UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

MEMORIA DE TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INFORMÁTICA

Posicionamiento de Oracle frente a los diferentes frameworks de movilidad híbrida del mercado

Autor: David Valencia Delgado-Corredor
Tutor: David Vallejo Fernández
Cotutor: Rafael Prada Gómez

Febrero, 2017

Posicionamiento de Oracle frente a los diferentes frameworks de movilidad híbrida del mercado

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

MEMORIA DE TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INFORMÁTICA

Posicionamiento de Oracle frente a los diferentes frameworks de movilidad híbrida del mercado

Febrero, 2017

Autor: David Valencia Delgado-Corredor

Fdo:

Tutor: David Vallejo Fernández
Cotutor: Rafael Prada Gómez

Fdo de autorización a defender:

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	**VOCAL**	**SECRETARIO**
Fdo.	**Fdo.:**	**Fdo.:**

Resumen

El número de usuarios que utilizan aplicaciones móviles está creciendo enormemente en los últimos años. En 2014, el número de consumidores de aplicaciones móviles había superado al de consumidores de aplicaciones de escritorio. A finales de 2015, la penetración de los dispositivos móviles ascendió al 97%, alcanzándose la cifra de 7,9 mil millones de dispositivos móviles.

La existencia de distintas plataformas para dispositivos móviles hace que muchas veces sea costoso para las empresas el desarrollo de aplicaciones para cada una de ellas. Es aquí donde el desarrollo de aplicaciones híbridas cobra una gran importancia. Este tipo de aplicaciones pueden desplegarse en varias plataformas mediante el desarrollo de un código único, lo que resulta en un menor coste para la empresa.

Este Trabajo Fin de Máster (TFM) se enmarca dentro un convenio FORTE de la Universidad de Castilla-La Mancha con la compañía avanttic Consultoría Tecnológica. Actualmente avanttic utiliza el framework Oracle Mobile Application Framework (MAF) para el desarrollo de proyectos de aplicaciones móviles híbridas. El gran número de frameworks de desarrollo híbrido existentes en el mercado ha creado la necesidad en avanttic de obtener un conocimiento que permita posicionar Oracle MAF con respecto al resto de frameworks.

En este trabajo se propone el desarrollo de una comparativa entre distintos frameworks de desarrollo híbridos existentes en el mercado, con el fin de obtener un conocimiento que permita posicionar los frameworks de Oracle con el resto.

Abstract

The number of users using mobile applications has grown in recent years. By 2014, the number of mobile application consumers had surpassed the number of consumer desktop applications. By the end of 2015, the penetration of mobile devices reached to 97%, reaching the quantity of 7.9 billion mobile devices.

The existence of different platforms for mobile devices makes it often expensive for companies to develop applications for each of them. This is where the development of hybrid applications is of great importance. These types of applications can be deployed on multiple platforms by developing a unique code, resulting in a lower cost for the company.

This Master Thesis is framed within a FORTE agreement of the University of Castilla-La Mancha with the company avanttic Consultoría Tecnológica. avanttic currently uses the Oracle Mobile Application Framework (MAF) framework for its developments of hybrid mobile application projects. The large number of hybrid development frameworks on the market has created the need in avanttic to create a knowledge base that allows positioning Oracle MAF with respect to other frameworks.

This work proposes a comparison between different application frameworks in the market, in order to create a knowledge that allows to position the Oracle frameworks with the rest.

Agradecimientos

Quisiera expresar todo mi agradecimiento a aquellas personas que me han apoyado durante todo este tiempo y que han hecho posible la realización de este proyecto.

En el plano académico, me gustaría dar las gracias a David Vallejo Fernández, director de este trabajo, por su tiempo, apoyo y confianza.

A todas las personas que forman parte de avanttic Consultoría Tecnológica SL, con los que he vivido momentos únicos y que se han convertido en una familia para mí. Mención especial a mi tutor dentro de la empresa, Rafael Prada, que me ha enseñado durante todo este tiempo y me ha hecho crecer como profesional.

En el ámbito personal, quisiera dar las gracias a mi familia y a mis amigos, por haberme apoyado siempre, por los buenos momentos que hemos pasado juntos y por haber estado a mi lado en los buenos y en los malos momentos. Para terminar, a mis compañeros de universidad, que han hecho que esta etapa de mi vida sea inolvidable.

A mi familia y amigos

Índice General

Índice de tablas

Índice de figuras

CAPÍTULO 1
INTRODUCCIÓN

Durante los últimos años el uso de las aplicaciones móviles ha crecido enormemente. Según un estudio realizado por Ditrendia [1] la penetración de los teléfonos móviles ascendió al 97% a finales de 2015[1] y el número de dispositivos móviles alcanzó los 7,9 mil millones, superando el número de personas del planeta. Además, el aumento de las capacidades de los dispositivos móviles permite que las personas puedan hacer uso de servicios desde cualquier lugar a través de Internet. Es por ello que los usuarios demandan cada vez más aplicaciones móviles para hacer uso tanto de nuevos servicios, como de servicios que ya consumían por otros medios. La Figura 1 muestra el crecimiento del número de usuarios que utilizan aplicaciones móviles.

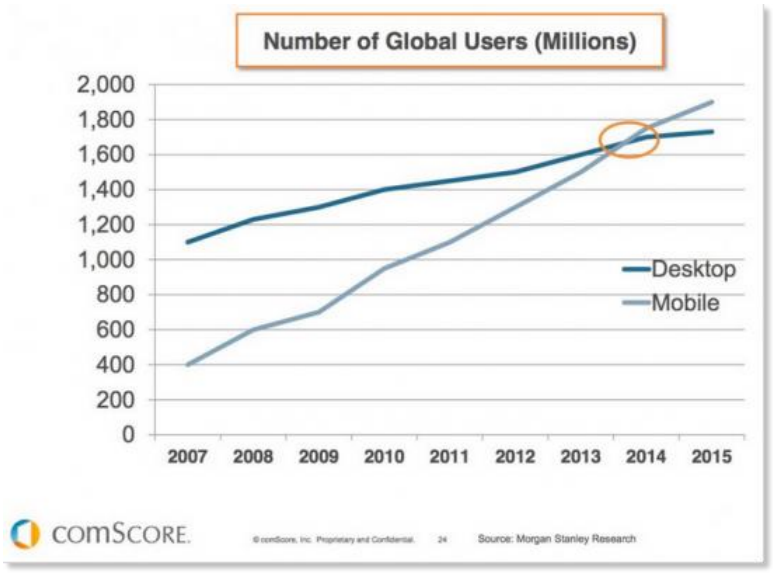


Figura 1: Crecimiento del número de consumidores de aplicaciones móviles

Fuente: *http://www.marketingenredes.com/marketing_movil/mobile-first-o-la-base-del-marketing-movil.html*

[1] No se han encontrado datos posteriores a 2015

Para las compañías, el desarrollo de estas aplicaciones no siempre es simple o barato. La existencia de distintas plataformas móviles hace que sea necesario desarrollar una misma aplicación para cada una de estas plataformas. El desarrollo híbrido ha surgido como una solución a este problema. Una aplicación híbrida aprovecha la versatilidad del desarrollo web y puede acceder a las funcionalidades del dispositivo, como la cámara, el GPS, los contactos o el bluetooth. Estas aplicaciones suponen un menor coste que las aplicaciones nativas, ya que se adaptan a distintas plataformas con un solo desarrollo.

En los últimos años el número de aplicaciones híbridas se ha disparado. En un estudio llevado a cabo por la compañía Gartner, se predijo que en 2016 más del 50% de las aplicaciones serían híbridas [2].

Debido al auge del desarrollo híbrido en los últimos años, existe una gran cantidad de frameworks que permiten el desarrollo este tipo de aplicaciones. Por tanto, es necesario que las empresas dedicadas al desarrollo software posean un conocimiento base sobre las distintas tecnologías y frameworks de desarrollo.

Para dar una solución a los problemas anteriormente mencionados, se pretende, mediante este Trabajo de Fin de Máster, el desarrollo de una comparativa de distintos frameworks de desarrollo de aplicaciones híbridas, para obtener un conocimiento que permita establecer el posicionamiento de los frameworks de Oracle con respecto a las alternativas actuales.

1.1 Contexto del proyecto

Este proyecto se ha llevado a cabo dentro del marco de un convenio FORTE, realizando las prácticas en empresa y el desarrollo de este Trabajo Fin de Máster.

Estas prácticas se han desarrollado en la empresa avanttic Consultoría Tecnológica S.L., con sedes en Barcelona (C/Aragó 182, 4ª) y Madrid (C/ Capitán Haya 38, 6º B). En concreto, estas prácticas se han llevado a cabo en la sede de Madrid.

La compañía avanttic Consultoría Tecnológica S.L. es una consultora tecnológica especializada en tecnología Oracle, y es Platinum Partner de esta compañía. Dentro de Oracle, se especializa en infraestructura, desarrollo y consultoría middleware (SOA, BPM y BI). La compañía ofrece ayuda a sus clientes en el uso adecuado, eficiente y sostenible de

las tecnologías de la información, lo que les permite incrementar la productividad, la calidad y los resultados. Además, está apostando por la transformación digital como herramienta clave de futuro, y lleva años transformándose para alinear sus servicios de consultoría mediante soluciones **smact** (social, mobile, analytics, cloud, internet of things). Estas soluciones, construidas sobre la plataforma e infraestructura de Oracle, mejoran los tiempos de implementación y facilitan el uso adecuado de la tecnología.

Debido al crecimiento del número de aplicaciones híbridas del mercado, los clientes demandan proyectos realizados con frameworks de desarrollo híbridos JavaScript. Actualmente, avanttic utiliza Oracle Mobile Application Framework[2] para desarrollar este tipo de aplicaciones. Sin embargo, los desarrollos de este framework se realizan en su mayor parte en lenguaje Java. Es por ello que la compañía ha decidido comenzar a utilizar Oracle JavaScript Extension Toolkit[3], cuyo principal lenguaje es JavaScript. Además, se quiere realizar un estudio sobre estos frameworks y otros disponibles en el mercado, para obtener un conocimiento que permita identificar las ventajas y desventajas de cada uno de ellos, pudiendo de esta forma ofrecer más alternativas a sus clientes. Esto ayudará a la empresa en el proceso de toma de decisiones a la hora de decidir qué tecnología es la más adecuada para un proyecto determinado. Este hecho se alinea con uno de los **objetivos principales del máster.**

La realización de este proyecto coincide con una fecha muy cercana al evento organizado por la compañía Oracle en Madrid, el Oracle Digital Day, detallado en el Anexo A de este documento. Es entonces cuando surge la idea de desarrollar una aplicación que se presentaría en dicho evento. Aprovechando esto, se decide que sea esta aplicación la que se desarrollaría para la realización este proyecto. La aplicación se denominó *#smact Try&Win*, y consistía en la realización de un sorteo digital de varios premios a elegir. Para que un usuario entrase en el sorteo, debía superar una serie de retos. Para conseguir un reto, el usuario debía buscar una serie de balizas bluetooth (beacons) que detectaba la aplicación, de modo que podía conseguir el reto cuando estaba lo suficientemente cerca.

[2] Oracle MAF: *http://www.oracle.com/technetwork/developer-tools/maf/overview/index.html*
[3] Oracle JET: *http://www.oracle.com/webfolder/technetwork/jet/index.html*

1.2 Estructura del documento

El presente documento se compone de 6 capítulos que se describen a continuación.

El capítulo actual corresponde al Capítulo 1 del documento, en el cual se introduce al lector en el tema del desarrollo híbrido, resumiendo sus principales ventajas, y el contexto en el que se enmarca el desarrollo de este proyecto.

En el Capítulo 2 se expone el objetivo principal de este TFM, se describen los objetivos parciales que se derivan del mismo y se incluyen una serie de limitaciones a tener en cuenta a la hora del desarrollo del proyecto.

El Capítulo 3 corresponde al estado del arte, que se centra en la evolución de las aplicaciones y dispositivos móviles desde su inicio hasta la actualidad, la evolución de los frameworks de desarrollo JavaScript, los tipos de aplicaciones que existen actualmente y algunas consideraciones a la hora de determinar qué tipo de desarrollo elegir al afrontar un proyecto.

En el Capítulo 4 se explica la metodología de trabajo que se ha utilizado para el desarrollo del proyecto y se detallan las tecnologías utilizadas por los frameworks que son objeto de comparación en este trabajo.

En el Capítulo 5 se recogen los resultados obtenidos en base a los objetivos propuestos. Estos resultados se han obtenido siguiendo la metodología de trabajo propuesta en el Capítulo 4.

