que este sea más estable.

También el elevado precio de los terminales hace que el desarrollo de aplicaciones para esta plataforma sea inferior a la de sus competidores. Esto también tiene sus ventajas e inconvenientes ya que al haber menos aplicaciones en su tienda oficial AppStore, provoca que haya menos competencia en la venta de aplicaciones y el poder adquisitivo de sus usuarios es mucho mayor a las que tiene su gran competidor Android.

Esto provoca que a pesar de la cantidad de descargas que obtiene el sistema operativo Android, el posible beneficio en ventas de aplicaciones se ve reducido con respecto al sistema operativo iOS.

La implementación o arquitectura de la plataforma iOS puede verse como un conjunto de 4 capas que se mantiene en iOS 9 al igual que en las versiones anteriores. Las más bajas pertenecen a los servicios en los que se basan todas las aplicaciones, las capas de alto nivel contienen servicios y tecnologías más sofisticadas. A la hora de escribir código se utilizaran los frameworks de alto nivel.



- Cocoa Touch: Conjunto de frameworks orientado a objetos que permiten el desarrollo de aplicaciones nativas para iOS. En el caso de Mac OSX se llama simplemente Cocoa. El lenguaje con el que se programa es Objective-C y en la actualidad el nuevo Swift. Dentro de esta capa podemos encontrar los siguientes servicios:

    - Eventos multi-touch
    - Acelerómetro y giroscopio
    - Jerarquía de vistas
    - Localización e internalización: Nos permite adaptar la aplicación a diferentes ididomas sin necesidad de cambios en el programa
    - Soporte para cámara

- Media:
    - OpenAL (Open Audio Library)
    - Mezcla de audio y grabación
    - Reproducción de video
    - Formatos de archivos de imágenes
    - Quartz: Framework para manipular gráficos en 2D
    - Core Animation: Framwork para la visualización de datos.
    - Open GL: Framwork para manipular gráaficos en 3D

- Core Services (Servicios básicos):

Contiene los servicios fundamentales todas las aplicaciones ya sean las que incluye Apple por defecto en iOS como en aplicaciones que puedas descargar desde la App Store

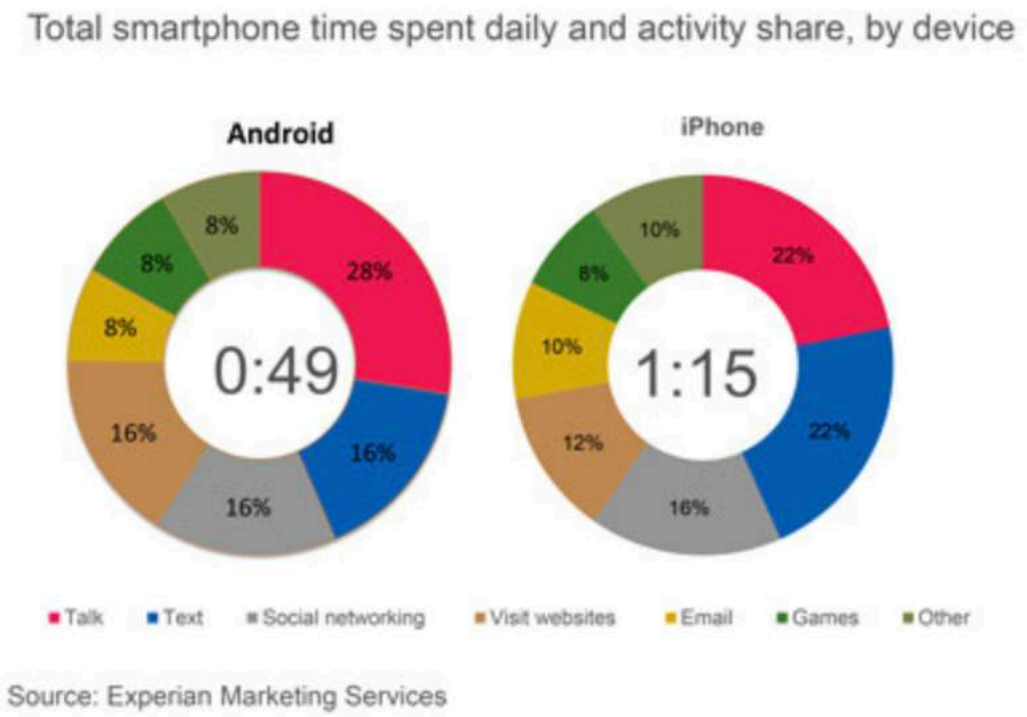
  - Networking
  - Base de datos: SQLite
  - Core Location: Framework que permite un fácil acceso al GPS del terminal
  - Threads
  - CoreMotion

- Core OS (Núcleo del sistema operativo)
  - TCP/IP
  - Sockets
  - Gestión de la batería
  - File System (Sistema de archivos)
  - Seguridad

VENTAJAS

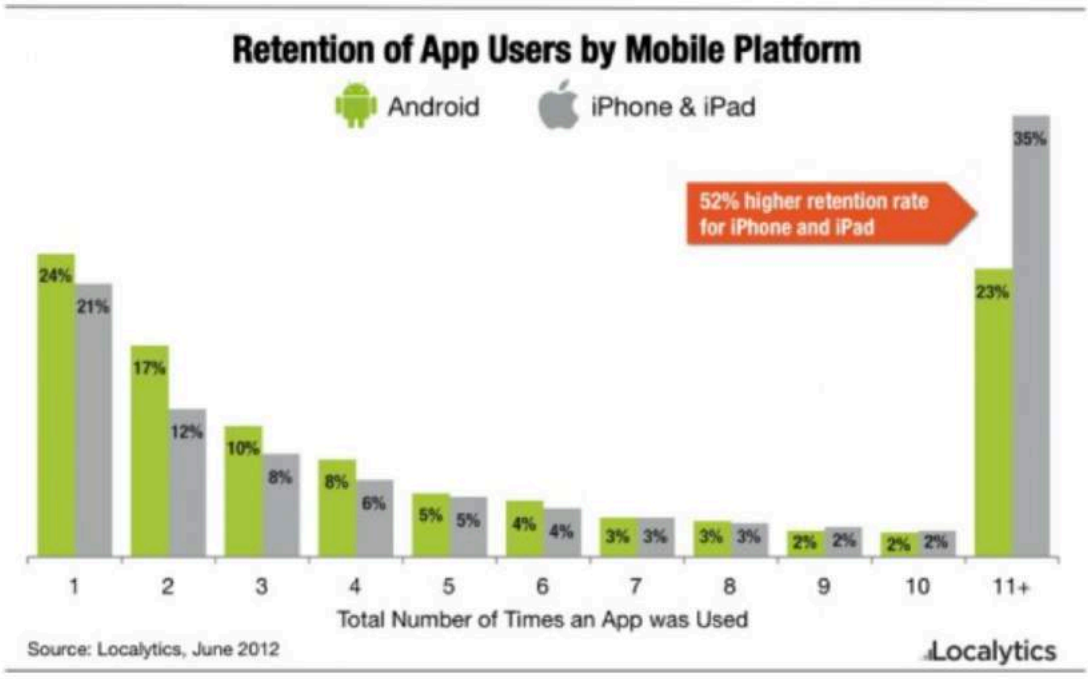
Existen varias razones por las que multitud de desarrolladores, tanto independientes como para empresas, toman la decisión de empezar a programar para iOS en vez de Android.

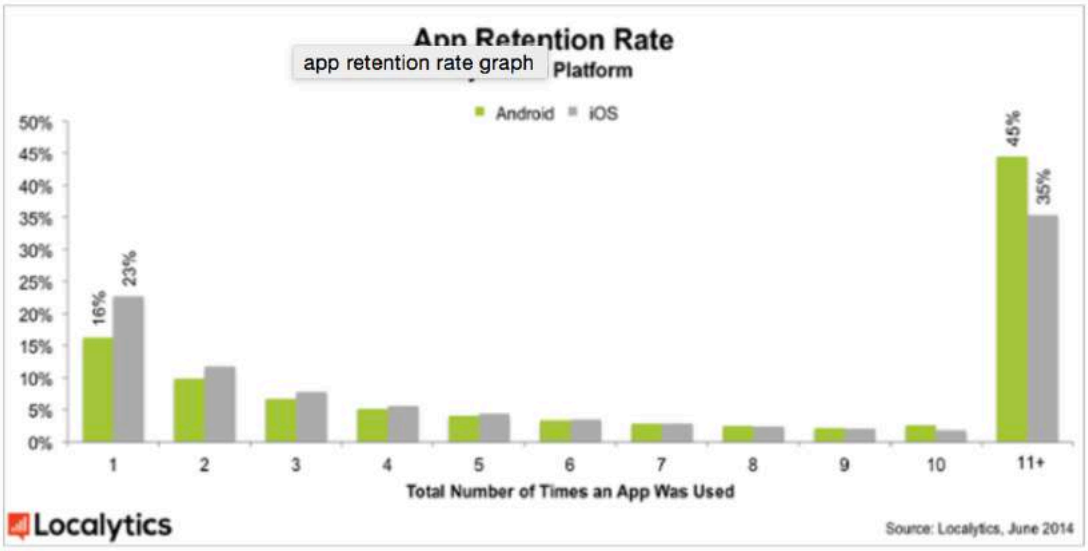
- Usuarios más activos: Según diferentes estudios, los usuarios de Apple tienden a navegar más por internet que los usuarios de Android (67% del tráfico web es de dispositivos móviles en Diciembre 2013) o hacer más compras con ellos.



También parece ser que los usuarios de iPhone pasan mucho más tiempo al día usando el teléfono que en Android (26 minutos más de media que un usuario de Android) pero no tanto para llamar sino como smartphone para todo lo demás.

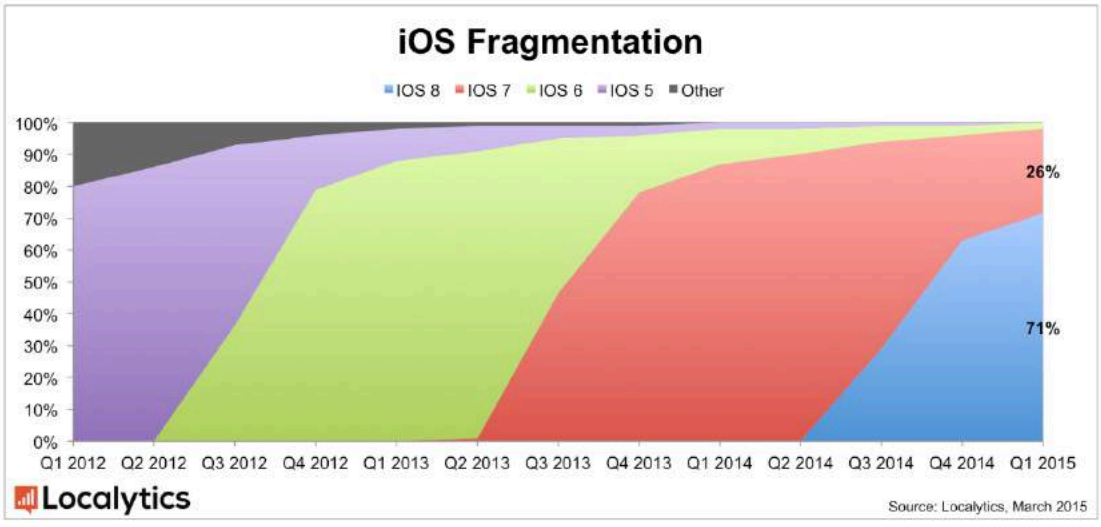
Hasta hace un par de años la retención de usuarios en aplicaciones de iOS era mayor que en aplicaciones Android, es decir, los usuarios eran más fieles a las aplicaciones iOS y usaban las aplicaciones más de 11 veces antes de dejar de usarlas. Aunque este mismo estudio realizado en 2015 da la vuelta y es ahora Android quien lidera esta característica.

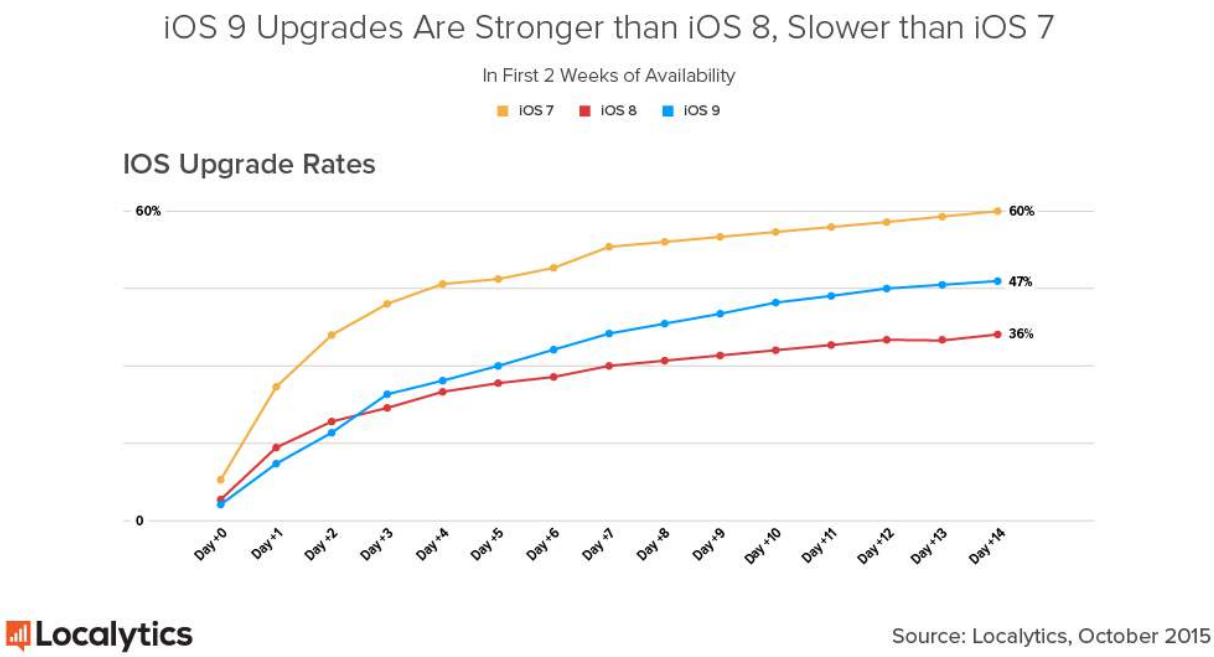




- Menos descargas pero más ingresos: Según afirman datos recogidos por AppAnnie afirman que Google Play obtiene hasta un 60% más de descargas frente a iOS AppStore basándose en datos recogidos en 2014, además se confirma que la comunidad de desarrolladores Android es mayor que la de iOS, pero en cambio, los desarrolladores de Apple generan hasta un 70% más de beneficio con aplicaciones de pago.

- La fragmentación es mínima: A diferencia de Android, la fragmentación de versiones en dispositivos iOS es mucho menor. Con la versión iOS 8 el 71% de todos los dispositivos iOS se han actualizado a fecha de Marzo de 2015. Con la salida al mercado de iOS 9 el 12.5% de usuarios ya se han actualizado pasadas 24h de su lanzamiento. En las primeras dos semanas ya el 47% han adoptado el nuevo iOS 9 en sus dispositivos.





- Multitud de Librerías: Existen en la actualidad multitud de librerías externas realizadas por usuarios muy bien documentadas que podemos reutilizar en nuestro código, esto nos ahorra mucho tiempo de programación.
- Aplicaciones revisadas: Todas las aplicaciones iOS que se suban a la tienda App Store son revisadas previamente a su publicación por la compañía para ver si cumple una serie de normas estipuladas por Apple. Unos requisitos mínimos que hacen que por un lado, las aplicaciones que salgan en la tienda para su descarga sea de una mínima calidad, pero por otro lado, impide al desarrollador cierto tipo de libertad para desarrollar, usar contenido con copyright, normas de estilo…

INCONVENIENTES

- Revisión de Aplicaciones: Como ya he comentado anteriormente, la revisión de aplicaciones por parte de un desarrollador Apple, da cierto tipo de restricciones a los desarrolladores de aplicaciones iOS. Aunque

realmente, este punto es algo mínimo, normalmente, una aplicación funcional, correcta, sin errores y con contenido legal no tiene porqué tener problemas en publicarse en la tienda oficial.

- Lenguaje de desarrollo: Tanto Objective-C como Swift son lenguajes basados en C, y por tanto, son de fácil aprendizaje, aún así, son lenguajes usados exclusivamente para el desarrollo en un entorno Apple. En cambio Java es usado en multitud de desarrollos de software libre.
- Entorno de desarrollo en Mac OS X: Para el desarrollo de aplicaciones iOS es condición indispensable tener un ordenador Mac, puesto que no es posible encontrar el IDE con el framework de iOS para cualquier otra plataforma. Es conveniente pensar por otra parte, que si un usuario desarrolla en un entorno iOS para Apple, es muy probable que tenga otros productos Apple como el Mac.
- Cuota desarrollador iOS: Existe una cuota de desarrollador iOS por la que es necesario pagar 99 euros anuales para disfrutar de las herramientas de desarrollo y subir aplicaciones a la tienda App Store.

ENTORNO DE DESARROLLO

Lo primero que debemos tener para empezar a programar en iOS es un Mac. Se puede simular también en una máquina virtual a través de Windows con el software VMWare e instalando la última versión de Mac OS X.

En mi caso, ya disponía de un ordenador Mac, con lo que ese paso no era realmente necesario. Además del Mac, de manera opcional, es recomendable tener otros dispositivos iOS para poder testear de forma natural la aplicación a desarrollar como un iPhone 6 o un iPad. En cualquier caso, podemos usar el propio simulador que la herramienta de desarrollo nos ofrece.

Para empezar a desarrollar aplicaciones deberemos inscribirnos en el iPhone Developer Program y pagar la cuota anual de 99€.

Desde hace unos años, Apple distribuye el software de desarrollo XCode de forma gratuita para todos los usuarios, siendo de forma exclusiva las versiones beta tanto del SDK como de XCode para los desarrolladores subscritos al iPhone Developer Program. Este paso es realmente importante para la gente que quiera empezar a desarrollar en esta plataforma, sin riesgo a que en un futuro quiera cesar la actividad sin pagar ninguna cuota.

Por lo tanto, en la actualidad, un desarrollador iOS que no pague la cuota establecida, podrá programar una aplicación para dispositivos iOS y probarla en el simulador sin problemas a través de un ordenador Mac. Pero no podrá, ni probarla en un dispositivo físico, ni subirla a la App Store para su comercialización.

Entre las herramientas que nos ofrece Apple para el desarrollo iOS, las más importantes son:

- XCode: Entorno de desarrollo integrado para gestionar proyectos. Nos permite editar, compilar, ejecutar y depurar el código. En la actualidad se integra con otras herramientas. Esta herramienta es la que se usa para la programación de la aplicación.
- Interface Builder: Ayuda a crear visualmente las vistas del proyecto. Los objetos usados en la interfaz de la aplicación se guardan en un fichero especial en XML y se carga en la aplicación en tiempo real
- Instruments: Analiza y depura el código en tiempo de ejecución para ver el rendimiento de la aplicación. Reúne información de uso de CPU, monitor de actividad, pérdidas de memoria… Nos permite detectar problemas potenciales en la aplicación.

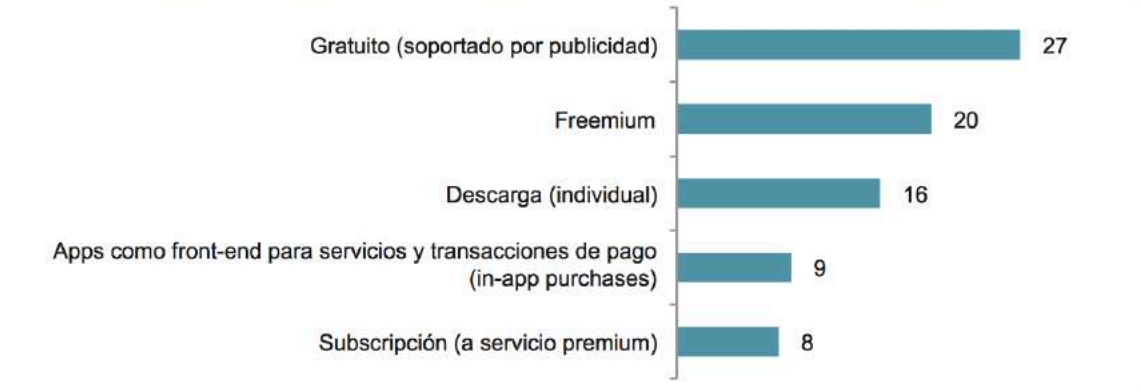
1.3.3 POSIBILIDAD DE COMERCIALIZACIÓN Y VENTA

Existen diferentes opciones de comercialización y venta de aplicaciones. Habrá una diferencia en la toma de decisiones a seguir dependiendo del sistema operativo donde desarrollemos la aplicación, ya que la cuota de mercado y el perfil de un usuario para un sistema Android es diferente a los del sistema iOS.

Como podemos ver en el siguiente gráfico, los modelos gratuitos se imponen como modelos de ingresos más adoptados en el negocio de aplicaciones en 2015. Según este informe de un estudio de Telecomunicaciones para el 2015 proporcionado por Altran, uno de los retos más importantes de los desarrolladores de aplicaciones móviles consiste en cómo capturar valor de los usuarios que se decantan por las aplicaciones gratuitas. Dichas aplicaciones constituyen a día de hoy entre un 20% y un 30% de la totalidad, pero representan casi el 90% de las descargas. Bajo estas condiciones se explica el atractivo de los modelos gratuitos, donde la publicidad representaría la única fuente de ingresos para los desarrolladores.

En cualquier caso, se nota cierta tendencia a buscar modelos de ingresos alternativos como los modelos freemium o in-app purchase, que pueden ser debidos entre otras causas a la caída de ingresos por publicidad.

5A. Modelos de ingresos para aplicaciones móviles con mayor impacto en el 2015
- Ranking Agregado -



1.3.3.1 APLICACIÓN DE PAGO

Esta opción es viable preferiblemente para aplicaciones desarrolladas en el sistema operativo iOS, aunque cada vez menos, los usuario con más poder adquisitivo están dispuestos a pagar por una aplicación que les proporcione un determinado servicio.

1.3.3.2 APLICACIÓN GRATUITA o FREEMIUM

Esta opción es la que se toma cada vez con más frecuencia, sobretodo para desarrolladores Android, ya que la competencia en su tienda oficial es sobradamente mayor que en la plataforma iOS.

Debido a esto, los desarrolladores se ven obligados a poner las aplicaciones gratuitas con algún sistema de venta dentro de esta para rentabilizarla de alguna manera.

Si se desea que una aplicación gratuita obtenga algún tipo de rendimiento económico, es necesaria la incorporación de publicidad extra en forma de banner dentro de la aplicación, ya sea de algún anunciante determinado o a través de plataformas de anuncios.

Se puede usar iAds (para sistemas iOS) o Google AdMob que son las más importantes, pero también, podemos encontrar muchas como: Chartboost, Madvertise, Millenial Media, Adfonic, Tapjoy, Aditic, Admoda, Flurry, Hunt, inMobi, Jumptap, Kiip, Mobfox, Mobpartner, MoPub, Mojiva, PlayHaven, RevMob y Smaato.

A continuación se muestra una pequeña comparativa entre las diferentes plataformas y modelos de rentabilidad.

Ad Networks Comparison Chart

Network	Known Model	Supported Ad Format	Media	House Ads
AdMob	CPC	Banner, Smart Banner, Interstitial, Search, Table	Yes	Yes
mMedia	CPC, CPM, CPA	Banner, Interstitial, Interactive Video, Rich Media	Yes	Yes
Adfonic	CPC, CPM	Banner, Interstitial, Video, Rich Media	Yes	Yes
Chartboost	CPC, CPI	Interstitial	No	Yes
Tapjoy	CPI	Banner, Interstitial, Video, Offer Wall	Yes	No
Aditic	CPC	Banner, Expandable, Interstitial, Splash Screen, Video	No	Yes
Admoda	CPC, CPM	Banner, Post-roll, Text, Click-to-Call	No	Yes
Flurry	CPC, CPM, CPI	Banner, Interstitial, Video, Takeover	Yes	Yes
HUNT	CPC, CPM, CPA	Banner, Expandable, Interstitial, Video	Yes	No
InMobi	CPC, CPM	Banner, Expandable, Interstitial, Video, Text, Rich Media	Yes	Yes
Jumptap	CPC	Banner, Interstitial, Video, Rich Media	Yes	Yes
Kiip	CPI	Offer Wall	No	No
Madvertise	CPC, CPM	Banner, Expandable, Overlay, Video Overlay, Flip, Engagement	Yes	Yes
Mobfox	CPC, CPM	Banner, Video	Yes	Yes
MobPartner	CPC, CPI, CPS, CPL	Banner, Interstitial, Video, Offer Wall	No	No
MoPub	CPC, CPM	Banner, Expandable, Interstitial, Video, ORMMArich Media Ads, Landing Page	Yes	Yes
Mojiva	CPC, CPM	Banner, Expandable, Interstitial, Video, Text, Table	Yes	No
PlayHaven	CPI	Interstitial, Offer Wall	No	No
RevMob	CPI	Banner, Interstitial	No	No
Smaato	CPC, CPM	Banner, Expandable, Interactive Banner, Floating Banner, Video, Text	Yes	Yes

Como **Known Models** se conocen:

- CPC (Cost Per Click): También conocido como PPC (Pay Per Click), implica que sólo se paga por el número de clicks que se hacen en un anuncio.
- CPM (Cost Per Mile Impressions): Se relaciona directamente con la cantidad de veces que se muestra un anuncio. Indica el valor que se da a cada 1000 impresiones de un mismo banner.
- CPA (Cost Per Action): Es el coste que se paga por una acción realizada por el usuario.

- CPI (Cost Per Install): Se refiere al pago por instalación de la aplicación. Se cuenta a partir de que el usuario abre la aplicación descargada.
- CPS (Cost Per Sale): Coste de venta de la aplicación. No requiere necesariamente que se abra o se use la aplicación.
- CPL (Cost Per Lead): Este modelo valora la conversión de un usuario lead, es decir, una persona que ha proporcionado información adicional, ha cumplimentado un registro, realizado alguna descarga o suscrito a algún servicio. Este modelo es interesante para conseguir una base de datos de contactos interesados en la aplicación.

En cuanto a **Media**, se considera las plataformas que pueden anunciar contenido en video o audio.

House Ads es una nueva funcionalidad que permite a la aplicación publicitar sus propias aplicaciones. Este servicio normalmente suele ser gratuito ya que simplemente se crea una red interna de anuncios que beneficia al propio desarrollador y sus aplicaciones publicadas en la tienda.

También es posible incluir alguna tienda dentro de la app, que a través de pequeños pagos, se puedan obtener o desbloquear ciertas funcionalidades de la aplicación o incluso también es posible implementar algún tipo de suscripción mensual o anual.

Estas opciones descritas anteriormente, también son posibles en las aplicaciones de pago, pero no son viables, ya que se le está cobrando al usuario de forma doble por un servicio que ya ha pagado en la descarga.

1.3.3.3 RETORNO DE INVERSIÓN (ROI)

Es importante tener en cuenta a la hora de escoger el sistema operativo en el cual desarrollar la aplicación, el retorno de la inversión, ya que no siempre es el mismo y ni si quiera es siempre proporcional a la inversión, para cada aplicación desarrollada puede ser diferente.

Las aplicaciones móviles de marca y servicio son aquellas que se pueden descargar y utilizar gratuitamente, pero que tampoco persiguen ningún tipo de ROI en la aplicación, ya que buscan potenciar la marca. Este ROI medirá a medio y largo plazo una mejora de la imagen y percepción de la marca e intención de compra, atracción y fidelización de nuevos clientes. También conseguirá mayor difusión y viralidad de la marca.

En aplicaciones de pago, el retorno de la inversión es inmediato, sin embargo este método no es el preferido por los usuarios que deben pagar antes de usar y ver la aplicación. Se crean unas expectativas altas cuando luego existe una posibilidad de decepcionar al usuario. Con este modelo podemos optar por precios bajos (menos de un euro) o por precios altos si el servicio que se ofrece es de gran calidad y necesario para el usuario. Este es el modelo a seguir preferiblemente para desarrolladores iOS, ya que el usuario tipo es más proclive al pago de aplicaciones, ya que suelen tener un poder adquisitivo más alto, mientras que el usuario móvil de Android apuesta más por aplicaciones gratuitas.

En aplicaciones gratuitas “Freemium” podemos obtener un retorno de inversión dentro de la aplicación una vez ha sido descargada por el usuario. Son las más abundantes y se basan en incluir publicidad, compras dentro de la aplicación para desbloquear servicios, funcionalidades… (In-Apps) o pagos por suscripción.

Tanto la Apple Store para iOS, como Google Play para Android, ofrecen herramientas para medir el retorno de la inversión, como contabilizar las descargas e ingresos percibidos, de donde proceden los usuarios, comentarios y opiniones, reporte de errores…

1.4 SWIFT VS OBJECTIVE-C

Para centrarnos en la elección de un lenguaje de programación u otro es necesario saber con anterioridad ciertos aspectos de cada lenguaje así como de donde provienen y cual es su historia.

1.4.1 HISTORIA DEL LENGUAJE PARA DISPOSITIVOS IOS

Hace 43 años (en 1972) que nace el lenguaje C, de donde se basa toda la estructura del lenguaje para el desarrollo de aplicaciones para dispositivos iOS. En aquel año se introdujo una sintaxis innovadora que acabó convirtiéndose en la más popular del mundo, y que después, ha influenciado a muchos lenguajes de la actualidad.

En 1980 aparece SmallTalk, éste fue el primer lenguaje orientado a objetos. En este momento se piensa que es posible crear un lenguaje como C pero orientado a objetos, de forma que nace C++ en 1983. Es durante los próximos años y más concretamente en 1986 cuando se crea Objective-C como una extensión del lenguaje C añadiendo cierto tipo de clases, objetos, métodos, protocolos, excepciones, propiedades y categorías.

Ya en 1988 Objective-C fue adoptado por NeXT (compañía que fundó Steve Jobs) base del nuevo sistema operativo que correrían sus computadores, NeXTStep.

En el año 1996 Apple compró NeXT y su sistema operativo NeXTStep pasó a llamarse Mac OS X en 2001. Es por esto, que las mayorías de las clases que integran la SDK de iOS mantienen el prefijo NS (de NeXTStep)

En 2007 se introduce el iPhone en el mercado de los teléfonos móviles inteligentes. Su sistema operativo estará basado en la tecnología Mac, por ello se usa Objective-C también para desarrollar aplicaciones nativas para este dispositivo.

En resumen Objective-C lleva casi 30 años de historia en la plataforma Apple y su evolución ha llevado muchas mejoras, sobre todo a nivel de gestión de memoria.

Durante el año pasado 2014, la compañía presenta un nuevo lenguaje llamado Swift por el que Apple ha dedicado multitud de recursos a lo largo de años anteriores. Este nuevo lenguaje coexiste de momento con Objective-C sin problemas y es utilizado actualmente para todos los dispositivos iOS.

Cuando Apple presentó Swift, aseguró que era un código “interactivo y divertido”, con una sintaxis concisa a la vez que expresiva y que las aplicaciones funcionan mucho más rápido con respecto al anterior lenguaje Objective-C con el incremento de rendimiento que se ha obtenido optimizando el compilador.

1.4.2 VENTAJAS DE SWIFT FRENTE A OBJECTIVE-C

Objective-C, hasta la fecha, ha supuesto la base del desarrollador iOS para programar aplicaciones para dispositivos Apple, y por lo tanto, muchos de estos desarrolladores son reacios a cambiar de lenguaje ya que la sintaxis que ofrece Swift es muy diferente. El cambio supone un salto bastante amplio aunque los beneficios son bastante importantes a pesar de estar en su versión 1.2

El aprendizaje del lenguaje es el único frente encontrado a la hora de adoptar la decisión de desarrollar en un nuevo lenguaje. En principio resulta difícil pero la estructura de programación se asemeja más al clásico C con lo que puede favorecer a nuestro desarrollo.

También existe la posibilidad de desarrollar en una mezcla de ambos lenguajes, aunque esta puede resultar muy engorrosa en el código una vez el

desarrollador se pone a programar, y puede dar lugar a errores si no dominamos bien el nuevo lenguaje.

Por ello, visto que son más las ventajas que inconvenientes encontrados a la hora de desarrollar en Swift, vamos a comentar ciertos puntos interesantes por los cuales debemos aprender este nuevo lenguaje:

- Es más fácil de leer: Para diferenciar las palabras clave del lenguaje Objective-C de los propios tipos de C se introdujeron palabras clave con el símbolo @. Debido a que Swift no se construye sobre C, puede unificar todas las palabras clave y eliminar las numerosas @.

  En Swift ya no es necesario el punto y coma en la terminación de líneas e incluso en los paréntesis de un if/else. Otro cambio importante es que las llamadas a métodos no se anidan una dentro de otra formando un conjunto ilegible de corchetes [[[]]]. Las llamadas a métodos y funciones en Swift utilizan una lista separada por comas y entre paréntesis para los distintos parámetros. El resultado es un lenguaje más limpio, más expresivo con una sintaxis y gramática simplificada.

  El código Swift se asemeja más al Inglés natural como otros lenguajes de programación modernos. Esta lectura hace que sea más fácil para los programadores existentes de Javascript, Java, Python, C# y C++

- Tiene fácil mantenimiento: Un requisito de Objective-C debido a su evolución en C es que necesita mantener dos archivos de código con el fin de mejorar el tiempo de compilación y la eficiencia en la creación del fichero ejecutable.

  En el caso de Swift, deja este requerimiento atrás para pasar a un único archivo de código. XCode y el compilador LLVM, reducen la carga de trabajo del programador, pueden encontrar las dependencias y realizar compilaciones incrementales en Swift 1.2, por lo tanto ya no es

necesario el uso de un archivo de cabecera (.h) y uno de implementación (.m). Se convierte todo en un único archivo (.swift). Esto ha supuesto hasta la fecha una labor adicional que impide al programador una vista global del proyecto. Con Swift, los programadores dedican menos tiempo a escribir código de cabecera y pueden dedicar más tiempo a la lógica de la aplicación a crear. Swift elimina el trabajo repetitivo y mejora la calidad del código, comentarios…

- Es más seguro: Un aspecto interesante de Objective-C es la manera en que se manejan los punteros, especialmente los punteros a nil (null). En Objective-C no pasa nada si se intenta llamar a un método con una variable que es nula (sin inicializar). La expresión o línea de código se convierte en un no-operation (no-op), y si bien puede parecer beneficioso que no se bloquee, esta es una gran fuente de errores. Un no-op lleva a un comportamiento impredecible, que es el enemigo de los programadores que tratan de encontrar y solucionar un error que no se sabe porqué se ha producido o detener un comportamiento errático.

  Aquí juegan un papel importante los nuevos tipos Opcionales de Swift. Estos se comportan como contenedores de un tipo de datos o contenedores de un nil. Con lo que puede generar un error de compilación a medida que se escribe mal el código. Esto genera un feedback a corto plazo mientras el código es escrito, lo que reduce en gran medida la cantidad de tiempo y dinero que se va a gastar en arreglar errores.

  Tradicionalmente, en Objective-C, si un valor es devuelto de un método, era responsabilidad del programador documentar el comportamiento del puntero devuelto (usando comentarios y reglas en el nombramiento de métodos). En Swift, los tipos Opcionales y los tipos con valor hacen que sea explícitamente claro en la definición del método si un valor existe o si debe ser Opcional (ya que el valor puede ser nulo).

Para proporcionar un comportamiento predecible, Swift desencadena un error en tiempo de ejecución si se lee una variable Opcional nula. Este error proporciona un comportamiento coherente, lo que facilita el proceso de corrección de errores, ya que obliga al programador a solucionar el problema de inmediato. El error en tiempo de ejecución, detendrá la aplicación en la línea de código donde se ha utilizado una variable opcional nula.

- Unifica la gestión de memoria: Swift unifica el lenguaje de una manera que Objetive-C no hace. El soporte de ARC (Automatic Reference Counting) se completa a través de rutas de código de procedimiento y orientado a objetos. En Objective-C, ARC se encuentra dentro de las APIs de Cocoa y el código orientado a objetos. No está disponible, sin embargo, para el código C procesal y APIs como el Core Graphics. Esto significa, que es responsabilidad del programador, manejar la gestión de memoria cuando se trabaja con la API de Core Graphics y otras APIs de bajo nivel disponibles en iOS. Las enormes pérdidas de memoria que un programador puede obtener en Objective-C son imposibles en Swift.

  Un programador no debería tener que pensar en la gestión de memoria para cada objeto que crea. Como ARC maneja toda la gestión de memoria en tiempo de ejecución, el capital humano que se hubiera dedicado a esta gestión se centra en la lógica de la aplicación y a crear nuevas características y funcionalidades. Como ARC en Swift trabaja tanto en código procesal como orientado a objetos, no requiere que estemos pensando en la gestión de memoria si se trabaja con APIs de más bajo nivel, un problema que sí ocurre en Objective-C.

- Requiere menos código: Swift reduce la cantidad de código que se requiere para los estados repetitivos y la manipulación de cadenas. En Objective-C el trabajo con cadenas de texto es muy detallado y requiere muchos pasos para combinar dos piezas de información. Swift adopta

las características de lenguajes de programación modernos como unir dos cadenas con un operador “+”, lo que falta en Objective-C. El soporte de combinar cadenas para cualquier lenguaje de programación es fundamental.

La inferencia de tipos en Swift reduce la complejidad de instrucciones de código, ya que, el propio compilador averigua el tipo que estas usando sin tener que definirlo previamente. A modo de ejemplo, Objective-C construye una cadena de tipos que previamente están definidos como :%s, %d, %@. Para Swift el tipo no es necesario, basta con: \(variable).

- Es más rápido: El rendimiento del código Swift sigue apuntando a la dedicación de Apple en la mejora de la velocidad a la que Swift puede ejecutar la lógica de la aplicación.

  Según Primate Labs, creadores de la popular herramienta de rendimiento GeekBench, Swift se acercaba a las características de rendimiento de C++ en diciembre de 2014 utilizando el algoritmo Mandelbrot. En febrero de 2015, Primate Labs descubrió que el XCode 6.3 beta mejoró el rendimiento en Swift del algoritmo GEMM (un algoritmo de memoria vinculado con el acceso secuencial de grandes arrays) por un factor de 1.4. La aplicación FFT (un algoritmo de memoria con acceso aleatorio de grandes arrays) tuvo una mejora de rendimiento de 2.6 veces mayor.

  Con el paso del tiempo se observaron mejoras sustanciales en Swift mediante la aplicación de mejores prácticas, lo que resultó un impulso de 8.5 veces el rendimiento anterior del algoritmo FFT. Las mejoras también permitieron a Swift superar a C++ en el algoritmo Mandelbrot por un factor de un 1.03

Con lo cual, Swift está ahora prácticamente a la par que C++, tanto para los algoritmos FFT y Mandelbrot. Según Primate Labs, el rendimiento de los algoritmos GEMM sugieren que el compilador de Swift no puede vectorizar código que el compilador de C++ si puede. Este último es un factor a mejorar que podría lograrse para las próximas versiones de Swift.

- Menos colisiones de nombre con los Espacios de nombres: Un problema que desde siempre ha venido sufriendo el código de Objective-C es su falta de soporte para los espacios de nombres, que era la solución de C++ a las colisiones de código en los nombres de archivo. Cuando la colisión ocurre en Objective-C es un error de vinculación y la aplicación no se puede ejecutar. Existen para ello soluciones temporales pero con grandes dificultades. La regla común es usar de dos a tres letras como prefijo para diferenciar el código que se escribe en Objective-C de otros códigos como por ejemplo Facebook.

  Swift ofrece espacios de nombres implícitos que permiten que el mismo archivo de código pueda existir en múltiples proyectos sin causar fallos en la compilación y sin requerir nombres como NSString o CGPoint. Por último, esta característica en Swift mantiene una productividad mayor en los programadores sin tener que complicarse en este tipo de conflictos. Por ello vuelven a aparecer los conocidos String, Dictionary, Array en lugar de NSString, NSDictionary o NSArray que nacieron por la falta de espacios de nombres en Objective-C.

  Con esto se facilita en gran medida la incorporación de proyectos de código abierto en tu propio código. Además permite a diferentes empresas de software crear los mismos nombres de archivos de código sin preocuparse de las colisiones al integrar los proyectos de código abierto. Por ejemplo, ahora tanto Facebook como Apple pueden utilizar un archivo de código de un objeto llamado FlyingCar.swift sin errores o fallos de compilación.

- Soporta bibliotecas dinámicas: El mayor cambio en Swift que no ha recibido tanta atención es el cambio de bibliotecas estáticas a bibliotecas dinámicas. Las bibliotecas dinámicas son partes ejecutables de código que pueden estar vinculados a una aplicación. Esta características permite a las aplicaciones actuales de Swift enlazar con otras nuevas versiones del lenguaje mientras evoluciona en el tiempo.

  El desarrollador sube a la App Store la aplicación con las librerías, las cuales están firmadas digitalmente con el certificado de desarrollo. Esto quiere decir que Swift puede desarrollarse más rápidamente que la versión iOS que es un requisito indispensable para un lenguaje de programación moderno. Los cambios en las bibliotecas pueden ser incluidos con la última actualización de una aplicación en la AppStore.

  Las bibliotecas dinámicas nunca habían sido soportadas por iOS hasta la salida de Swift con iOS 8, a pesar de que Mac ya las había soportado hace un tiempo atrás. Las bibliotecas dinámicas son externas al ejecutable, pero se incluyen en el paquete de la aplicación descargada desde la App Store. Esto reduce el tamaño inicial de una aplicación ya que el código externo se carga simplemente cuando se va a usar.

  Las posibilidad de aplazar la carga de bibliotecas en una aplicación móvil mejorarán el rendimiento percibido por el usuario. Apple se centra más en la carga de imágenes, recursos… ahora compila y enlaza código sobre la marcha. Esto reduce los tiempos iniciales de espera.

  Las bibliotecas dinámicas en Swift hacen que los cambios y mejoras en el lenguaje de programación se noten con una simple actualización de la aplicación sin esperar al estreno de una nueva versión iOS.

- Nueva funcionalidad: Swift Playgrounds para probar código. Esto es en cierta medida una bendición para los desarrolladores experimentados. Estos permiten probar nuevos algoritmos sin tener que crear una nueva aplicación entera en iPhone. Esto resulta muy beneficioso a los programadores que en un proceso de escritura de código desean probar cierto algoritmo y ver su funcionalidad antes de añadirlo al código de la aplicación. De manera que agiliza mucho el proceso y el trabajo del programador haciendo su trabajo más productivo.

- Swift tiene futuro. Capacidad de mejora: Objective-C continuará a pesar de la entrada del nuevo lenguaje Swift, pero no tendrá muchos más cambios de los que ya tiene. Obviamente algunas características nuevas de Swift migrarán a Objective-C dotándolo de más funcionalidad.

Apple está enfocada en ofrecer la mejor experiencia al usuario en las aplicaciones y esto viene de la mano de Swift, con lo que se esperan muchos cambios en las próximas versiones. En la ultima WWDC 2015 ya se ha anunciado Swift 2 para desarrolladores con multitud de mejoras.

1.4.3 VISTA RÁPIDA DE CÓDIGO SWIFT

La intención de este punto no es explicar cómo se procede con el lenguaje Swift en detalle, sino mostrar por encima algunos ejemplos de código para poner en valor la sencillez y la fácil lectura que resulta el uso de este lenguaje.

Tipos Básicos

Int: Número entero de 32 bits (-2.147.483.648, 2.147.483.648)
Double: Número flotante de 64 bits con hasta 15 decimales de precisión
Float: Número flotante de 32 bits con hasta 6 decimales de precisión
Bool: Número booleano que puede ser 0 (false) o 1 (true)
String: Cadena de caracteres, tratada como un array de caracteres.

Variables y Constantes

Se usa **let** para hacer una constante y **var** para hacer una variable. El valor de una constante no se tiene por qué saber en tiempo de compilación, pero hay que asignarle un valor una vez. Esto significa que puede usar constantes para nombrar un valor que se determina una vez, pero que se puede utilizar en muchos lugares.

La inferencia de tipos nos ayuda a reducir la cantidad de código necesaria para realizar declaraciones de variables… El compilador determina automáticamente de qué tipo se trata. Para evitar confusiones al compilador podemos optar por determinar qué tipo estamos tratando.

```
// Variables (var) y Constantes (let)
var mensaje: String = "Hola ¿qué tal?"
let animales: Int = 8

// Variables y Constantes (inferencia de tipos)
var nombre = "Antonio"
var numero = 9.2, apellido = "Perico"
let peras = 8
```

Como podemos observar no se añade el ; al final de cada línea.

Conversión de Tipos

En el siguiente ejemplo vemos como convertir un tipo Float a un tipo Int

```
var numero: Int = Int(numero)
```

Lectura / Escritura por Terminal

Existe una forma mas simple de introducir valores en strings, simplemente se añade \ antes del paréntesis.

```
let nombre = "Alberto"
let peras = "cinco"

println("Su nombre es \(nombre) y tiene \(peras) peras")
```

Colecciones: Arrays

Existen diferentes maneras de declarar un Array. También podemos añadir objetos al Array con el método **append**. Es importante saber que es necesario que todos los objetos del Array sean del mismo tipo.

```
var arrayVacio: [Int] = [] // definimos array vacío
var arrayCadenas = ["Uno", "Dos", "Tres", "Cuatro"]
var cadena = arrayCadenas[2] // leer

arrayCadenas.append("Cinco") // añadir
arrayCadenas[1...3] = ["Dos_", "Tres_", "Cuatro_"]
```

Colecciones: Diccionarios [Key:Value]

Las colecciones se tratan del mismo modo que un Array, simplemente se componen de un índice y un valor. Si en la asignación de un valor no existe el objeto, este se creará automáticamente y se añadirá al diccionario.

```
var dictionaryMovies = ["Iron Man": 3, "Capitans": 2, "Hulk": 2,
"Thor": 2]
var   dictionaryMovies2:   [String:Int]   =   ["Iron   Man":   3,
"Capitans": 2, "Hulk": 2, "Thor": 2]

dictionaryMovies["Thor"] = 4
dictionaryMovies["Viuda Negra"] = 3 // como no existe, se añade
```

Lectura de Diccionarios

La lectura de diccionarios la podemos realizar mediante **Tuplas** (explicado más adelante), de manera que obtenemos en una sola línea el valor y el índice.

```
for (valor, indice) in dictionaryMovies {
    println(valor)
    println(indice)
}
```

Controles de Flujo

El uso de controles de flujo es imprescindible en cualquier programación. A continuación se detalla la forma en la que se usa cada una de las sentencias If, Switch, For in y Do while a modo de ejemplo.

En una sentencia **If** no son necesarios los **paréntesis ()**. El compilador reconoce que desde if hasta el corchete { se está tratando una condición.

Para el caso de **Switch**, no son necesarios los **break** después de cada caso.

```
// Controles de flujo: If
var peliculas: [String] = ["Hulk", "Cskd", "aksdk"]

if peliculas[1] == "Hulk" {
    println("Si, el vengador es \(peliculas[2])")
}

// Controles de flujo: Switch
switch peliculas {
    case "Iron Man":
        println("Es Iron Man")
    case "Hulk", "Viuda Negra":
        println("Tenemos varios case")
    default:
        println("Opcion default para el resto")
}

// Switch con Tuplas
var centro = (160,100)
let coordenada = (50,50)

switch coordenada {
    case (0,0):
        println("Point 0")
    case (0,_):
        println("Pegado a la izquierda")
    case (_,0):
        println("Pegado abajo")
    case let (x,y) where x == y
        println("Pasa por la diagonal")
    case let (x,y) where x == centro.0 && y == centro.1:
        println("En el centro")
    case  (centro.0-2...centro.0+2,centro.1-2...centro.1+2):  //
se define un rango
        println("Cerca del centro")
    default:
```

```
        println("default")
}

// Controles de flujo: For in
for v in vengadores {
    println(v)
}
for n in 1...6 {
    println("Número \(n)")
}
for num in stride(from:1, to:10, by:2){
    println("Numero de dos en dos")
}
for (index, cadena) in enumerate(cadenas) {
    println("La cadena \(cadena) tiene el índice o posición
\(index)")
}

// Controles de flujo: Do while
var i = 1
do{
    println("Hace algo repetidamente")
    i++
} while i < 8
```

Opcionales

Un valor opcional puede contener un valor o puede ser *nil* para indicar que el valor no se encuentra. Se escribe un signo de interrogación (?) después del tipo de valor para marcar el valor como opcional. Para poder ver el valor que contiene el opcional deberemos realizar *if let* del valor. Si el valor opcional es *nil*, la condición es *false* y el código entre llaves se salta.

Por otra parte, el valor opcional se desenvuelve y se asigna a la constante después de *let*, lo que hace que el valor sin envolver este disponible dentro del bloque de código.

```
// Es un contenedor que puede contener un objeto o un nil
var valor:Int?

// Desempaquetado o unwrap
if let valorSeguro = valor {
    println("Leemos \(valorSeguro) de forma segura") // SEGURO
}
```

```
println("Leemos \(valor!) de forma no segura") // NO SEGURO
```

Clases

```
// Clases - se pasa por referencia
//  Asignación  con  'let'  se  permite  también  cambiar  las
propiedades de la misma

class Enemigo {
    let nombre:String
    var fuerza:Int
    var vida:Int
}

class Orco:Enemigo {
}
```

Structs

```
// Structs - se realiza una copia de esta en cada asignación.
// Asignación con 'let' no permite cambiar las propiedades de la
misma

struct valoresVitales {
    var fuerza:Int
    var vida:Int
}

class Enemigo {
    let nombre:String
    let vital:valoresVitales

    init(nombre:String, vital:valoresVitales){
    self.nombre = nombre
    self.vital = vital
    }
}

// Crear objeto de estructura e instancia de la clase con ese
objeto
var vitalUruk = valoresVitales(30,50)
let urukHai = Enemigo("Shagrat", vitalUruk)
```

Inicializadores

Podemos diferenciar inicializadores designados e inicializadores de conveniencia. Los inicializadores designados inicializan los parámetros de la clase. Los inicializadores de conveniencia permiten inicializar otros parámetros aunque no sean de la clase, estos no pueden llamar a la superior *super.init* si no a la de la propia clase *self.init*

```
// Inicializadores Designados
init(nombre:String, vital:valoresVitales){
    self.nombre = nombre
    self.vital = vital
}

// Inicializadores de Conveniencia
convenience init(name:String, fuerza:Int, vida:Int){
    var vitalUruk = valoresVitales(fuerza:fuerza, vida:vida)
    self.init(nombre:name, vital:vitalUruk)
}

// Crear instancia fuera de la clase
let urukHai = Enemigo(name:“Shagrat”, fuerza:30, vida:50)
```

Herencia de Clases

A continuación se propone un ejemplo simple de la herencia de clases

```
class Orco:Enemigo {
    let clan:String

    init(nombre:String, vital:valoresVitales, clan:String){
        self.clan = clan
        super.init(nombre:nombre, vital:vital) // ya que no es
de conveniencia
    }

    convenience   init(nombre:String,   fuerza:Int,   vida:Int,
clan:String){
        var vitalOrco = valoresVitales(fuerza:fuerza, vida:vida)
        self.init(nombre:nombre, vital:vitalOrco, clan:clan)
    }
}
```

Extensiones

También se pueden extender las funciones de una clase. Este método se utiliza también para mantener un código limpio extendiendo clases en diferentes archivos Swift.

```
// Extensiones: Clases

extension Enemigo {
    var valia:Int {
    return (self.vital.fuerza * self.vital.vida)
    }
}

let orcoEnemigo = Orco(nombre:"Thrum", fuerza:20, vida:15, clan:"Skullsplitter")
orcoEnemigo.valia // muestra esa operación
```

Funciones

La peculiaridad que tienen las funciones es que no existe la inferencia de tipos (reconocimiento automático de tipo de variable) en los parámetros de entrada

```
// Tipo devuelto mediante '->'
func tipoDispositivo() -> (){ // Podemos usar también -> Void
}
func tipoDispositivo(size:Int) -> Bool {
    return true
}
func tipoDispositivo(width:Float, height:Float = 3.4) -> (Int, String) {
    return (Int(height), "i")
}
tipoDispositivo(32.7, height:6.5) // El parámetro por defecto debe ser llamado por su nombre

func tipoDispositivo(width:Float, height:Float) -> (valor:Int, cadena:String) {
    return (3, "i")
}

tipoDispositivo.0 // tipoDispositivo.valor
tipoDispositivo.1 // tipoDispositivo.cadena
```

Enumeraciones

Se usa *enum* para crear una enumeración. Al igual que las clases y todos los otros tipos, las enumeraciones pueden tener métodos asociados en ellos.

```
enum deviceType {
    case   iPhone4s,   iPhone5,   iPhone6,   iPhone6Plus,   iPad,
iPadRetina, iUnknown
}

var device = .iPhone5

// DEMO: Función que devuelve el tipo de dispositivo que estamos
usando

func iDevice() -> deviceType {
    let height = UIScreen.mainScreen().bounds.size.height
    var device:deviceType

    switch height {
        case 480: device = .iPhone4s
        case 560: device = .iPhone5
        case 667: device = .iPhone6
        case 736: device = .iPhone6Plus
        case 1024:

            if(UIScreen.mainScreen().scale == 2.0){
                device = .iPadRetina
            } else{
                device = .iPad
            }
        default: device = .iUnknown
    }

    return device
}
```

Polimorfismo

El polimorfismo es la capacidad de una función de ser definida de diferentes formas y tener diferentes comportamientos en función de cómo es invocada. Podemos definir 2 funciones con el mismo nombre y el asistente de código nos dará a elegir.

```
func sumarPuntos(c1:CGPoint, c2:CGPoint) -> CGPoint {
    return CGPointMake(c1.x + c2.x, c1.y + c2.y)
}
```

```
func    sumarPuntos(c1x:CGFloat,    c1y:CGFloat,    c2x:CGFloat,
c2y:CGFloat) -> CGPoint {
    return CGPointMake(c1x + c2x, c1y + c2y)
}
```

Tuplas

Es una agrupación de varios tipos de datos. Son ideales para crear tipos compuestos rápidos y fáciles. En muchos casos, es una mejor opción usar un *struct* con mayor seguridad de tipos y una mayor funcionalidad.

```
var film = (“Interestellar”, 2014)
film.0
film.1

// Mejor nombrar los componentes para no caer en índices que no
existen
var film = (pelicula:"Interestelar", estreno:2014)
film.pelicula
film.estreno

// También se puede construir la tupla con asignación de tipos
// Para este caso hay que nombrar los componentes en los tipos
var film: (pelicula:String, estreno:Double, valoracion:Float) =
("Interestellar", 2014, 10.0)

// Deconstrucción de la tupla: Asignación a variables
let (pelicula, estreno, valoracion) = film

// Si solo nos interesa un valor, creamos un placeholder o tipo
vacio que el compilador ignora
let (pelicula, _, _) = film
```

1.5 SOLUCIONES ESCOGIDAS

Vistas las distintas opciones, decido realizar una aplicación enfocada a la programación de los diferentes canales de televisión de forma nativa ya que no va a ser necesaria una actualización constante de la interfaz de la aplicación.

Ésta se servirá directamente de diferentes servidores web para recoger el contenido. El objetivo será desarrollar la aplicación para dispositivos iOS ya que ofrecen más posibilidades de comercialización y venta, además las previsiones de futuro de esta plataforma son muy buenas con la introducción de nuevos dispositivos como Apple Watch o el previsible Apple Car.

Habiendo escogida esta plataforma, la opción de comercialización escogida es la de una aplicación Freemium ya que los contenidos ofrecidos son libres y están a disposición del usuario a través de internet. La aplicación ordenará estos resultados de forma lógica para ofrecer un mejor servicio al usuario, por ello no creo conveniente poner a la venta una aplicación que ofrece un servicio gratuito. En cambio, sí que es posible poner un sistema de publicidad dentro de alguna vista para obtener ingresos, y desbloquear a través de pago (in-app) de algún tipo de funcionalidad extra como: avisos a través de notificaciones.

2. MATERIAL Y MÉTODOS

2.1 ¿QUÉ APP HE ELEGIDO?

2.1.1 MOTIVACIÓN

La forma de ver la televisión está cambiando en los últimos años. Cada vez es más normal que los telespectadores naveguen en internet o comenten en las redes sociales a la vez que disfrutan de contenidos televisivos. La popularidad y evolución de los smartphones y tabletas hace que estos dispositivos se conviertan en una segunda pantalla complementaria a la televisión.

Los telespectadores se están haciendo cada vez más activos y participativos lejos de su hasta ahora papel de receptor pasivo y su medio de comunicación cada vez más usado son los dispositivos móviles. Un estudio realizado por la compañía Nielsen revela que uno de cada tres usuarios de Twitter ha enviado al menos un “tuit” sobre lo que estaba viendo en ese momento en televisión.

Estos datos recopilados por la compañía en 2012 para Estados Unidos, sirven como referencia a la creciente tendencia de la incorporación de las nuevas tecnologías a los contenidos televisivos. Revelan un crecimiento del 27% respecto a los cinco meses anteriores. El informe afirma que un 38% de los usuarios de smartphones y un 41% de tablets navega por internet a la vez que ve la televisión. Se confirma entonces que la “doble pantalla” es cada vez más común, de hecho, las cadenas de televisión ya incorporan hashtags para dirigir el contenido en Twitter.

Según un informe proporcionado por Barlovento Comunicación con datos de Kantar Media, el pasado mes de Enero de 2015 alcanzó los 263 minutos por persona. Además el aumento del número de canales de televisión tanto en la TDT como canales de pago, hacen que los usuarios demanden una aplicación

que organice todo el contenido que proporciona la parrilla de televisión y así poder estar al tanto y organizar la lista de series y películas que quieres ver, saber la programación en cada instante y estar informado de los detalles de las emisiones de los diferentes programas que se emiten en televisión.

Como consecuencia de todas estas necesidades vi la oportunidad de crear una aplicación que organice todo el contenido de la televisión y filtre las películas, series, programas y documentales en base a filtros que el usuario puede usar dependiendo de sus necesidades.

2.1.2 OBJETIVOS GENERALES

Entre los objetivos de esta aplicación, resulta interesante usar el nuevo lenguaje de programación SWIFT que Apple ha desarrollado para la plataforma iOS. La idea es comprobar las ventajas que supone este lenguaje, y a la vez, poner en funcionamiento una aplicación que sirva a los usuarios cuando necesiten consultar u organizar la parrilla de televisión.

La aplicación obtendrá la información de la programación y detalles de las diferentes películas, series, programas o documentales de diferentes servicios web externos. Para ello, la aplicación tendrá que conectar con estos servicios a través de internet.

Además, haremos uso de la técnica web-scrapping para recoger el contenido de páginas interesantes y mostrarlo de forma nativa dentro de la aplicación.

2.1.3 OBJETIVOS ESPECÍFICOS

Podemos resumir los diferentes objetivos en los puntos siguientes:

- Desarrollar esta aplicación en lenguaje SWIFT y probar las ventajas que tiene frente a otros lenguajes de programación iOS como OBJECTIVE-C
- Usar aplicaciones externas para el prototipado inicial de la aplicación, de forma que se pueda obtener un modelo inicial con los bocetos de las diferentes vistas.
- Utilizar una herramienta de control de versiones GIT para evitar imprevistos de borrado, errores o cambios de diseño.
- Construir una herramienta bien estructurada apta para usuarios convencionales, sin muchos conocimientos de uso de dispositivos móviles, que puedan moverse fácilmente por la interfaz sin ayuda de ningún tipo. Cuidar el diseño y obtener una aplicación atractiva y manejable para el usuario.
- Conectar con bases de datos externas para obtener toda la información necesaria para desarrollar el sistema.
- La aplicación, no debe precisar continuamente de una conexión a internet para su uso, si no que debe guardar o cachear la información para su utilización posterior cuando no haya conexión.
- Debe vincular de alguna forma con redes sociales, para poder compartir y recomendar el contenido con otros usuarios. Esta será una medida de marketing que debe integrar la propia aplicación.
- Posibilidad de seleccionar los canales que tiene el usuario para uso posterior en diferentes filtros de la aplicación. De esta forma se mostrará sólo el contenido de los canales que el usuario le interese.
- Mostrar las películas, series, programas y documentales que van a emitir próximamente en televisión y aplicar filtros a los resultados encontrados como:
    - Ver sólo el contenido de mis canales o de otros canales
    - Seleccionar los años de emisión de las películas

    - Filtrar por género de las películas: Acción, Drama, Aventura…

- Ver en una pantalla diferente los detalles del programa que van a emitir como la carátula, sinopsis, valoraciones de usuarios, actores, presupuesto, año de emisión, tráiler
- Desarrollar una vista en la que se pueda buscar otras emisiones de la misma película, serie, programa o documental en la parrilla de televisión para los próximos días.
- Opciones en la vista detallada del programa para compartir o publicar en redes sociales.
- Habilitar un sistema de notificaciones o avisos para que la aplicación recuerde al usuario 5 minutos antes de comenzar mediante una notificación, que el programa va a emitirse y el canal donde lo van a hacer.
- Ordenar en forma de calendario todos estos avisos o notificaciones que el usuario ha ido añadiendo en su dispositivo.
- Mostrar la programación de televisión por tiempos y por canales:
    - Por tiempos: Se incluirán en una tabla los canales con los programas que se están emitiendo en ese mismo instante de tiempo (que puede ser modificado por el usuario deslizando la tabla horizontalmente)
    - Por canales: Se visualizará toda la programación de un solo canal resaltando el programa que se está emitiendo en ese momento.
- La aplicación debe ser capaz, en una vista o sección separada, de realizar web-scrapping de una página en internet y recogiendo esos datos, mostrar de forma nativa en la aplicación el contenido interesante de datos de audiencias.
- Añadir en la aplicación un sistema de recogida de valoraciones de usuarios, para vincularlo a la AppStore y recoger buenas críticas para incrementar las ventas. Usar un sistema de atención al cliente, para que hagan llegar sus dudas antes de realizar una mala crítica en la AppStore.

- Realizar pruebas de uso con otros usuarios, con la ayuda de aplicaciones externas como Crashlytics, para comprobar el buen funcionamiento de la aplicación y posibles mejoras.
- Incluir estadísticas, con la ayuda de librerías externas como Google Analytics, para comprobar el flujo de visitas dentro de la aplicación, de esta manera, se podrá observar qué personas, cómo y cuando utilizan la aplicación una vez se publique en la AppStore.

2.1.4 DESCRIPCIÓN

La aplicación MyZap muestra la guía de televisión, realiza búsquedas en base a filtros definidos por el usuario, crea avisos y notificaciones para los diferentes programas, ordena y organiza las películas o series pendientes de ver y analiza las audiencias para toda la semana.

2.1.5 ALCANCE

El proyecto se considerará completo cuando el usuario sea capaz de filtrar y obtener la información deseada de la programación de la televisión o audiencias y crear notificaciones o avisos de los programas que le interese. De la misma manera, también deberá ser capaz de navegar por la aplicación de forma clara e intuitiva.

2.1.6 RIESGOS

Durante la realización de este trabajo pueden surgir diferentes imprevistos, que puedan retrasar la fecha de finalización. A continuación se explicarán los riesgos y se analizarán las posibles soluciones.

2.1.6.1 DETECCIÓN DE POSIBLES RIESGOS

- Insuficiente conocimiento en alguna de las áreas:. En un principio, la falta de conocimiento en alguna de las áreas de desarrollo hace más lento todo el proceso.
- Posibilidad de errores en el entorno de desarrollo: El desarrollo para plataformas móviles es muy cambiante, el lenguaje está en su versión 1.2 y todavía está en desarrollo, con lo que es posible encontrarse algunos errores de lenguaje. Además pueden encontrarse errores en el software XCODE de programación a pesar de las continuas actualizaciones por la salida de este nuevo lenguaje.
- Errores en el diseño del proyecto: Se intentará realizar un diseño correcto que permita al usuario un movimiento intuitivo por dentro de la aplicación, pero puede ser que este diseño conlleve una serie de errores al no tener una percepción externa de otros usuarios. Para lo que al desarrollador le puede parecer una acción sencilla, para el usuario es posible que no lo sea.

2.1.6.2 PLANES DE CONTINGENCIA

- Insuficiente conocimiento en alguna de las áreas: Dicho problema se tendrá en cuenta antes del comienzo de desarrollo del mismo y se dedicará una parte del tiempo a adquirir nuevos conocimientos.
- Posibilidad de errores en el entorno de desarrollo: Este riesgo es poco probable debido a la cantidad de actualizaciones del lenguaje y software de programación usado para el desarrollo. En caso de ocurrir, sería de gran impacto con lo que se recomienda el uso de control de versiones GIT o sistemas de guardado en la nube como DropBox o Google Drive.
- Errores en el diseño del proyecto: En el caso de que esto suceda, se tendrá en cuenta y habrá que replantear el diseño para corregir de inmediato ya que puede llevar al abandono del usuario y una mala crítica en la tienda AppStore, esto supondría una repercusión directa en las ventas de la aplicación.

2.1.7 TECNOLOGÍAS A USAR

Para realizar esta aplicación necesitamos del uso de diferentes tecnologías entre ellas:

XCode: Este es el software utilizado para el desarrollo de la aplicación. Para ello necesitaremos un ordenador Mac.

Photoshop e Illustrator: Estos dos son los software más utilizados junto con XCode para el diseño de la aplicación. Es importante para el prototipado de la aplicación y para el diseño final. Su uso es importante en la creación de vistas, botones y demás objetos de la aplicación.

Control de versiones GIT: En este tipo de desarrollo avanzado de aplicaciones, es muy interesante el uso de control de versiones GIT, de forma que podamos mantener en la nube las versiones desarrolladas con detalle de los cambios realizados para llevar un control y mantener segura la aplicación en un servidor externo. De esta forma, con este software GIT tenemos controlada la aplicación ante cualquier imprevisto, error o borrado inesperado de código. El control de versiones en GIT también nos permite compartir el código de nuestra aplicación en la red, de forma que otros usuarios, de forma pública, puedan incorporar ideas y nuevo código al proyecto realizando diferentes versiones que puedan resultar interesantes para una versión final estable.

Servicios Web: El uso de servicios web es primordial para el contenido de la aplicación. A través de estos servicios se recogerán los datos requeridos para mostrar, como la guía de televisión, detalles de películas, información de audiencias u otros datos que sean interesantes.

Beta-Testing y control de errores: La tecnologías de control de errores es importante antes de poner una aplicación a la venta en la App Store. Es importante, comprobar que el funcionamiento de la aplicación es correcta, y

que va a pasar los filtros que Apple requiere para sus aplicaciones de la App Store. Es importante también, de cara a la venta de la aplicación, la usabilidad y la funcionalidad de la misma. Controlar estos aspectos son clave para que una aplicación funcione, por ello, esta fase de Beta-Testing es un paso importante previo a su publicación.

Análisis estadísticos: Para ello, usaremos plataformas de análisis de visitas y de uso de la aplicación. Hay distintas aplicaciones que se dedican al análisis detallado de estos datos, pero principalmente destaca el SDK de Google Analytics. Esta API permite un control absoluto sobre la visita de un usuario a la aplicación en tiempo real. Más adelante, se detallará su funcionamiento. Este análisis, nos permite saber las intenciones del usuario y los posibles errores que hemos cometido en la funcionalidad de la aplicación para una posterior revisión y actualización.

2.1.8 CONOCIMIENTOS REQUERIDOS

Lenguaje Swift: Es imprescindible el conocimiento previo del lenguaje. En este caso, ha sido necesario realizar diferentes cursos, tutoriales y ejemplos previos para realizar un amplio estudio de este lenguaje partiendo desde cero y sus diferencias con Objective-C que era el lenguaje ya conocido.

Es importante también saber usar un patrón de diseño Modelo-Vista-Controlador para el desarrollo de la aplicación. Esto nos ayudará a obtener una programación estructurada.

XCode: También son necesarios conocimientos de la plataforma de desarrollo en su última versión, pues es sobre esta donde vamos a crear la aplicación. Para este proyecto no ha sido necesario el aprendizaje ya que se poseían conocimientos previos de este software en sus versiones anteriores.

Photoshop e Illustrator: El conocimiento del uso de estas tecnologías es importante, ya que a pesar que podemos obtener recursos e imágenes de la red, siempre necesitaremos realizar algún retoque o crear nuestros propios diseños. Debido al conocimiento adquirido en la creación de aplicaciones previas a este proyecto, no ha sido necesario el aprendizaje, basta con un conocimiento medio de esta tecnología.

Control de versiones GIT: Para el uso de control de versiones GIT privadas, usaremos la plataforma Bitbucket con el software SourceTree como es en el caso de este proyecto y para el control de versiones publicas, se usará la plataforma GitHub. Es importante el conocimiento del software GIT para su instalación y funcionamiento. GIT almacena la información como instantáneas del proyecto a lo largo del tiempo, con lo que es bastante interesante conocer su uso.

Servicios Web: Es necesario conocer en detalle los diferentes servicios web y las APIs que estos ofrecen. Con esto, obtendremos los datos para mostrar qué requerimos en la aplicación. Realizaremos un estudio sobre las API de Miguiatv, Themoviedb y Thetvdb, que son las principales y más completas bases de datos de programación de televisión, películas y series.