En el Capítulo 6 se exponen las conclusiones obtenidas tras la realización el trabajo, además de una serie de propuestas de trabajo futuro y una conclusión personal.

CAPÍTULO 2
OBJETIVOS

En este capítulo se expone tanto el objetivo principal de este TFM como los objetivos parciales en los que puede desglosarse. Finalmente se exponen una serie de limitaciones y condicionantes que pueden encontrarse a la hora de desarrollar este proyecto.

2.1 Objetivo principal

El objetivo principal de este TFM consiste en la **realización de un estudio** y **una comparativa** de distintos frameworks de desarrollo de aplicaciones híbridas, que permita a la compañía avanttic establecer una base sobre el posicionamiento de los frameworks de Oracle con respecto a otras alternativas del mercado. Para ello, se desarrollará una misma aplicación con tres frameworks diferentes:

- Dos frameworks de Oracle: Oracle Mobile Application Framework (MAF) y Oracle JavaScript Extension Toolkit (JET).
- Ionic mobile app framework. Para este desarrollo, se ha decidido utilizar la versión 2 de Ionic, actualmente en versión beta, debido a la expectación que se ha creado alrededor de Angular 2, utilizado en esta versión del framework.

2.2 Objetivos parciales

Del objetivo principal podemos derivar una serie de objetivos parciales que deben ser completados para la consecución del objetivo principal. Estos objetivos se realizarán de una forma progresiva para que la realización del proyecto sea satisfactoria. Podemos dividir los objetivos parciales en tres apartados principales.

Revisión exhaustiva del estado del arte de los frameworks

Una **revisión exhaustiva** de la tecnología debe ser el primer paso a llevar a cabo. Esto es fundamental para la consecución de posteriores requisitos y por lo tanto, del objetivo general. Para lograrlo se seguirán los siguientes pasos:

- Realización de un estudio inicial sobre diferentes frameworks de desarrollo híbrido existentes en el mercado.
- Preparación de los entornos de desarrollo de Oracle JET, Oracle MAF e Ionic.
- Creación de ejemplos iniciales con cada uno de los frameworks.

Se pretende, con este objetivo, tener un conocimiento previo de la tecnología que es fruto de comparación en este trabajo, que permita determinar qué características de los frameworks introducir en la comparativa.

Creación de una aplicación prototipo

Para realizar la comparativa entre los frameworks se desarrollará una aplicación involucrada en un proyecto real con cada uno de los frameworks. Esto permite hacerse una idea de los problemas que puede surgir con el desarrollo de un determinado framework durante un proyecto real. El principal problema durante la creación de esta aplicación fue la falta de tiempo y la poca experiencia que se tenía con la tecnología de desarrollo, ya que ningún miembro del equipo había utilizado antes el framework Oracle JET en un proyecto real.

Por tanto, se desarrollará la aplicación *#smact Try&Win* con JET, MAF e Ionic. Esta aplicación surgió como un proyecto para el Oracle Digital Day, un evento organizado por Oracle, y que tuvo lugar el 27/09/2016. La aplicación presentada fue la versión desarrollada con Oracle JET. La experiencia de este evento se detalla en el Anexo A.

Comparación de los frameworks

Una vez alcanzados los objetivos anteriores, el último paso es la **comparación** de los tres frameworks, para poder determinar cuál de los tres es más completo. A pesar del resultado obtenido en la comparación, cabe desatacar que se debe tener en cuenta las características y restricciones de la aplicación que se quiera desarrollar, ya que cada framework puede ser

más adecuado que otro dependiendo de la aplicación. Para la consecución de este objetivo parcial se deben realizar los siguientes pasos:

- Realización de pruebas que permitan medir aspectos del rendimiento de la aplicación desarrollada con cada framework.
- Comparativa de las principales características de los frameworks:
    - Integración con las características del dispositivo.
    - Seguridad.
    - Velocidad de desarrollo.
    - Sistema de enlace de datos.
    - Sistema de navegación.
    - Interfaz de usuario.
    - Reconocimiento de gestos en pantalla.
    - Documentación y comunidad de soporte.
- Realización de una ponderación sobre los resultados obtenidos en los puntos anteriores.

2.3 Limitaciones

A la hora de observar el comportamiento de la aplicación desarrollada con los distintos frameworks podemos encontrar obstáculos debido a la falta de dispositivos de varias plataformas, en concreto para las plataformas iOS y Windows Phone.

Por otro lado, el poco margen que existía para la realización de la aplicación, ya que debía estar finalizada para el Oracle Digital Day, hace que no se pueda experimentar a fondo con todas las características ofrecidas por los frameworks.

CAPÍTULO 3
ESTADO DEL ARTE

En este capítulo se expone un estudio sobre la evolución de las aplicaciones y dispositivos móviles, los frameworks de desarrollo JavaScript y los tipos de aplicaciones existentes. Para finalizar se expone una serie de consideraciones a tener en cuenta a la hora de determinar el tipo de desarrollo que se debe elegir cuando se afronta un nuevo proyecto.

En el capítulo 5 se expone un estudio sobre el estado del arte de diferentes frameworks de desarrollo de aplicaciones híbridas, centrándose en los tres frameworks que se comparan en este trabajo. La decisión de incluir este estudio en el Capítulo 5, y no en el capítulo actual, se debe a que dicho estudio forma parte de los objetivos parciales de este trabajo, por lo que se ha considerado más oportuno incluirlo en el capítulo de resultados.

3.1 Evolución de las aplicaciones y dispositivos móviles

Los dispositivos móviles se han convertido en dispositivos esenciales en la vida de las personas en muy pocos años. Actualmente existe una gran cantidad de modelos y de fabricantes de estos dispositivos que continúan innovando en este mercado.

Fue en 1973 cuando surgieron los dispositivos móviles de primera generación. Estos dispositivos eran pesados y de un gran tamaño, y solo podían ser utilizados para realizar llamadas de voz. La seguridad que ofrecían era muy baja, lo que implicaba que otra persona pudiese escuchar las llamadas de una forma muy sencilla.

La segunda generación de móviles supuso el paso de la tecnología analógica a la digital. Fue en esta generación de dispositivos, a finales de los años 90, cuando surgen las primeras aplicaciones móviles. Estas aplicaciones eran las que conocemos como la agenda, juegos

arcade, editores de tonos de llamada, etc. Estas aplicaciones cumplían funciones muy elementales y tenían un diseño muy simple.

A la llegada de la tecnología WAP (Wireless Application Protocol) hacia el año 2000 le acompañó un fuerte incremento en el desarrollo de móviles y aplicaciones. Fue en esta época cuando surgieron los móviles de tercera generación, en la cual aparecieron dispositivos que incluían pantallas LCD a color. Estos dispositivos incorporaban aplicaciones como la cámara y juegos en 3D, y permitían la realización de videoconferencias gracias a la mejora en la transferencia de datos. Sin embargo, las restricciones que imponían los fabricantes sobre sus sistemas operativos hacía muy difícil el desarrollo de aplicaciones por parte de desarrolladores externos.

Todo esto cambió con el lanzamiento en 2008 del App Store de Apple, que permitía a los desarrolladores publicar sus aplicaciones para Iphone en esta plataforma. Con la llegada del market de Android, al ser una plataforma Open Source, permitió una mayor libertad y la llegada de smartphones de bajo coste. La Figura 2 muestra la cuota de mercado de cada plataforma móvil, según Net Market Share [3].

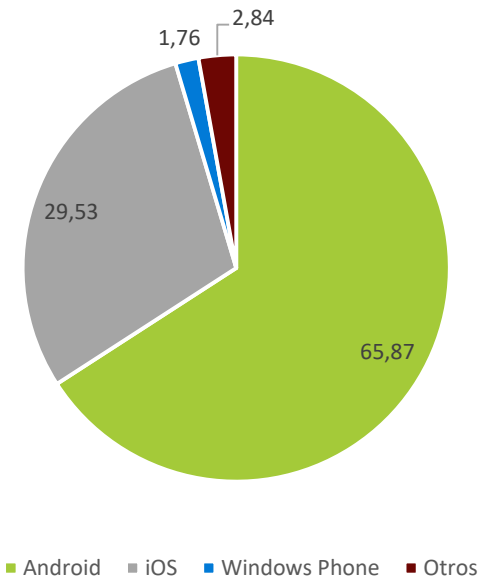


Figura 2: Cuota de mercado por plataforma móvil

En el año 2010 se lanzaron los primeros servicios basados en tecnología 4G, dando paso a la cuarta generación. La principal característica de esta tecnología es que tiene la capacidad

de proveer velocidades de acceso muy elevadas, ofreciendo una alta calidad de servicio, con una alta seguridad y con el mínimo coste posible.

El hecho de que podamos acceder desde cualquier lugar y en cualquier momento al contenido de Internet a través del móvil creó la necesidad de adaptar los sitios Web a estos dispositivos, surgiendo así las Web App. Además, la existencia de diferentes plataformas para dispositivos móviles ha dado lugar al incremento del número de aplicaciones híbridas en el mercado, que pueden ejecutarse en varias plataformas con un solo desarrollo. De esta forma, podemos distinguir en la actualidad entre tres tipos de aplicaciones para dispositivos móviles: nativas, híbridas y web, en las cuales nos centramos en el capítulo siguiente.

3.2 Evolución de los frameworks de desarrollo JavaScript

El desarrollo de aplicaciones ha evolucionado mucho desde sus inicios hasta la actualidad. En este apartado nos centramos en la evolución del desarrollo de aplicaciones con JavaScript desde su inicio hasta los frameworks que podemos encontrar hoy en día.

Los inicios de JavaScript

En un principio la programación en páginas web con **JavaScript** se realizaba utilizando únicamente el propio lenguaje, es decir, se utilizaba JavaScript sin apoyarse en ninguna librería o framework. Con el tiempo, las páginas web comenzaron a requerir más funcionalidades y más complejas, por lo que programar de este modo se hizo muy pesado.

En ese momento nació **jQuery**, una librería JavaScript que facilitaba el desarrollo de este lenguaje. Esta librería permitía interactuar con los elementos HTML, manipular el árbol DOM, el manejo de eventos, el desarrollo de animaciones y la utilización de AJAX. Después se lanzó **jQuery Mobile**, que se trata de una versión de jQuery optimizada para dispositivos móviles.

Posteriormente surgieron los frameworks CSS, como **Bootstrap** y **Foundation**, que facilitaban el uso de las estructuras CSS, ya que su uso comenzaba a resultar complejo.

Los frameworks JavaScript

Las funcionalidades y la complejidad de las páginas web continuaron creciendo y el uso de jQuery se hizo inmanejable. Fue entonces cuando surgió el patrón de diseño MVC (Modelo Vista Controlador), que separaba la interfaz de usuario de las capas de lógica y datos. Entre este tipo de frameworks destacó **Backbone.js**, que debía su popularidad, en parte, a que era un framework muy ligero (algo más de 7 kb) y flexible.

Un framework que ganó popularidad rápidamente y que acabó superando a Backbone es **AngularJS.** Fue un proyecto iniciado por Google que incorporaba funcionalidades integradas, que requerirían de librerías externas si se utilizase simplemente JavaScript o Backbone. Angular proporciona su propio sistema de módulos y de plantillas, además de servicios para obtener datos del servidor utilizando AJAX sin necesidad de utilizar librerías externas.

La llegada de NodeJS

La llegada de **Node.js** supuso una revolución en el desarrollo web. Se trata de un entorno de ejecución para JavaScript para la capa del servidor. El hecho de que con Node.js se pudiese programar en el lado del servidor con el lenguaje JavaScript permitía crear un desarrollo más homogéneo entre el cliente y el servidor.

Además de la parte del servidor, con Node.js surgieron una gran cantidad de herramientas y módulos que permitieron mejorar los desarrollos y hacerlos más ágiles.

Las herramientas de automatización de tareas

El surgimiento de **Grunt** permitió la automatización de tareas tales como el minificado de archivos, la compilación y la realización de pruebas unitarias.

Posteriormente llegó **Gulp** que utilizaba *Streams* proporcionados por Node.js, facilitando la implementación y la ejecución de las tareas que hacíamos con Grunt.

Las aplicaciones híbridas

El auge de las aplicaciones móviles hizo que los frameworks evolucionaran y surgiesen nuevas herramientas. En un primer momento surgieron herramientas, como **Phonegap**, que

permitían que una aplicación web fuese empaquetada como una aplicación móvil. Sin embargo, la experiencia que ofrecía al usuario no era muy buena.

Phonegap evolucionó a **Cordova**, y surgieron frameworks como **Ionic**, que se basaba en Angular, y permitía crear aplicaciones para móviles para diferentes plataformas, programando con JavaScript y empaquetando el proyecto con Cordova.