Beta-Testing y control de errores: Para ser un buen Beta-Tester de una aplicación es necesario saber ejecutar la aplicación en desarrollo por todos los caminos y probar todas sus funciones. Además, se deberán reportar hallazgos importantes, errores encontrados o posibles mejoras en funcionalidad o diseño. Se debe explicar en detalle qué se debe arreglar y qué deberá mejorar. Es importante encontrar diferentes usuarios para realizar las pruebas.

Para el uso de estas técnicas necesitaremos conocer el uso de la plataforma Crashlytics o Testflight (comprada por Apple).

Análisis estadísticos: El análisis de las visitas anónimas a la aplicación también es una forma de testear la aplicación y ver qué camino toma el usuario frente a alguna acción en concreto. Será bueno el conocimiento sobre tratamiento de datos estadísticos de Google Analytics y el uso de esta plataforma online que nos proporciona estadísticas en tiempo real.

2.2 MATERIAL UTILIZADO

Para comenzar el desarrollo de aplicaciones iOS y en concreto esta aplicación, es necesario cierto material indispensable.

Un Mac: Será necesario disponer de un ordenador Mac, ya que las herramientas usadas para la creación de este tipo de aplicaciones sólo estarán disponibles para el sistema operativo Mac OSX.

iOS SDK: Este SDK incluye por una parte, el entorno de desarrollo de XCode con el que desarrollaremos la aplicación haciendo uso del nuevo lenguaje Swift, y por otra parte el simulador de iPhone incluido en este software para ejecutar nuestras aplicaciones sin necesidad del dispositivo de forma física. Esta herramienta nos permitirá usar diferentes librerías y diferentes interfaces que nos permitirán estructurar y diseñar la aplicación.

2.2.1 IOS DEVELOPER PROGRAM

Apple ofrece a desarrolladores la posibilidad de adquirir una licencia de desarrollador iOS. La razón por la que es realmente importante tener esta licencia es, entre otras cosas, porque podemos probar aplicaciones en el dispositivo iOS físico, y según que versión de licencia se tenga, puedes distribuir la aplicación o tener acceso libre al software beta de Apple que todavía no ha salido al mercado.

Es muy importante realizar estas pruebas de funcionamiento de la aplicación además del simulador en un dispositivo físico real, ya que el simulador que Apple proporciona en su iOS SDK, permite hacer un uso limitado de todas sus funcionalidades. Por ejemplo, no podemos hacer uso de la cámara de fotos o de algunas aplicaciones integradas de nuestro dispositivo o las preferencias de configuración que el dispositivo físico tiene. Por ello, debemos ejecutar la

aplicación en un dispositivo iOS real, para comprobar que el funcionamiento es correcto.

Existen tres tipos de licencias de desarrollo iOS, cada uno de los cuales, nos ofrece una serie de servicios.

iOS Developer Program: Este programa tiene un coste de 99 euros anuales, y permite a los desarrolladores distribuir las aplicaciones en la App Store como personas individuales, como empresa u organización.

iOS Developer Enterprise Program: Este es un programa que tiene un coste asociado de 299 euros anuales y permite desarrollar aplicaciones propias para la distribución interna dentro de una empresa, organización o institución

iOS Developer University Program: Este resulta ser un programa gratuito para instituciones educativas de nivel superior que quieran introducir el desarrollo de iOS en su plan de estudios. Esta, en cambio, tiene bastantes limitaciones, ya que, no permite realizar una distribución Ad Hoc de las aplicaciones ni distribuirlas en la App Store.

	Developer Program	Enterprise Program	University Program
Acceso a las versiones preliminares del iOS SDK	✓	✓	✗
Acceso a las versiones Gold Master del iOS SDK (últimas versiones antes del lanzamiento)	✓	✓	✓
Pruebas en el iPhone, iPad y iPod touch	✓	✓	✓
Soporte técnico a nivel de código	✓	✓	✗
Distribución Ad Hoc (distribucón de la aplicación entre un número limitado de colaboradores)	✓	✓	✗
Distribución en la App Store	✓	✗	✗
Distribución personalizada B2B	✓	✗	✗
iAd Network y iAd producer (plataforma de publicidad de Apple)	✓	✗	✗
Distribución In-house (permitiendo distribuir e instalar aplicaciones sin sincronización por iTunes a los dispositivos registrados con la licencia)	✗	✓	✗
Coste	99$ anuales	299$ anuales	Gratuito
Requisitos	Tarjeta de crédito válida para la compra	Un número válido de DUNS de Dun y Bradstreet (número identificativo de la empresa en el mercado)	El programa universitarios sólo está disponible para las instituciones de educación superior

En este gráfico se muestra una comparativa sobre los distintos programas de desarrollo iOS que se ha comentado anteriormente. Indica qué tipo de programa se necesita para realizar cada una de las funciones.

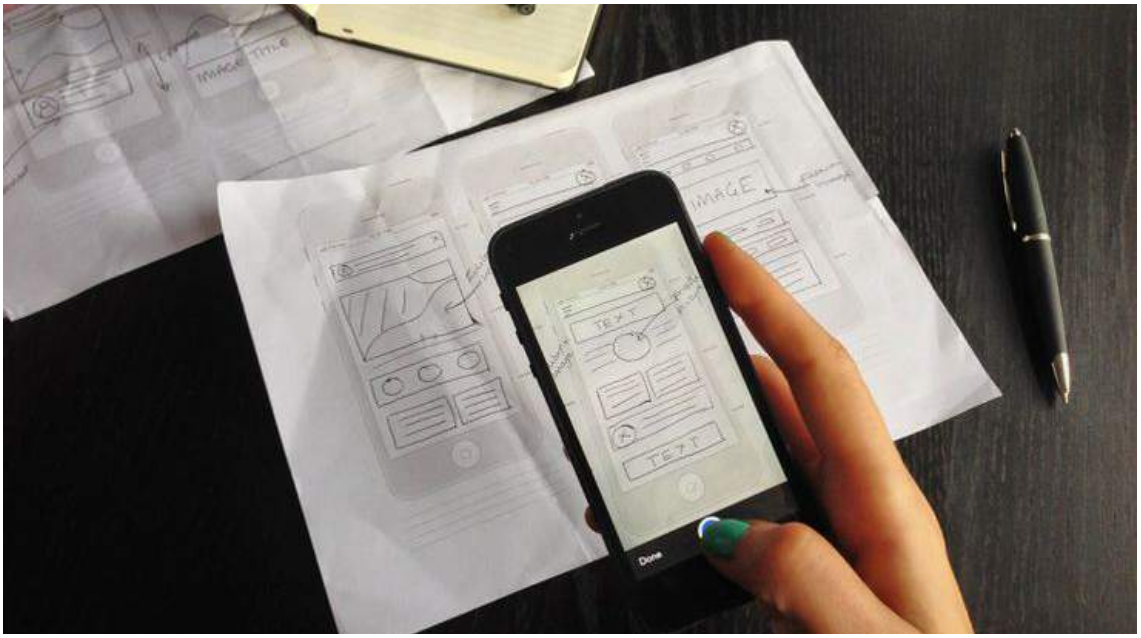
2.3 DESARROLLO DE LA APLICACIÓN

El desarrollo de esta aplicación conlleva una serie de pasos que se deberán realizar para la creación del proyecto.

2.3.1 PROTOTIPADO DEL PROYECTO

Es importante definir correctamente, previo a la programación, un proyecto de aplicación móvil de acuerdo con los objetivos creados, para poder realizar así un presupuesto ajustado en el caso de venta de una aplicación a un cliente, o incluso, para presentar una buena propuesta como desarrollador y exponer el diseño de una futura aplicación.

El prototipado de aplicaciones es una de las técnicas más sencillas para hacer este trabajo y poder previsualizar cómo va a quedar la aplicación a desarrollar en cuestión de diseño sin programar una línea de código. Esto ahorrará mucho tiempo y trabajo de forma que se ahorra en el presupuesto del desarrollo y podremos ver de manera fiel como va a resultar la aplicación.



Ejemplo de realización de prototipos

Posteriormente se podrá modificar, optimizar el diseño, la apariencia y usabilidad para mejorar la aplicación y crear nuevas ramas de desarrollo de la aplicación. Es importante de cara al cliente, si requiere de un cambio en el diseño o vistas de la aplicación, que se realice en esta fase de diseño, pues si se realiza posteriormente, el coste del trabajo será mucho mayor.

Existen multitud de herramientas dedicadas al diseño de prototipos. A continuación se muestran las imágenes prototipo del proyecto realizadas en papel y se detallan algunas de estas herramientas más importantes y fáciles de usar.

2.3.1.1 MONTAJE DE PROTOTIPOS

Para el montaje de estos prototipos se han investigado ciertas herramientas importantes, alguna de ellas gratuitas que ayudarán al diseño de la aplicación.

2.3.1.1.1 PROTOTYPER

Prototyper fue creado por la empresa Justinmind, y es una de las herramientas más potentes para la creación de prototipos de forma interactiva sin necesidad de escribir nada de código. Se puede obtener una versión gratuita y otra versión de pago tanto para Mac como para Windows.

Se usa para el diseño de prototipos tanto para iOS como para Android. Es fácil e intuitivo. Se pueden importar imágenes de Photoshop o Illustrator y realizar un buen montaje del proyecto.

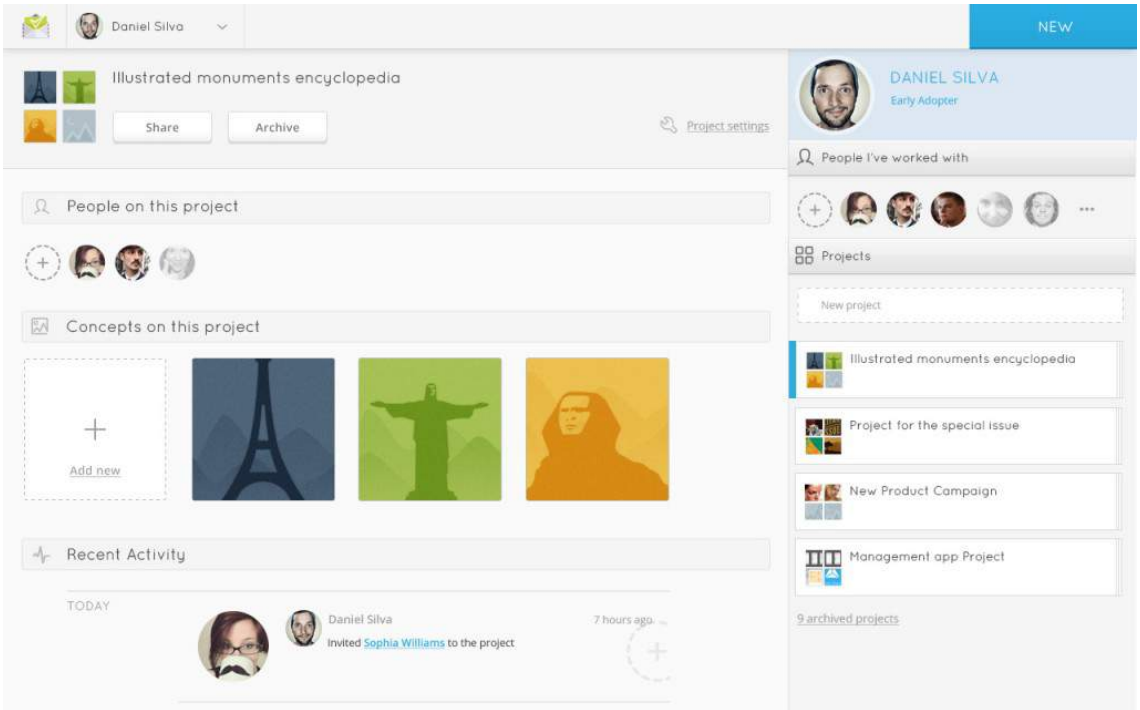
Una vez importadas las imágenes del proyecto se añade la interactividad a las distintas vistas para darle funcionalidad.

2.3.1.1.2 CONCEPT INBOX

Esta herramienta proporciona multitud de recursos para el desarrollador. Mientras la mayor parte de las herramientas está en inglés, ésta se encuentra también en español.

El concepto de esta herramienta es más colaborativo entre un equipo de desarrolladores, diseñadores y clientes, de forma que todos usan una plataforma común para expresar sus ideas. Te permite comentar sobre nuevos avances, realizar anotaciones o señalar sobre el diseño realizado, interactuar con diferentes prototipos y todo esto en tiempo real.

El primer proyecto realizado con esta herramienta es gratuito, para los siguientes debes crear una cuenta de pago dependiendo del número de proyectos que necesites desarrollar.

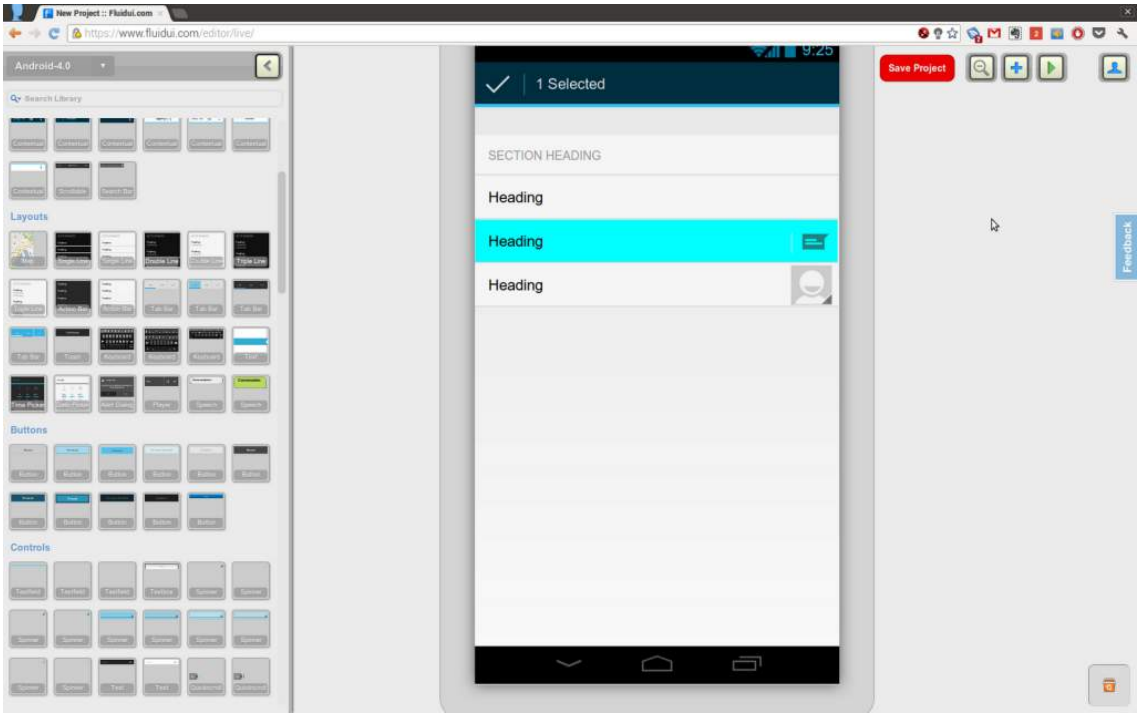


2.3.1.1.3 FLUID UI

Ésta es una herramienta online que puede utilizar cualquiera de forma sencilla e intuitiva. El diseño de la plataforma permite al usuario conocer el entorno y da multitud de opciones para la creación del prototipo. Puedes usarlo para diferentes plataformas iOS o Android.

La versión gratuita permite poder crear un total de diez vistas del prototipo, por lo que puede quedar un poco corto en nuestro caso, ya que necesitamos crear varias vistas al ser una app más compleja. En la versión de pago se pueden contratar planes a partir de 12$ / mes.

Una de las peculiaridades de esta herramienta es que mediante un código QR, puedes escanearlo con un dispositivo y ejecutar el prototipo para ver su funcionalidad real y testear el diseño que se ha creado.



2.3.1.1.4 ANTETYPE

Antetype es una herramienta que busca un nivel más alto de diseño. La calidad de esta es un poco superior a Fluid. El estilo es similar a Keynote y podemos comprarla en la propia App Store ya que sólo está disponible para Mac.

El precio es el mayor inconveniente, ya que, aunque existe una versión gratuita de 30 días, esto no es suficiente y la versión de pago es de 149 euros.

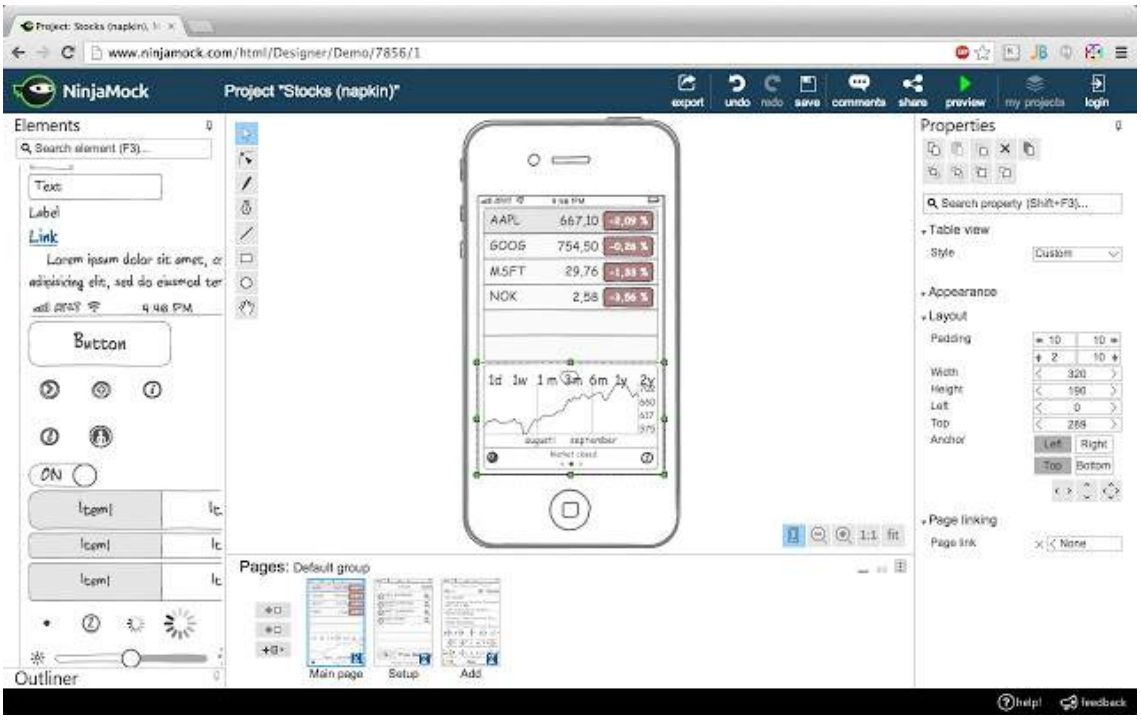
A pesar de esto, es una herramienta muy completa, con ella también se pueden realizar webs y descargar opciones y recursos extras. Adaptado para desarrolladores iOS, Android, Windows Phone…

2.3.1.1.5 NINJAMOCK

NinjaMock es una de las herramientas que más fama tiene por crear muy buenos prototipos de forma muy rápida aunque con algunas limitaciones. Es una plataforma online de diseño de prototipos para apps que incorpora dibujos con acabados a mano alzada. Este tipo de diseño es muy básico, ya que no

muestra una vista real de la aplicación, sino que, muestra un diseño en forma de boceto. No ofrece más que la interacción entre distintas vistas.

Una de la principales ventajas de NinjaMock es que es totalmente gratuita, por lo que puede servir para realizar un primer boceto de la aplicación.



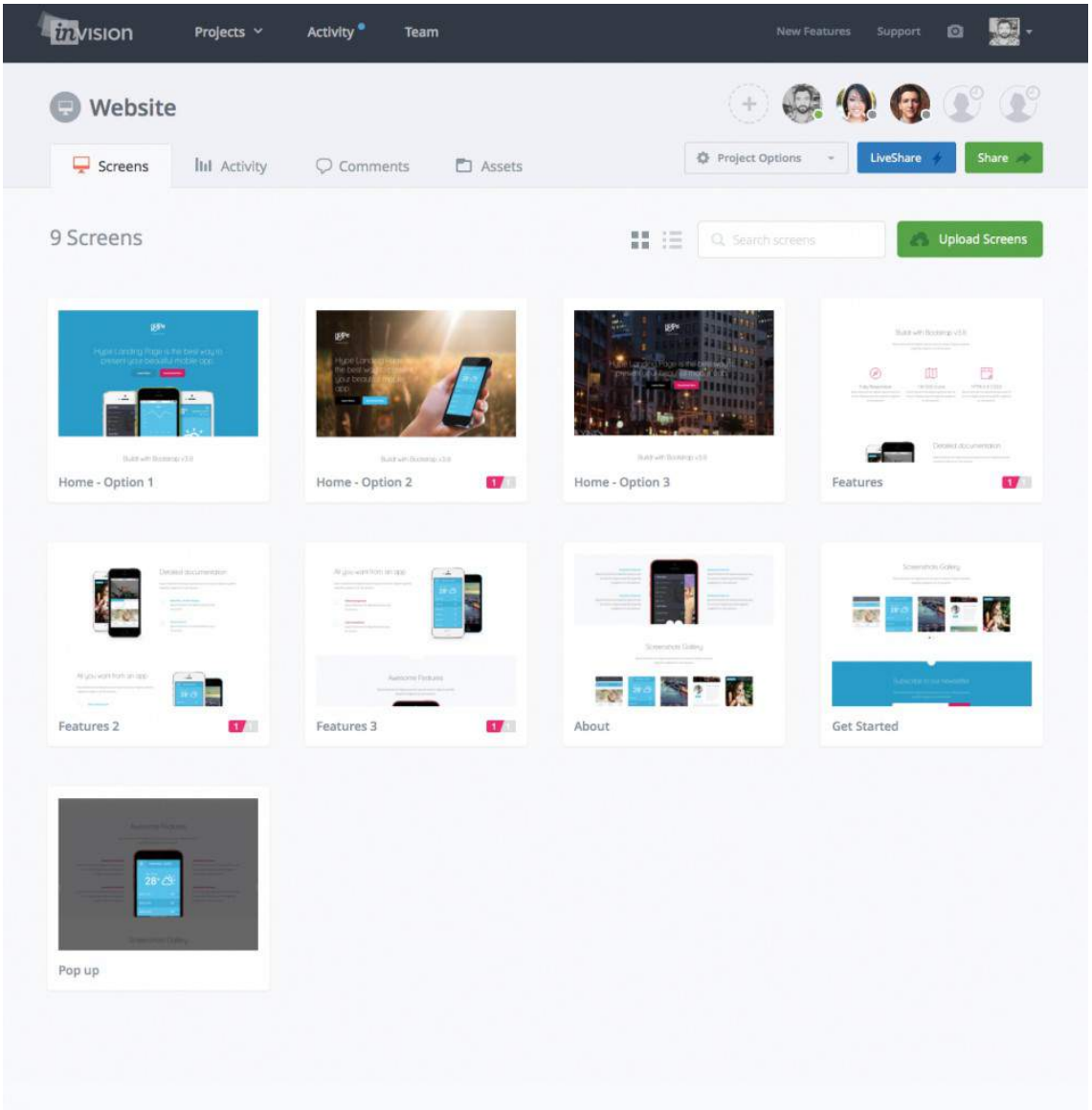
2.3.1.1.6 INVISION

Esta es una de las herramientas más potentes que podemos encontrar actualmente de forma gratuita. Además de poder realizar prototipos completos y bocetos funcionales, se puede trabajar en equipo en un mismo prototipo y en tiempo real.

Existe una pequeña limitación en la versión gratuita que no pone impedimentos a la hora de realizar el prototipos, ya que, sólo podremos crear un prototipo al mismo tiempo. En una versión de pago se pueden realizar varios prototipos a la vez (hasta 100 proyectos al mismo tiempo para proyectos con grandes equipos). El coste aún así es más asequible que con Antetype.

Esta herramienta es muy útil para el desarrollo en pequeños equipos, formados por pocos miembros que trabajan en pocos proyectos al mismo tiempo. Ofrece un entorno perfecto para la colaboración entre desarrolladores, diseñadores y clientes. Y para terminar, existe una versión en entorno Mac donde puedes sincronizar los archivos que sean necesarios.

Esta es sin duda, la mejor opción para el desarrollo de la aplicación de este TFG.



2.3.2 HERRAMIENTAS DE CONTROL DE VERSIONES

El control de versiones, es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones anteriores específicas más adelante.

Te permite revertir archivos a un estado anterior, revertir un proyecto entero, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo…

Existen diferentes métodos de control de versiones, uno de ellos, y el más sencillo pero poco usual, es copiar los archivos a otro directorio (indicando fecha y hora de la copia). Con este método es fácil cometer errores, guardar archivos donde no hay que guardarlos o liarse con los directorios.

Por esto, se usa otro método más simple que programadores desarrollaron hace tiempo, los VCS locales que contienen una simple base de datos que lleva un registro de todos los cambios realizados sobre los archivos.

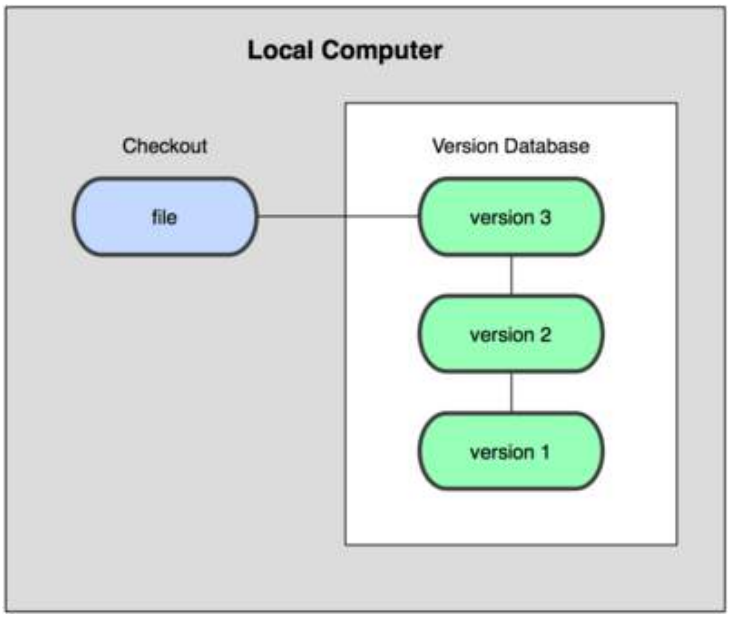


Diagrama de control de versiones local

En el caso de un equipo de desarrolladores, se usan sistemas centralizados de control de versiones (Centralized Version Control System → CVCS). Estos sistemas, como CVS, Subversion y Perforce tienen un servidor donde residen todos los archivos que han sido versionados, y diferentes clientes los descargan desde ese lugar central. Este sistema se ha ido usando durante muchos años.

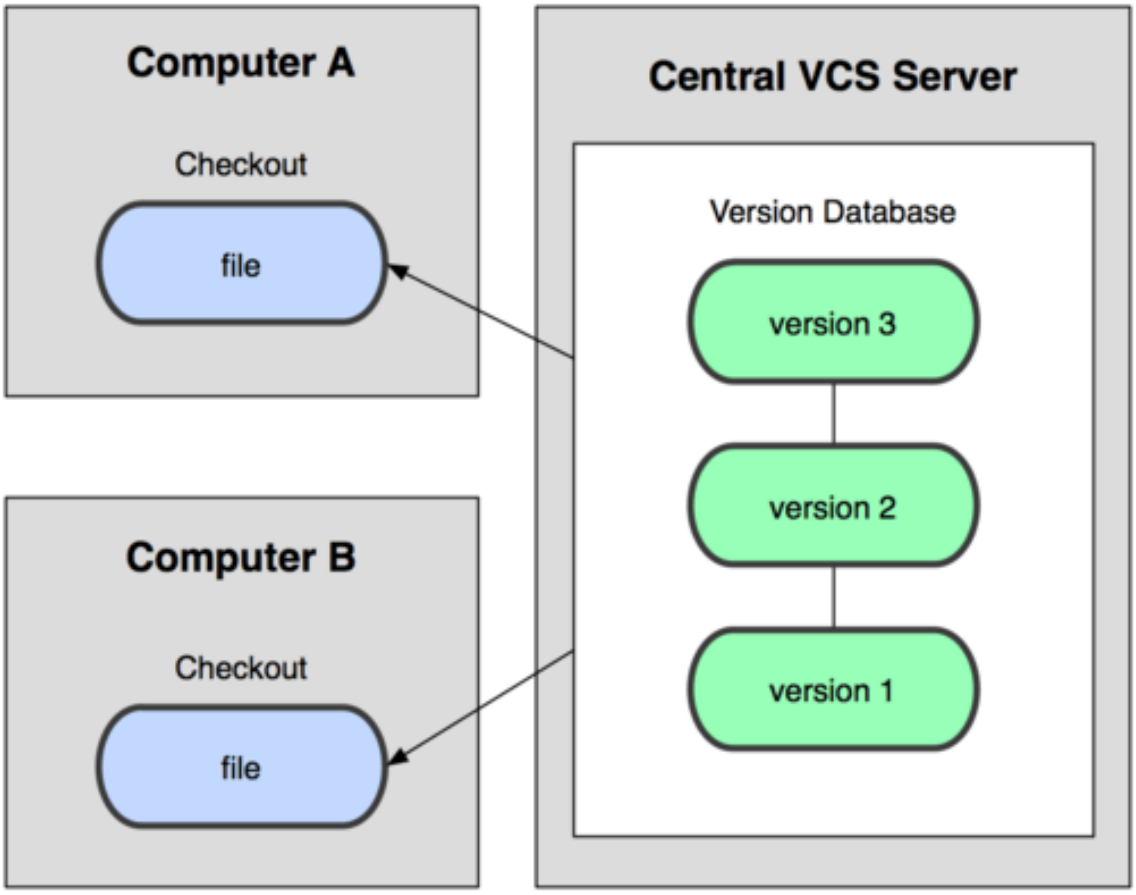


Diagrama de control de versiones centralizado

Con este sistema todo el mundo sabe en qué están trabajando los otros colaboradores del proyecto, y los administradores tienen un control detallado de los que puede hacer cada uno de ellos.

En cambio, este sistema también tiene una desventaja: Todo depende del servidor. Con lo cual, si existe un problema con el servidor, ningún colaborador

puede trabajar o guardar cambios, o incluso, si el disco duro del servidor se corrompe, no tenemos ninguna copia disponible para continuar el proyecto.

En un sistema de control de versiones distribuido no existe este problema, ya que los clientes, no sólo descargan la última instantánea de los archivos, sino que replican completamente el repositorio. Así, si un servidor cae, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Cuando se hace una descarga instantánea, en realidad se hace una copia de seguridad de todos los datos.

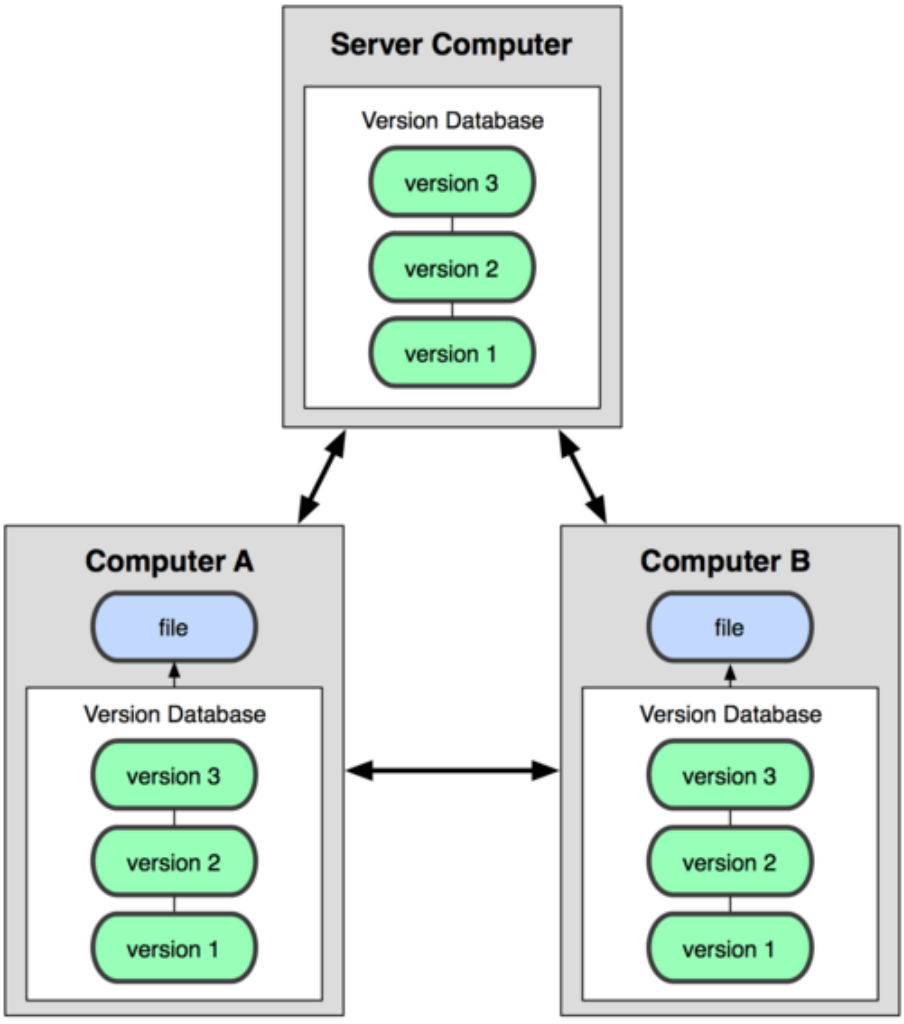


Diagrama de control de versiones distribuido

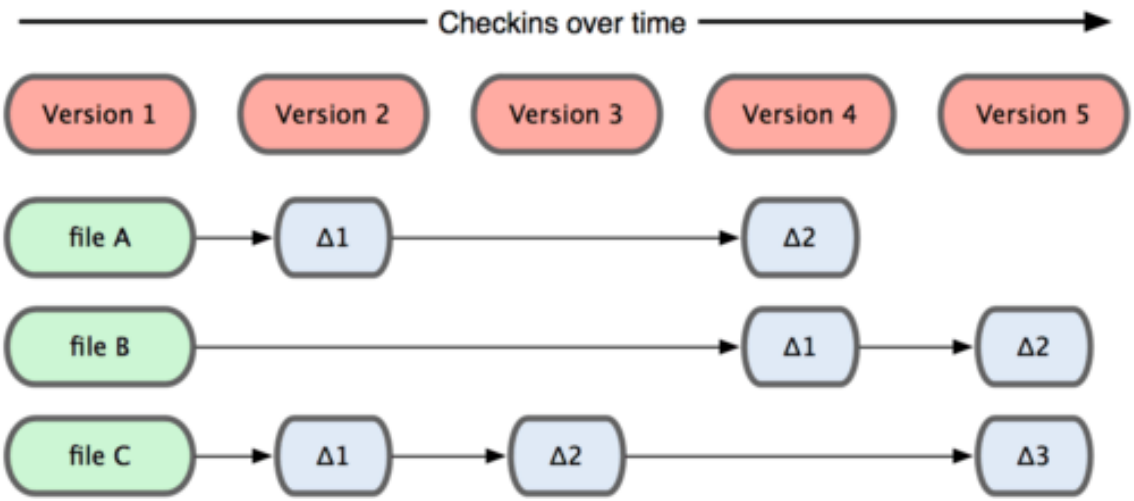
2.3.2.1 GIT

Desde su nacimiento en 2005, GIT ha evolucionado de manera importante, de forma que conserva sus cualidades iniciales:

- Velocidad
- Diseño sencillo
- Fuerte apoyo al desarrollo no lineal (miles de ramas paralelas)
- Completamente distribuido
- Capaz de manejar grandes proyectos de manera eficiente

Existe una diferencia principal por la cual podemos distinguir el sistema GIT de cualquier otro VCS, y es cómo GIT modela sus datos.