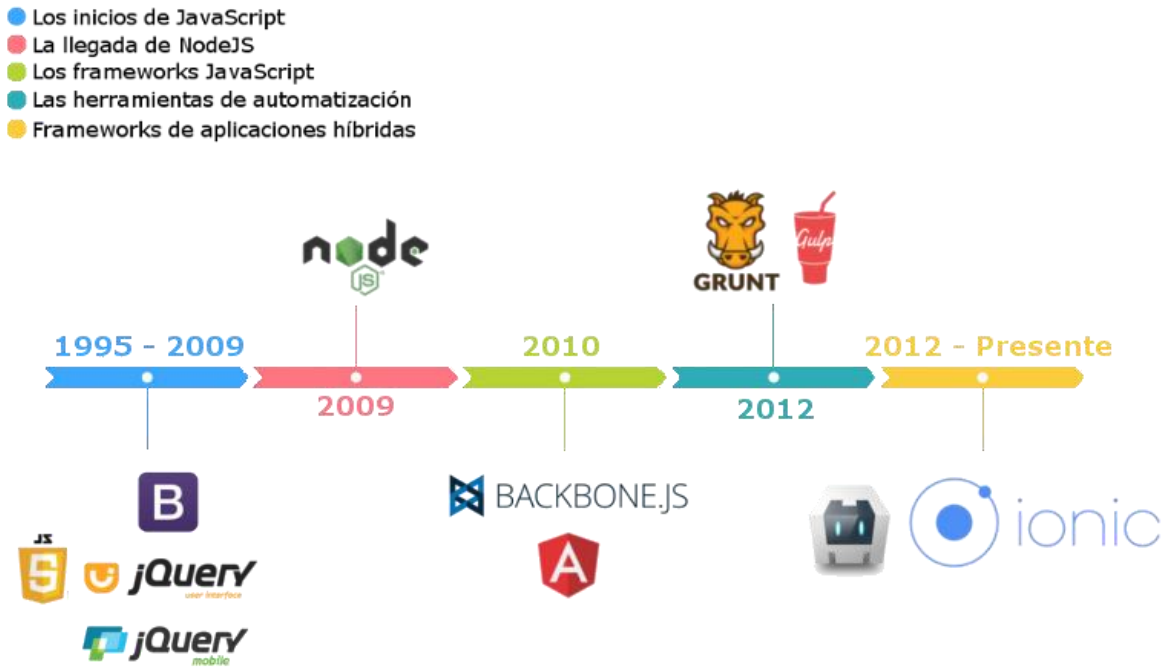


Figura 3: Evolución de los frameworks JavaScript

3.3 Tipos de aplicaciones

En este apartado se explican las principales características de los tipos de aplicaciones móviles que podemos encontrar, y cuáles son las ventajas y desventajas de cada una de ellas. Finalmente, se habla sobre qué tipo de desarrollo elegir para nuestras aplicaciones en función de sus características y necesidades.

3.3.1 Aplicaciones nativas

Una aplicación nativa es aquella que se instala en el dispositivo y que está desarrollada específicamente para cada plataforma: Android, iOS, Windows Phone, etc [4]. Cada una de estas plataformas tiene un sistema diferente, por lo que es necesario desarrollar la aplicación para cada una de las plataformas en las que queramos que esté disponible. La Tabla 1 muestra las principales plataformas móviles del mercado.

Algunos ejemplos de aplicaciones nativas son *camera+*, desarrollada específicamente para iOS, y *KeePassDroid* para el sistema Android.

Plataforma	Desarrollador	Lenguaje de programación
Android	Google	Java
iOS	Apple	Objective-C
Windows Phone	Microsoft	C#

Tabla 1: Principales platadormas móviles

Ventajas y desventajas de las aplicaciones nativas

Las ventajas de una aplicación nativa son las siguientes [5]:

- **Acceso a los servicios del dispositivo.** Al ser una aplicación que se instala en el dispositivo, puede hacer uso de sus servicios, tales como la cámara, el GPS, el bluetooth, etc.
- **Mejor experiencia de usuario.** Cada plataforma sigue unas líneas de diseño. Los SDK proporcionados por cada plataforma nos permiten adaptar las aplicaciones nativas a dicha plataforma. Esto facilita el aprendizaje de la aplicación al usuario.
- **Mayor visibilidad.** La posibilidad de subir la aplicación al market hace que tenga una mayor visibilidad y que sea más fácil de encontrar por los usuarios.

- **Mejor rendimiento.** Con una aplicación nativa se puede lograr el mayor rendimiento posible, ya que estas se ejecutan a un nivel más bajo que las aplicaciones híbridas, que se ejecutan en un navegador.

Las desventajas de este tipo de aplicaciones son:

- **Conocimiento más amplio.** Es necesario conocer las herramientas de desarrollo para cada plataforma específica.
- **Su desarrollo suele ser más caro.** La necesidad de mantener un proyecto por cada plataforma hace que el coste del desarrollo y mantenimiento de este tipo de aplicaciones sea mayor.
- **El código de desarrollo no puede reutilizarse en otras plataformas.** Las aplicaciones nativas se desarrollan con el lenguaje específico para cada plataforma, por lo que no podemos reutilizar el código desarrollado en plataformas distintas.
- **Necesidad de actualización por parte del usuario.** Es responsabilidad del usuario actualizar la aplicación cuando una actualización esté disponible en el market. En una aplicación web, el usuario no debe preocuparse de actualizar la aplicación.

3.3.1.1 El desarrollo de aplicaciones nativas

En este apartado se exponen las principales plataformas para dispositivos móviles y los requerimientos para desarrollar aplicaciones en cada una de ellas.

Android

Android es un sistema operativo basado en Linux, diseñado principalmente para dispositivos móviles con pantalla táctil, como smartphones y tablets. Inicialmente fue desarrollado por Android Inc. y posteriormente fue comprado por Google.

La Figura 4 muestra la arquitectura de Android, que se compone de los siguientes elementos [6]:

- **El núcleo Linux.** El núcleo de Android está basado en el sistema operativo Linux. Esta capa proporciona servicios como la seguridad, la gestión de memoria, gestión de procesos, la pila de protocolos y el soporte de drivers para dispositivos.

- **El Runtime de Android.** Se basa en el concepto de máquina virtual de Java. Dadas las limitaciones de los dispositivos donde debía ejecutarse Android, Google decidió crear una nueva máquina virtual que respondiera mejor a estas limitaciones, la máquina virtual Dalvik.
- **Librerías nativas.** Incluye una serie de librerías en C/C++ usadas en varios componentes de Android.
- **Entorno de trabajo de las aplicaciones.** Android proporciona una plataforma de desarrollo libre de aplicaciones, que permite a los desarrolladores el acceso completo a las APIs usadas por las aplicaciones base. Esta capa ha sido diseñada para facilitar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades, de manera que otras aplicaciones puedan hacer uso de ellas.
- **Aplicaciones.** Este nivel se compone de todas las aplicaciones que están instaladas en la máquina virtual de Android. Las aplicaciones se ejecutan en la máquina virtual Dalvik para garantizar la seguridad del sistema.

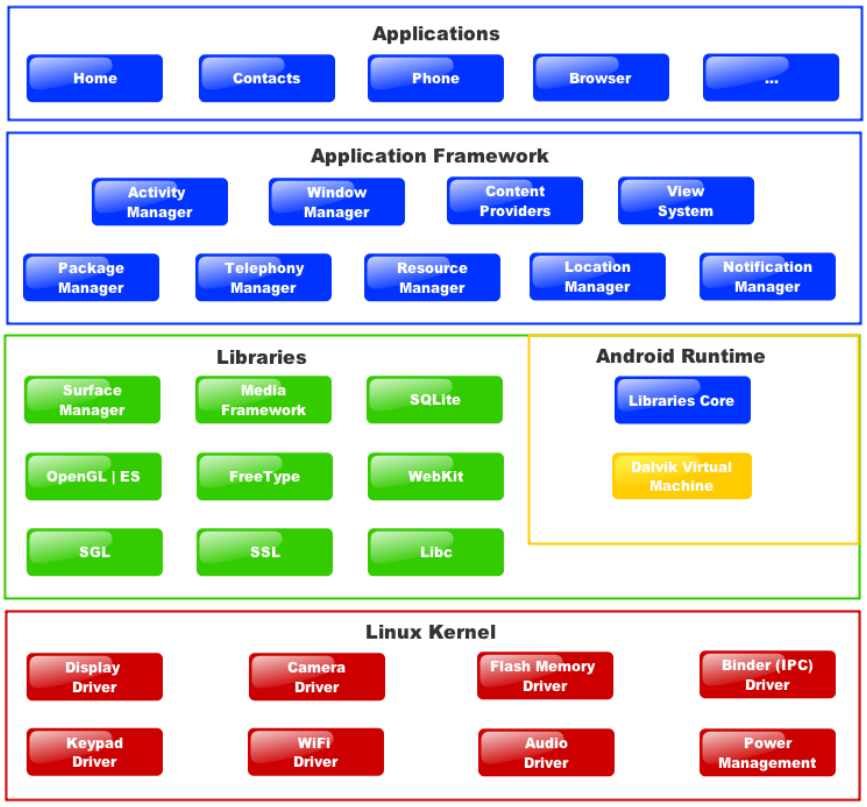


Figura 4: Arquitectura de Android

Fuente: *https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android*

Requerimientos para el desarrollo en Android

Para desarrollar aplicaciones en Android es necesario que los desarrolladores preparen su entorno de desarrollo con una serie de programas que permitirán manejar algunas funcionalidades del sistema operativo Android.

Android SDK

El SDK de Android permite a los desarrolladores crear aplicaciones para la plataforma Android. El SDK incluye proyectos de ejemplo, herramientas de desarrollo, emuladores y un conjunto de librerías utilizadas para la compilación de las aplicaciones para Android.

Java Runtime Enviroment

El Java Runtime Enviroment (JRE) es un conjunto de utilidades que permite ejecutar el código java en el equipo.

El JRE actúa como un intermediario entre el sistema operativo y Java. Está compuesto por la Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes que son necesarios para que las aplicaciones escritas en lenguaje Java puedan ser ejecutadas.

IDE

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés) es una aplicación que proporciona al desarrollador una serie de servicios para facilitar el desarrollo de software. Entre los IDEs más populares para el desarrollo en Android podemos destacar Eclipse[4], NetBeans[5] y Android Studio[6].

iOS

iOS es un sistema operativo desarrollado por la compañía Apple. Este sistema operativo fue desarrollado originalmente para el iPhone, aunque actualmente se utiliza en otros dispositivos como el iPod touch o iPad.

La arquitectura de iOS está compuesta por cuatro capas [7]:

[4] Eclipse: *https://eclipse.org*
[5] NetBeans: *https://netbeans.org*
[6] Android Studio: *https://developer.android.com/studio/index.html*

- **Capa del Núcleo del Sistema Operativo (Core OS).** Es el núcleo del sistema. Realiza la gestión de controladores, memoria virtual, sistema de ficheros, TCP/IP, sockets, seguridad, gestión de memoria y comunicación entre procesos.
- **Capa de Servicios principales (Core Services).** Es la capa que proporciona los servicios principales del sistema, los cuales van a ser utilizados por las aplicaciones.
- **Capa de Medios de Comunicación (Media).** Esta capa contiene una serie de frameworks y tecnologías que proporcionan acceso a los ficheros multimedia como audio, gráficos, videos, etc.
- **Capa Cocoa Touch.** Contiene herramientas que permiten crear y acceder a objetos y a estructuras de datos básicos. Permite crear interfaces de usuario, y conectarlas con controladores para manejar eventos. Es la capa que permite al usuario la interacción con las aplicaciones.

La Figura 5 muestra la arquitectura de iOS.

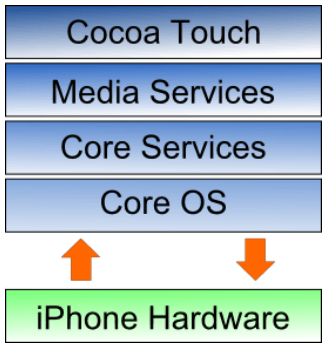


Figura 5: Arquitectura de iOS

Fuente: *http://so1il141.blogspot.com.es/2015/05/architectura-del-sistema-operativo-ios-8.html*

Requerimientos para el desarrollo en iOS

Para comenzar a desarrollar aplicaciones en iOS necesitamos una serie de requisitos mínimos:

- Un **Mac**. Será necesario disponer de un Mac para el desarrollo de aplicaciones en iOS, ya que las herramientas necesarias para su desarrollo solo están disponibles para el sistema operativo de Mac.
- Una **cuenta de desarrollador de Apple**. Será necesario que dispongamos de una cuenta de desarrollador, lo que nos permitirá acceder a los recursos disponibles para

los desarrolladores, como el iPhone SDK y los certificados para firmar nuestras aplicaciones.

- **iPhone SDK**. Es un kit de desarrollo software para iOS que permite el desarrollo de aplicaciones nativas para esta plataforma. En el SDK se incluye el entorno de desarrollo Xcode, en el cuál se desarrollan las aplicaciones mediante el lenguaje Objective-C, además de un emulador de iPhone para ejecutar las aplicaciones.

El entorno de desarrollo Xcode

Xcode es un entorno de desarrollo integrado (IDE) desarrollado por Apple y que ofrece las herramientas necesarias para programar tanto en Mac OS como en iOS. Está muy integrado con los frameworks Cocoa y Cocoa Touch, los cuales nos permiten crear la interfaz para las aplicaciones, creando un entorno de desarrollo productivo y fácil de usar.

Xcode se centra en el rendimiento. Mientras se desarrolla la aplicación, se muestra inmediatamente los posibles errores de código. Además permite compilar y ejecutar la aplicación en el emulador, o lanzarla directamente a tu dispositivo móvil. También dispone de un depurador integrado.

Windows Phone

Windows Phone es un sistema operativo desarrollado por Microsoft, como sucesor de Windows Mobile, para dispositivos móviles. A diferencia de Windows Mobile, que se centraba en el ámbito empresarial, Windows Phone se enfoca en el mercado de consumo. En el año 2015 Microsoft abandonó Windows Phone y se enfocó en un nuevo sistema para dispositivos móviles, denominado Windows 10 Mobile.

La arquitectura de Windows Phone tiene los siguientes componentes [8]:

- **Modelo de aplicación**. En Windows Phone las aplicaciones se despliegan en un paquete XAP. Es un archivo comprimido que contiene todos los recursos de nuestra aplicación. Las aplicaciones solo pueden ser instaladas desde el Market Place de Windows, y los desarrolladores deben registrarse para poder ofrecer sus aplicaciones.
- **Modelo de UI**. El modelo de interfaz de usuario se compone de elementos, páginas y sesiones. Los elementos son los controles que se le muestran al usuario, como por ejemplo, un botón. Una página es una agrupación de elementos y una sesión es el

conjunto de interacciones que el usuario realiza sobre la aplicación, pudiendo involucrar a otras aplicaciones, por ejemplo, cuando una aplicación abre la galería de imágenes para seleccionar una fotografía.

- **Integración con la nube**. Windows Phone está enfocado en la integración con la nube. Por defecto se integra con servicios como *Exchange, Google Mail, Hotmail, Xbox Live, Skydrive, Facebook, Twitter o Bing*.
- **Runtime de aplicaciones**. El Runtime de aplicaciones especifica dónde y cómo se ejecutan las aplicaciones, las limitaciones que vamos a encontrar y los frameworks de los que disponemos para desarrollar.

La Figura 6 muestra esta arquitectura.

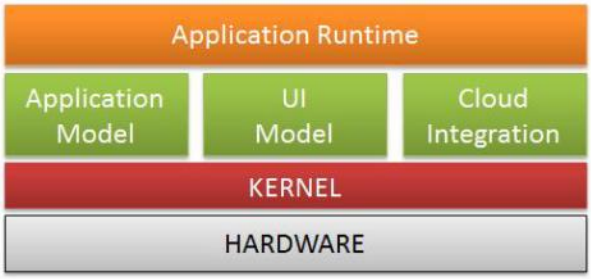


Figura 6: Arquitectura de Windows Phone

Fuente: *http://tutocisc.bligoo.com/conocimientos-basicos-arquitectura-de-windows-phone-7*

Requerimientos para el desarrollo en Windows Phone

Para desarrollar aplicaciones para Windows Phone necesitamos una serie de requisitos [8]:

- El SDK de Windows Phone.
- Un ordenador con un sistema operativo Windows Vista o superior.
- Un espacio de 4 GB disponibles.
- 3 GB de RAM.
- Una tarjeta gráfica con funcionalidad DirectX 10 o superior, y un controlador WDDM 1.1.

Windows Phone SDK

El SDK de Windows Phone es un conjunto de herramientas que permite a los desarrolladores crear aplicaciones para Windows Phone. Las herramientas que incluye el SDK son:

- **Microsoft Visual Studio para Windows Phone**. Es la herramienta principal para el desarrollo de las aplicaciones.
- **Microsoft Expression Blend para Windows Phone**. Es una herramienta para diseñadores, pensada para el diseño de la interfaz de usuario.
- **Emulador de Windows Phone**. Un emulador que nos permitirá ejecutar nuestras aplicaciones.
- **Application Deployment**. Nos permite desplegar nuestras aplicaciones en el emulador o en un dispositivo móvil.
- **Windows Phone Developer Registration**. Esta herramienta nos permite registrar un dispositivo con nuestra cuenta de desarrollador de Marketplace, de modo que podamos depurar y desplegar aplicaciones en él.
- **Windows Phone Marketplace Test Tool**. Es una herramienta que nos permite emular las pruebas que se harán posteriormente cuando enviemos la aplicación al Marketplace para su publicación.

3.3.2 Aplicaciones web

Las WebApp son aquellas que no se encuentran instaladas en el dispositivo y que se desarrollan en HTML, CSS y JavaScript. La principal diferencia con una página web estándar es que están diseñadas para verse en dispositivos móviles [9]. Este tipo de aplicaciones necesitan de conexión a internet para poder acceder a ellas desde el dispositivo y pueden ejecutarse desde cualquier plataforma.

Dos ejemplos de WebApp son Gmail y Youtube, que vistas desde el navegador del dispositivo móvil ofrecen un aspecto y una funcionalidad muy parecida a la que ofrecen sus correspondientes aplicaciones móviles.

Ventajas y desventajas de las aplicaciones web

Las ventajas de las aplicaciones web son [10]:

- **No ocupan espacio en el dispositivo.** La aplicación no se encuentra instalada en el dispositivo, y por lo tanto no consume sus recursos.

- **Proceso de desarrollo más sencillo y con un coste menor.** Solo es necesario desarrollar la aplicación una vez. Las distintas plataformas pueden acceder a ella a través del uso de un navegador.
- **Un solo código para múltiples plataformas.** Esto evita la necesidad de mantener varios proyectos al mismo tiempo.
- **No requiere actualización por parte del usuario.** Cuando el desarrollador realiza cambios en la aplicación, los usuarios tienen acceso a ellas inmediatamente.

Las principales desventajas de las aplicaciones web son:

- **Necesita conexión a Internet.** La única posibilidad de acceder a la aplicación es mediante un navegador web, por lo que es necesario conexión a Internet.
- **Acceso limitado** a los servicios del dispositivo.
- **Menor rendimiento,** al no estar integrada en el dispositivo.
- **Peor experiencia de usuario,** al no poder hacer uso de componentes nativos del dispositivo.
- **Menor visibilidad.** Los usuarios no pueden encontrar la aplicación en el market.

Adaptando la Web al dispositivo móvil

Dado el incremento de smartphones y tablets resulta vital que nuestro sitio web pueda adaptarse a estos dispositivos para una correcta visualización. Si la visualización no es correcta, los usuarios pueden sentirse frustrados, corriendo el riesgo de que no vuelvan a acceder a nuestro sitio web.

Para asegurarnos de que un sitio web se muestre correctamente en un dispositivo móvil podemos encontrar dos opciones [11]:

- Construir nuestro sitio web utilizando un diseño responsive o un diseño adaptativo.
- Desarrollar un sitio web dedicado.

Sitios web adaptativos/responsive

Un sitio web con diseño adaptativo se adaptará a un conjunto predefinido de diferentes resoluciones de pantalla. Esto requiere tener un número diferente de plantillas para las

diferentes resoluciones y tiene el riesgo de que la salida de un nuevo dispositivo requiera el uso de una plantilla nueva.

Los sitios web con diseño responsive se adaptan a cualquier resolución de pantalla utilizando una única plantilla, y debe ser diseñada de manera que los elementos puedan redimensionarse o moverse según sea necesario.

Sitios web dedicados

En algunos casos podemos encontrarnos que hacer nuestro sitio web con un diseño responsive o, crear uno nuevo, puede ser muy costoso debido a la complejidad del sitio y de su funcionalidad. En estos casos, una mejor opción puede ser crear un sitio web específico para dispositivos móviles. Generalmente, estos sitios web solo presentan las características principales, y no todas las características que pueda presentar la página web.

Los sitios web móviles tienen la ventaja de ser más baratos a corto plazo, ya que su diseño resulta más sencillo que el de un sitio web responsive. Sin embargo, resultan más caros a largo plazo debido a su mantenimiento.

Un sitio web dedicado tendrá una URL diferente al sitio web convencional. Cuando un usuario accede al sitio web, el sistema comprobará el dispositivo y la resolución de pantalla para ofrecer la versión más apropiada.

3.3.3 Aplicaciones híbridas

Las aplicaciones híbridas combinan características de las aplicaciones web y las aplicaciones nativas [12]. Estas aplicaciones se instalan en el dispositivo y se ejecutan dentro de un contenedor web nativo. Se desarrollan utilizando tecnologías de desarrollo web, como HTML, JavaScript y CSS.

Un ejemplo de aplicación híbrida que podemos destacar es Instagram. Siendo una aplicación híbrida, Instagram consigue el soporte de HTML5 que soporta datos offline, como vídeos. Tiene sentido que una aplicación para compartir fotos y vídeos esté hecha con una tecnología que permita a los usuarios acceder incluso cuando se encuentra offline.

Ventajas y desventajas de las aplicaciones híbridas

Las aplicaciones híbridas presentan las siguientes ventajas [13]:

- **Un solo código para múltiples plataformas.** Al igual que las aplicaciones web, solo es necesario desarrollar la aplicación una única vez para todas las plataformas.
- **Acceso a los servicios del dispositivo.** Las aplicaciones híbridas se instalan en el dispositivo, por lo que al igual que las aplicaciones nativas, pueden hacer uso de las características del dispositivo. Sin embargo, requieren de plugins para acceder a estos servicios.
- **Visibilidad en los markets.** El desarrollador puede subir este tipo de aplicaciones a los markets de las distintas plataformas.
- **Menor coste de desarrollo que las aplicaciones nativas.** El desarrollo para todas las plataformas es mediante un único código, por lo que no requiere gestionar más que un proyecto al mismo tiempo.

Las desventajas de las aplicaciones híbridas son:

- **Necesidad de actualización por parte del usuario,** al igual que sucede con las aplicaciones nativas, el usuario debe encargarse de su actualización cuando esta esté disponible.
- **Menor rendimiento que una aplicación nativa.** Las aplicaciones híbridas se ejecutan dentro de un *WebView*, por lo que el rendimiento siempre será menor que en una aplicación nativa.

El WebView

Como se ha comentado antes, una de las principales diferencias que presentan las aplicaciones nativas con respecto a las aplicaciones híbridas es que estas últimas se ejecutan en un contenedor web nativo. Este contenedor es lo que se conoce como WebView.

El sistema WebView es un componente nativo proporcionado por el sistema operativo que es capaz de cargar contenido web. Permite a las aplicaciones híbridas acceder a las capacidades del dispositivo móvil, tales como la cámara, el bluetooth, el GPS, etc. Esto es posible gracias a los frameworks que utiliza el WebView para representar el contenido, como Apache Cordova, que incrusta código HTML5 en el WebView para proporcionar un "puente" que permita el acceso a los recursos nativos del dispositivo.

La Figura 7 muestra la interacción de los diferentes tipos de aplicaciones con el dispositivo. Como muestra la figura, las aplicaciones nativas utilizan un *wrapper* para interactuar con el dispositivo.

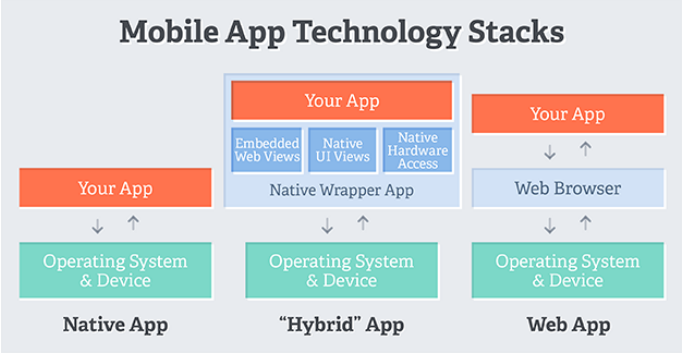


Figura 7: Interacción del dispositivo con los distintos tipos de aplicaciones

Fuente: http://www.sysfore.com/mobile-apps-overview.aspx

3.3.4 Qué tipo de desarrollo elegir

Tras ver los tipos de aplicaciones existentes, la duda que surge a la hora de comenzar un nuevo proyecto es qué tipo de desarrollo llevar a cabo. La elección de un tipo de desarrollo u otro depende de diversos factores que se deben tener en cuenta.