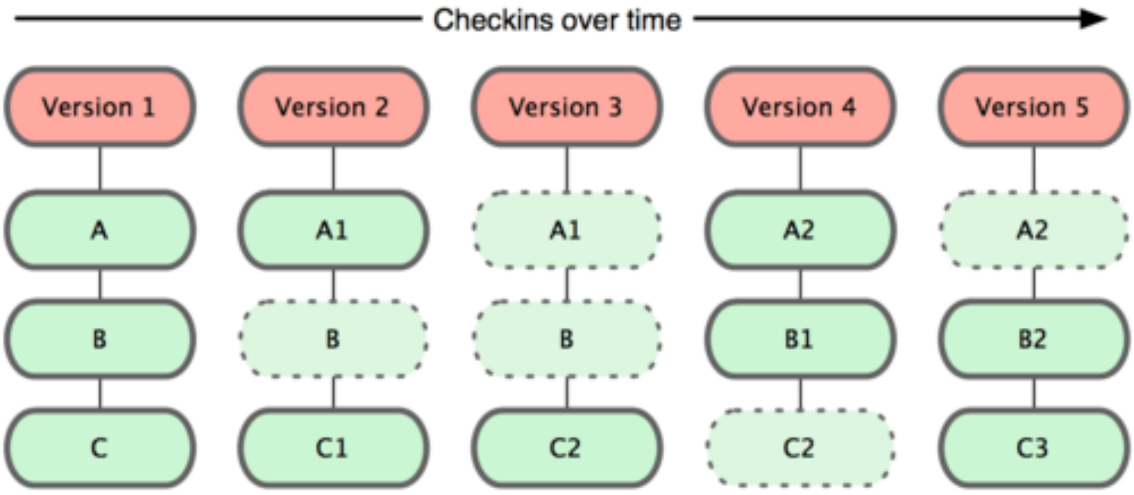
Otros sistemas de VCS, modelan la información como un conjunto de archivos y las modificaciones que se hacen sobre cada uno de ellos a lo largo del tiempo.



Modelo de almacenamiento de datos de otros sistemas VCS

Git modela sus datos de forma distintas, como un conjunto de instantáneas. Cada vez que se confirma un cambio, él básicamente hace una foto del aspecto de los archivos en ese momento y guarda una referencia a esa

instantánea. Si los archivos no se han modificado, Git no almacena el archivo de nuevo, sólo un enlace al archivo anterior que ya tiene almacenado.



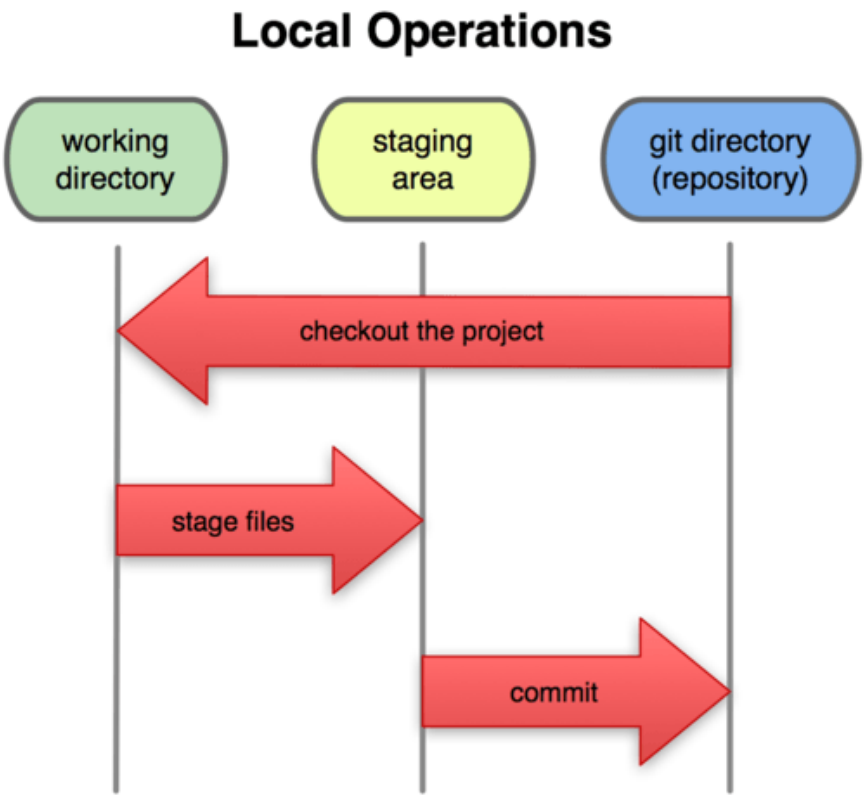
Almacenamiento como instantáneas en GIT

Los beneficios que se obtienen con este sistema se encuentran cuando se realizan ramificaciones de las distintas versiones.

Funcionamiento de GIT

Git tiene tres estados en los que se pueden encontrar los archivos: Confirmado (commited), Modificado (modified) y Preparado (staged).

- Confirmado significa que los datos están almacenados de manera segura en la base de datos local.
- Modificado significa que has modificado el archivo pero no lo has confirmado todavía.
- Preparado significa que has marcado un archivo modificado para que vaya en tu próxima confirmación



El flujo de trabajo en GIT es de la siguiente manera:

- Modificas una serie de archivos en el directorio de trabajo
- Se preparan los archivos, los añades a tu área de preparación
- Confirmas los cambios, toma los archivos tal y como están en el área de preparación y almacena esas instantáneas de manera permanente en el directorio de GIT

Comandos básicos de GIT más utilizados para el TFG

```
$ git init
```

Inicializa y crea los archivos necesarios para usar git en el proyecto que hay dentro del directorio

```
$ git commit –m 'versión inicial del proyecto'
```

Guarda una nueva versión de las modificaciones realizadas en el proyecto

```
$ git clone git://github.com/schacon/grit.git
```

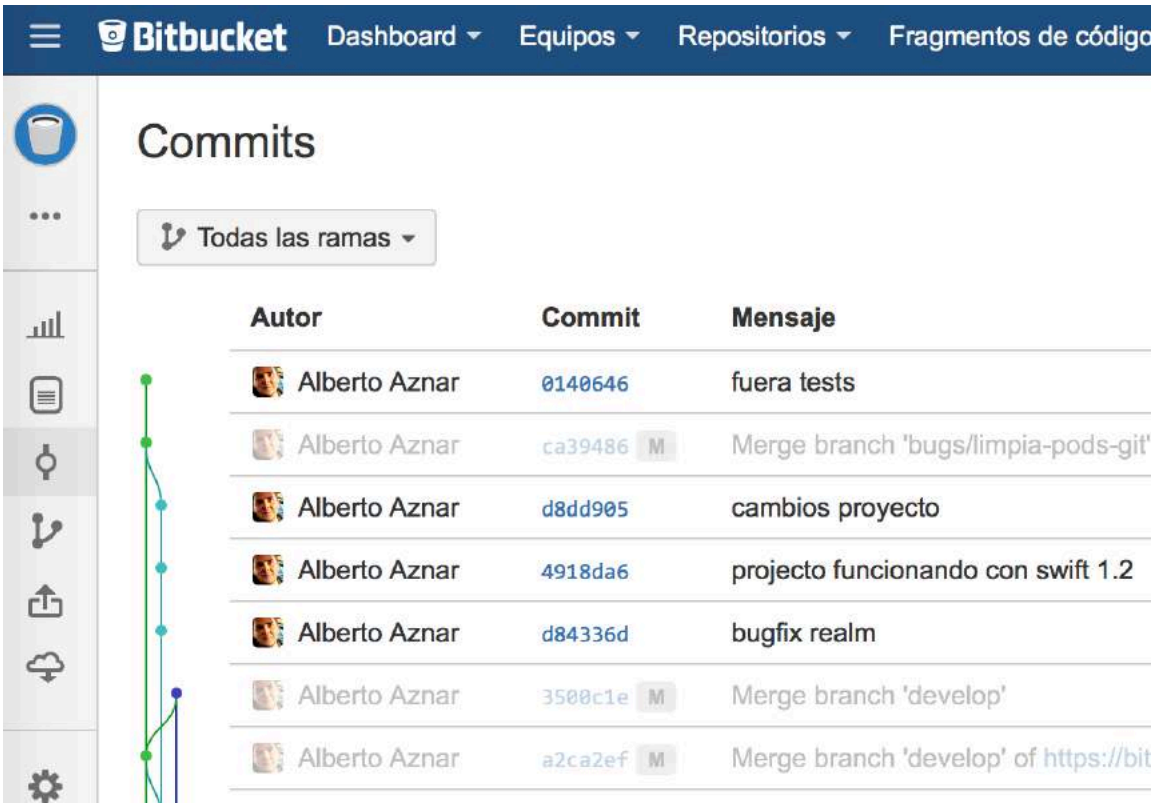
Clona la versión del servidor en nuestro directorio para poder trabajar

```
$ git status
```

Muestra el status del proyecto, si hay alguna modificación pendiente…

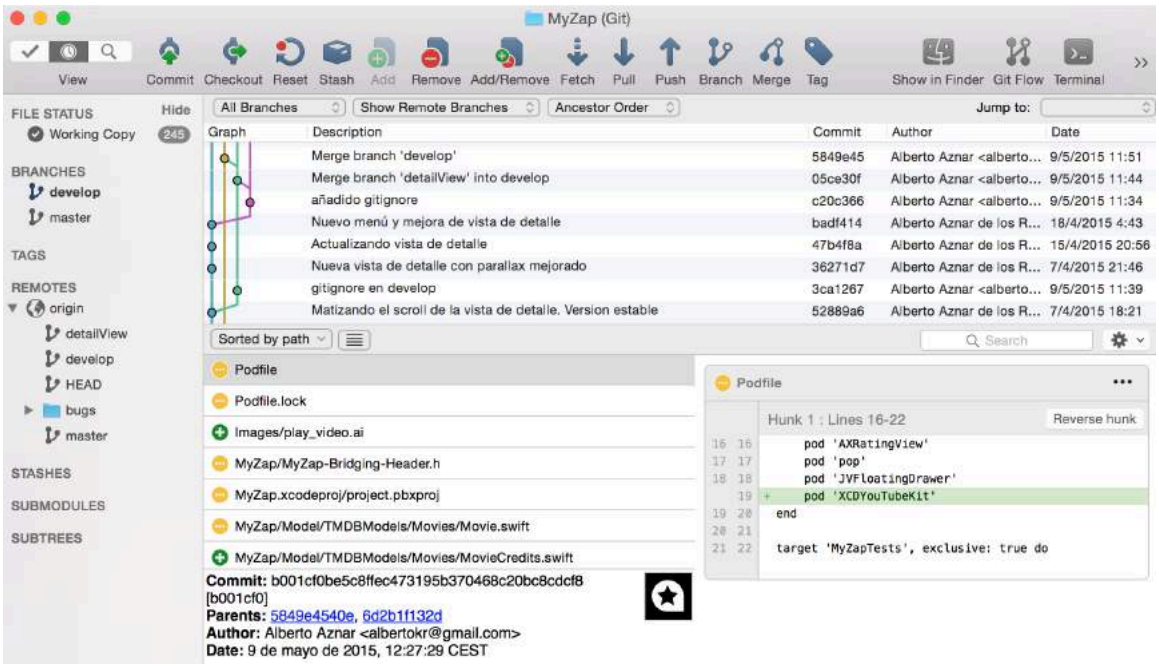
Bitbucket para control de versiones

Para este TFG se ha usado una plataforma de control de versiones llamada Bitbucket. Este es un servicio de alojamiento web para proyectos que utilizan el sistema GIT. Bitbucket tiene planes gratuitos que ofrecen multitud de servicios aunque con un número limitado de repositorios privados. Es similar a Github, pero a diferencia de que Github es totalmente público en su versión gratuita.



SourceForge: Software de Bitbucket

Para esta plataforma, se ha usado el software de control de versiones SourceForge con el que se realizan automáticamente todos los comandos Git sin tener que usar el terminal para ejecutarlos. Directamente con este software se realiza el control y la sincronización con el servidor de manera que se mantiene el proyecto totalmente actualizado y versionado.



2.3.3 PATRÓN DE DISEÑO

Existen muchos patrones de diseño para organizar el código y la forma de programar aplicaciones. Para programar aplicaciones iOS el patrón de diseño más usado es el MVC (Model View Controller)

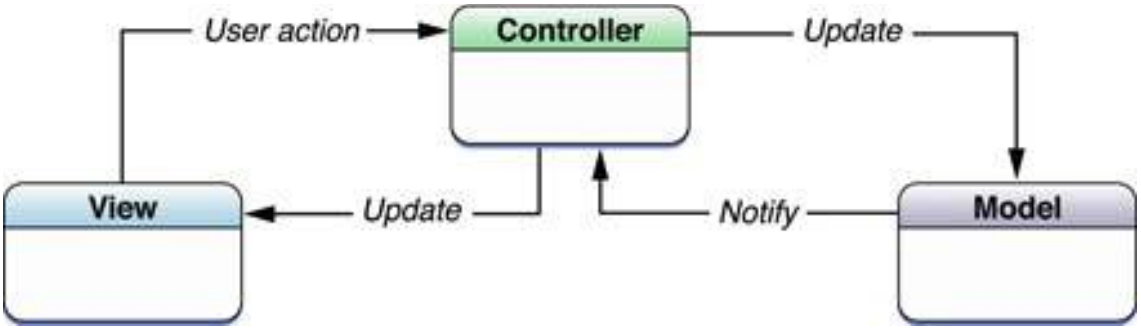
2.3.3.1 MODELO – VISTA – CONTROLADOR

Este patrón de diseño consiste en dividir la aplicación en estas tres capas, de modo que cada una de ellas cumple su función.

La Vista es todo lo que se puede ver en la aplicación. El usuario puede interactuar con esta y dentro de la cual encontramos botones, campos de texto, etiquetas, tablas… De forma general, se incorporan en estas la mayoría de los objetos que son subclases de UIView

El Modelo es la capa que incluye todos los objetos que permiten almacenar y manipular datos. Está separada de la interfaz de usuario y gestiona tanto los accesos a esa información, como consultas o actualizaciones. Envía a la vista la información que se solicita en cada momento para que sea mostrada.

El Controlador es el cerebro del patrón de diseño. El controlador actúa como de intermediario entre la vista y los diferentes modelos de datos. En éste se implementa la lógica de la aplicación. Éste hace funcionar tanto la vista como el modelo de forma sincronizada para mostrar los datos en la aplicación.



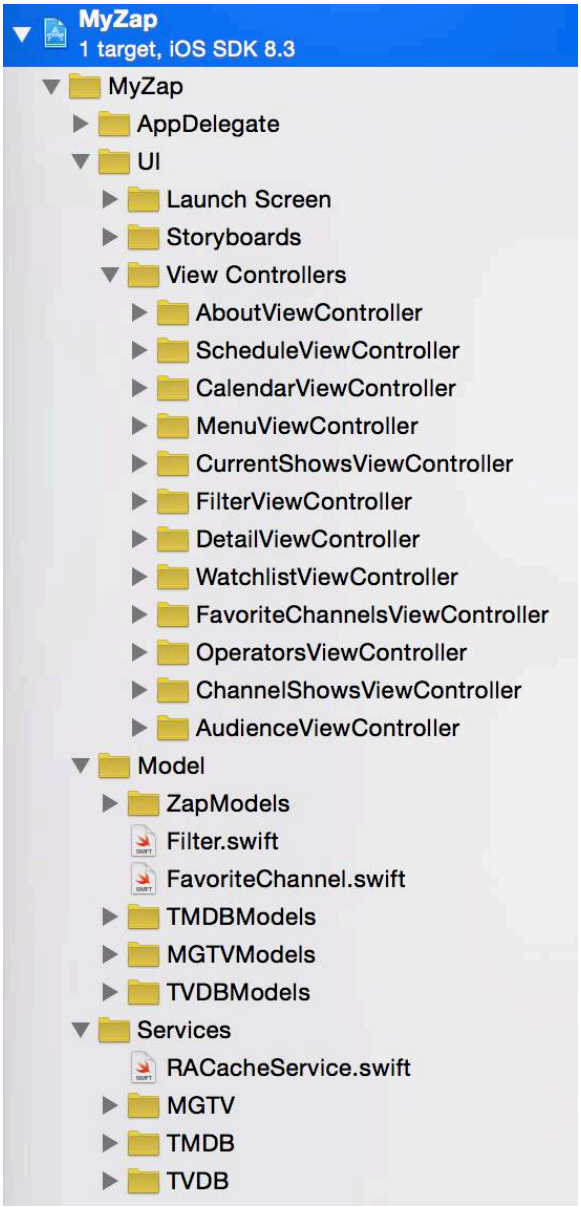
Flujo de interacción de capas

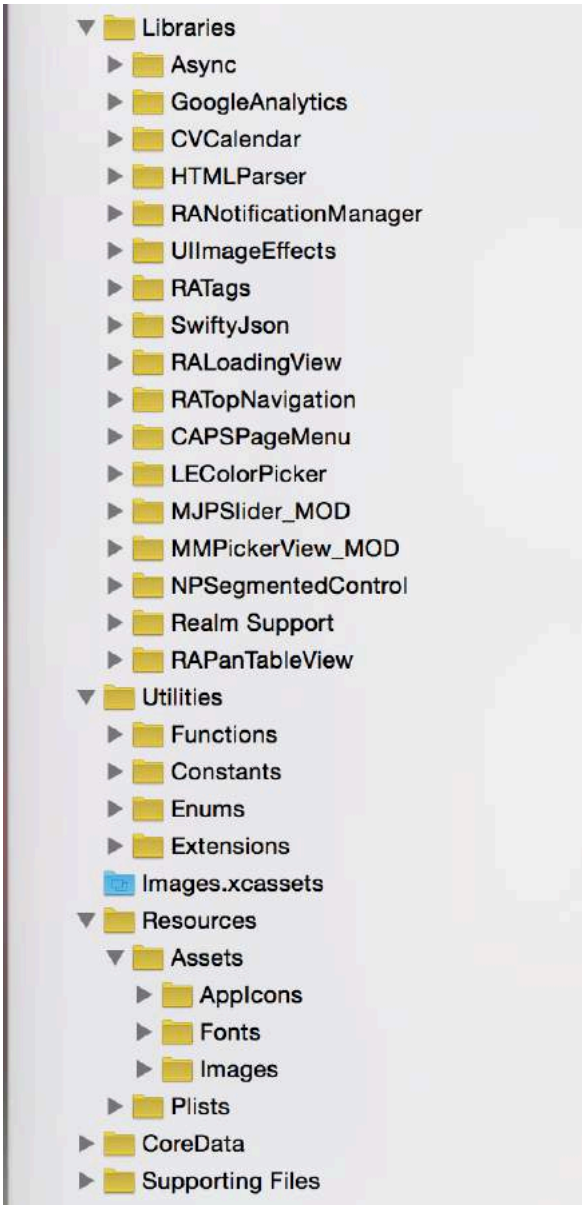
- El usuario interactúa con la vista de alguna forma (pulsa un botón, realiza una acción)
- El controlador recibe esa notificación solicitada por el usuario, normalmente mediante un gestor de eventos o “callback” y realiza las acciones oportunas.
- El controlador actualiza el modelo de forma adecuada a la acción que se solicita.
- El modelo notifica al controlador que se ha actualizado o modificado

- El controlador envía el modelo de datos actualizado a la vista, actualizándola de forma ordenada para mostrar al usuario lo que se ha solicitado.

2.3.3.2 ESQUEMA DEL PROYECTO

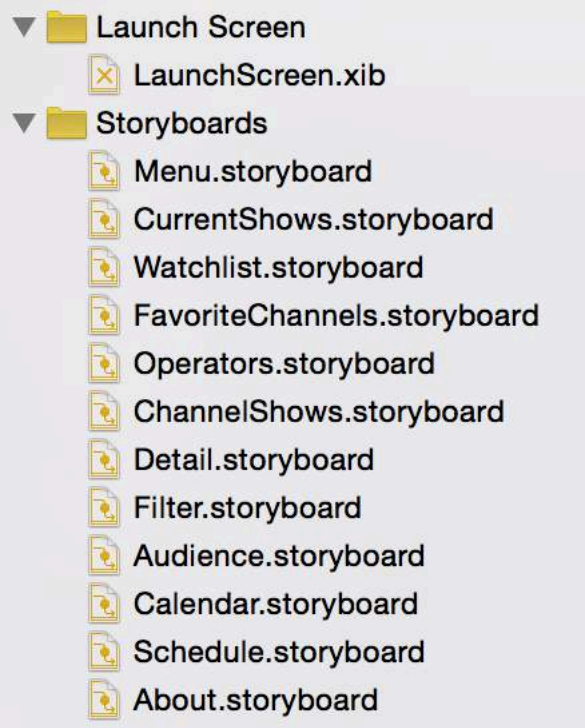
En la aplicación a desarrollar en este TFG se ha organizado el proyecto en XCode teniendo en cuenta el patrón de diseño MVC.





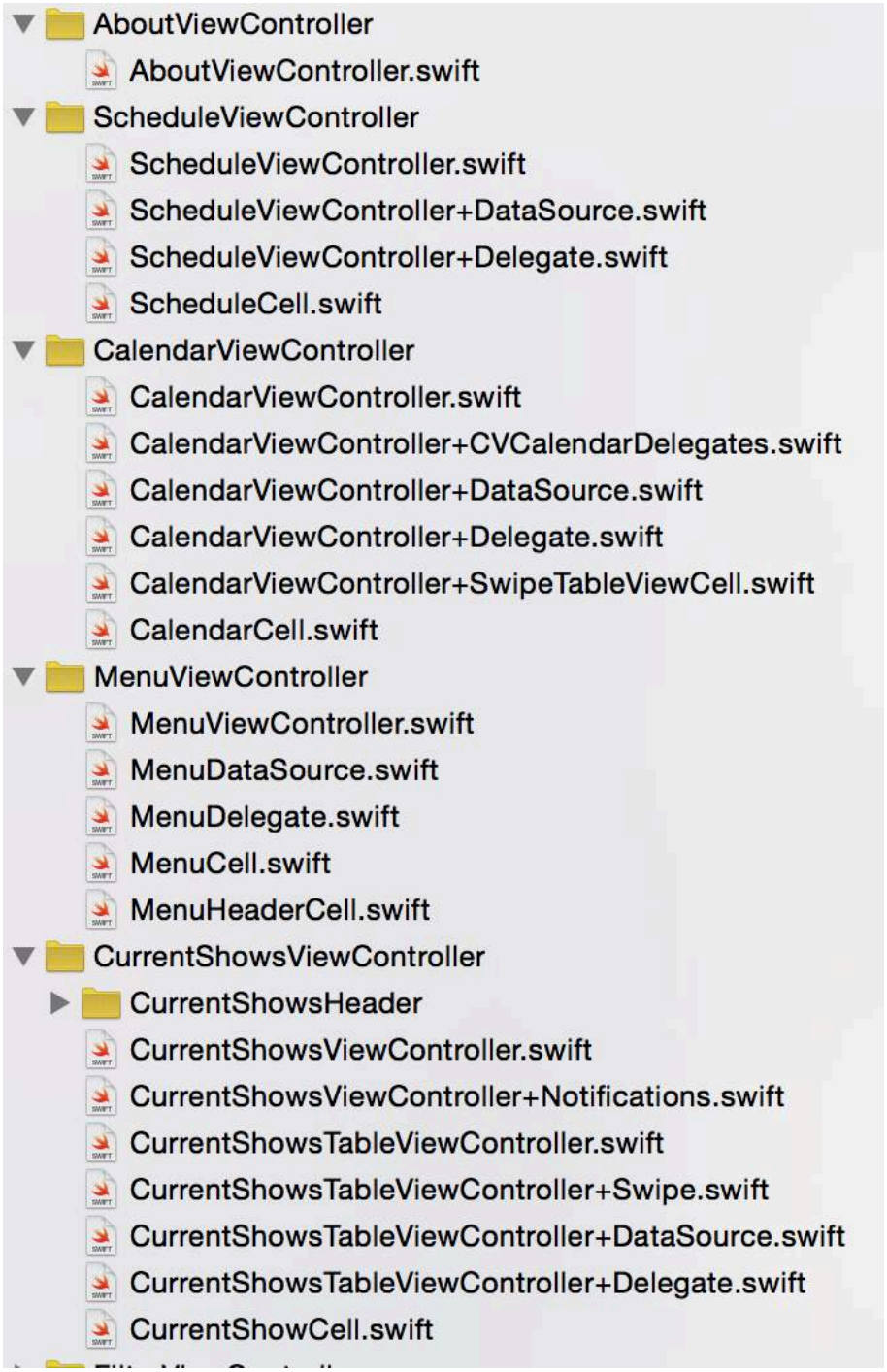
Vistas

Las vistas del proyecto se han realizado en diferentes Storyboards de XCode de forma que cada vista se separa en un archivo diferente.



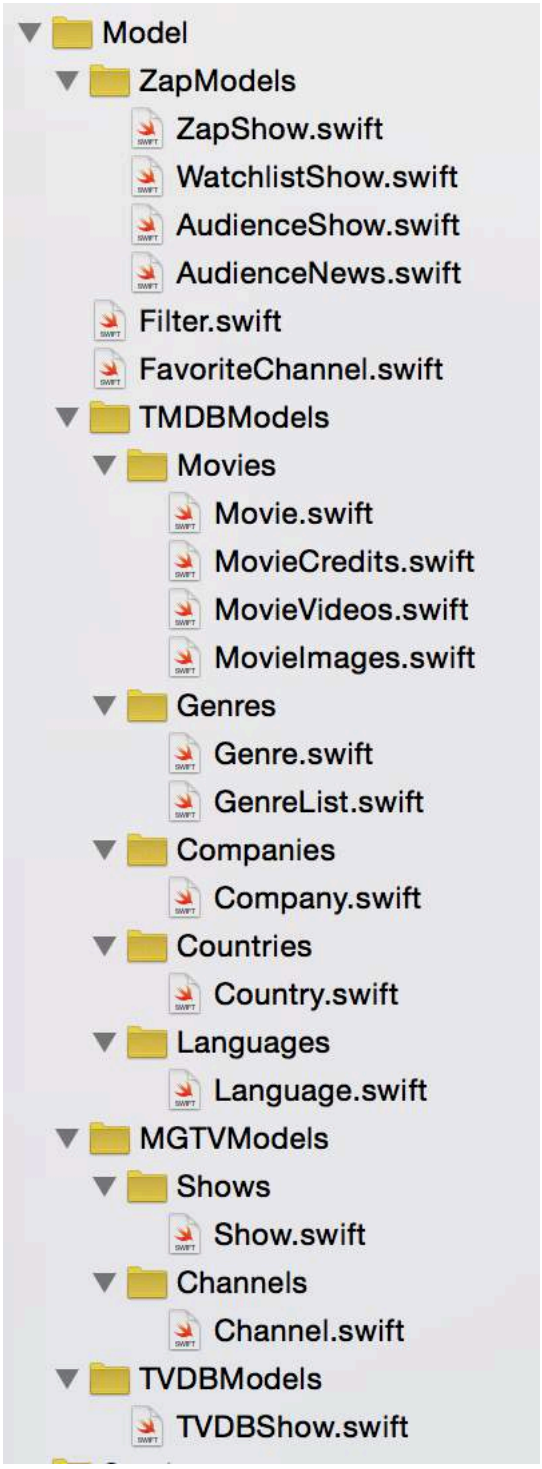
Controladores

En la carpeta View Controllers encontramos los diferentes controladores que conectan los modelos con las vistas. Cada uno de estos controladores extienden sus funcionalidades con otras para organizar el código de forma ordenada.



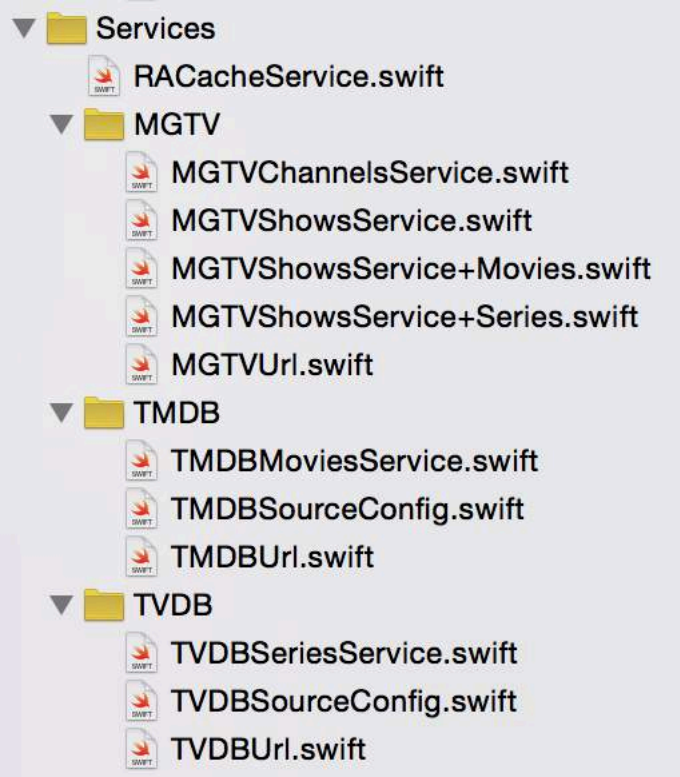
Modelos

La mayoría de los modelos de la aplicación corresponden con los modelos de los diferentes servicios web. La aplicación usa la base de datos local para almacenar estos objetos en el dispositivo.



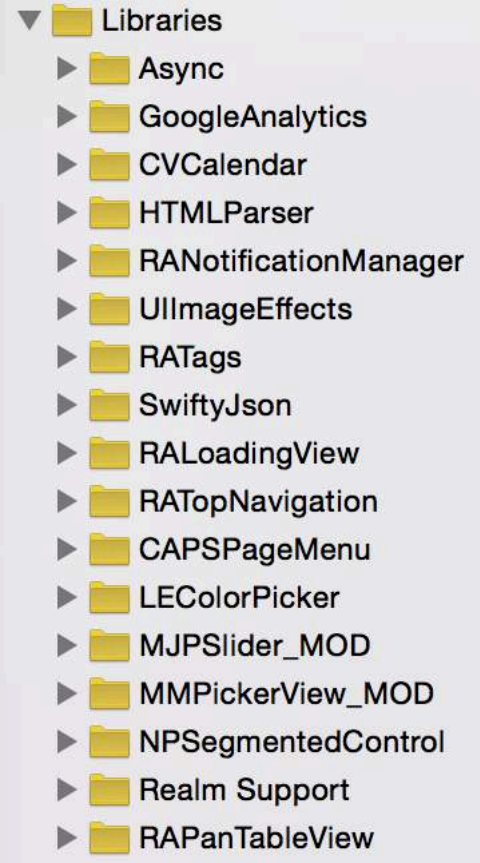
Servicios

En una carpeta separada, se ha colocado todos los archivos que corresponden con la conexión a los diferentes servicios web. Estos servicios son llamados desde el controlador para obtener los datos y trasladarlos a un modelo determinado para guardarlos en la base de datos local de la aplicación.



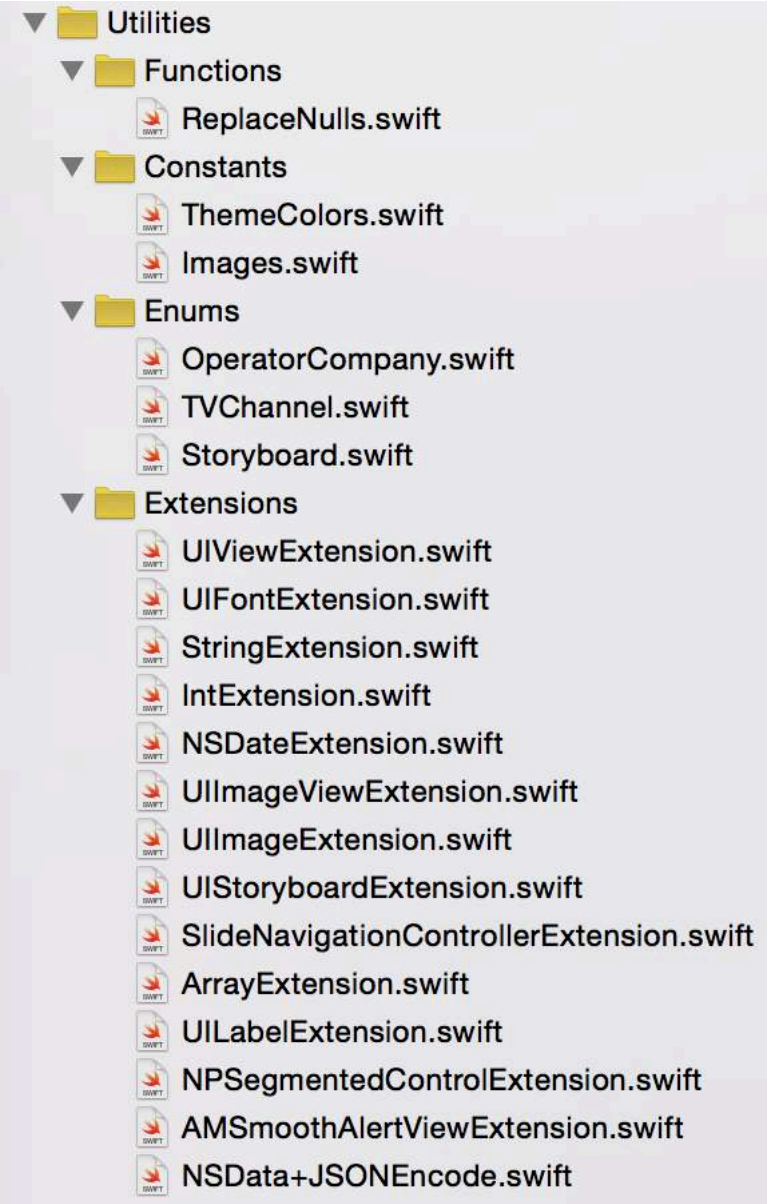
Librerias

En esta carpeta se han incluido las diferentes librerías que usa el proyecto que no están administradas por “Cocoapods”.