Coste del desarrollo

Uno de los principales factores a tener en cuenta es el coste que supondrá el proyecto. Lo primero que debemos hacer es especificar las plataformas para las que queremos que la aplicación esté disponible, y tener en cuenta que un desarrollo nativo requerirá que cada plataforma tenga un desarrollo propio. Además, un desarrollo nativo requerirá un mantenimiento y soporte separados para cada plataforma. Este tipo de desarrollo también necesita de desarrolladores con conocimientos específicos en el desarrollo para cada plataforma. Por lo tanto, si el coste resulta un factor clave y se necesita dar soporte a múltiples plataformas, el desarrollo híbrido puede ser la mejor elección.

Rendimiento

En cuanto al rendimiento, las aplicaciones nativas siempre podrán lograr un mejor resultado ya que se realizan exclusivamente para cada plataforma, lo que permite que la aplicación se

integre mejor con el dispositivo y ofrecer una mejor experiencia al usuario. Si el rendimiento es un factor clave, una aplicación nativa es la mejor opción.

Acceso a las características del dispositivo

Las aplicaciones híbridas y nativas nos permiten hacer uso de las características del dispositivo, tales como la cámara, el GPS, los contactos, etc. Sin embargo, no siempre es necesario el acceso a estos servicios. En estos casos, puede resultar mejor el desarrollo de una Web App.

Seguridad

La seguridad es otro factor clave a tener en cuenta, y donde las aplicaciones nativas tienen ventaja. Si las necesidades de la aplicación requieren de una alta seguridad, sin duda una aplicación nativa es la mejor opción.

CAPÍTULO 4
MÉTODO DE TRABAJO

A la hora de desarrollar un proyecto resulta fundamental la elección de una metodología de trabajo que se adapte bien a sus necesidades. Una buena elección de la metodología de trabajo puede marcar la diferencia entre el éxito y el fracaso de un proyecto. Durante este capítulo se describirá la metodología elegida y cómo se ha adaptado para este proyecto.

Dada la naturaleza del proyecto, enmarcado dentro de una empresa real, se necesitaba una metodología que fuese lo suficientemente ágil, que se adaptase a los requisitos de la aplicación *Try&Win* y permitiese trabajar de manera dinámica, cumpliendo con la fecha de presentación de la aplicación.

Por todo esto, se estableció un método de trabajo basado en hitos que se debían cumplir en modo de iteraciones y de forma incremental. Durante el desarrollo del proyecto se realizaban reuniones semanales para asegurar el cumplimiento de las fechas previstas.

4.1 Scrum

Scrum puede definirse como un marco de trabajo en el que las personas se pueden adaptar a problemas complejos, mientras que de una manera productiva y creativa se entregan productos con el mayor valor posible [14].

La metodología Scrum consiste en un *Equipo Scrum* y sus roles asociados, eventos, artefactos y reglas. Cada uno de estos componentes tiene un propósito específico para el éxito del proyecto.

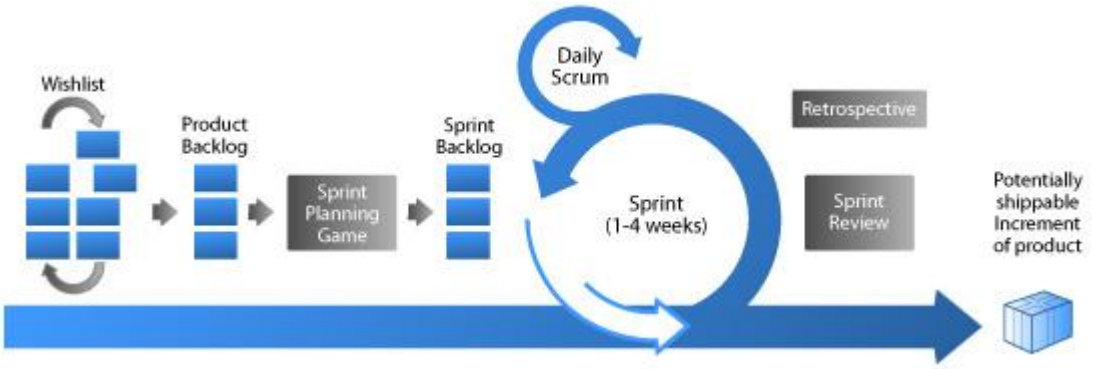


Figura 8: Metodología Scrum

Fuente: *http://dotnetsolutions.cloudapp.net/about-us/agile-development/how-does-it-work*

4.1.1 Roles de trabajo

El *Equipo Scrum* consta de tres roles principales: **Product Owner**, **Equipo de Desarrollo** y **Scrum Master** [14]. El *Equipo Scrum* se auto-organiza y elije la mejor forma de realizar el trabajo, en lugar de ser dirigido desde fuera del equipo.

Product Owner

Es el responsable de maximizar el valor del producto y el trabajo del Equipo de Desarrollo. El *Product Owner* es la única persona responsable de la gestión del *Product Backlog*. La gestión del *Product Backlog* incluye:

- Identificar claramente los ítems del Product Backlog.
- Ordenar los ítems del *Product Backlog* para alcanzar los objetivos de la mejor forma posible.
- Optimizar el valor del trabajo que realiza el *Equipo de Desarrollo*.
- Asegurarse de que el *Product Backlog* sea visible, transparente y claro para todos, y mostrar al equipo que es lo siguiente en lo que se trabajará.
- Asegurar que el *Equipo de Desarrollo* entiende los ítems del *Product Backlog*.

El *Product Owner* puede realizar el mismo este trabajo, o hacer que el *Equipo de Desarrollo* lo haga.

La organización debe respetar las decisiones del *Product Owner* y sus decisiones son visibles en el *Product Backlog*. El *Equipo de Desarrollo* no puede actuar de acuerdo a lo que

diga otra persona y a nadie se le permite decirle al equipo que trabaje en otro conjunto de requisitos diferente al especificado por el *Product Owner*.

En el caso de este proyecto, el rol de *Product Owner* está representado por Jesús García Hernández, director general de avanttic.

Equipo de desarrollo

El *Equipo de Desarrollo* está formado por un conjunto de profesionales encargados de la entrega de incrementos del producto al final de cada *Sprint*. Solo los integrantes del equipo pueden crear estos incrementos.

Los equipos de desarrollo están estructurados y habilitados por la organización para administrar su propio trabajo.

El *Equipo de Desarrollo* tiene las siguientes características:

- Se auto-organizan. Nadie, ni siquiera el *Scrum Master*, pude decirle al *Equipo de Desarrollo* cómo convertir el *Product Backlog* en incrementos funcionales del producto.
- Son multifuncionales, con todas las habilidades necesarias para crear un incremento de producto.
- Scrum no reconoce otro título dentro del *Equipo de Desarrollo* que no sea el de desarrollador, independientemente del trabajo que realice cada miembro.
- Scrum no reconoce sub-equipos de trabajo dentro del *Equipo de Desarrollo*, independientemente de dominios particulares que necesiten ser abordados como ‘pruebas’ o ‘análisis de negocio’.
- Cada miembro del *Equipo de desarrollo* puede estar especializado en ciertas habilidades o áreas, sin embargo, la responsabilidad pertenece al *Equipo de Desarrollo* como un todo.

El tamaño del *Equipo de Desarrollo* debe ser los suficientemente pequeño como para ser ágil, y lo suficientemente grande como para poder realizar un trabajo significativo en un *Sprint*. Los equipos con menos de tres miembros suelen ser menos productivos, ya que pueden encontrar limitaciones dentro de las habilidades del equipo, lo que puede derivar en la no entrega de un incremento potencial del producto. Por el contrario, un equipo muy grande puede ser complejo de gestionar. Un equipo de más de nueve personas requiere de

mucha coordinación. El *Product Owner* y el *Scrum Master* no se cuentan a no ser que realicen trabajo del *Sprint Backlog*.

En este proyecto, este rol lo desempeña el equipo de desarrollo de la aplicación *Try&Win.* Se compone de cinco personas: Rafael Prada Gómez, Jacinto Calderón Collado, Ana Belén Rubio, Rubén Rodríguez Santiago y David Valencia Delgado-Corredor.

Scrum Master

Es el responsable de asegurar que la metodología Scrum es entendida y utilizada. Esto se lleva a cabo asegurándose de que el *Equipo de Desarrollo* se adhiera a la teoría, prácticas y reglas de Scrum.

El *Scrum Master* se encarga de ayudar a las personas que están fuera del *Equipo Scrum* a entender cuáles deben ser sus interacciones con el equipo y cuáles no. Entre el trabajo del *Scrum Master* se encuentra:

Servicio al Product Owner

El Scrum Master sirve al *Product Owner* de varias formas:

- Encontrar técnicas para la gestión eficaz del *Product Backlog.*
- Ayudar al equipo a entender la necesidad de crear ítems del *Product Backlog* claros y concisos.
- Asegurar que el *Product Owner* sabe cómo organizar el *Product Backlog* para maximizar el valor.
- Entender y practicar el desarrollo ágil.
- Facilitar los eventos de Scrum según se solicitan o necesitan.

Servicio al Equipo de Desarrollo

El Scrum Master sirve al *Equipo de Desarrollo* de la siguiente manera:

- Entrena al *Equipo de Desarrollo* en la auto-organización y la multifuncionalidad.
- Ayuda al *Equipo de Desarrollo* a crear productos de alto valor**.**
- Elimina impedimentos para el progreso del *Equipo de Desarrollo.*
- Facilitar los eventos de Scrum según se solicitan o necesitan.
- Entrena al *Equipo de Desarrollo* en entornos organizacionales en los que Scrum aún no ha sido totalmente adoptado y comprendido.

Servicio a la Organización

El Scrum Master sirve a la organización de la siguiente manera:

- Entrena a la organización para la adopción de Scrum.
- Planificar las implementaciones de Scrum dentro de la organización.
- Realizar cambios que incrementen la productividad del *Equipo Scrum*.
- Trabajar con otros Scrum Masters para aumentar la efectividad de la aplicación de Scrum en la organización.

En este caso, el rol de *Scrum Master* estaría desempeñado por Rubén Rodríguez Santiago, consultor de la compañía avanttic y especialista en soluciones cloud.

4.1.2 Eventos de Scrum

Los eventos en Scrum se utilizan para crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Cada evento tiene una duración máxima. Cuando un *Sprint* comienza, se fija su duración, y no puede ser acortada o alargada.

A continuación se describen los diferentes eventos definidos por Scrum.

Sprint

Un *Sprint* es el corazón de Scrum, un intervalo de tiempo de un mes o menos durante el cual se crea un incremento usable del producto.

Cada *Sprint* puede considerarse como un proyecto con un límite de un mes máximo. Cada *Sprint* consta de una definición de lo que se va a construir, un diseño y un plan que guiará la construcción, el trabajo y el producto resultante.

Los *Sprints* están limitados a un mes. Cuando la duración es más larga, la definición de lo que se está construyendo puede cambiar, y por lo tanto la complejidad y el riesgo pueden aumentar.

Sprint Planning

El trabajo que se va a realizar en un *Sprint* se planea en el *Sprint Planning*. Este plan es creado de forma colaborativa por el *Equipo Scrum*.

Los *Sprint Plannings* son eventos que tienen un intervalo de tiempo de un máximo de ocho horas para *Sprints* de un mes. Si los *Sprints* son más cortos, estos eventos suelen serlo también. El *Scrum Master* se asegura de que el evento tenga lugar y de que los asistentes entiendan su propósito.

Daily Scrum

El *Daily Scrum* es un evento de 15 minutos en el cual el *Equipo de Desarrollo* sincroniza actividades y crea un plan para las próximas 24 horas. Esto se hace inspeccionando el trabajo desde el último *Daily Scrum* y pronosticando el trabajo que se podría realizar antes del siguiente. Durante la reunión los miembros del *Equipo de Desarrollo* explican:

- Qué hicieron ayer que ayudara al *Equipo de Desarrollo* a conseguir la meta del *Sprint*.
- Qué van a hacer hoy para ayudar al *Equipo de Desarrollo* a conseguir la meta del *Sprint.*
- Si ve algún impedimento para que él o el equipo de desarrollo alcance la meta del *Sprint*.

Los *Daily Scrum* mejoran las comunicaciones, eliminan otras reuniones, identifican los impedimentos para el desarrollo, destacan y promueven la toma de decisiones rápidas y mejoran el nivel de conocimiento del *Equipo de Desarrollo*.

Sprint Review

El *Sprint Review* se lleva a cabo al final del *Sprint* para inspeccionar el incremento realizado y adaptar el *Product Backlog* si fuese necesario. Durante el *Sprint Review* el *Equipo Scrum* colabora sobre lo que se ha realizado en el *Sprint* y lo que se podría hacer para optimizar el valor.

El *Sprint Review* es una reunión de cuatro horas para *Sprints* de un mes. Para *Sprints* más pequeños la reunión suele ser más corta.

Como resultado del *Sprint Review* se obtiene un *Product Backlog* revisado que define los ítems probables para el próximo *Sprint*.

Sprint Retrospective

El *Sprint Retrospective* es una oportunidad para el *Equipo Scrum* de crear un plan de mejoras que se ejecutará en el próximo *Sprint*.

Esta etapa se produce después del *Sprint Review* y antes del siguiente *Sprint Planning* y tiene una duración de tres horas para *Sprints* de un mes. Para *Sprints* de menor duración, esta etapa suele ser más corta.

El *Scrum Master* se asegura de que el *Equipo Scrum* mejore los procesos y prácticas de desarrollo para hacer más efectivo el próximo *Sprint*. Durante esta etapa, el *Equipo Scrum* planea maneras de incrementar la productividad.

Al final de esta etapa, el *Equipo Scrum* debería haber identificado las mejoras que implementarán en el siguiente *Sprint*. La implementación de esas mejoras implica la adaptación del equipo a Scrum. Aunque estas mejoras pueden ser implementadas en cualquier momento, el *Sprint Retrospective* proporciona una oportunidad formal de enfocarse en la adaptación a Scrum.

4.1.3 Artefactos de Scrum

Los artefactos de Scrum representan trabajo o valor para proporcionar transparencia y oportunidades de inspección y adaptación. Estos artefactos están diseñados para maximizar la transparencia de la información de modo de todos tengan la misma compresión del artefacto.

A continuación se describen los diferentes artefactos definidos por Scrum.

Product Backlog

El *Product Backlog* es una lista ordenada de todo lo que sería necesario en el producto y es la única fuente de requisitos para cualquier cambio que se realice en el producto.

Un *Product Backlog* nunca está completo. Su primer desarrollo solo establece los requisitos inicialmente conocidos y mejor entendidos. Va evolucionando conforme evoluciona el producto y el entorno en el que se utilizará.

El *Product Backlog* enumera todas las características, funciones, requisitos, mejoras y correcciones que se harán en el producto en futuras versiones. Los ítems del *Product Backlog* tienen una descripción, un pedido, una estimación y un valor.

A medida que el producto se utiliza y gana valor, el *Product Backlog* se convierte en una lista más extensa. Los requisitos nunca dejan de cambiar, por lo que el *Product Backlog* es un artefacto vivo. Los cambios en los requisitos de negocio, las condiciones del mercado o la tecnología pueden causar cambios en el *Product Backlog*.

Sprint Backlog

El *Sprint Backlog* es un conjunto de ítems del *Product Backlog* seleccionados para un *Sprint*, más un plan para entregar el incremento del producto y conseguir el objetivo del *Sprint*. El *Sprint Backlog* es una previsión del *Equipo de Desarrollo* sobre qué funcionalidad estará en el próximo incremento del producto y el trabajo necesario para entregar esa funcionalidad.

Cuando se requiere un nuevo trabajo el *Equipo de Desarrollo* lo añade al *Sprint Backlog*. Cuando el trabajo es completado, se actualiza la estimación del trabajo restante. El *Sprint Backlog* es una imagen en tiempo real del trabajo que el *Equipo de Desarrollo* planea realizar durante el *Sprint*, y pertenece únicamente al *Equipo de Desarrollo*. Solo el *Equipo de Desarrollo* puede cambiar el *Sprint Backlog* durante un *Sprint*.

4.2 Marco tecnológico de trabajo

En esta sección se exponen las herramientas, tecnologías y librerías utilizadas para el desarrollo de este proyecto.

4.2.1 Herramientas de gestión de proyectos

SVN

Apache Subversion[7] (SVN), es una herramienta de control de versiones basada en un repositorio. Está distribuida bajo licencia open source. SVN utiliza el concepto de revisión para guardar los cambios que se producen en el repositorio.

[7] Apache Subversion: *https://subversion.apache.org*

En este proyecto se ha utilizado subversión para mantener un control del software desarrollado con cada uno de los frameworks.

TortoiseSVN

TortoiseSVN[8] es un cliente subversión implementado como una extensión de Shell de Windows.

En este proyecto se ha utilizado TortoiseSVN para el acceso y control de los repositorios.

4.2.2 Herramientas para el modelado

Visual Paradigm

Visual Paradigm[9] es una herramienta CASE (Computer Aided Software Engineering o Ingeniería de Software Asistida por Computador) profesional para el modelado UML (Unified Modeling Language o Lenguaje Unificado de Modelado). Visual Paradigm ha sido realizada con el propósito de soportar el ciclo de vida completo del proceso de desarrollo del software, a través de la realización de una gran variedad de tipos de diagramas.

En este proyecto se utiliza esta herramienta para realizar los diagramas de secuencia de la aplicación *#smact Try&Win*.

Balsamiq Mockups

Balsamiq Mockups[10] es una aplicación que se utiliza para el diseño de borradores y bocetos de las interfaces de usuario. Permite hacer representaciones de todos los elementos utilizados para construir webs y aplicaciones, como las pantallas, los títulos, menús, imágenes, vídeos, etc., de modo que nos permite tener una rápida aproximación de la solución que se quiere desarrollar.

[8] TortoiseSVN: *https://tortoisesvn.net*
[9] Visual Paradigm: *https://www.visual-paradigm.com*
[10] Balsamiq Mockups: *https://balsamiq.com/products/mockups*

Microsoft Project

Microsoft Project[11] es un software para la administración de proyectos que está diseñado, desarrollado y comercializado por Microsoft, para asistir a administradores de proyectos. Este software ayuda en el desarrollo de planes, la asignación de recursos a tareas, el seguimiento del progreso del proyecto, la administración de presupuestos y el análisis de las cargas de trabajo.

4.2.3 Herramientas y tecnologías para el desarrollo del proyecto

4.2.3.1 Editores y entornos de desarrollo

Sublime text

Sublime text[12] es un editor de texto y de código multiplataforma, escrito en C++ y Phyton para el soporte de plugins. Entre sus características principales destacan: soporte nativo para varios lenguajes, soporte de snippets y plugins, configuración total de keybindings, paleta de comandos y búsqueda dinámica.

NetBeans

NetBeans[13] es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) libre, creado principalmente para el lenguaje de programación Java. NetBeans permite desarrollar las aplicaciones a partir de un conjunto de componentes software llamados *módulos*, permitiendo que las aplicaciones sean extensibles mediante la agregación de nuevos módulos.

JDeveloper

JDeveloper[14] es un entorno de desarrollo integrado desarrollado por Oracle pensado para simplificar el desarrollo de aplicaciones basadas en Java, abordando cada paso del ciclo de vida de la aplicación, desde el diseño hasta la codificación, optimización y creación de

[11] Microsoft Project: *https://products.office.com/es-es/project*
[12] Sublime Text: *https://www.sublimetext.com*
[13] NetBeans: *https://netbeans.org*
[14] JDeveloper: *http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html*

perfiles de despliegue. JDeveloper se centra en proporcionar al desarrollador un enfoque visual y declarativo para el desarrollo de las aplicaciones.

4.2.3.2 Tecnologías de Oracle JET

Hammer.js

Hammer.js[15] es una librería open source que puede reconocer gestos mediante el tacto en la pantalla o el uso del ratón. Tiene soporte para el reconocimiento de gestos tales como pulsar sobre la pantalla, zoom, presión de pantalla durante un tiempo definido, rotación, desplazamiento y click con varios dedos al mismo tiempo.

jQuery UI

jQuery UI[16] es una librería JavaScript que proporciona un conjunto de interacciones con la interfaz, efectos, widgets y temas. Permite añadir comportamientos complejos como Drag&Drop, redimensionado de elementos, selección de elementos en listas, ordenación de listas, seleccionar el tamaño de los elementos, etc.

Knockout.js

Knockout[17] es una implementación JavaScript del MVVM (Model-View-ViewModel) con plantillas. Está pensado para ayudar a los desarrolladores a crear interfaces de usuario adaptables. Permite realizar bindings declarativos, refresco automático de la interfaz de usuario y detección de dependencias.

RequireJS

RequireJS[18] es una librería con la cual cargar archivos y módulos JavaScript. Permite aislar mediante módulos los componentes de nuestra aplicación cliente y resolver las dependencias de los mismos. Está optimizado para el uso en el navegador, aunque puede utilizarse en otros entornos JavaScript, como Rhino y Node.

[15] Librería Hammer.js: *http://hammerjs.github.io*
[16] jQueryUI: *https://jqueryui.com*
[17] Knockout: *http://knockoutjs.com*
[18] RequireJS: *http://requirejs.org*

Grunt

Grunt[19] es una herramienta utilizada para realizar tareas frecuentes de forma automática, como la minificación, compilación, pruebas unitarias, etc. Permite definir tareas personalizadas en un archivo llamado *Gruntfile* y ejecutarlas a través de la línea de comandos. Grunt está escrito en Node.js[20] y se distribuye a través de npm[21].

4.2.3.3 Tecnologías de Oracle MAF

Java

Java[22] es un lenguaje de programación de propósito general, concurrente y orientado a objetos. La sintaxis de Java deriva en gran medida de C y C++, aunque proporciona utilidades de mayor nivel que estos lenguajes. Las aplicaciones de Java se compilan generalmente a bytecode que puede ejecutarse en la máquina virtual Java (JVM, por sus siglas en inglés).

AMX

MAF Application Mobile XML (AMX) es un lenguaje de marcas extensible para la creación de interfaces de usuario. Las aplicaciones implementadas con AMX proporcionan una experiencia nativa al usuario.

4.2.3.4 Tecnologías de Ionic

Angular

AngularJS[23] es un framework MVC (Model-View-Controller) de JavaScript de código abierto para desarrollar aplicaciones SPA (Single-Page Applications o aplicaciones de página única) en el lado del cliente.

[19] Grunt: *http://gruntjs.com*
[20] Node.js: *https://nodejs.org/es*
[21] npm: *https://www.npmjs.com*
[22] Java: *https://www.java.com/en*
[23] Angular: *https://angularjs.org*

TypeScript

TypeScript[24] es un lenguaje de programación libre y de código abierto desarrollado por Microsoft. Es un superconjunto de JavaScript que añade tipado estático y objetos basados en clases. TypeScript extiende la sintaxis de JavaScript, por lo que cualquier código JavaScript debería funcionar sin problema. El compilador de TypeScript traduce el código a JavaScript original.

Gulp

Gulp[25] es una herramienta JavaScript de código abierto y extensible usada como un sistema de construcción de front-end. Está basado en Node.js y npm, y se usa para la automatización de tareas repetitivas como la minificación, concatenación, búsqueda en caché, tests unitarios, optimización, etc.

4.2.3.1 Tecnologías comunes a los tres frameworks

En este apartado se describen las tecnologías que son usadas por los tres frameworks: JET MAF e Ionic.

HTML

HTML[26] (HyperText Markup Language) es un lenguaje de marcas que se utiliza para el desarrollo de páginas en Internet. Es de formato abierto y permite ordenar y etiquetar diversos documentos dentro de una lista. HTML es un estándar a cargo de la World Wide Web Consortium (W3C), organización que se dedica a la estandarización de casi todas las tecnologías ligadas a la web.

JavaScript

JavaScript (abreviado como JS) es un lenguaje ligero e interpretado. Se trata de un lenguaje script multi-paradigma, basado en prototipos, dinámico, con soporte para estilos de programación funcional, orientado a objetos e imperativo. Se utiliza principalmente en el

[24] TypeScript: *https://www.typescriptlang.org*
[25] Gulp: *http://gulpjs.com*
[26] World Wide Web Consortium, HTML: *https://www.w3.org/html*

lado del cliente, implementado como parte de un navegador web para la creación de páginas web dinámicas, aunque existen formas de JavaScript para el lado del servidor, como Node.js.