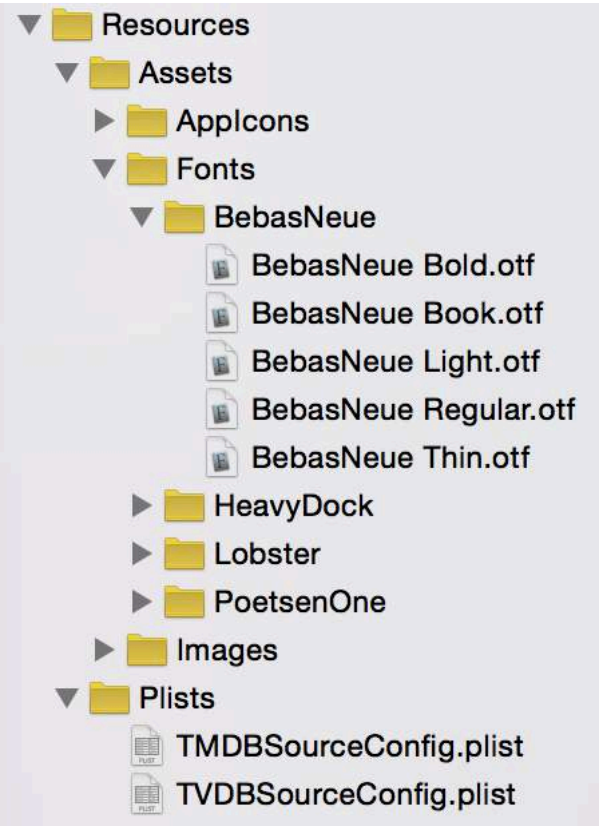
Utilidades

Estos archivos son constantes usadas de forma general en la aplicación y diferentes extensiones de clases para aumentar su funcionalidad.



Recursos

Aquí se incluyen todos los recursos usados generalmente en las vistas como imágenes, fuentes, sonidos, iconos…



2.3.4 SERVICIOS WEB

Los servicios web que se usan en la aplicación usan el protocolo de comunicación REST basado en XML o JSON. Este protocolo es el más moderno y se basa en realizar una petición HTTP al URL del servicio web donde se encuentran los datos que nos interesan.

Esta petición la debemos hacer de forma asíncrona para que la aplicación no se quede congelada hasta que el servidor responda o se produzca un timeout. Además el tratamiento de datos también lo realizaremos de forma asíncrona, para ello trabajaremos en un hilo independiente que va a estar pendiente de la respuesta para realizar una determinada acción.

En este caso, usaremos una petición síncrona, pero no en el hilo principal. Para ello usaremos el Grand Central Dispatch.

Grand Central Dispatch (GCD) está disponible desde la versión 10.6 de OS X y iOS 4.0. Es una tecnología para optimizar el uso de procesadores multinúcleo. El sistema operativo se encarga de gestionar los hilos de ejecución de las aplicaciones.

Se basa en:

- Block objects
- Dispatch queues
- Synchronization
- Event sources

Para la petición HTTP usaremos la librería AFNETWorking que administra y realiza de forma asíncrona la llamada a la URL.

2.3.4.1 API MIGUIATV

Miguatv es un servicio web que ofrece la programación de todos los canales de televisión, imágenes, horarios, descripción, últimos programas y más detalles de series y películas.

DISEÑO DE CLASES

Show

type	Tipo	Define el tipo de programa que se trata: película, serie, programa, documental
showID	Id de programa	Id de referencia del programa
channelID	Id del canal	Id de referencia del canal de televisión
Inittime	Tiempo inicial	Tiempo en el cual empieza el programa
Endtime	Tiempo final	Tiempo en el cual finaliza el programa
Subtype	Subtipo	Subtipo de programa
Title	Título	Título de la película, serie, programa…
Subtitle	Subtitulo	Frase descriptiva del programa
Text	Texto	Descripción del programa
Rate	Valoración	Valoración general realizada por usuarios
Backdrop	Imagen de fondo	Imagen de fondo del programa

```
class Show: RLMObject {

    dynamic var type = ""
    dynamic var showID = 0
    dynamic var channelID = 0
    dynamic var inittime = 0
    dynamic var endtime = 0

    // Get from TMDB
    dynamic var subtype = ""
    dynamic var title = ""
    dynamic var subtitle = ""
    dynamic var text = ""
    dynamic var rate: String? = ""
    dynamic var backdrop = ""

    override class func primaryKey() -> String! {
        return "inittime"
    }
}
```

Channel

Id	Id	Id de referencia del canal de televisión
Name	Nombre	Nombre o título del canal de televisión
Type	Tipo	Define el tipo de canal: TDT, Nacional..
Language	Lenguaje	Lenguaje del canal
Image	Imagen	Icono del canal de televisión

```
class Channel: RLMObject {

    dynamic var id = 0
    dynamic var name = ""
    dynamic var type = ""
    dynamic var language = ""
    dynamic var image = ""

    override class func primaryKey() -> String! {
        return "id"
    }

    class func ById(id:Int) -> Channel? {

        let channels: RLMResults = self.objectsWhere("id == \(id)")

        return channels.lastObject() as? Channel
    }

    func getShows() -> RLMResults? {

        return ZapShow.objectsWhere("channelID == \(self.id)").
            sortedResultsUsingProperty("inittime", ascending: true)
    }
}
```

2.3.4.2 API THEMOVIEDB

Themoviedb utiliza una gran base de datos de películas y series creada y actualizada por usuarios. En ella se puede obtener mucha información sobre las series y películas. Es análoga a Imdb (Internet Movie Data Base).

Se utilizará Themoviedb para obtener información de las películas mostradas en la aplicación debido a que la id de las películas obtenidas del servicio de Miguiatv corresponden a la id de películas de Themoviedb.

DISEÑO DE CLASES

Movie

adult	Adultos	Contenido para adultos
Backdrop_path	Imagen de fondo	Ruta URL de la imagen de fondo
Budget	Presupuesto	Presupuesto de la película
Credits	Créditos	Actores y directores que salen en la película
Videos	Trailers	Trailers de la película
Genres	Géneros	Lista de géneros descriptivos de la película
Homepage	Web oficial	Web oficial de la película
Id	Id	Id de referencia de la película
Imdb_id	Id de IMDB	Id de referencia de IMDB de la película
Original_language	Lenguaje original	Lenguaje original de la película
Original_title	Título original	Título original de la película
Overview	Resumen	Resumen de la película
Popularity	Popularidad	Puntuación de popularidad
Poster_path	Poster	URL del poster de la película
Production_companies	Productoras	Productoras que han intervenido en la película

Production_countries	Países	Países de las productoras
Release_date	Fecha de lanzamiento	Fecha de estreno de la película
Revenue	Beneficios	Beneficios que ha obtenido la película
Runtime	Duración	Duración de la película
Spoken_langauges	Lenguajes	Lenguajes en los que se emite la película
Status	Estado	Estado de la película
Tagline	Tags	Tags asociados a la película
Title	Título	Título de la película
Video	Video	La película tiene tráiler o no
Vote_average	Valoración media	Valoración media de usuarios
Vote_count	Cuenta de valoraciones	Número de valoraciones realizadas de usuarios

```
class Movie: RLMObject {

    dynamic var adult = false
    dynamic var backdrop_path = ""
//    dynamic var belongs_to_collection: String?
    dynamic var budget = 0
    dynamic var credits = MovieCredits()
    dynamic var videos = MovieVideos()
    dynamic var genres = RLMArray(objectClassName: Genre.className())
    dynamic var homepage = ""
    dynamic var id = 0
    dynamic var imdb_id = ""
    dynamic var original_language = ""
    dynamic var original_title = ""
    dynamic var overview = ""
    dynamic var popularity = 0.0
    dynamic var poster_path = ""
    dynamic var production_companies = RLMArray(objectClassName: Company.className())
    dynamic var production_countries = RLMArray(objectClassName: Country.className())
    dynamic var release_date = ""
    dynamic var revenue = 0
    var runtime: Int?
    dynamic var spoken_languages = RLMArray(objectClassName: Language.className())
    dynamic var status = ""
    dynamic var tagline = ""
    dynamic var title = ""
    dynamic var video = false
    dynamic var vote_average = 0.0
    dynamic var vote_count = 0

    override class func primaryKey() -> String! {
        return "id"
    }
}
```

MovieCredits

Cast	Casting	Actores que salen en la película
Crew	Equipo	Equipo de producción de la película

Cast

Cast_id	Id	Id de referencia de casting
Character	Nombre	Nombre en la película
Credit_id	Id	Id de referencia de créditos
Id	Id	Id de referencia
Name	Nombre	Nombre real del actor
Order	Orden	Orden de importancia en la película
Profile_path	Imagen	Foto del actor

Crew

Credit_id	Id	Id de referencia de créditos
Department	Departamento	Departamento con el que se le asocia
Id	Id	Id de referencia
Job	Trabajo	Trabajo en la película
Name	Nombre	Nombre de la persona
Profile_path	Imagen	Foto de la persona

```
class MovieCredits: RLMObject {

    //dynamic var id = 0
    dynamic var cast = RLMArray(objectClassName: Cast.className())
    dynamic var crew = RLMArray(objectClassName: Crew.className())
}

class Cast: RLMObject {

    dynamic var cast_id = 0
    dynamic var character = ""
    dynamic var credit_id = ""
    dynamic var id = 0
    dynamic var name = ""
    dynamic var order = 0
    dynamic var profile_path = ""

    override class func primaryKey() -> String! {
        return "id"
    }
}

class Crew: RLMObject {

    dynamic var credit_id = ""
    dynamic var department = ""
    dynamic var id = 0
    dynamic var job = ""
    dynamic var name = ""
    dynamic var profile_path = ""

    override class func primaryKey() -> String! {
        return "id"
    }
}
```

MovieVideos

Results	Resultados	Array de resultados de MovieVideo

```
class MovieVideos: RLMObject {

    //dynamic var id = 0
    dynamic var results = RLMArray(objectClassName: MovieVideo.className())
}

class MovieVideo: RLMObject {

    dynamic var id = ""
    dynamic var iso_639_1 = ""
    dynamic var key = ""
    dynamic var name = ""
    dynamic var site = ""
    dynamic var size = 0
    dynamic var type = ""
}
```

MovieImages

Id	Id	Id de referencia
Backdrops	Imágenes de fondo	Array de imágenes de fondo
Posters	Posters	Array de posters de la película

MovieImage

File_path	Ruta de archivo	URL del archivo de imagen
Width	Ancho	Ancho de Imagen
Height	Alto	Alto de Imagen
Aspect_ratio	Ratio	Ratio de Imagen
Vote_average	Valoracion Media	Valoración Media de usuarios
Vote_count	Numero de Valoraciones	Numero de valoraciones de usuarios

```
class MovieImages: RLMObject {

    dynamic var id = 0
    dynamic var backdrops = RLMArray(objectClassName: MovieImage.className())
    dynamic var posters = RLMArray(objectClassName: MovieImage.className())

    override class func primaryKey() -> String! {
        return "id"
    }

    func getImagesArray() -> [String] {

        var imagesArray: [String] = []

        for backdrop in self.backdrops {
            imagesArray.append((backdrop as! MovieImage).file_path)
        }
        for backdrop in self.posters {
            imagesArray.append((backdrop as! MovieImage).file_path)
        }

        return imagesArray
    }
}

class MovieImage: RLMObject {

    dynamic var file_path = ""
    dynamic var width = 0
    dynamic var height = 0
//    dynamic var iso_639_1 = ""
    dynamic var aspect_ratio = 0.0
    dynamic var vote_average = 0.0
    dynamic var vote_count = 0
}
```

Genre

Id	Id	Id de referencia
Name	Nombre	Nombre del género

```
class Genre: RLMObject {

    dynamic var id = 0
    dynamic var name = ""

    override class func primaryKey() -> String! {
        return "id"
    }
}
```

GenreList

Id	Id	Id de referencia
Genres	Lista de géneros	Array de géneros

```
class GenreList: RLMObject {

    dynamic var id = 0
    dynamic var genres = RLMArray(objectClassName: Genre.className())

    override class func primaryKey() -> String! {
        return "id"
    }
}
```

Company

Headquarters		
Homepage	Página Oficial	Página web oficial
Id	Id	Id de referencia
Logo_path	Logo	Imagen del logo de la compañía
Name	Nombre	Nombre de la compañía
Parent_company	Compañía padre	Nombre de la compañía a la que pertenece

```
class Company: RLMObject {

    //dynamic var description = ""
    dynamic var headquarters = ""
    dynamic var homepage = ""
    dynamic var id = 0
    dynamic var logo_path = ""
    dynamic var name = ""
    dynamic var parent_company = ""

    override class func primaryKey() -> String! {
        return "id"
    }
}
```

Country

Iso_3166_1	Iso	Nombre ISO
Name	Nombre	Nombre del país

```
class Country: RLMObject {

    dynamic var iso_3166_1 = ""
    dynamic var name = ""

    override class func primaryKey() -> String! {
        return "iso_3166_1"
    }
}
```

Language

Iso_639_1	Iso	Nombre ISO
Name	Nombre	Nombre del lenguaje

```
class Language: RLMObject {

    dynamic var iso_639_1 = ""
    dynamic var name = ""

    override class func primaryKey() -> String! {
        return "iso_639_1"
    }
}
```

2.3.4.3 API THETVDB

El servicio web de TheTVDB es otra base de datos abierta y de actualización pública de series y programas de televisión. Ofrece bastante información y recursos como posters, banners, fotos de actores, resúmenes y datos técnicos de todos los episodios de todas las temporadas de multitud de series de televisión.

Usaremos esta api para la obtención de información sobre series y programas de la aplicación.

DISEÑO DE CLASES

TVDBShow

ShowID	Id	Id de referencia del programa
Title	Título	Título del programa
Summary	Resumen	Resumen descriptivo del programa
Banner	Imagen	Imagen del programa
BannerThumbnail	Imagen pequeña	Imagen pequeña del programa
imdbID	Id de IMDB	Id de referencia de IMDB del programa
premiereDate	Fecha de estreno	Fecha en la que se estrena el programa
Status	Estado	Estado del programa
Genre	Género	Array de géneros que describen el programa
Actors	Actores	Array de actores que participan en el programa
Poster	Poster	Imagen del programa
PosterThumbnail	Poster pequeño	Imagen pequeña del programa
Fanart	Fanart	Imagen del programa
FanartThumbnail	Fanart pequeño	Imagen pequeña del programa

airDay	Fecha que se emite	Fecha que se emite
airTime	Hora que se emite	Hora en la que se emite
runTime	Tiempo de duración	Duración del programa
Network	Red	Red en la que se emite
contentRating	Valoración del contenido	Valoración del contenido
Rating	Valoración	Valoración

```
class TVDBShow: RLMObject {

    // basic properties of a show
    dynamic var showID = 0
    dynamic var title = ""
    dynamic var summary = ""
    dynamic var banner = ""
    dynamic var bannerThumbnail = ""
    dynamic var imdbID = ""
    dynamic var premiereDate = ""

    // extra properties of a show
    dynamic var status = ""
    dynamic var genre = RLMArray(objectClassName: StringObject.className())
    dynamic var actors = RLMArray(objectClassName: StringObject.className())
    dynamic var poster = ""
    dynamic var posterThumbnail = ""
    dynamic var fanart = ""
    dynamic var fanartThumbnail = ""
    dynamic var airDay = ""
    dynamic var airTime = ""
    dynamic var runtTime = ""
    dynamic var network = ""
    dynamic var contentRating = ""
    dynamic var rating: Float = 0
//    dynamic var episodes = NSMutableArray()

    override class func primaryKey() -> String! {
        return "showID"
    }
}
```

2.3.5 FUNCIONES INTERESANTES DEL PROYECTO

A continuación, se explican algunos de los métodos más interesantes de este proyecto.

La conexión con bases de datos externas, es importante para la extracción de los datos que se van a mostrar en la aplicación. Una vez extraídos los datos, se mapean para añadirlos a la base de datos local.

La técnica web-scrapping, es otro método para extraer información de una página web. Se analiza el código html de la página y se extraen los datos interesantes para mostrarlos luego de forma nativa dentro de la aplicación.

2.3.5.1 CONEXIÓN CON BASES DE DATOS EXTERNAS

Para realizar la conexión a una base de datos externa se han de seguir una serie de pasos:

- Usar un servicio web intermedio a modo de pasarela entre la app y la base de datos
- Serializar los datos a través de JSON
- Guardar estos objetos en la base de datos local para su uso posterior sin necesidad de conexión a internet

Para seguir cada uno de estos pasos se han realizado una serie de clases y métodos organizados que realizan todo este proceso.

MGTVUrl.swift

Esta clase se encarga de generar la URL para realizar la llamada a la API de MiGuiaTV y obtener los datos requeridos.

```
class MGTVUrl: NSObject {

    let MGTVHost = "http://api.miguia.tv/1/"
    let language = "es_ES"
    let channelFormat = "/channel/"

    func showsByChannel(channel: Int) -> String {

        return MGTVHost + language + channelFormat + String(channel) + ".json"
    }

    func channels() -> String {

        return MGTVHost + language + "/channels.json"
    }

    func currentShows() -> String {

        return MGTVHost + language + channelFormat + "now.json"
    }
}
```

RACacheService.swift

Esta es una clase creada para guardar la fecha de la última extracción de los datos de la API. De esta forma, comprueba si la llamada a esta API se ha realizado en un tiempo menor a 2 horas (60 * 60 * 2 en segundos). Si es así, no se realizará la llamada a la API y se obtendrán los datos guardados en la base de datos local del propio iPhone.

El guardado de la fecha de extracción se realiza a través de NSUserDefaults.

```
class RACacheService: NSObject {

    var disabled = false
    var cachedTime: Int = 60 * 60 * 2

    func getCache(cacheString: String) -> Bool {

        if disabled {
            return false
        }

        let defaults = NSUserDefaults.standardUserDefaults()
        let cachedDate = defaults.objectForKey(cacheString) as? NSDate
        if let cachedDate_ = cachedDate
        {
            let secondsBetween = Int(NSDate().timeIntervalSinceDate(cachedDate_))
            if (secondsBetween >= cachedTime) {
                return false
            }
            else {
                return true
            }
        }
        else {
            return false
        }
    }

    func saveCache(cacheString: String) {

        let defaults = NSUserDefaults.standardUserDefaults()
        defaults.setObject(NSDate(), forKey: cacheString)
    }
}
```

MGTVChannelsService.swift

Esta clase se encarga de administrar el servicio web, tanto la llamada a la API como la recogida de datos, serialización JSON y guardado en la base de datos local.

Cuando se obtienen y guardan los resultados, es necesario avisar a las diferentes vistas que han terminado la llamada al servicio web y los datos están listos para ser mostrados. Para esto se usan los protocolos y delegados.

Para ello, se ha creado un protocolo llamado MGTVChannelsServiceDelegate

```
protocol MGTVChannelsServiceDelegate
{
    func didSuccessChannels()
    func didFailChannels(error:NSError)
}
```

En la clase que da nombre al archivo se crea un método llamado fetchChannels() que se encarga de realizar la llamada a la API siempre que no se haya realizado en un tiempo de 2 horas (mediante cacheService explicado anteriormente) .

Para ello, se usa la librería AFNetWorking que gestiona las llamadas web con una serie de parámetros y obtiene de forma asíncrona la respuesta obtenida de la API. Si la respuesta se ha obtenido correctamente llamamos a una nueva función que ordena los resultados. Si ha surgido algún error, llamamos a otra función que conecta con el delegado correspondiente.

```
class MGTVChannelsService: NSObject {

    let manager = AFHTTPRequestOperationManager()
    var delegate: MGTVChannelsServiceDelegate?
    let url = MGTVUrl()
    let cacheService = RACacheService()

    func fetchChannels() {

        let cacheString = "fetchChannels"
        if !cacheService.getCache(cacheString) {

            self.manager.GET(url.channels(), parameters: "", success:
                {
                    (operation, responseObject) -> Void in
                    self.cacheService.saveCache(cacheString)
                    self.didSuccessChannels(responseObject)

                }) {
                    (operation, error) -> Void in
                    self.didFailChannels(error)
            }
        }
        else {
            delegate?.didSuccessChannels()
        }
    }
}
```

Debido a la cantidad de información recibida que hay que gestionar tras la llamada a la API, es necesario usar de nuevo un método asíncrono que realice en background tanto la serialización JSON como el guardado en la base de datos local.

Para ello se ha usado el Grand Central Dispatch (GCD) a través de una librería que simplifica enormemente el código llamada Async. El GCD se usa para ejecutar tareas pesadas en segundo plano. Si no la usáramos, el usuario de la aplicación podría ver como esta tarea hace bloquear toda la ejecución del sistema. El GCD nos permite lidiar con aplicaciones multihilo de forma más sencilla y controlada.

```
func didSuccessChannels(responseObject: AnyObject!) {

    Async.background {
        self.saveInRealm(responseObject)
        } .main {
        self.delegate?.didSuccessChannels()
    }
}

func didFailChannels(error: NSError!) {

    println("ERROR (MGTVChannelsService): \(error)")
    delegate?.didFailChannels(error)
}
```

Como vemos, una vez se ha ejecutado todo el proceso en background, se llama al delegado didSuccessChannels(). En cambio, si se ha obtenido un error llamamos al delegado didFailChannels()

Para guardar los datos obtenidos en background se ha creado un método llamado saveInRealm(). En este se usa la librería JSON que serializa (mapea) un objeto de texto tipo JSON en un objeto diccionario.

Con la librería Realm gestionamos la base de datos local. Creamos un objeto de tipo Channel() que hereda propiedades de Realm y lo completamos con los

datos obtenidos del objeto JSON. Una vez completado, se hace commit para guardar los cambios del objeto en la base de datos.

Si no usáramos esta librería, el guardado en la base de datos se realizaría de un modo más complicado que ensuciaría el código haciéndolo ilegible, por ello, es recomendable el uso, siempre que se pueda, de este tipo de librerías.

```
func saveInRealm(responseObject: AnyObject) {

    let json = JSON(responseObject)
    let realm: RLMRealm = RLMRealm.defaultRealm()

    if (!realm.inWriteTransaction) {
        realm.beginWriteTransaction()
    }

    for var i = 0; i < json["data"].count; ++i {

        let channel = Channel()

        channel.id = json["data"][i][0].intValue
        channel.name = json["data"][i][1].stringValue
        channel.type = json["data"][i][2].stringValue
        channel.language = json["data"][i][3].stringValue
        channel.image = json["data"][i][4].stringValue
        Channel.createOrUpdateInDefaultRealmWithValue(channel)
    }

    realm.commitWriteTransaction()
}
```

2.3.5.2 WEB SCRAPPING

En este proyecto ha resultado interesante extraer algunos datos de audiencias de televisión para poderlos mostrar dentro de la aplicación. Para ello, se ha necesitado usar la técnica de web scrapping que nos permite extraer datos que no están estructurados y guardarlos de forma estructurada en la base de datos local.

De esta forma, la técnica web scrapping consigue leer los contenidos de una página web y haciendo uso de librerías como HTMLParser se consigue mapear de forma ordenada todo el contenido extraído y buscar coincidencias de una cadena. Esta misma técnica es muy usada por otras páginas web programadas en php.

Algunos de los ejemplos más conocidos del uso de web scrapping es usado por Google para su motor de búsqueda, en el cual, analiza e indexa contenido de diferentes sitios web. Es importante que el contenido extraído de estos sitios web sea público, de forma que no se violen términos y condiciones.

En este TFG se han usado varios métodos para realizar esta técnica dentro de la clase que muestra la vista de “Audiencias de Televisión”.

Como se puede comprobar a continuación, todo este proceso, se realiza en background para que no suponga un bloqueo en tiempo de ejecución. De forma que la aplicación se puede seguir usando y cuando se complete la extracción de información, la vista dibujará todos los datos extraídos.

```
func updateAudiencesWithDate(date: NSDate) {

    Async.background {

        var err1 : NSError?
        var encod = NSStringEncoding()
        var string = String(contentsOfURL: self.getAudiencesURLByDate(date),
            usedEncoding: &encod, error: &err1)

        println("ENCODING: \(encod.description)")
        println("ERROR IS: \(err1)")

        var htmlString = String(contentsOfURL: self.getAudiencesURLByDate(date),
            encoding: encod, error: &err1)
        println("ERROR IS: \(err1)")

        if let htmlString_ = htmlString {

            var err2 : NSError?
            let option = CInt(HTML_PARSE_NOERROR.value | HTML_PARSE_RECOVER.value)
            let parser = HTMLParser(html: htmlString_, encoding:
                NSUTF8StringEncoding, option: option, error: &err2)
            if err2 != nil {
                println(err2)
            }

            var bodyNode = parser.body
            if let bodyNode_ = bodyNode {

                self.parseNewsWithNode(bodyNode_)
                self.parseAudiencesWithNode(bodyNode_)
            }
        }

        } .main {
            self.tableView.reloadData()
            self.newsCollectionView.reloadData()
    }
}
```

2.3.6 LIBRERÍAS USADAS

Para la realización de esta aplicación se han investigado numerosos repositorios de código abierto con numerosos controles y librerías que pudieran resultar útiles para mejorar las funcionalidades y mantener un código limpio para obtener una lectura fácil.

La mayor parte de estas librerías han sido realizadas por diferentes usuarios que publican libremente su código para que sea testeado y mejorado en internet a través de versiones de código abierto en github.

2.3.6.1 COCOAPODS

Cuando hacemos frente a una multitud de librerías, la instalación, la actualización o modificación de estas se convierte en un proceso muy tedioso. Una forma de gestionar fácilmente las diferentes librerías es el uso de Cocoapods (http://cocoapods.org) un gestor de dependencias de librerías de Swift y Objective-C.

Para el uso de este gestor debemos instalarlo en nuestro Mac. Un vez instalado, dentro de la carpeta del proyecto usaremos un fichero de dependencias llamado “Podfile”. Las librerías usadas en este archivo se llaman “Pods”. Todos los pods usados en este archivo deben estar dados de alta en el repositorio de Cocoapods.

Cocoapods nos crea un Workspace en XCode que es el que vamos a usar para crear nuestro proyecto.

Podfile

```
# Uncomment this line to define a global platform for your project
platform :ios, '7.0'

#use_frameworks!

    pod 'AFNetworking'
    pod 'iOS-Slide-Menu'
    pod 'Colours'
    pod 'SDWebImage'
    pod 'UIActivityIndicator-for-SDWebImage'
    pod 'DWTagList'
    pod 'RMSwipeTableViewCell'
    pod 'AXRatingView'
    pod 'JVFloatingDrawer'
    pod 'XCDYouTubeKit'
    pod 'ARSpeechActivity'
    pod 'AMSmoothAlert'
    pod 'hpple'
    pod 'DIDatepicker'
    pod 'iTVDb'
    pod 'pop'
    pod 'Ask4AppReviews'
    pod 'Async'
    pod 'XMLDictionary'
```

2.3.6.1.1 AFNETWORKING

AFNetworking es una librería para la gestión de tareas de networking para iOS y OSX que funciona a modo de envoltorio de las librerías del **Foundation URL Loading System** (es decir, NSURLConnection y NSURLSession) para la realización de peticiones a servicios web.

NSURLConnection hace referencia a: NSURLRequest, NSURLResponse, NSURLCache, NSURLPeotocol, NSHTTPCookieStorage, NSURLCredentialStorage y a NSURLConnection además de los protocolos y delegados correspondientes.

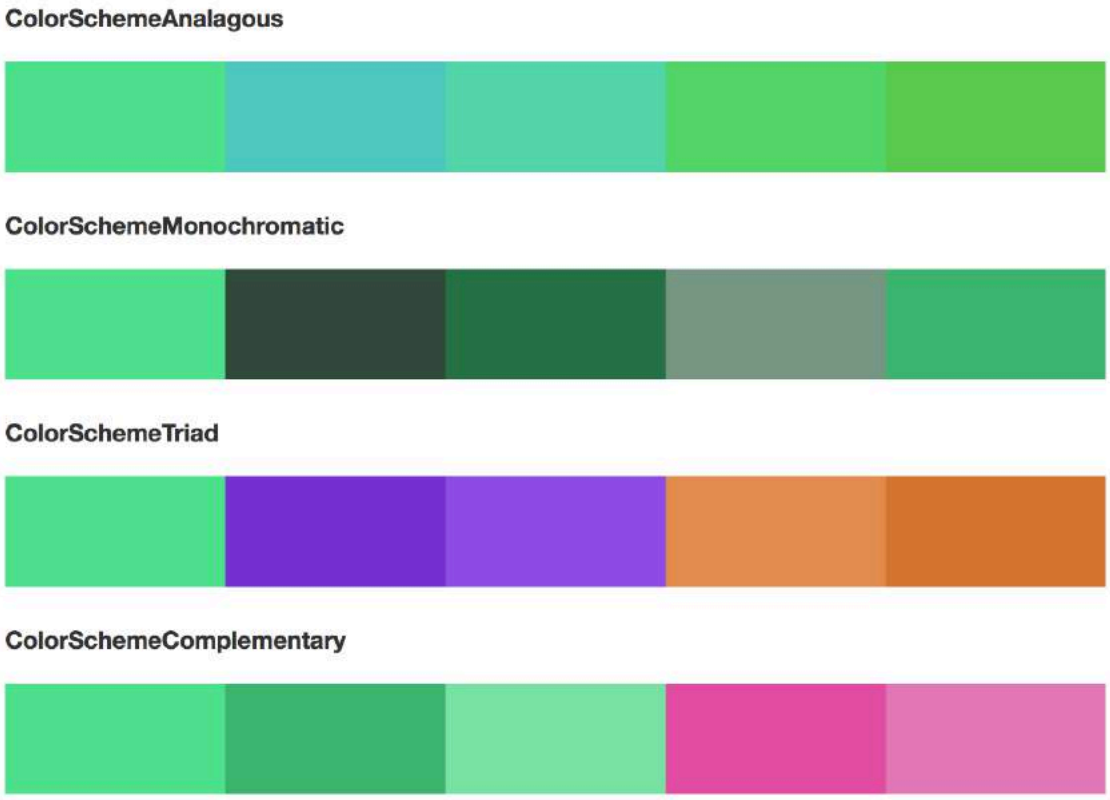
NSURLSession hace referencia al mismo conjunto de clases añadiendo NSURLSessionConfiguration y a las subclases de NSURLSessionTask.

AFNetworking hace uso de todas estas clases ofreciendo en ocasiones una sintaxis más clara y fácil de leer y en otras ocasiones se encarga de tareas más tediosas como la conversión de datos JSON, XML, PLIST, gestión de estados, solicitudes y respuestas.

2.3.6.1.2 COLOURS

Colours usa la clase UIColor para definir diferentes paletas de colores de forma fácil y sencilla. De este modo, se genera, a partir de un color, una paleta de diferentes colores para usar en la aplicación.

Por ejemplo: Del verde obtenemos estas paletas de colores en forma de Array




```
cell.iconImageView.tintColor = primaryColorTheme.colorSchemeOfType(ColorScheme.Monochromatic)[1]
```

Esta librería da una enorme cantidad de posibilidades para el cambio del tema de colores de la aplicación de forma fácil y rápida.

También contiene multitud de funciones para el calculo de distancias entre colores, conversión de UIColor a Arrays o Diccionarios, uso de formatos HSBA, CIELAB, RGBA, CMYK, Color Components, modificación de colores de forma dinámica (aclarar, oscurecer, contraste, colores complementarios…)

2.3.6.1.3 SDWEBIMAGE

Esta librería añade una extensión a UIImageView para soportar la carga de imágenes de internet. Además posee otras funcionalidades como:

- Gestionar la cache para la carga rápida de imágenes.
- Descarga asíncrona de imágenes
- Gestión de expiración de caché automática
- Soporte de gifs animados
- Soporte de formato WebP
- Descompresión de imágenes

2.3.6.1.4 UIACTIVITYINDICATOR FOR SDWEBIMAGE

Complementa la librería anterior con el uso de un indicador de actividad para mostrar que una imagen está descargándose.

```
extension UIImageView {

    func setImageURL(url:String) {

        self.setImageWithURL(NSURL(string: url)!, placeholderImage: UIImage(named:
            "placeholder_logo")!, usingActivityIndicatorStyle: UIActivityIndicatorViewStyle.
            White)
    }
}
```

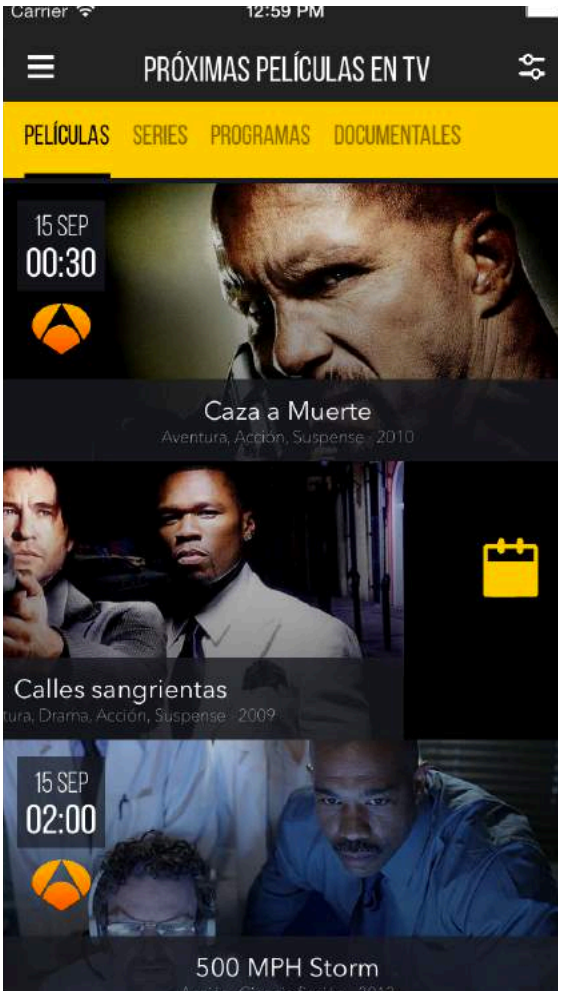
2.3.6.1.5 DWTAGLIST

DWTagList genera una lista de tags desde un array de strings en una vista totalmente customizable con fuentes de diferentes tipos y colores.

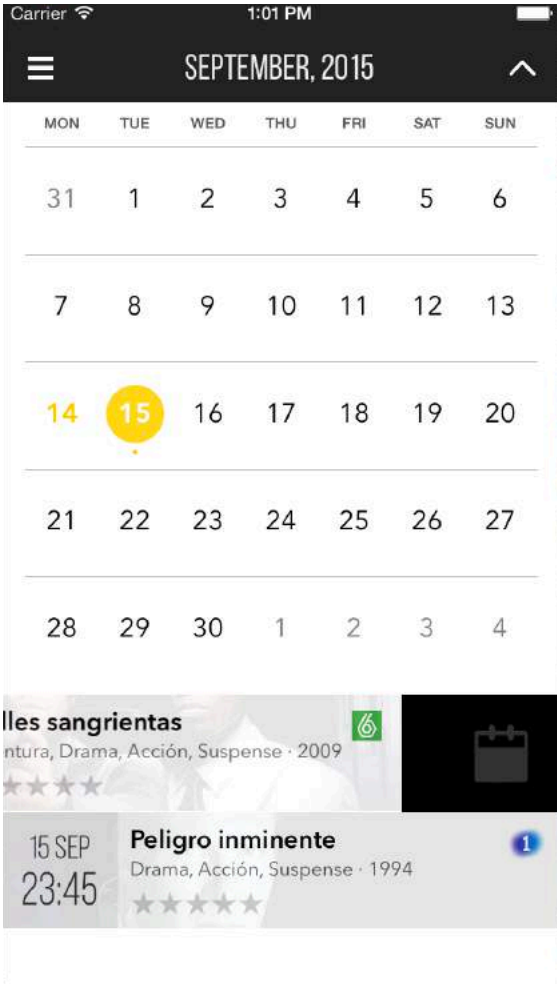
2.3.6.1.6 RMSWIPETABLEVIEW CELL

Esta librería añade una nueva funcionalidad a la celda de una tabla. Con RMSwipeTableViewCell cualquier celda puede deslizarse de forma horizontal para realizar cualquier acción determinada.

En el caso de esta aplicación, la usamos para añadir o eliminar una serie o película a la agenda.



Vista principal con deslizamiento en una celda de la tabla para la incorporación de la película en la agenda



Vista de la agenda que muestra el calendario y una tabla con las películas pendientes de ver

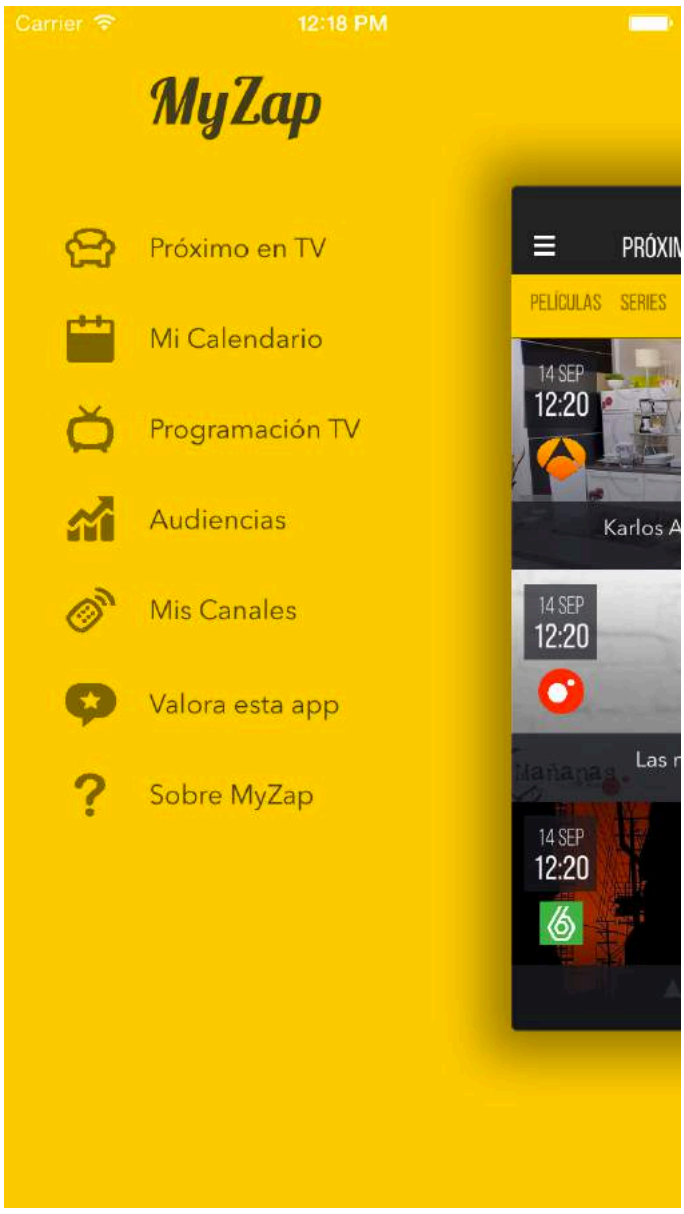
2.3.6.1.7 AXRATINGVIEW

AXRatingView es un UIVIew que muestra un sistema de valoración por estrellas.

2.3.6.1.8 JVFLOATINGDRAWER

JVFloatingDrawer usa el controlador UINavigationController para mostrar la vista del menú cuando llamamos a la función correspondiente en la presión de un botón en la parte superior de la barra de navegación.

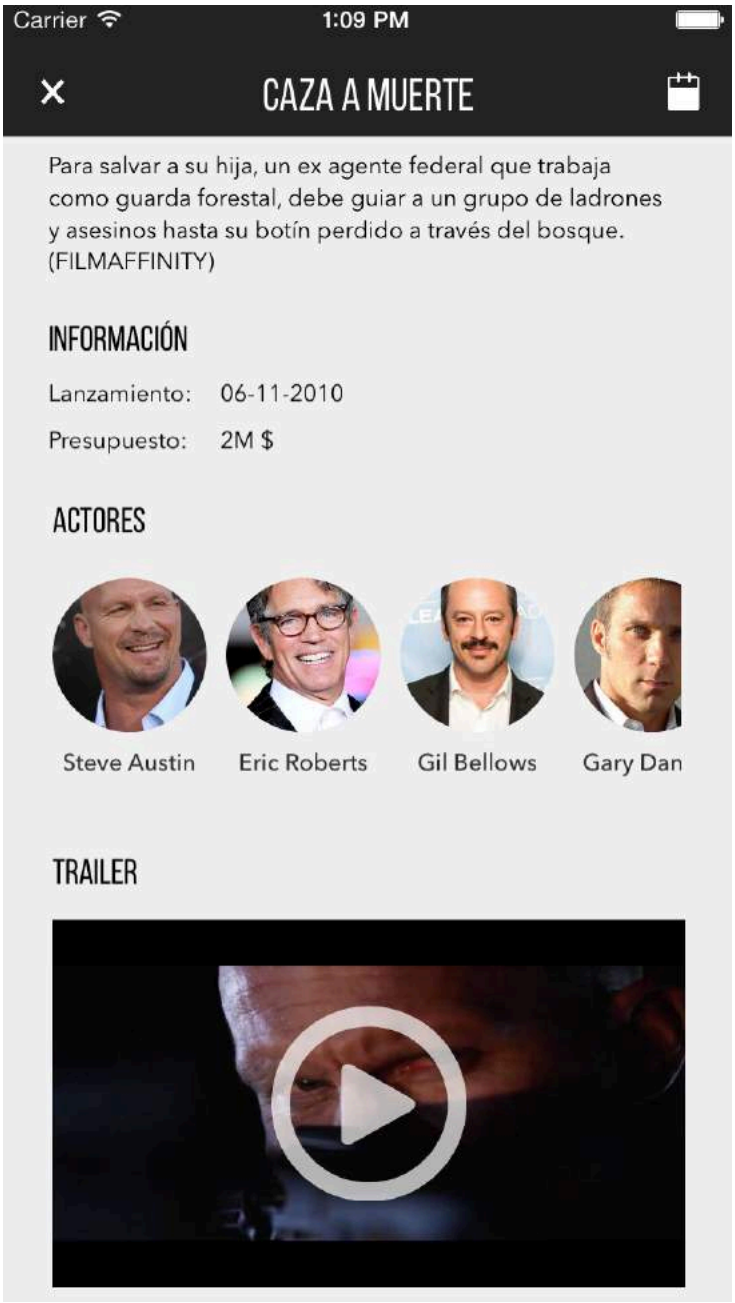
También hace uso de animaciones para mostrar u ocultar el menú de una forma dinámica.



Vista que muestra el menú de la aplicación

2.3.6.1.9 XCDYOUTUBEKIT

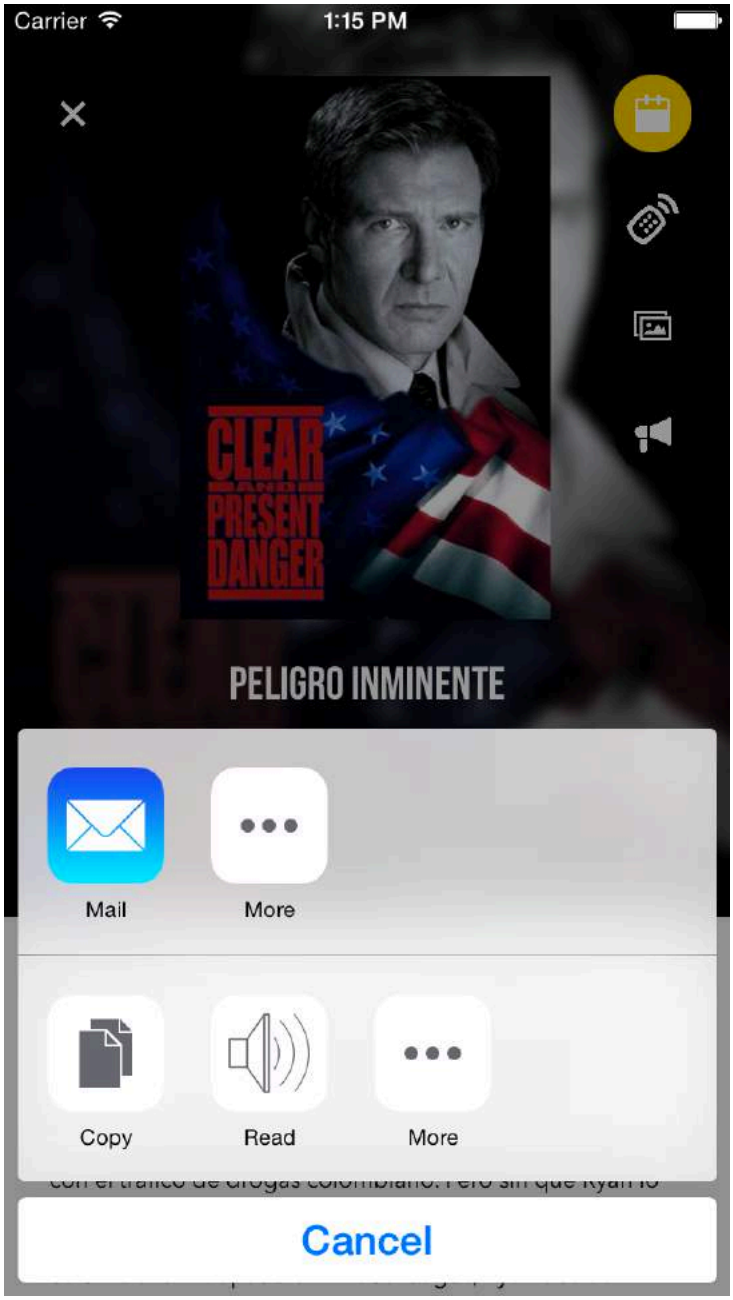
Esta librería ofrece un reproductor de videos Youtube para mostrar en la aplicación de una forma sencilla. Con esta podremos reproducir los trailers de las series o películas que aparecen en la información detallada de la aplicación.



Vista de detalle de la película donde se muestra la información sobre la película y tráiler

2.3.6.1.10 ARSPEECHACTIVITY

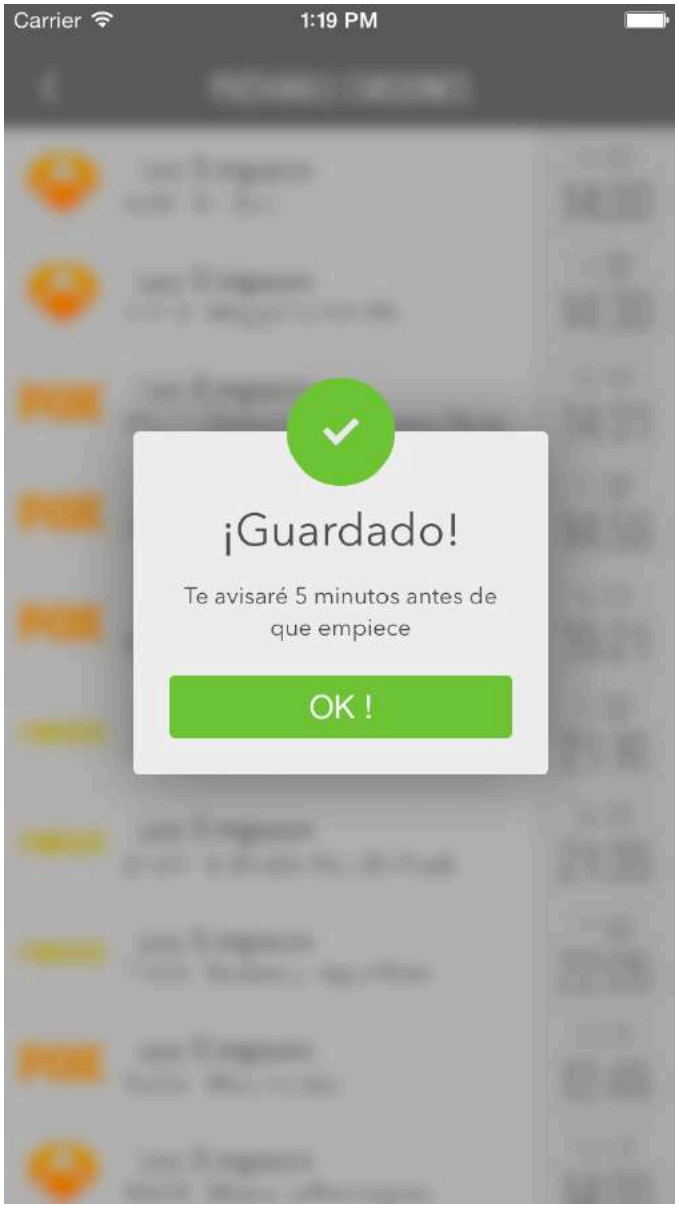
ARSpeechActivity ofrece una funcionalidad nueva para la aplicación que resulta muy útil para que la aplicación nos lea los textos de las descripciones. Hace uso de UIActivity para mostrar esta funcionalidad en la lista de acciones disponibles.



Vista de la aplicación que muestra el panel de actividad de la película

2.3.6.1.11 AMSMOOTHALERT

AMSmoothAlert ofrece un UIAlertView personalizado para ofrecer notificaciones instantáneas dentro de la aplicación.



Vista que muestra el guardado de la notificación para el aviso previo a la emisión de la película

2.3.6.1.12 DIDATEPICKER

DIDatePicker muestra una pequeña tabla en la parte superior de la vista que permite seleccionar un día determinado y con ello mostrar lo que queramos. En el caso de esta aplicación, se muestran las audiencias correspondientes a un día específico.

Vista que muestra las audiencias de televisión de las últimas dos semanas

2.3.6.1.13 ITVDB

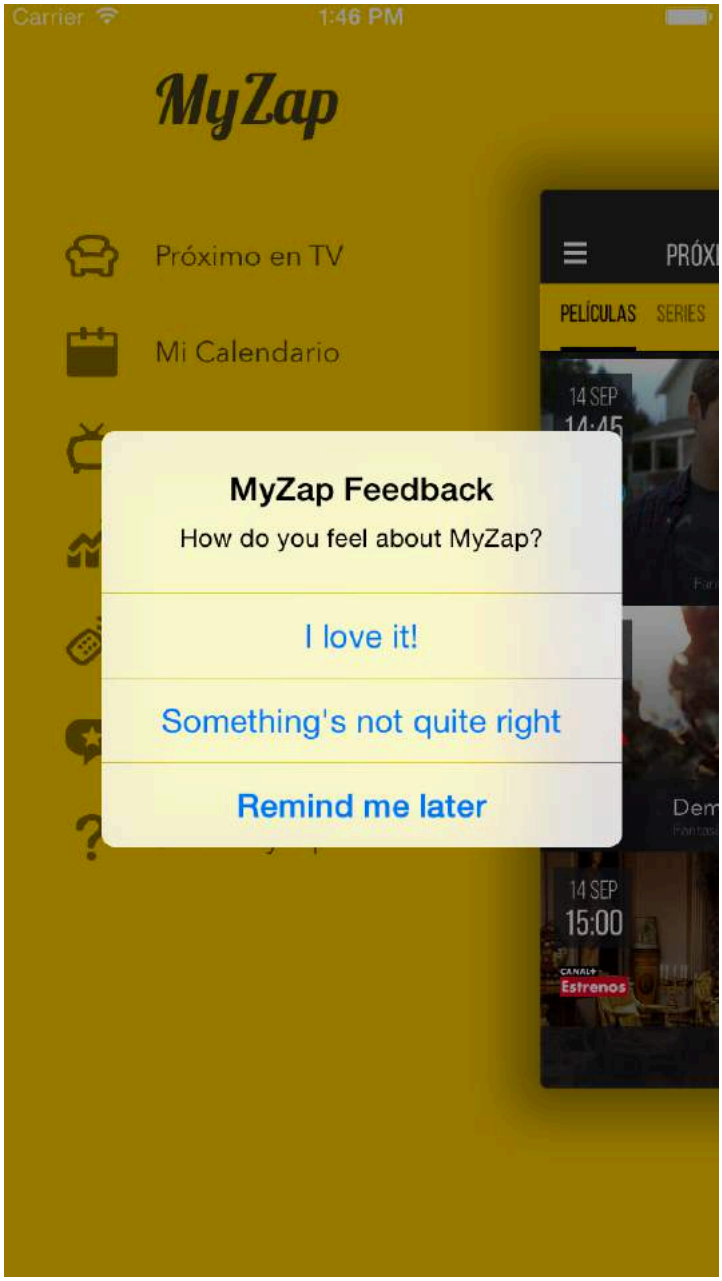
iTVDb es una librería que ofrece un servicio completo de conexión a la API de thetvdb para realizar una llamada y obtener la respuesta de forma parseada.
Con esta librería reducimos una enorme cantidad en el código y obtenemos la información relacionada con las series, programas y documentales de forma rápida y ordenada.

2.3.6.1.14 ASK4APPREVIEWS

Esta es una utilidad muy importante para cualquier aplicación, ya que resulta una vía fácil para generar reseñas de la aplicación en la App Store resultando una acción muy importante para el ASO (App Store Optimization). De esta forma generamos interés y posicionamos la aplicación para obtener más descargas.

Esta librería pregunta a los usuarios qué les parece la aplicación y que la valoren, si la valoración es correcta, se les pide que publiquen su reseña en la AppStore. En cambio, si su valoración es negativa, se les pide que escriban un email al desarrollador para ofrecer soporte al usuario y mejorar su valoración con respecto a la aplicación.

Considero esta librería muy importante ya que a través de una pregunta dirige a usuarios con valoraciones positivas a la App Store y con valoraciones negativas al desarrollador para así mejorar esta valoración.



Vista que muestra la funcionalidad de Ask4AppReviews

2.3.6.1.15 ASYNC

Esta librería hace uso del GCD (Grand Central Dispatch) para realizar acciones asíncronas en la aplicación. Como se ha comentado previamente, el GCD se usa para ejecutar tareas pesadas en segundo plano.

2.3.6.1.16 XMLDICTIONARY

Con XMLDictionary se puede parsear o generar de foma muy directa un XML a un NSDictionary. Esta librería resulta muy útil para parsear la respuesta que obtenemos en formato XML de las diferentes APIs usadas en la aplicación.

2.3.6.2 OTRAS LIBRERIAS O FRAMEWORKS

Existen en el proyecto otras librerías y frameworks que se han incorporado directamente y necesitan actualización manual ya que no están incluidos en Cocoapods.

2.3.6.2.1 REALM

Realm es un motor de bases de datos que nace con la intención de sustituir a SQLite o Core Data, para implementar bases de datos de manera sencilla en apps desarrolladas con Android y el sistema iOS.

Con esta librería se obtienen los métodos necesarios para manejar objetos Realm y acceder a la base de datos.

Se explica con más detalle en un punto posterior.

2.3.6.2.2 CRASHLYTICS

Esta herramienta nos simplifica mucho la tarea de seguir los crashes de nuestra aplicación y nos permite tener centralizada toda la información de una manera muy intuitiva.

Se explica con más detalle en un punto posterior.

2.3.6.2.3 GOOGLE ANALYTICS

Google Analytics es una herramienta que nos permite analizar lo que hacen los usuarios que se descargan la aplicación detectando las funcionalidades que más se utilizan, las visitas que se realizan en cada vista y un gran número de datos interesantes para el análisis y mejora de la app.

Se explica con más detalle en un punto posterior.

2.3.6.2.4 CVCALENDAR

CVCalendar es una librería que muestra un calendario completo para mostrar eventos que hemos guardado previamente. Esta vista se incluye en la sección Agenda de la aplicación para mostrar las películas y series que se han guardado para que nos avise antes de empezar.

Calendario y tabla que muestra las películas guardadas

Tiene multitud de propiedades totalmente customizables como marcadores de colores, vista de calendario en meses o semanas, separación en píxeles entre días y semanas, color de background, etiquetas, animaciones en la selección de días…

Contiene un conjunto de propiedades y protocolos que se deben implementar en el proyecto para recoger todas las acciones que proporciona esta librería.

2.3.6.2.5 HTMLPARSER

Esta librería se usa para el parseo html en la sección de audiencias. Con HTMLParser obtenemos un conjunto de arrays y diccionarios con las diferentes etiquetas y contenido de la web. Podemos realizar búsquedas para obtener el código html que más nos interese para mostrarlo en la aplicación.

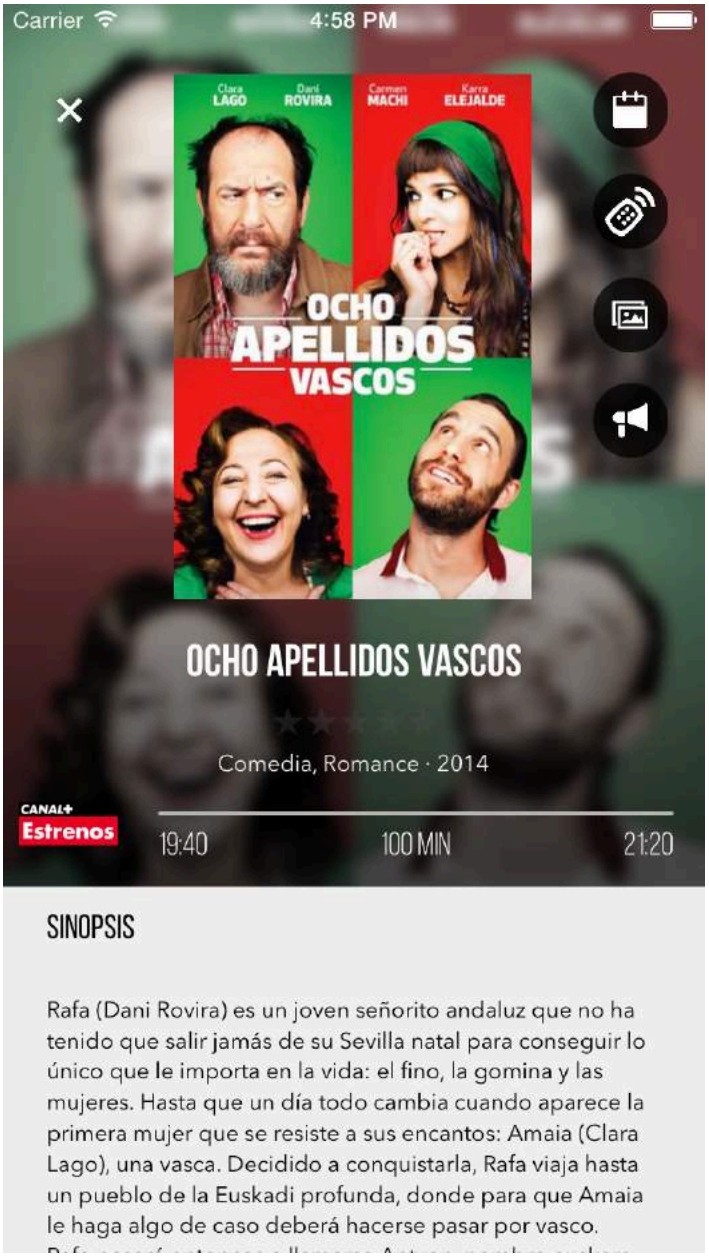
2.3.6.2.6 UIIMAGEEFFECTS

UIImageEffects es una librería escrita en Swift que extiende la clase UIImage y se encarga del tratamiento de imágenes para aplicar efectos Blur y Tint usando frameworks de Apple como vImage, Quartz y UIKit. vImage es usado para el procesamiento de imágenes sin necesidad de escribir código adicional.

Las funciones definidas en UIImageEffects son:

```
public extension UIImage {
    public func applyLightEffect() -> UIImage? {...}

    public func applyExtraLightEffect() -> UIImage? {...}

    public func applyDarkEffect() -> UIImage? {...}

    public func applyTintEffectWithColor(tintColor: UIColor) -> UIImage? {...}

    public func applyBlurWithRadius(blurRadius: CGFloat, tintColor: UIColor?,
        saturationDeltaFactor: CGFloat, maskImage: UIImage? = nil) -> UIImage? {...}
}
```

y son usadas en diferentes partes de la aplicación como el fondo de la vista de detalle de una serie o película.



Uso de UIImageEffects para el efecto Blur del fondo de vista

2.3.6.2.7 SWIFTYJSON

SwiftyJSON se encarga de realizar la conversión de un texto en formato JSON a un array. Es usado principalmente para mostrar y guardar de una forma organizada en la base de datos, las respuestas que se reciben de las llamadas a las diferentes APIs.

```
let json = JSON(responseObject)
let realm: RLMRealm = RLMRealm.defaultRealm()

if (!realm.inWriteTransaction) {
    realm.beginWriteTransaction()
}
println(json)

for var i = 0; i < json["data"]["channels"].count; ++i {

    let channel = Channel()

    channel.id = json["data"]["channels"][i][0].intValue
    channel.name = json["data"]["channels"][i][1].stringValue
    channel.type = json["data"]["channels"][i][2].stringValue
    channel.image = json["data"]["channels"][i][3].intValue
    Channel.createOrUpdateInDefaultRealmWithValue(channel)
}

realm.commitWriteTransaction()
```

Como se muestra en la imagen extraida, la respuesta de la API para obtener los canales de televisión es convertida mediante JSON() obteniendo un array que posteriormente se guarda en Realm.

2.3.6.2.8 CAPSPAGEMENU

CAPSPageMenu usa diferentes vistas para realizar un paginado de forma dinámica que podemos mover con *Pan Gestures*. Además incorpora un menú superior para cambiar de vista de forma rápida.

Este menú es totalmente customizable y está construido sobre un *ScrollView* con paginado para poder cambiar de vista mediante un *Swipe* con el dedo.

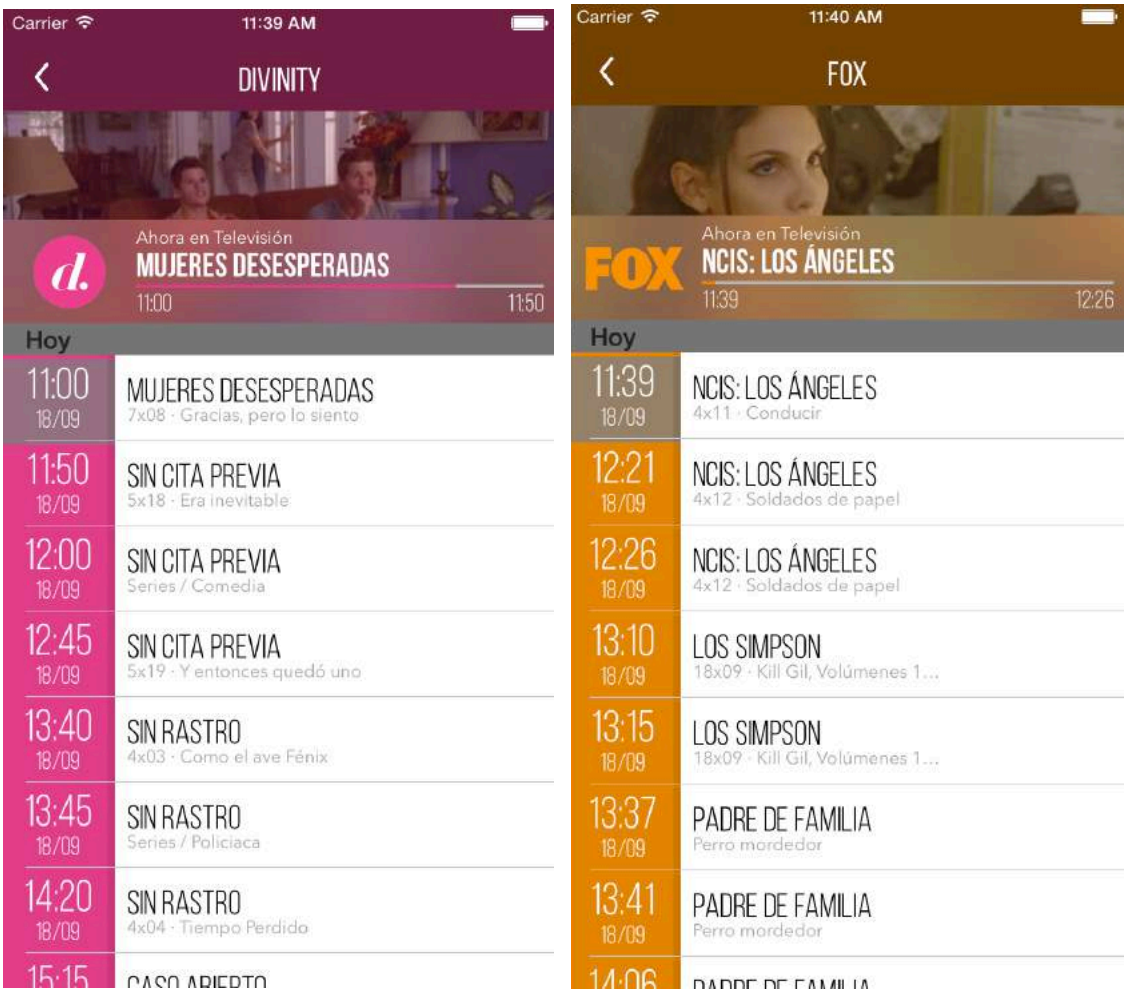
Paginado horizontal en la vista principal que muestra una lista de películas, series, programas o documentales

Como se puede comprobar en la imagen, esta librería se usa para la página inicial de la aplicación donde diferenciamos varias secciones: Películas, Series, Programas y Documentales.

2.3.6.2.9 LECOLORPICKER

LEColorPicker es una librería realmente útil para la mejora del diseño de la aplicación. Esta nos permite obtener la un color medio en función de la imagen. Resulta muy útil para obtener una gama de colores a partir de una imagen.

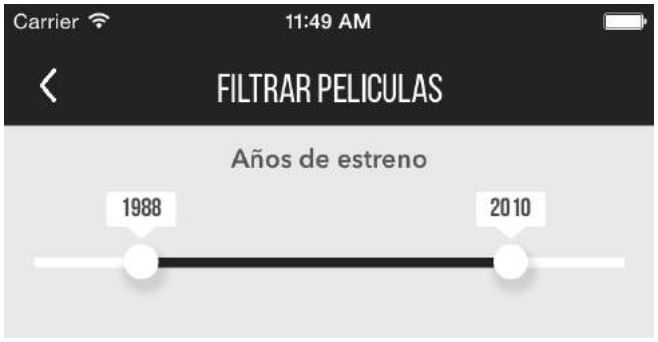
Esta funcionalidad es usada para el diseño de la vista de programación de cada canal. Se obtiene un tema de colores dependiendo de la imagen del logo del canal que sirve para pintar la vista.



Mediante LEColorPicker obtenemos una paleta de colores diferente dependiendo del icono del canal

2.3.6.2.10 MJPSLIDER

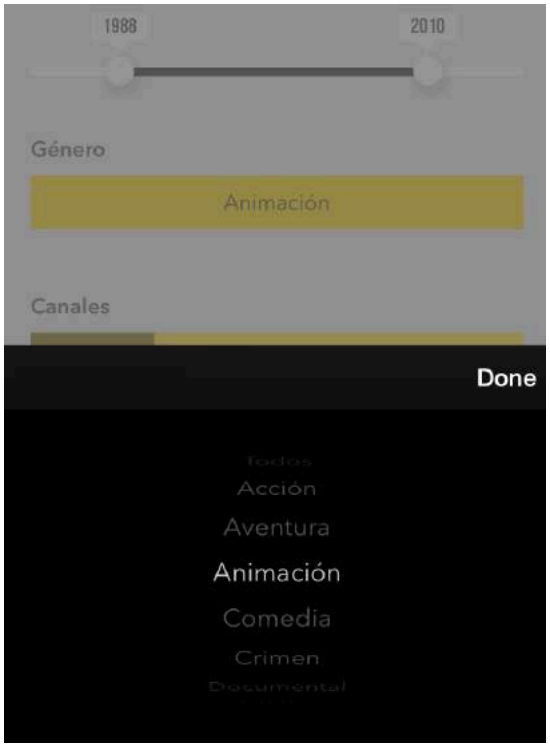
MJPSlider es un objeto UISlider customizado que muestra los valores máximo y mínimo.



UISlider customizado por la librería MJPSlider

2.3.6.2.11 MMPICKERVIEW

MMPickerView es un objeto UIPickerView customizable



UIPickerView customizado por la librería MMPickerView

2.3.6.2.12 NPSEGMENTEDCONTROL

Esta librería permite obtener un selector customizable a partir de un Array de Strings.

2.3.7 BASES DE DATOS (REALM)

El uso de *CoreData* desarrollado por Apple para almacenar información en una base de datos, es un framework maduro y confiable, pero complicado de usar. Deja de ser intuitivo y difícil de usar para varios threads a la vez. Por ello se crean otras herramientas que facilitan el uso de una base de datos para el guardado de información.