CSS

CSS[27] es un lenguaje para definir y crear la presentación de documentos estructurados en HTML o XML. La idea del desarrollo de CSS surgió de separar la estructura de los documentos de su presentación. Al igual que con HTML, la W3C se encarga de la especificación de las hojas de estilo, entre las que se encuentra CSS, y que sirven de estándar para los navegadores.

Cordova

Apache Cordova es un entorno de desarrollo de aplicaciones móviles que permite el uso de tecnologías web estándar para el desarrollo multiplataforma. Permite el desarrollo para Android, iOS y otros sistemas operativos sin la necesidad de programar en lenguajes nativos como Java, Objective-C, etc.

Figura 9: Herramientas y tecnologías utilizadas

[27] World Wide Web Consortium, CSS: *https://www.w3.org/Style/CSS*

CAPÍTULO 5
RESULTADOS

En este capítulo se mostrarán los resultados obtenidos tras el estudio sobre los distintos frameworks de desarrollo de aplicaciones móviles híbridas. En primer lugar se presentará la planificación del trabajo, mostrando los hitos planificados mediante un diagrama de Gantt. Posteriormente se presentará un análisis comparativo sobre las características ofrecidas por los frameworks JET, MAF e Ionic v2. Por último, se realiza una ponderación de los resultados obtenidos en la comparativa, para terminar con una conclusión sobre estos resultados.

Es necesario que algunas de las características de estos frameworks se analicen sobre aplicaciones reales desarrolladas con dichos frameworks. Es por ello que para la realización de este proyecto ha sido necesario el desarrollo de una misma aplicación móvil con cada uno de ellos. Ya que este TFM se ha realizado en la compañía avanttic, se ha aprovechado para utilizar una aplicación desarrollada en un proyecto real dentro de la empresa. El hecho de que se trate de un proyecto real hace que la experiencia obtenida mediante su desarrollo sirva para tener un mejor conocimiento sobre las fortalezas y debilidades de cada framework.

En el capítulo 6 se exponen las conclusiones obtenidas tras el análisis mostrado en este capítulo.

5.1 Planificación

El desarrollo de este proyecto se ha dividido en 6 iteraciones que se han completado progresivamente. En la Figura 10 se muestra el diagrama de Gantt con la planificación de cada iteración.

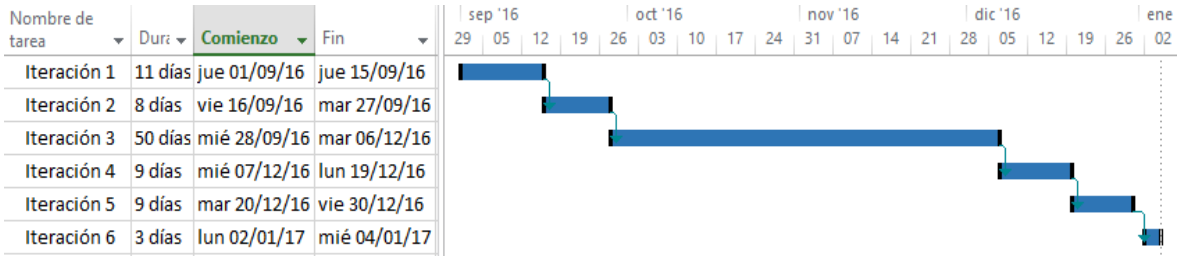


Figura 10: Diagrama de Gantt del plan del proyecto

A continuación se detalla el trabajo realizado en cada una de las iteraciones para llevar a cabo este proyecto.

Iteración 1

En esta primera iteración se realizó un **estudio sobre los frameworks** que se han utilizado para desarrollar la aplicación *#smact Try&Win* y sobre los que se centra la comparativa realizada en el proyecto. En el estudio se incluyó otros frameworks existentes que resultan de interés.

Como resultado de este estudio se obtuvo una mejor comprensión de los frameworks existentes, y un conocimiento sobre su arquitectura y funcionamiento que sentaron las bases para el posterior desarrollo de la aplicación.

Esta iteración se llevó a cabo entre las fechas 01/09/2016 y 15/09/2016.

Iteración 2

Una vez que se tenía un conocimiento base sobre los frameworks que se iban a comparar, el siguiente paso fue la preparación de los **entornos de desarrollo** de cada uno de ellos.

Una vez preparados los entornos de desarrollo, se realizaron una serie de ejemplos con cada framework que incluían el desarrollo de páginas de ejemplo y el acceso a distintos servicios del dispositivo (cámara, GPS, bluetooth) con la integración de algunos plugins Cordova. Esto sirvió como primera toma de contacto para familiarizarse en el desarrollo en estos frameworks.

Esta iteración se llevó a cabo entre el 16/09/2016 y el 27/09/2016.

Iteración 3

El siguiente paso fue el **desarrollo de la aplicación** *#smact Try&Win* con cada uno de los frameworks. Este desarrollo sirvió para conocer más en profundidad las características de cada framework y los problemas que pueden surgir a la hora de desarrollar un proyecto real para una compañía.

Las etapas de esta iteración se pueden dividir en 3 etapas:

- Desarrollo de la aplicación con Oracle JET. Este periodo estuvo comprendido entre el 28/09/2016 y el 26/10/2016. La aplicación desarrollada fue presentada en el Oracle Digital Day, evento que fue presentado por Oracle en Madrid el 27/10/2016. Esta experiencia se encuentra detallada en el Anexo A.
- Desarrollo de la aplicación con Ionic 2. Este periodo estuvo comprendido entre 27/10/2016 y 15/11/2016.
- Desarrollo de la aplicación con Oracle MAF. Este periodo estuvo comprendido entre el 16/11/2016 y el 06/12/2016.

Iteración 4

En esta iteración se llevó a cabo la **medición** de distintas características de las aplicaciones desarrolladas para determinar el rendimiento de cada framework. Los resultados de esta iteración están detallados más adelante en este capítulo.

Esta etapa se realizó entre las fechas 07/12/2016 y 19/12/2016.

Iteración 5

En esta iteración se realizó una **comparativa** entre otros aspectos y características ofrecidos por el framework, tales como el sistema de *binging*, el acceso a los servicios de los dispositivos, la seguridad, la documentación y comunidad de soporte, etc. Los resultados de esta iteración se detallan en este capítulo.

Esta iteración se realizó entre el 20/12/2016 y el 30/12/2016.

Iteración 6

Una vez terminada la comparativa sobre las características de cada framework, se llevó a cabo una **ponderación de los resultados** para determinar cuál de los frameworks ofrecía unas mejores características en su conjunto. Al igual que en las iteraciones 5 y 6, puede verse este resultado más adelante en este mismo capítulo.

Con la obtención de estos resultados, se pretende ayudar a la compañía avanttic a tener una visión más amplia de las ventajas e inconvenientes que ofrecen cada uno de estos frameworks, y la capacidad de analizar cuál de ellos es más adecuado para un desarrollo determinado, lo que le permitirá a la compañía ofrecer más alternativas a sus clientes y un mejor servicio.

Esta iteración se realizó entre el 02/01/2017 y el 04/01/2017.

5.2 Estudio de los frameworks de desarrollo

En este apartado se presenta un estudio de los tres frameworks de desarrollo en los cuales se centra este TFM y presenta las características de algunos otros frameworks que pueden encontrarse en el mercado.

5.2.1 Oracle JavaScript Extension Toolkit

Oracle JavaScript Extension Toolkit (JET) [15] es un framework JavaScript orientado al desarrollo de aplicaciones cliente, ya sean aplicaciones web o aplicaciones híbridas para dispositivos móviles. Oracle JET se compone de un conjunto de librerías JavaScript de código abierto, además de un conjunto de servicios y funcionalidades añadidas por Oracle, con el fin de ayudar a los desarrolladores a construir aplicaciones de una forma mejor y más rápida.

Tiene una arquitectura MVVC (Model-View-ViewModel). En este tipo de arquitecturas el modelo representa los datos de la aplicación, la vista es la presentación de los datos y el ViewModel se encarga de procesar los datos de forma que puedan presentados. La Figura 11 muestra este tipo de arquitectura.

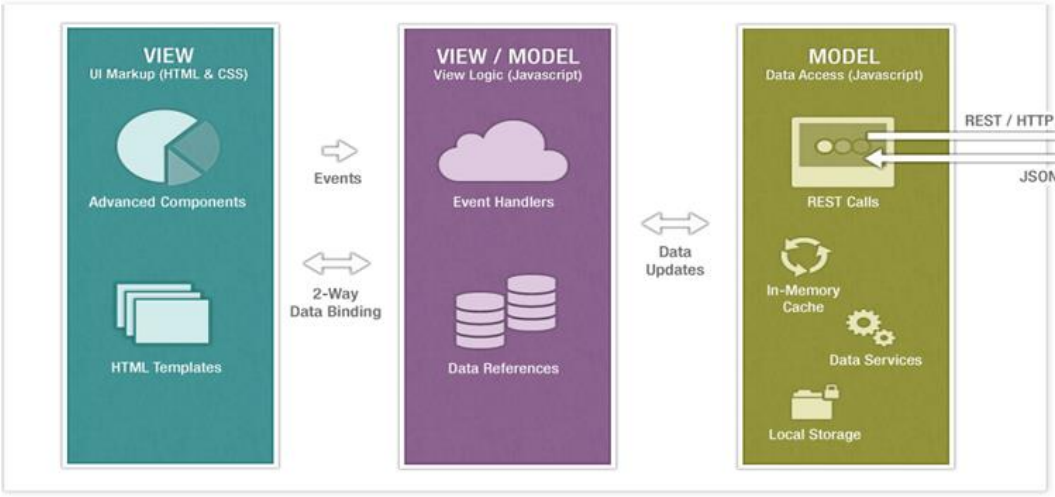


Figura 11: Arquitectura MVVC

Fuente: *https://blog.avanttic.com/tag/mvvm*

Las principales características de JET son:

- Desarrollo multiplataforma.
- Uso de plugins Cordova[28] para el acceso a las características del dispositivo.
- Gran variedad de componentes.
- Compatibilidad de diseño de interfaz para varias plataformas.
- Sistema de enrutamiento avanzado, que proporciona soporte para el historial de navegación HTML5.
- Enlace bidireccional avanzado entre la capa de modelo y la interfaz de usuario.
- Gestión de recursos inteligentes. Mediante la librería RequirJS, JET aumenta el rendimiento de la aplicación.
- Arquitectura modular.
- Internacionalización con soporte para 27 idiomas.

Entre las librerías de las que se compone Oracle JET destacan:

- RequireJS[29], usada para para facilitar la gestión de las librerías. Proporciona un mecanismo de carga asíncrona a los módulos y sus dependencias.
- JQueryUI[30], que proporciona un conjunto de interacciones con el usuario, efectos, temas y widgets para la creación de interfaces de usuario.

[28] Apache Cordova: *https://cordova.apache.org*
[29] Require.js: *http://requirejs.org*
[30] jQueryUI: *https://jqueryui.com*

- Knockout[31], que permite enlazar elementos HTML con el modelo de datos.
- Hammer.JS[32], para el reconocimiento de gestos en la pantalla (touch gestures) del dispositivo.

La arquitectura de Oracle JET se muestra en la Figura 12.

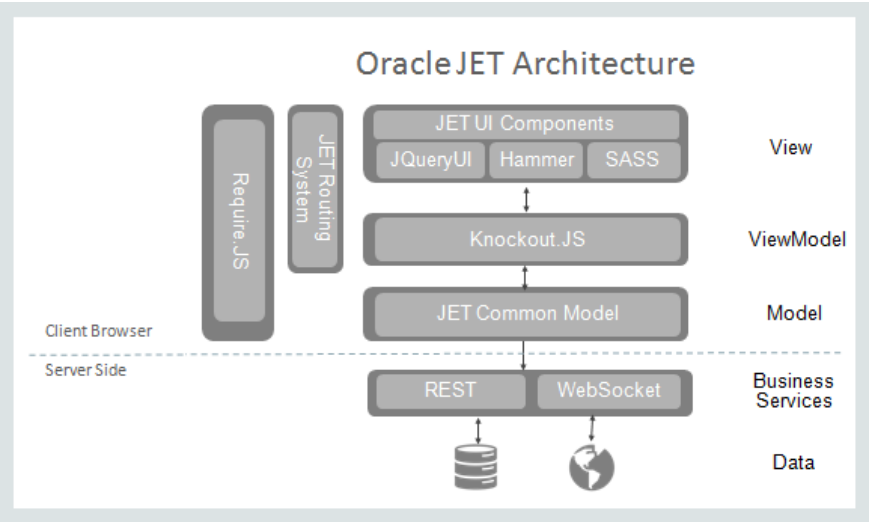


Figura 12: Arquitectura de Oracle JET

Fuente: *https://docs.oracle.com/middleware/jet200/jet/developer/GUID-293CB342-196F-4FC3-AE69-D1226A025FBB.html*

5.2.1.1 Estructura del proyecto

Para crear una nueva aplicación híbrida con JET podemos utilizar Yeoman[33]. Yeoman es un conjunto de herramientas construidas sobre Node.js[34], que permite a los desarrolladores generar el esqueleto de una aplicación con unos pocos comandos.