Realm es un motor de base de datos pensado para ser utilizado en el desarrollo de aplicaciones móviles tanto para Android como para iOS. Este sistema permite crear base de datos relacionales de forma sencilla y gratuita.

Una de las ventajas más importantes de Realm es que puede ser usado en diferentes lenguajes (Java, Objective-C y Swift). Realm además trabaja con ORM (Object Relational Mapping) que crea una base de datos orientada a objetos virtual, sobre la base de datos relacional, esto posibilita el uso de las características propias de la orientación de objetos, sin necesidad de usar lenguaje SQL para realizar consultas a la base de datos.

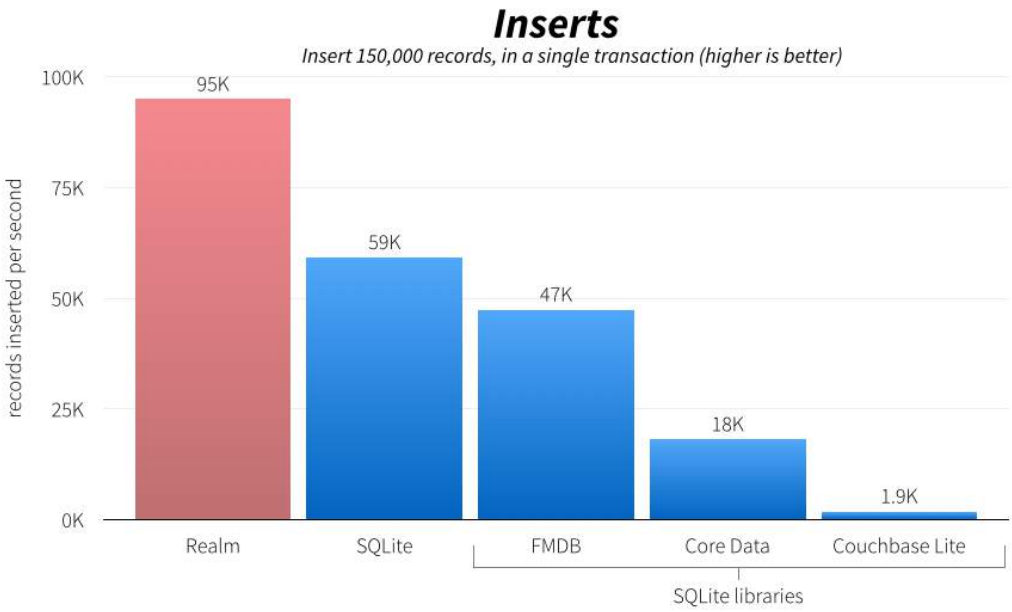
Para el proceso de escritura en la base de datos obtenemos *defaultRealm* para usar el Realm por defecto (si no hemos creado varias bases de datos). Este método crea (u obtiene si ya existe) una base de datos llamada *default.realm* en el directorio “Documents” de la aplicación.

Lo importante es que si mandamos llamar al mismo Realm varias veces en el mismo thread, se reusa la instancia que se creó la primera vez.

```
func saveInRealm(responseObject: AnyObject) {

    let json = JSON(responseObject)
    let realm: RLMRealm = RLMRealm.defaultRealm()

    if (!realm.inWriteTransaction) {
        realm.beginWriteTransaction()
    }

    for var i = 0; i < json["data"]["channels"].count; ++i {

        let channel = Channel()

        channel.id = json["data"]["channels"][i][0].intValue
        channel.name = json["data"]["channels"][i][1].stringValue
        channel.type = json["data"]["channels"][i][2].stringValue
        channel.image = json["data"]["channels"][i][3].intValue
        Channel.createOrUpdateInDefaultRealmWithValue(channel)
    }

    realm.commitWriteTransaction()
}
```

Como vemos en la imagen, iniciamos la transacción con el método *beginWriteTransaction()* de modo que bloqueamos a Realm en modo escritura (desbloqueado se encuentra en modo lectura). Con el método *addObject()* o *createOrUpdateInDefaultRealmWithValue()* se agregan objetos al Realm. Por último con *commitWriteTransaction()* concluye la transacción y se libera el bloqueo y Realm vuelve a ser solo de lectura.

El hecho de no usar SQLite, hace que el equipo de Realm pudiera optimizar la velocidad al acceso a la base de datos incluyendo las escrituras.



Para la obtención de objetos guardados en Realm se usa el método *objectsWhere()*

```
class Channel: RLMObject {

    dynamic var id = 0
    dynamic var name = ""
    dynamic var type = ""
    dynamic var language = ""
    dynamic var image = 0

    override class func primaryKey() -> String! {
        return "id"
    }

    class func ById(id:Int) -> Channel? {

        let channels: RLMResults = self.objectsWhere("id == \(id)")

        return channels.lastObject() as? Channel
    }

    func getShows() -> RLMResults? {

        return ZapShow.objectsWhere("channelID == \
            (self.id)").sortedResultsUsingProperty("inittime", ascending: true)
    }
}
```

Como se puede observar en la imagen. Se ha incluido un método llamado *getShows()* que obtiene los objetos guardados en Realm. En este método se hace uso de la función *objectsWhere()* para realizar la búsqueda de objetos guardados. Se usa *sortedResultsUsingProperty()* para ordenar los resultados. Para obtener todos los objetos se usa *allObjects().*

El tipo de objetos *Channel* que se guarda en Realm deberá heredar obligatoriamente de *RLMObject*.

2.4 PRUEBAS

2.4.1 BETA TESTING y CONTROL DE ERRORES

Las pruebas y control de errores es de las partes más importantes en el desarrollo de una app. Esta fase de depuración permite corregir fallos en el diseño, la usabilidad y errores de código en la aplicación.

2.4.1.1 TESTFLIGHT

Testflight es una plataforma que permite el testeo de aplicaciones sin necesidad de subirlas a la AppStore. En la actualidad se permite la prueba de aplicaciones hasta con 25 usuarios de manera interna y hasta 10 dispositivos cada uno, así como invitar hasta 1.000 usuarios externos por app, sin necesidad de registrarlos en nuestro entorno de desarrollo previa revisión de la aplicación por Apple, en un proceso similar al que se pasa cuando se sube una app en la App Store.

Hasta este año, el servicio dependía de dar de alta uno a uno los dispositivos para pruebas en el programa de desarrolladores online, generar un perfil, firmar la app e incluirla en el entorno de desarrollo *XCode*. Tras la compra del servicio por parte de Apple, todo este proceso se ha eliminado y es más transparente y ha cambiado el funcionamiento con *iTunes Connect* haciéndolo más intuitivo.

Una vez subida la aplicación al *iTunes Connect*, podemos habilitar el beta testing mediante "**Enable Testflight Beta Testing**" y en "**Internal Testing**" elegimos a quién de nuestros usuarios internos registrados queremos enviar la app marcándolo simplemente. Para añadir más usuarios, necesitaremos darlos de alta en "**Users and Roles**" del propio *iTunes Connect*.

Los usuarios, a su vez, deberán tener la app Testflight para iOS instalada en su dispositivo.

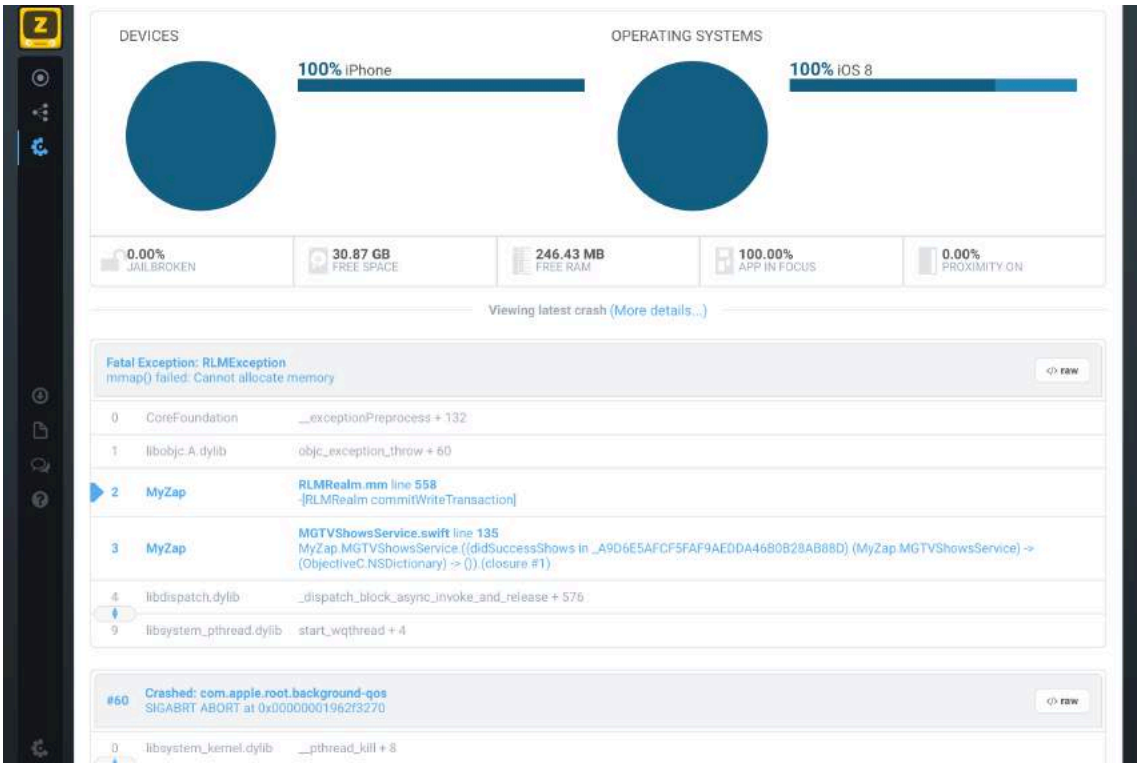
2.4.1.2 CRASHLYTICS

Una de las cosas más interesantes que como desarrolladores nos interesa saber es cuando falla nuestra aplicación y sobre todo el lugar exacto donde está fallando. Esto lo podemos conseguir teniendo un sistema de envío de crashes que son ficheros que el dispositivo genera cuando se produce una excepción no controlada.

Entre la información que nos aporta podemos destacar el “identifier”, la versión, la fecha y hora en la que se produjo, la versión del dispositivo y la excepción. También vemos una pila con una serie de posiciones de memoria.

Entonces tenemos dos problemas que resolver manualmente. El primero es crear un envío del fichero de crash cuando este se produce y el segundo descifrar la información del crash.

Aquí es donde **Crashlytics** nos puede echar una buena mano, ya que se trata de un servicio que nos proporciona una forma muy elegante y transparente de recibir los crashes de los usuarios y una forma visual de descifrarlos en forma de dashboard en su aplicación web.



Además proporciona estadísticas del número de crashes divididos por distintas características como versión, sistema operativo, etc…

Simplemente deberemos distribuir entre los distintos Beta Testers (registrados previamente en Crashlytics) la aplicación para la búsqueda de errores.

2.4.2 ANÁLISIS ESTADÍSTICOS DE USO

Las herramientas de supervisión de aplicaciones miden la experiencia de los usuarios, el rendimiento de la app, el tiempo de respuesta del usuario ante determinadas acciones. De esta manera podemos conseguir saber dónde se encuentra el usuario y qué es lo que quiere hacer.

Por ello, es importante saber en un informe detallado por qué los usuarios abandonan la aplicación o qué vistas son las que más éxito tienen, de forma,

que se pueda dirigir al usuario para la comercialización de productos, anuncios o ventas in-app.

2.4.2.1 GOOGLE ANALYTICS

Google Analytics es la herramienta más conocida que nos permite analizar qué hacen los usuarios detectando las funcionalidades que más se utilizan, las visitas que se realizan en cada vista y un gran número de datos interesantes para el análisis y mejora de la app. Entre otras cosas nos permitirá:

- Seguir el número de usuarios activos que utilizan la app
- Desde qué zonas geográficas acceden
- Uso que hacen los usuarios de determinadas funcionalidades
- Compras in-app
- El número y tipo de errores que hagan cerrar la app
- Muchas otras métricas…

Esta herramienta es la que hemos usado para el análisis de estos parámetros. Para integrar Google Analytics en la aplicación, hemos seguido una serie de pasos.

Primero, se ha dado de alta la app en el soporte web de Google Analytics. Con ello, obtenemos un Código de Seguimiento que nos permitirá registrar los datos y accesos a la aplicación.

Descargamos el SDK y lo incorporamos al proyecto añadiendo también los frameworks necesarios para su ejecución:

- libGoogleAnalytics.a
- CoreData.framework
- SystemConfiguration.framework

Para iniciar el seguimiento desde la aplicación se ha de indicar en el código la llamada a la librería y así registrar todos los eventos.

```
func setupGoogleAnalytics() {

    // Optional: automatically send uncaught exceptions to Google Analytics.
    GAI.sharedInstance().trackUncaughtExceptions = true

    // Optional: set Google Analytics dispatch interval to e.g. 20 seconds.
    GAI.sharedInstance().dispatchInterval = 20

    // Optional: set Logger to VERBOSE for debug information.
    GAI.sharedInstance().logger.logLevel = GAILogLevel.Verbose

    // Initialize tracker. Replace with your tracking ID.
    GAI.sharedInstance().trackerWithTrackingId("UA-35382675-6")
}
```

En cada vista generada de la aplicación incorporaremos una llamada para el registro de esa visita.

```
override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)
    self.screenName = "Schedule View"
}
```

Con Google Analytics podremos obtener los siguientes informes:

- **Audiencia**: Con este informe, se puede segmentar a los usuarios que entran en la aplicación dependiendo de la demografía, su localización, comportamiento (si son usuarios recurrentes), el dispositivo desde el cual acceden, el proveedor de servicios (Jazztel, movistar, Vodafone…) y mucho más.

- **Adquisición**: Nos proporciona información de quienes son y de donde vienen los usuarios. Si se han creado campañas de publicidad puede resultar muy útil si el flujo de usuarios procede desde estos medios u otros. Se trata de analizar el *mobil user* y actuar en consecuencia.

- **Comportamiento**: Se comprueba de forma meticulosa el flujo de comportamiento dentro de la aplicación. Una vez el usuario aterriza, se observará por qué vistas pasa, cuanto tiempo dedica a ellas y obtener una analítica completa de todo este comportamiento. Podemos

comprobar si este comportamiento del usuario es el esperado o si se pierde por la aplicación o debemos cambiar algo para mejorar la usabilidad.

- **Conversiones**: Podemos integrar los pagos e in-apps que se encuentran dentro de la aplicación para analizar las ventas y en consecuencia, las conversiones de visitas-ventas.

2.4.2.2 APP ANNIE

App Annie ofrece otro servicio web en la que analiza el rendimiento de las tiendas de aplicaciones y no tanto el de la propia aplicación. En este ámbito permite ver el rendimiento tanto de la venta de aplicaciones en la App Store como de las operaciones in-app. También permite analizar la captación de tráfico hasta la descarga de la aplicación.

En este caso, App Annie no cubre la parte de análisis de comportamiento del usuario o de errores.

Captura de pantalla de ejemplo de la plataforma App Annie

2.4.2.3 FLURRY

Flurry es una herramienta de Yahoo que analiza aplicaciones de iOS, Android, Blackberry, Windows Phone y web. Es una herramienta totalmente gratuita. Con ella podemos analizar todas la funcionalidades de la aplicación a través de eventos con los que estudiaremos las rutas que sigue cada usuario, segmentaremos la ubicación, versión del dispositivo…

Además, con Flurry, podemos analizar a la competencia en base a parámetros como la frecuencia de uso, la retención de usuarios…

2.5 PLAN DE COMERCIALIZACIÓN

Actualmente hay multitud de aplicaciones en la tienda App Store que finalmente compiten con la aplicación que se ha desarrollado. El objetivo del plan de comercialización o plan de marketing es tratar llegar a la cima y destacar entre todas las apps que compiten por la atención del consumidor.

Para conseguir la atención de los usuarios que circulan por la red, debemos hacer una serie de estrategias de marketing creativas.

En este apartado incluiremos los procesos de marketing más sencillos para hacer crecer el número de descargas de la aplicación.

2.5.1 NOMBRE DE LA APLICACIÓN Y CATEGORÍA

El nombre de la aplicación móvil debe de ser único y fácil de pronunciar. Es importante que el nombre incluya las keywords (palabras clave) más importantes del tema que estamos tratando y que sea en parte descriptivo de la aplicación.

En nuestro caso, la aplicación se llama **MyZap** (derivado de Zapping). Pero a este nombre podemos añadir: **La guía de televisión inteligente.**

Elegir la categoría correcta y la subcategoría es a veces complicado, ya que normalmente, las aplicaciones no se ajustan a una sola, y por ello, debemos hacer la pregunta: ¿Dónde se encuentran los clientes potenciales?. En este caso, esta claro que la categoría idónea será la de **Entretenimiento**.

2.5.2 ICONO DE LA APLICACIÓN

Crear un icono y capturas de pantalla es un aspecto fundamental en un plan de comercialización, ya que, la primera impresión será determinante en la elección de una u otra aplicación. El diseño visual es clave en la decisión del cliente.

El icono es lo que diferenciará la aplicación, y por ello, debe ser atractivo para destacar entre miles de iconos similares.

Para este proyecto, se ha desarrollado un icono mediante Photoshop e Illustrator que muestra un poco de qué va la aplicación.

Icono aplicación desarrollada MyZap

En la imagen se muestra una televisión con un carácter divertido y una Z (de Zap) en el centro.

2.5.3 LANDING PAGE Y/O BLOG

Para promocionar la aplicación se desarrollará una landing page de forma que se explique de forma clara todo lo que ofrece la aplicación y el potencial que tiene para convencer al cliente que la descargue.

Una landing page debe ser clara, simple y directa.

Otra opción que podemos añadir a la landing page, es crear un blog y compartir los contenidos.

El marketing de contenidos es muy importante y más hablando de ecommerce enfocado a las aplicaciones móviles. Es vital redactar contenidos interesantes para los posibles clientes. En el caso de este proyecto, es importante generar contenido sobre televisión, películas que van a emitir, programas, series…

Todo esto atraerá a un público interesado en descargar la aplicación.

Una buena forma de empezar a promocionar la aplicación es construir un blog un mes antes del lanzamiento, y comentar en ella, el proceso de construcción, las ventajas de la aplicación, metas, objetivos y generar un debate y expectativas entre los usuarios que visiten el blog.

2.5.4 REDES SOCIALES Y RESEÑAS EN BLOGS

Compartir y promocionar la aplicación en las redes sociales como Facebook, Twitter, Linkedin y Google Plus es interesante para publicar diariamente contenido interesante y un enlace a la aplicación a la App Store.

Conseguir reseñas en medios de comunicación y blogs es un ejercicio a largo plazo que trae mucho esfuerzo, pero no por ello es menos importante. Esta es una de las mejores formas para promocionar la aplicación. Esta totalmente probado que se consiguen aumentar las descargas apareciendo en páginas con alta reputación. Con una sola reseña de un experto se puede convertir la aplicación en todo un éxito.

Para conseguir esto, se enviarán notas de prensa tanto a los medios online como offline, se contactará con blogs interesantes sobre aplicaciones y se desarrollará todo el material necesario para difundir como: imágenes, notas de prensa, biografía del desarrollador, logos…

2.5.5 VIDEO PRESENTACIÓN

Un video de presentación es interesante para mostrar al usuario qué es lo que realiza la aplicación. No debe ser un video tutorial, más bien, debe de ser divertido, creativo, que invite a la reflexión o cuente una historia. Es importante que el usuario se quede con ganas de descargar y saber más sobre la aplicación.

2.5.6 ASO (APP STORE OPTIMIZATION)

ASO (App Store Optimization) es el SEO (Search Engine Optimization) adaptado al móvil. El hecho de que las tiendas de aplicaciones sean cada vez más competitivas, hace que el desarrollador deba esforzarse cada vez más en hacer visible su aplicación y dispute los primeros puestos en su categoría. Para ello deberemos seguir todos estos puntos que hemos hablado anteriormente y buscar las keywords oportunas que sirvan para promocionar la aplicación.

El uso de herramientas de análisis como las descritas anteriormente nos ayudará a matizar ciertos aspectos de la aplicación para la mejora de las descargas.

Todo este proceso de marketing entra dentro de lo que se llama ASO.

3. CONCLUSIONES Y TRABAJOS FUTUROS

3.1 CONCLUSIÓN

Con la elaboración de este documento, se ha conseguido explicar la necesidad de desarrollar una aplicación para dispositivos móviles que permita organizar y ordenar la programación de televisión en base a nuestros gustos, operador contratado y tipo de contenido requerido (series, películas, documentales, programas).

A continuación, se han analizado las diferentes tecnologías y plataformas que permiten desarrollar la aplicación, y con los estudios realizados, se ha determinado crearla de forma nativa para evitar realizar actualizaciones constantes de la interfaz. El uso de servidores web, ayuda a proveer todos los contenidos necesarios para completar la información necesaria.

Tras un estudio pormenorizado, decidí desarrollar la aplicación para dispositivos iOS. El objetivo de comercialización y obtención final de ingresos, es más sencillo en esta plataforma y el desarrollo de nuevos dispositivos dentro de la compañía como el Apple Watch o Apple Car hace posible el desarrollo, actualización y comercialización de esta aplicación para nuevos consumidores potenciales.

Una vez escogida la plataforma, se decidió desarrollar un método Freemium para la obtención de ingresos dentro de la aplicación. Otras opciones de pago no eran viables, ya que el contenido de esta aplicación es gratuito en internet, y el servicio que da la aplicación es de organización de resultados de búsqueda en la programación de televisión. Por ello, se decide, incluir un sistema de pago dentro de la aplicación para eliminar anuncios de publicidad que aparecen de forma aleatoria y desbloqueo de otros servicios (avisos, notificaciones…)

Por último se ha decidido usar el lenguaje Swift frente a Objective-C debido a la mejora sustancial que supone en el desarrollo de aplicaciones móviles de plataforma Apple. Con Swift, los programadores dedican menos tiempo a escribir código de cabecera y pueden dedicar más tiempo a la lógica de la aplicación a crear. Swift elimina el trabajo repetitivo y mejora la calidad del código, comentarios…

El desarrollo de la aplicación ha supuesto el aprendizaje previo del lenguaje Swift, diseño previo mediante prototipados de la aplicación y el estudio de la usabilidad para la mejora de la navegación dentro de la aplicación. Ha sido importante el uso de herramientas de versiones git para el control y seguridad en las diferentes etapas de desarrollo. El uso de servicios web, y diseño de clases y objetos, así como la estructura MVC (Modelo Vista Controlador) escogida ha sido importante para el desarrollo del código de la aplicación.

Se ha realizado una técnica de scrapping para la obtención de información de una página web, y se han usado y desarrollado diferentes librerías que permiten, reducen y facilitan el código necesario para mejorar las funcionalidades de la aplicación.

Posteriormente, se ha elaborado un manual de uso con el fin de que sirva de guía a los posibles usuarios de la aplicación y que muestra en detalle las funcionalidades más importantes.

Las pruebas y control de errores de la app, han permitido corregir ciertas funcionalidades de la aplicación que no estaban claras. En diversos momentos del desarrollo, se han conseguido corregir ciertos errores importantes previos a la publicación de la app en la tienda App Store. Aunque se ha completado una primera fase de test, es necesaria una mayor fase de testing con diferentes usuarios que permitan descubrir nuevos errores y mejorar de esta forma la aplicación.

Finalmente, se han detallado los aspectos más importantes del plan de comercialización que permiten alcanzar los puestos más altos en los rankings de descargas de la tienda App Store y destacar sobre otras aplicaciones de la competencia.

En definitiva, cabe destacar, que se han alcanzado los objetivos que se pretendían en un principio de forma satisfactoria con el desarrollo de este TFG. Por un lado, el aprendizaje del lenguaje Swift usado para el desarrollo y por otro el cumplimiento y uso de todas las fases que conlleva el desarrollo de la aplicación, desde su diseño inicial hasta el plan de comercialización. Se han conseguido desarrollar diferentes técnicas importantes para la obtención de datos como el Web-scrapping y el uso de APIs de forma satisfactoria. Todo esto ha llevado como resultado una aplicación útil, fácil de usar y para todos los usuarios que hagan uso de la televisión como canal de entretenimiento.

3.2 TRABAJOS FUTUROS

Es importante en este punto del desarrollo, analizar cuales pueden ser las mejoras que se pueden llevar a cabo en la aplicación para mejorar sus funcionalidades y su comercialización.

De forma general, hubiera sido interesante adaptar esta misma aplicación para plataformas Android. Aunque este proceso supone un nuevo desarrollo de software, la mayor parte del trabajo de investigación, diseño y test ya está realizado y por lo tanto, sólo se dedicaría el esfuerzo en traducir la aplicación en lenguaje Java.

En cuanto a la aplicación en particular, es interesante realizar una mejora en la obtención de datos sobre los programas de televisión de los diferentes canales. Hasta ahora se ha usado un servicio web ajeno que permite la obtención de toda esta información en su conjunto. Pero puede suceder que el servicio web deje de funcionar o deje de prestar servicio. En este caso, la aplicación sería inservible. Es interesante desarrollar un servicio web desde un servidor propio, que realice técnicas de scrapping de diferentes canales por separado o de un blog donde obtengamos la programación por completo. De esta manera, si se modifica la página, desde el propio servidor se puede arreglar sin tener que recurrir a una actualización de la aplicación de forma urgente.

Otras funcionalidades que por falta de tiempo no se han podido realizar, son “Recomendación del día” donde la aplicación sugiera en base a nuestros gustos una película, serie, programa o documental. Por otro lado “Ver programas emitidos” nos permitiría visualizar desde la app los programas emitidos por otras cadenas. Esta última funcionalidad está pensada para realizar scrapping de las diferentes webs oficiales para obtener los videos de contenidos de televisión.

También es interesante mejorar la promoción en redes sociales que permiten una mayor valoración y nuevas reseñas en la red. Para ello se ha pensado en integrar Twitter para realizar comentarios de programas, de manera que en la vista de detalle del programa podamos comentar en una ventana aparte lo que ocurre en tiempo real. Esto generará comentarios en las redes sociales y un flujo de entrada de usuarios a través de estas.

Ya se ha desarrollado una sección para conocer la audiencia oficial de los diferentes programas de televisión en los diferentes días, pero sería interesante conocer lo que se llama como *Audiencia Social*. Mediante Twitter podemos calcular el grado de interés de un programa de televisión en base a unos cálculos proporcionales al número de veces que un comentario se ha compartido y se ha marcado como favorito. Para ello, se calcula el *Engagement Rate* (ER), un ratio que sirve para valorar la acogida de los contenidos, para comparar con otros competidores y para saber si el crecimiento de la comunidad de usuarios es cualitativo. Este ER se calcula en base a una serie de fórmulas que deben de ser estudiadas.

4. ANEXOS

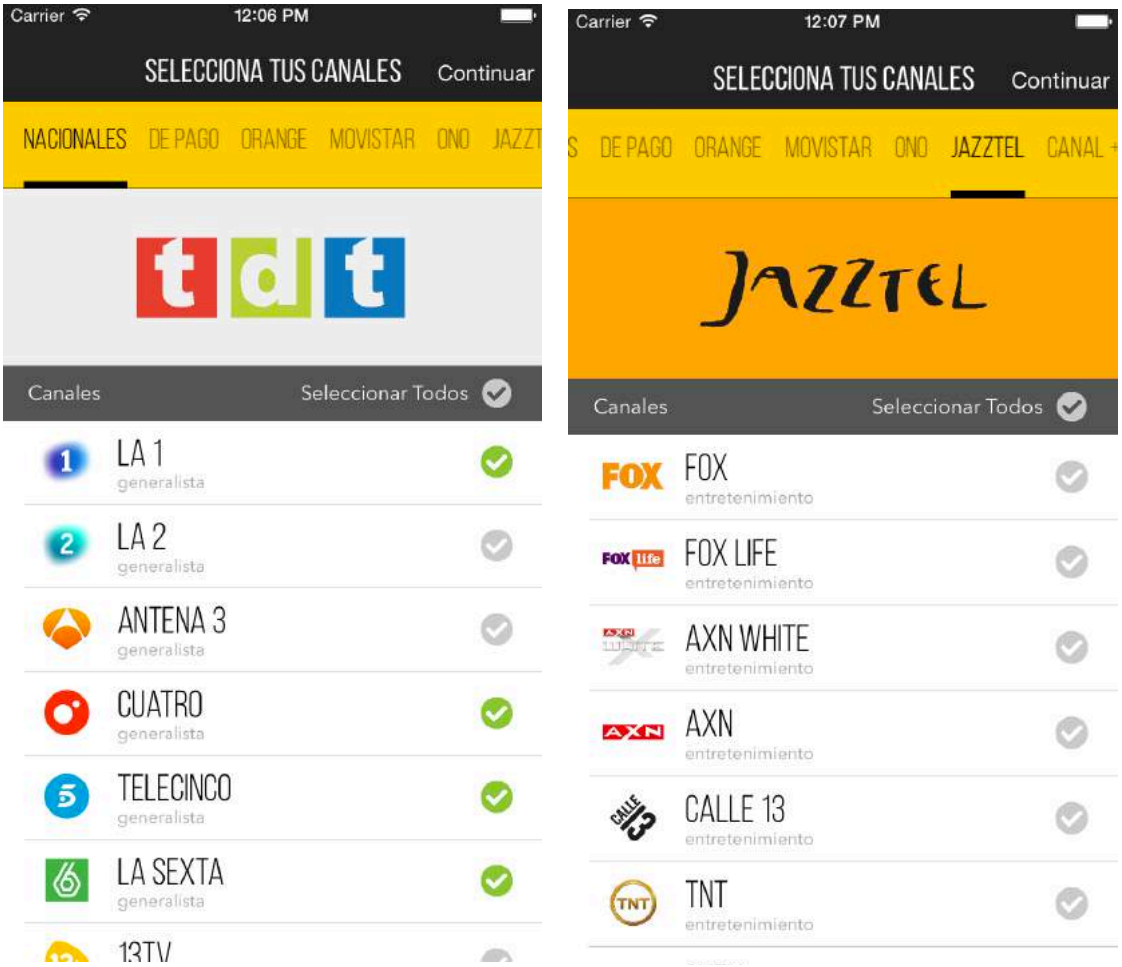
4.1 MANUAL DE USUARIO

Este capítulo pretende que sirva de guía para el uso de la aplicación. A continuación, se describen las distintas funcionalidades y características de las diferentes vistas de la aplicación desarrollada en este TFG sin entrar en aspectos técnicos.

Pantalla inicial de la aplicación

4.1.1 PRIMERA PUESTA EN MARCHA

Nada más abrir la aplicación, nos encontraremos con la selección de nuestros canales favoritos o canales del operador que tengamos contratado.

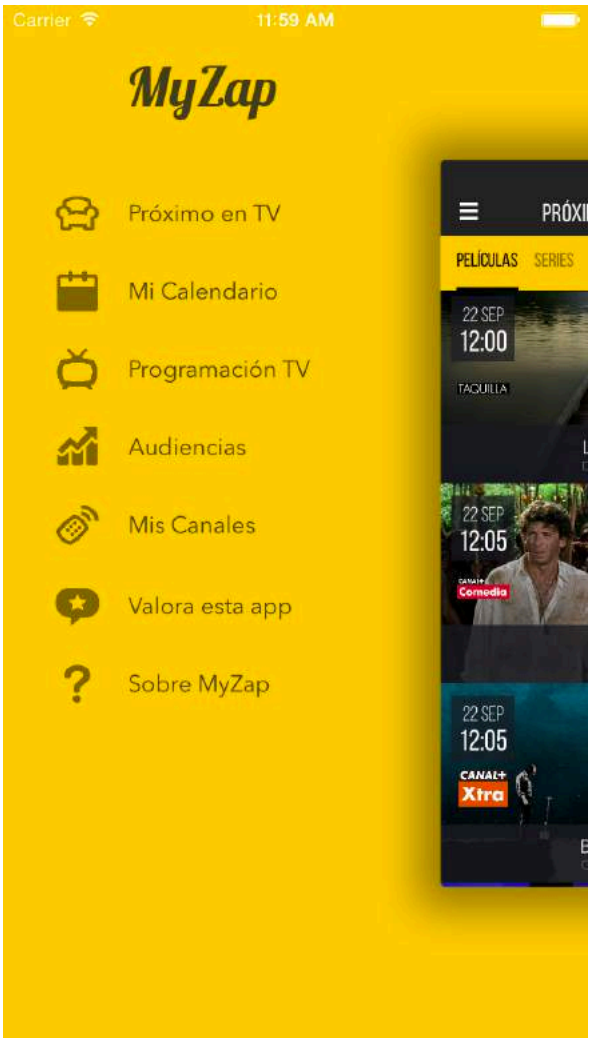


Vistas de selección de canales de televisión. Canales Nacionales en la primera y canales del operador Jazztel en la segunda

Nada más seleccionar los canales, la aplicación filtrará todo su contenido con respecto a estos canales. Esta selección también se puede cambiar desde la sección Mis Canales.

4.1.2 NAVEGACIÓN Y MENÚ INICIAL

Desde la vista inicial, se puede acceder al menú mediante un botón de navegación en la parte superior izquierda. Desde este menú, se puede acceder a todo el contenido de la aplicación de forma rápida y fácil.

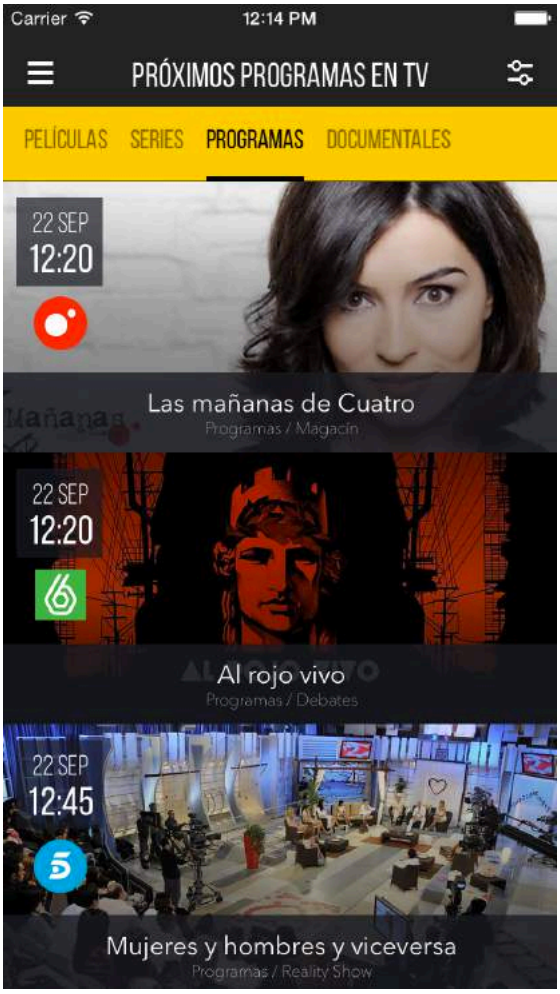


Vista que muestra el menú de la aplicación

En los próximos apartados se detallarán cada una de las vistas desarrolladas para una mejor usabilidad.

4.1.3 PRÓXIMO EN TELEVISIÓN

En esta vista, el usuario podrá observar las películas, series, programas y documentales que van a hacer próximamente y anotárselas en el calendario para verlas posteriormente en televisión. Estos programas están ordenador por fecha y hora.

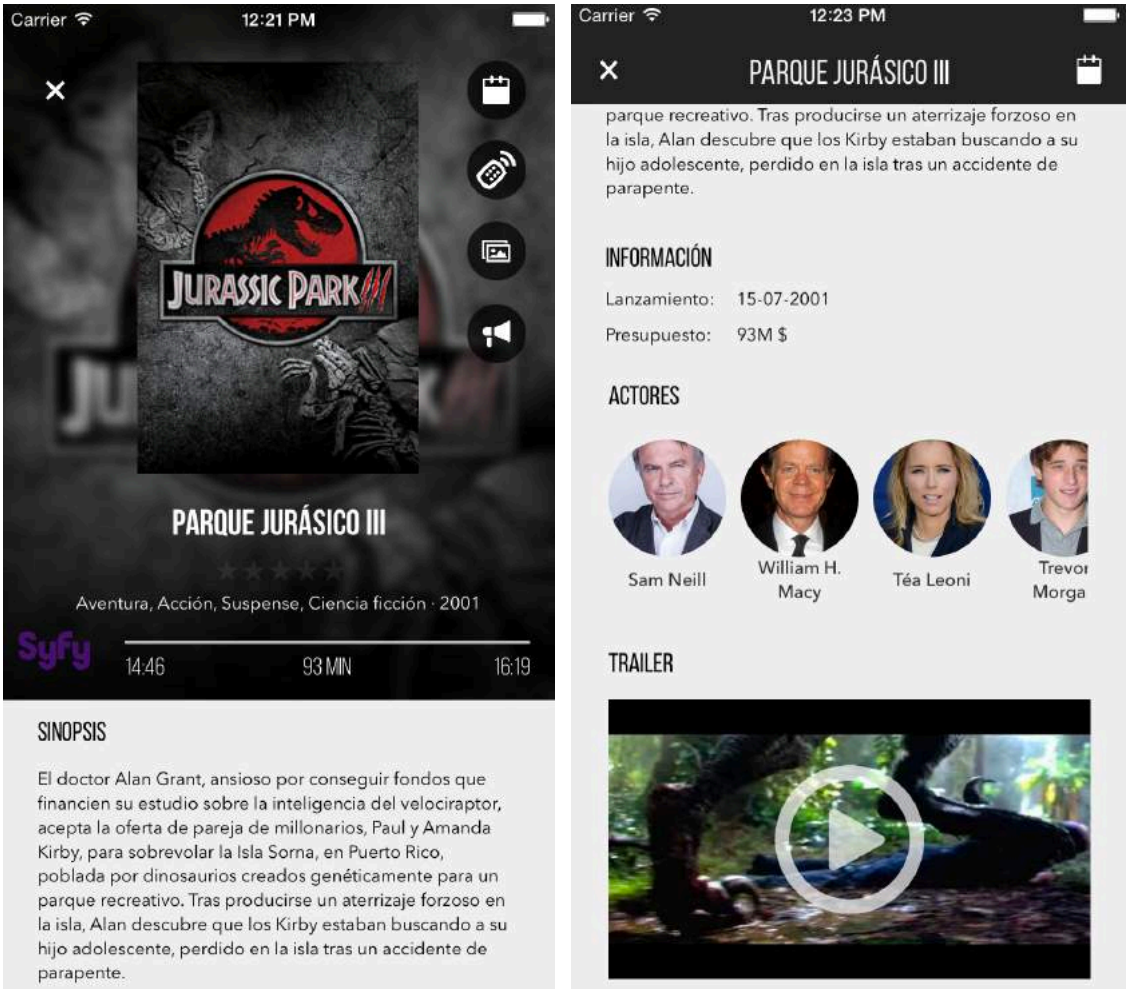


Vista principal donde se muestran los programas

Desde el botón superior derecho se pueden filtrar los contenidos que más nos interesen

Vista que muestra los filtros que se pueden aplicar

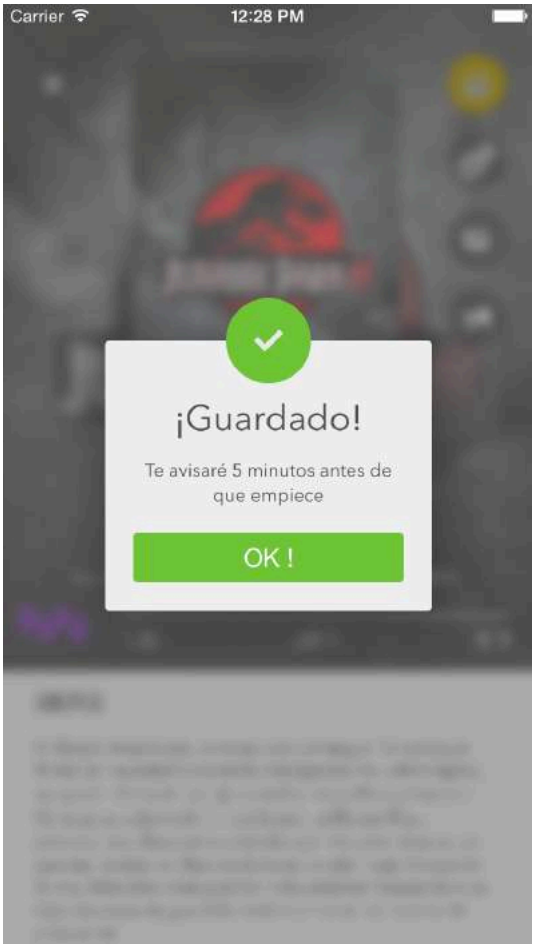
Si presionamos cada uno de los programas mostrados, podremos obtener información detallada, duración, géneros, sinopsis de la película, actores, tráiler que podremos ver en el momento…



Vista detallada de una película seleccionada

Desde esta vista además podremos realizar una serie de acciones directas desde los botones que tenemos en la parte superior derecha.

En las próximas imágenes, podemos observar las acciones del primer y segundo botón.



Guarda el aviso y lo añade al calendario de la aplicación

En el primer botón podremos guardar o anotar en el calendario la película o programa para que, mediante una notificación a través del móvil, nos avise 5 minutos antes de empezar.

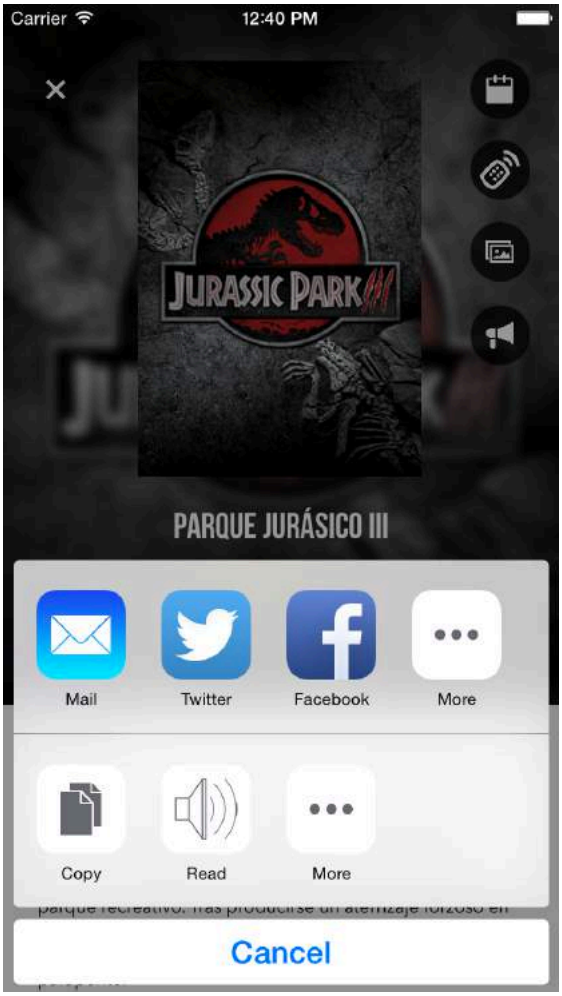
Vista que muestra la lista de próximas emisiones de la misma serie

En el segundo botón podremos ver las próximas emisiones que van a realizar la película o serie, de forma que no nos perdamos ningún capítulo.

En las siguientes imágenes podemos ver la acción del tercer y cuarto botón

Vista que muestra una lista de imágenes de la película

El tercer botón muestra una colección de imágenes o logotipos de la película o programa en una tabla.

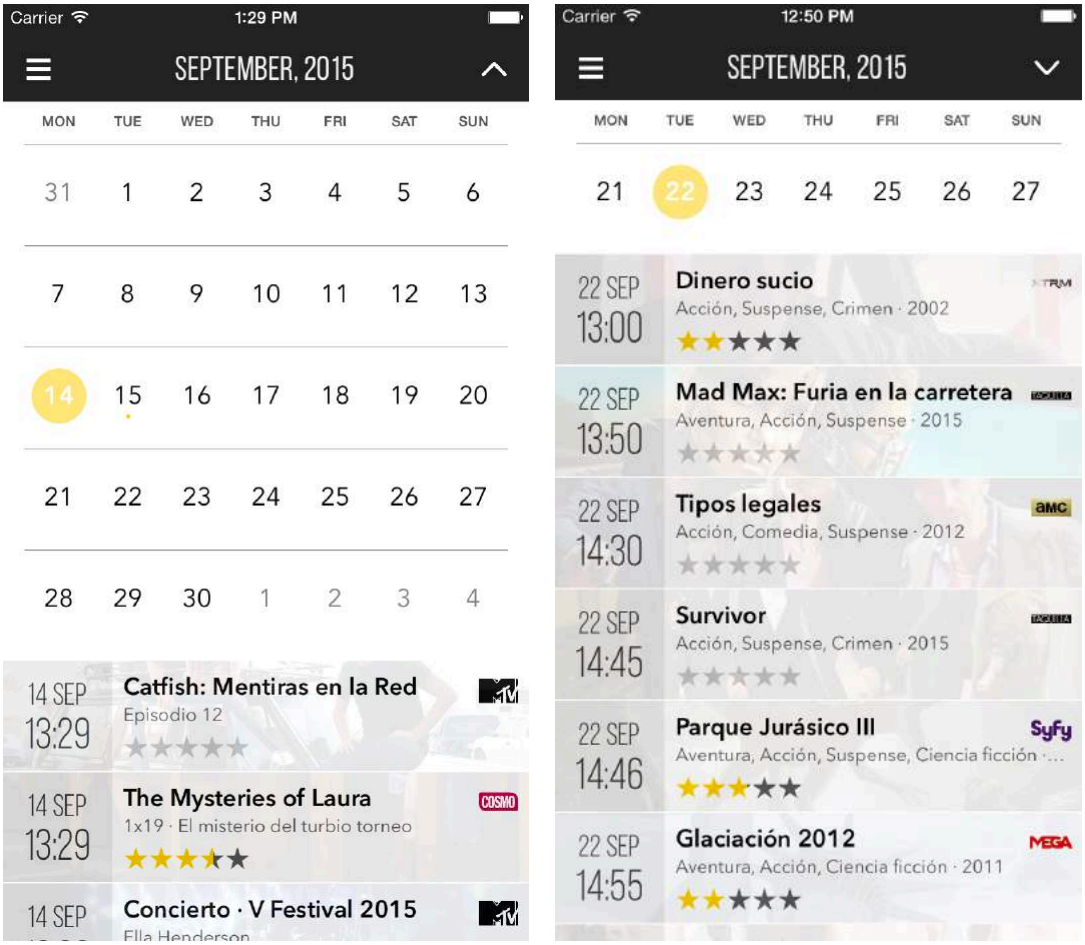


Botón de actividad que muestra las distintas opciones que podemos hacer con esta película

El cuarto botón sirve para compartir en las diferentes redes sociales. Además se añade en esta vista la función de lectura para que lea de forma automática el contenido de la sinopsis del programa.

4.1.4 MI CALENDARIO

La vista de Mi Calendario podremos ver todas las películas, programas, series o documentales que hemos guardado para que nos notifique posteriormente. Es una forma organizada para ver los contenidos que tenemos pendientes de ver en televisión.



El calendario se puede expander o contraer sin cambiar de vista

Desde el botón superior derecho podemos expandir o contraer el menú de forma dinámica para dar más espacio a la tabla inferior con los detalles de los programas.

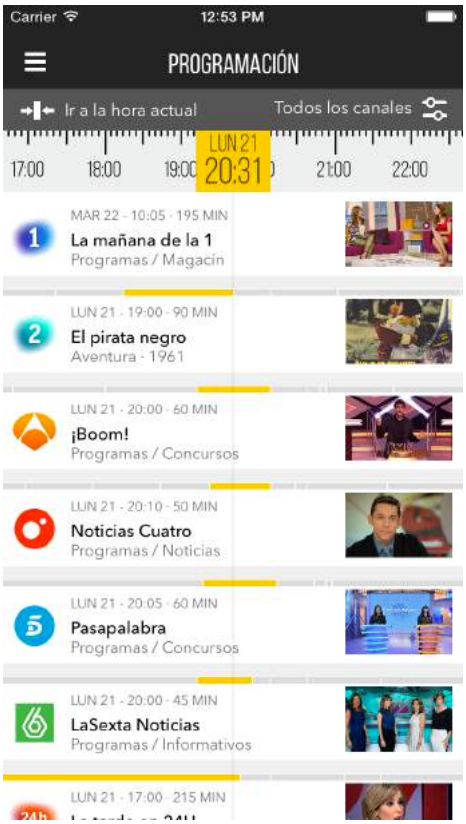
Seleccionando el día podremos observar los programas anotados para ver. También podemos deslizar el calendario a la derecha o izquierda para cambiar la semana o el mes, dependiendo si tenemos el calendario expandido o

contraído. Desde esta vista también se puede seleccionar uno de los programas para ver la vista de detalle de cada uno de ellos.

4.1.5 PROGRAMACIÓN DE TELEVISIÓN

En esta vista podemos ver de forma rápida toda la programación de forma dinámica. Deslizando con el dedo podemos mover el tiempo para ver qué van a emitir en ese momento en cada canal de televisión.

En esta misma vista también tenemos la opción de “Ir a la hora actual”, ya que una vez nos movemos en el tiempo, resulta tedioso volver deslizando exactamente hasta la hora actual. La otra opción es filtrar por “Todos los canales” o “Mis canales”.



Programación completa de televisión en forma horaria

Si seleccionamos cada una de las celdas de esta tabla, nos detallará la programación de cada canal con una vista personalizada, mostrando en la parte superior lo que se está emitiendo en este momento en televisión desde ese canal con el tiempo que le queda para finalizar.

Programa detallado de un canal de televisión

4.1.6 AUDIENCIAS

La vista de Audiencias muestra datos extraídos de la famosa página formulatv.com sobre las audiencias que ha habido las últimas dos semanas de los diferentes programas emitidos en televisión y ordenados por número de espectadores.

Vista que muestra las audiencias de televisión de las últimas dos semanas

La parte superior es un scrollView, en él podemos deslizarnos y seleccionar el día que nos interesa para ver los datos de audiencia.

4.1.7 MIS CANALES

Como se ha explicado al inicio de esta guía, si se quiere, se pueden seleccionar una serie de canales que filtran el contenido de la aplicación. De modo que solo obtenemos información que nos interesa de toda la programación de televisión.

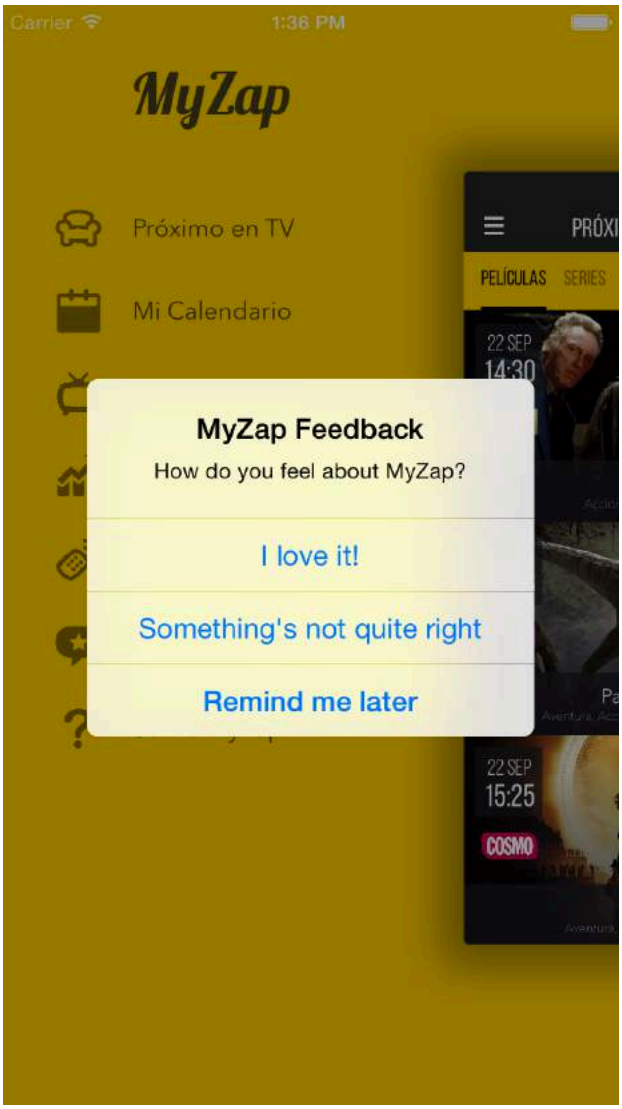
Desde esta vista podemos ver los canales elegidos como favoritos. En el botón superior derecho podemos editar esta lista llevándonos a la vista inicial.

Lista de canales que se han marcado como favoritos

También desde esta vista podemos seleccionar el canal para ver su programación de forma directa como se ha mostrado anteriormente.

4.1.8 VALORAR ESTA APLICACIÓN

Esta funcionalidad es muy importante, ya que permite pedir la opinión de los usuarios de forma directa, y si esta es buena, llevará al usuario directamente a la App Store para dejar su valoración a la vista del público.

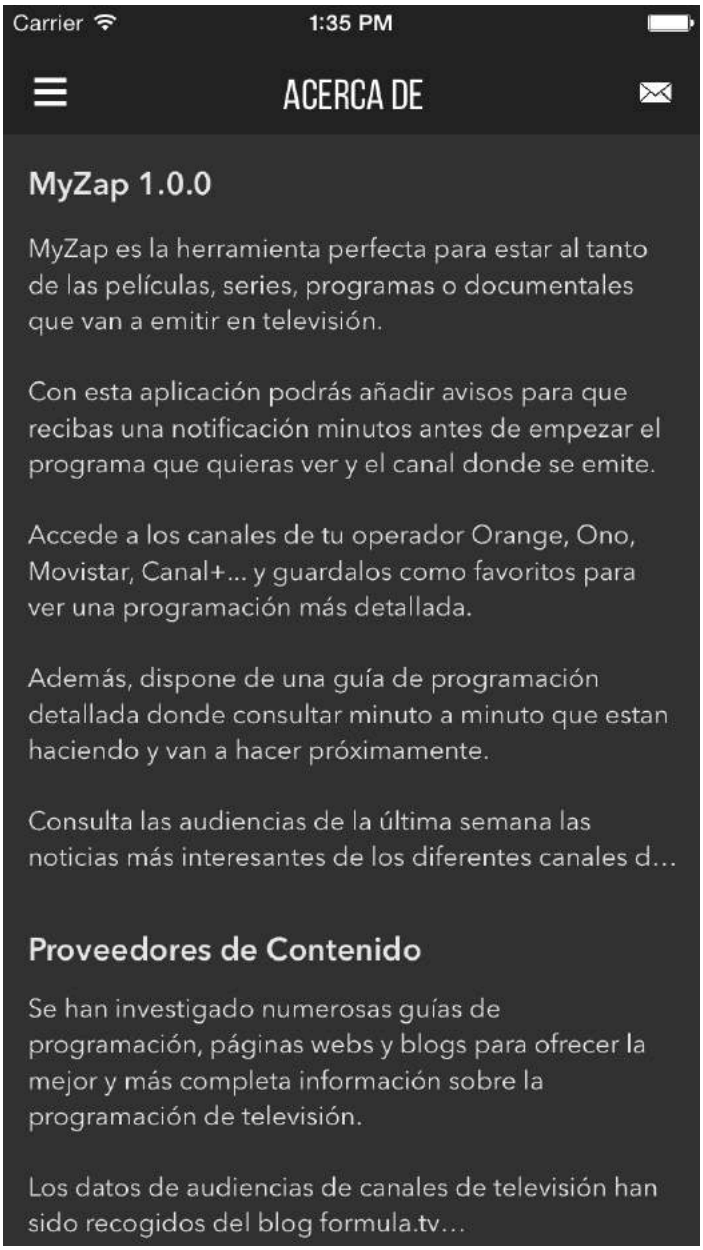


Valoración por parte del usuario al presionar el botón de "valorar esta aplicación"

Si la opinión no es buena, tendremos la opción de enviar un email de soporte al desarrollador para modificar errores o realizar nuevas funcionalidades.

4.1.9 SOBRE MY ZAP

En esta vista se realiza un pequeño texto descriptivo de la versión de la aplicación, de dónde se obtienen los contenidos y un método de contacto con el desarrollador para ayudar al usuario a usar la aplicación o corregir errores.



Vista que muestra una descripción breve de la aplicación

5. BIBLIOGRAFÍA

TECHNET. *Where The Jobs Are. Dr. Michael Mandel*, 2012 [Documento PDF] http://www.technet.org/wp-content/uploads/2012/02/TechNet-App-Economy-Jobs-Study.pdf

THEAPPDATE. *5º Informe estado de las apps en España*, 2014 [Infografía] http://www.theappdate.es/v-informe-estado-apps-espana/

PRNEWSWIRE. *Strategy Analytics*, 2014. http://www.prnewswire.com/news-releases/strategy-analytics-android-captures-record-85-percent-share-of-global-smartphone-shipments-in-q2-2014-269301171.html

KANTAR WORLDPANEL TECHCOM. *Smartphone OS Market Share*, 2015 [Plataforma Web] http://www.kantarworldpanel.com/global/smartphone-os-market-share/

KANTAR WORLDPANEL. *El iPhone 6 el smartphone más vendido en España*, 2015 [Contenido Web] http://www.kantarworldpanel.com/es/Noticias/El-iPhone-6-el-smartphone-mas-vendido-en-Espana

YEEPLY. *Economía App: Nuestros hábitos de uso y consumo de aplicaciones. Ana Mocholí*, 2014 [Contenido Web] https://www.yeeply.com/blog/economia-app-habitos-y-uso-de-aplicaciones-moviles/

ANDROID DEVELOPERS. *Platform Version Distributions*, 2015 [Contenido Web] http://developer.android.com/about/dashboards/index.html

LOCALYTICS. *Top of the line Android pones show 40% higher app engagement than iphones. Dave Hoch*, 2015 [Contenido Web] http://info.localytics.com/blog/top_of_the_line_android_phones_show_40_percent_higher_engagement_than_iphones

EVE INGSISTEMAS. *Sistemas Operativos Móviles iOS*. Eve Porras, 2012 [Contenido Web] http://eve-ingsistemas-u.blogspot.com.es/2012/04/sistemas-operativos-moviles-ios.html

APPLESFERA WEBLOGS SL. *Cinco razones por las que los desarrolladores siguen prefiriendo iOS frente Android. Miguel Michán*, 2013 [Contenido Web] http://www.applesfera.com/aplicaciones-ios-1/cinco-razones-por-las-que-los-desarrolladores-siguen-prefiriendo-ios-frente-a-android

LOCALYTICS. *App retention increasing: iPhone ahead of Android*. Bernd Leger, 2012 [Contenido Web] http://info.localytics.com/blog/app-user-loyalty-increasing-ios-beats-android

LOCALYTICS. *App retention improves – Apps used only once declines to 20%*. Dave Hoch, 2014 [Contenido Web] http://info.localytics.com/blog/app-retention-improves

APP ANNIE. *The Top App Trends of 2014*, 2014 [Contenido Web y PDF] http://blog.appannie.com/app-annie-index-retrospective-2014/

ALTRAN. *Evolución del macro-sector de las Telecomunicaciones en España 2012-2015* [Contenido PDF] https://www.altran.es/fileadmin/medias/ES.altran.es/documents/Ecosistema/EvolucionTelecomunicaciones2012-2015.pdf

HONGKIAT. *20 Advertising networks to monetize your mobile app*. Alvaris Falcon, 2013 [Contenido Web]
http://www.hongkiat.com/blog/mobile-app-monetizing-networks/

YEEPLY. *El retorno de la inversión en el desarrollo de aplicaciones móviles. Ana Mocholí*, 2015 [Contenido Web]
https://www.yeeply.com/blog/roi-en-el-desarrollo-de-aplicaciones-moviles/

APPLE INC. *The Swift Programming Language*, 2015 [Contenido Web]
https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/

APPLE INC. *Automatic Reference Counting. How ARC Works*, 2015 [Contenido Web]
https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift_Programming_Language/AutomaticReferenceCounting.html#//apple_ref/doc/uid/TP40014097-CH20-ID49

CODEHERO. *Objective-C desde cero: Conceptos Básicos*. Oscar González, 2013 [Contenido Web]
http://codehero.co/objective-c-desde-cero-conceptos-basicos/

INFOWORLD. *Swift VS Objective-C: 10 reasons the future favors Swift*. Paul Solt, 2015 [Contenido Web]
http://www.infoworld.com/article/2920333/mobile-development/swift-vs-objective-c-10-reasons-the-future-favors-swift.html

BARLOVENTO COMUNICACIÓN. *Kantar Media Studio. Audiencias Enero* 2015 [Contenido PDF]
http://www.barloventocomunicacion.es/images/publicaciones/barlovento-audiencias-enero-2015.pdf

NOPUEDESPASAR.COM. *¿Qué necesitamos para desarrollar iOS?.* Thais, 2013 [Contenido Web]
http://www.nopuedespasar.com/desarrollo-ios-que-necesitamos-para-desarrollar-ios/

YEEPLY. *Cómo definir tu aplicación móvil: Hacer un prototipo de app*. Ana Mocholí, 2014 [Contenido Web]
https://www.yeeply.com/blog/como-definir-tu-aplicacion-movil-hacer-prototipo-de-app/

GIT SCM. *Reference Manual Documentation*. 2015 [Documentos PDF]
https://git-scm.com/doc

THXOU. *Patrón de diseño MVC al detalle*. Luis Cardenas, 2014 [Contenido Web]
http://www.thxou.com/2011/11/04/programando-para-ios-patron-de-diseno-mvc/

UC3M. *Modelo-vista-controlador*, 2015 [Contenido Web]
http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html

APPLE DEVELOPER LIBRARY. *Cocoa Core Competencies. Model-View-Controller*, 2015 [Contenido Web]
https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

AXELHZF. *Curso iOS: Web Services*, 2014 [Contenido Web]
http://axelhzf.com/ios-curso/webservice.html

COCOAPODS. *Cocoapods Library Search & Reference Guide*, 2015 [Contenido Web]
http://cocoapods.org

PROGRAMACION.NET. *Realm, un motor de base de datos para aplicaciones móviles*. Ángel Carrero, 2014 [Contenido Web]
http://programacion.net/noticia/realm_un_motor_de_base_de_datos_para_aplicaciones_moviles_2197

WIKIPEDIA. *Mapeo objeto-racional*, 2015 [Contenido Web]
https://es.wikipedia.org/wiki/Mapeo_objeto-relacional

CÓDIGO AMBAR. *Tutorial: Realm una nueva base de datos para iOS*, 2014 [Contenido Web]
http://codigoambar.io/blog/2014/08/09/tutorial-realm-una-nueva-base-de-datos-para-ios-parte-1/

APPLE CODING. *Guía de Testflight beta testing para usuarios internos y externos*. Julio César Fernandez, 2014 [Contenido Web]
https://applecoding.com/guias/guia-testflight

YEEPLY. *Las 3 herramientas que nos ayudarán a usar el mobile user a fondo*. Vanessa Estorach, 2015 [Contenido Web]
https://www.yeeply.com/blog/3-herramientas-que-nos-ayudaran-analizar-el-mobile-user/

MIGUELDIAZRUBIO. *Integra Google Analytics en tu app. Miguel Díaz Rubio*, 2013 [Contenido Web]
http://www.migueldiazrubio.com/2013/04/30/desarrollo-ios-integra-google-analytics-en-tu-app/

ADICTOSALTRABAJO. *Tutorial Crashlytics iOS*. Rubén Aguilera, 2014 [Contenido Web]
http://www.adictosaltrabajo.com/tutoriales/crashlytics-ios/

LANCETALENT. *Como promocionar una aplicación móvil en 12 pasos*. Alba López, 2014 [Contenido Web]
http://www.lancetalent.com/blog/como-promocionar-una-aplicacion-movil/

JUSTINMIND. *Prototyper Tour*, 2015 [Software Guide]
http://www.justinmind.com/overview

CONCEPT INBOX. *Prototype Projects*, 2015 [Software Guide]
http://conceptinbox.com/es/

FLUID UI. *Fast and friendly mobile prototyping*, 2015 [Software Guide]
https://www.fluidui.com

ANTETYPE. *Layout aware User Interface Design*, 2015 [Software Guide]
http://www.antetype.com/learn.php

NINJAMOCK. *Wireframe Editor*, 2015 [Software Guide]
https://ninjamock.com/features

INVISION. *Free Web & Mobile Prototyping*, 2015 [Software Guide]
http://www.invisionapp.com/new-features

ATLASSIAN. *Bitbucket Documentation*, 2015 [Contenido Web]
https://confluence.atlassian.com/bitbucket/bitbucket-101-221448818.html

ATLASSIAN. *SourceTree. A free git & Mercurial client for Windows or Mac*, 2015 [Software Guide]
https://www.sourcetreeapp.com

RAYWENDERLICH. *Multithreading and Grand Central Dispatch on iOS for Beginners Tutorial*. Ray Wenderlich, 2011 [Contenido Web]
http://www.raywenderlich.com/4295/multithreading-and-grand-central-dispatch-on-ios-for-beginners-tutorial

THEMOVIEDB. *Database Documentation*, 2015 [API Documentation]
http://docs.themoviedb.apiary.io/

THETVDB. *Database Documentation*, 2015 [API Documentation]
http://thetvdb.com/wiki/index.php/Main_Page

GITHUB. *AFNetWorking*. Alamofire, 2015 [GIT Documentation]
https://github.com/AFNetworking/AFNetworking

GITHUB. *Colours*. Ben Gordon, 2015 [GIT Documentation]
https://github.com/bennyguitar/Colours

HACKEMIST. *SDWebImage*. Olivier Poitrey, 2015 [Contenido Web]
http://hackemist.com/SDWebImage/doc/

GITHUB. *UIActivity Indicator for SDWebImage*. Giaccomo Sacardo, 2015 [GIT Documentation]
https://github.com/JJSaccolo/UIActivityIndicator-for-SDWebImage

GITHUB. *DWTagList*. Dominic Wroblewski, 2015 [GIT Documentation]
https://github.com/domness/DWTagList

GITHUB. *RMSwipeTableViewCell*. Rune Madsen, 2015 [GIT Documentation]
https://github.com/runmad/RMSwipeTableViewCell

GITHUB. *AXRatingView*. Hiroki Akiyama, 2015 [GIT Documentation]
https://github.com/akiroom/AXRatingView

GITHUB. *JVFloatingDrawer*. Julian Villella, 2015 [GIT Documentation]
https://github.com/JVillella/JVFloatingDrawer

GITHUB. *XCDYoutubeKit*. Cedric Luthi, 2015 [GIT Documentation]
https://github.com/0xced/XCDYouTubeKit

ALEXRUPEREZ. *ARSpeechActivity*. Alex Ru Perez, 2015 [Contenido Web]
http://www.alexruperez.com/repos/24-arspeechactivity

GITHUB. *AMSmoothAlert*. Antonie Marlac, 2015 [GIT Documentation]
https://github.com/mtonio91/AMSmoothAlert

GITHUB *DIDatePicker*. Dmitry Ivanenko, 2015 [GIT Documentation]
https://github.com/noxt/DIDatepicker

GITHUB. *iTVDb*. Kevin Tuhumury, 2015 [GIT Documentation]
https://github.com/kevintuhumury/itvdb

GITHUB. *Ask4AppReviews*. Luke Durrant, 2015 [GIT Documentation]
https://github.com/LukeDurrant/Ask4AppReviews

GITHUB. *Async*. Caolan McMahon, 2015 [GIT Documentation]
https://github.com/caolan/async

GITHUB. *XMLDictionary*. Nick Lockwood, 2015 [GIT Documentation]
https://github.com/nicklockwood/XMLDictionary

GITHUB. *CVCalendar*. Eugene Mozharovsky, 2015 [GIT Documentation]
https://github.com/Mozharovsky/CVCalendar

GITHUB. *SwiftyJSON*. Alamofire, 2015 [GIT Documentation]
https://github.com/SwiftyJSON/SwiftyJSON

GITHUB. *CapsPageMenu*. Center For Advanced Public Safety, 2015 [GIT Documentation]
https://github.com/uacaps/PageMenu

GITHUB. *LEColorPicker*. Luis Enrique Espinoza Severino, 2015 [GIT Documentation]
https://github.com/luisespinoza/LEColorPicker

GITHUB. *MJPSlider*. Mike Platt, 2015 [GIT Documentation]
https://github.com/mikeplatt/MJPSlider

GITHUB. *MMPickerView*, Madjid Mahdjoubi, 2015 [GIT Documentation]
https://github.com/madjid/MMPickerView

COCOACONTROLS. *NPSegmentedControl for iOS*, Appetize.io, 2015 [Contenido Web]
https://www.cocoacontrols.com/controls/npsegmentedcontrol

AMAZON DIGITAL SERVICES. *App Store Optimization Bible: Learn How to ASO your apps,* Gabriel Machuret, 2014 [Libro]

LA METRICA. *10 Herramientas de analítica para apps*. Oscar G. Peinado, 2014 [Contenido Web]
http://lametrica.com/10-herramientas-de-analitica-para-apps/