Mediante Yeoman, podemos indicar una serie de opciones que permiten, entre otras cosas, elegir una serie de plantillas. Por ejemplo, podemos indicarle que queremos generar una aplicación utilizando la plantilla "navdrawer". De este modo se nos genera una aplicación de ejemplo que trae un menú desplegable por defecto.

Una vez generado, la estructura del directorio de la aplicación tendrá un aspecto similar al de la Figura 13.

[31] Knockout: *http://knockoutjs.com*
[32] Hammer.JS: *http://hammerjs.github.io*
[33] Yeoman: *http://yeoman.io*
[34] Node.js: *https://nodejs.org/es*

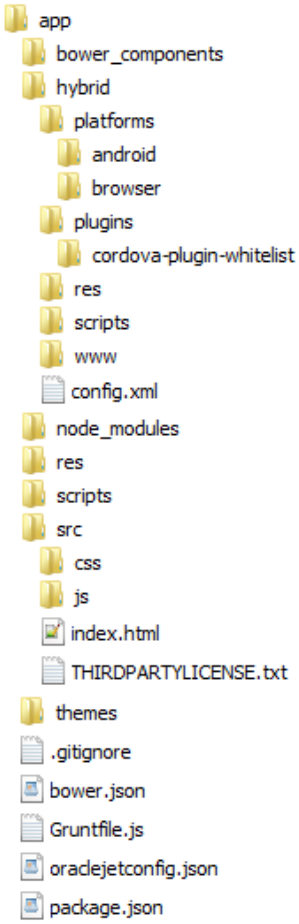


Figura 13: Estructura de un proyecto JET

Las carpetas y ficheros más importantes son:

- Carpeta **bower_components.** Contiene las librerías de terceros y paquetes de Oracle JET gestionados por Bower[35].
- Carpeta **hybrid.** Contiene archivos específicos de las plataformas para las que se ha compilado el proyecto.
- Archivo **hybrid/config.xml.** Contiene las opciones de configuración globales de Cordova. Se puede editar este archivo para especificar las características del API de Cordova, los plugins y las opciones de una plataforma específica.
- Carpeta **node_modules.** Contiene los módulos de Node.js usados por el framework.
- Carpeta **res.** Contiene los iconos de la aplicación y las imágenes de la pantalla de bienvenida.

[35] Bower: *https://bower.io*

- Carpeta **scripts.** Contiene los scripts de compilación de Oracle JET.
- Carpeta **src.** Esta carpeta es la raíz de nuestra aplicación. Contiene, entre otros, los archivos que componen los módulos de nuestra aplicación. En esta carpeta se encuentra el archivo index.html, que sirve como punto de entrada de la aplicación.
- Carpeta **themes.** Contiene los temas del Look & Feel utilizados por cada plataforma.

5.2.2 Oracle Mobile Application Framework

Oracle Mobile Application Framework (MAF) [16] es un framework de desarrollo híbrido que permite desarrollar aplicaciones para las plataformas Android, iOS y Microsoft Windows Phone a partir de un código único.

Figura 14: Oracle Mobile Application Framework

Fuente: *http://rene-ace.com/event/conociendo-maf*

MAF se integra con *JDevelper* y *Oracle Enterprise Pack for Eclipse*. Estos IDEs ofrecen un desarrollo visual y declarativo, haciendo el proceso más rápido y sencillo para el desarrollador. Entre las características de estos IDEs se encuentran:

- Editores de flujos y páginas visuales.
- Paleta de componentes con soporte Drag & Drop.
- Paleta de *Data Control* para un uso simplificado de servicios, lógica de backend y características del dispositivo.
- Panel de estructura interactivo, que permite manipular la estructura de las páginas.
- Inspector de propiedades para la manipulación de características y atributos.

Además de estas características, MAF ofrece:

- Desarrollo multiplataforma.
- Gran variedad de componentes.
- Uso de plugins para acceder a los servicios del dispositivo.
- Compatibilidad de interfaz de usuario para varias plataformas.
- Enlace bidireccional entre el modelo y la interfaz de usuario.

La Figura 15 muestra la arquitectura de Oracle MAF. Como puede verse, MAF es un contenedor nativo que se despliega en el dispositivo. Tiene un enfoque Modelo Vista Controlador (MVC), que separa la capa de presentación de la capa lógica. Este contenedor nativo permite a la aplicación MAF comportarse como una aplicación nativa tanto en Android como en iOS. MAF utiliza el API de Cordova para acceder a los recursos del dispositivo, permite interactuar con la base de datos local SQLite y el acceso a los recursos de la parte el servidor. También permite el uso de notificaciones push.

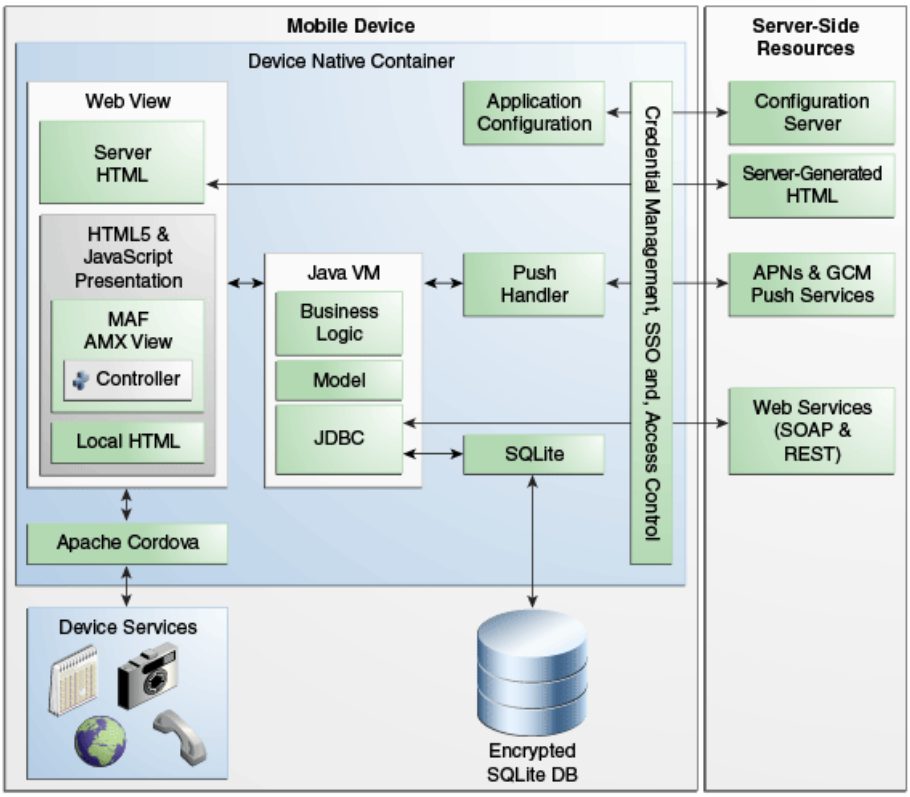


Figura 15: Arquitectura de Mobile Application Framework

Fuente: *http://docs.oracle.com/middleware/mobile200/mobile/develop-oepe/oepe-maf-about.htm*

5.2.2.1 Estructura del proyecto

Oracle MAF se integra con JDeveloper[36], un entorno de desarrollo integrado pensado para facilitar el desarrollo de aplicaciones basadas en Java. Para crear una aplicación MAF con JDeveloper debemos seleccionar File > New > Application, y elegir la opción *Mobile Application Framework Application*.

La Figura 16 muestra la estructura generada para el proyecto.

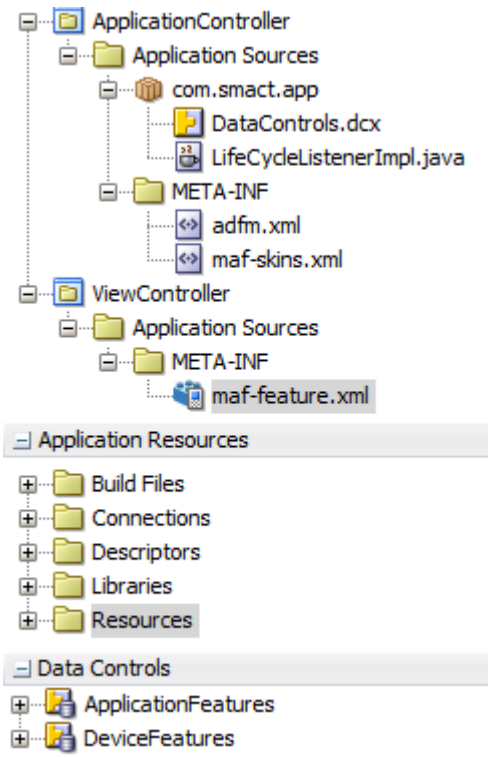


Figura 16: Estructura de un proyecto MAF

Entre los elementos del proyecto podemos destacar:

- Archivo **LifeCycleListenerImpl.java.** Proporciona a los desarrolladores la estructura básica que necesitan para incluir su propia funcionalidad durante las diferentes etapas de la aplicación.
- Archivo **maf-feature.xml.** Archivo donde se especifican las *features* o módulos de la aplicación.

[36] JDeveloper: *http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html*

- **Data Controls.** Un data control es una abstracción de la implementación de un servicio usando interfaces que describen las operaciones del servicio, sus colecciones, información sobre sus propiedades, métodos y tipos implicados. Por defecto encontramos dos Data Control en una aplicación:
    - **ApplicationFeatures.** Proporciona métodos para implementar la navegación de una aplicación MAF.
    - **DeviceFeatures.** Proporciona acceso a características del dispositivo que pueden ser utilizadas en la aplicación MAF.
- En la carpeta *Descriptors* de la ventana *Application R*esources podemos encontrar el archivo **maf-application.xml,** en el que podemos configurar el nombre de la aplicación, los menús de navegación, la seguridad y las opciones de acceso al dispositivo por parte de la aplicación.
- Carpeta **Resources.** Podemos encontrar imágenes que utilizará la aplicación, como los iconos y las imágenes de bienvenida.

5.2.3 Ionic

Ionic [17] es un framework de desarrollo de aplicaciones híbridas con soporte para las plataformas Android, iOS y Windows Phone. Es un proyecto open source, bajo licencia MIT. Este framework cuenta con dos versiones. La versión 1 de Ionic utiliza JavaScript y Angular[37], la versión 2, actualmente en beta, utiliza TypeScript y Angular 2. Debido a la expectación creada alrededor de Angular 2, este trabajo se centra en Ionic V2.

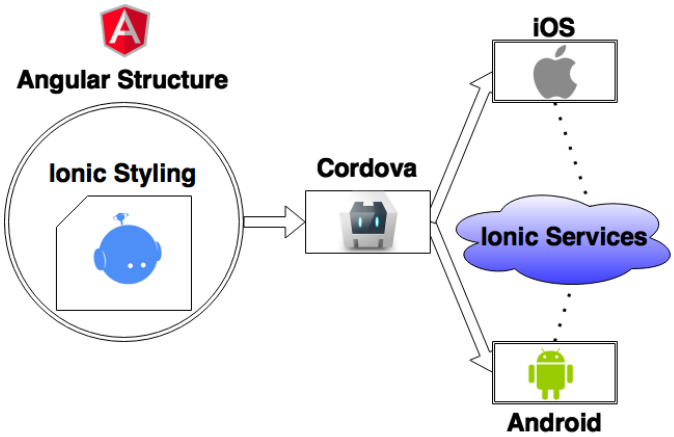


Figura 17: Ionic Framework

Fuente: *http://www.objectivetruth.ca/projects/2015/08/30/ionic-post-review.html*

[37] AngularJS: *https://angularjs.org*

Las principales características principales de Ionic son:

- Desarrollo multiplataforma.
- Uso de plugins Cordova para acceder a las características del dispositivo.
- Wrappers de TypeScript para el uso de plugins Cordova.
- Gran variedad de componentes.
- Enlace bidireccional entre el modelo y la interfaz de usuario.
- Varios sistemas de navegación.
- Compatibilidad en el diseño de la interfaz de usuario para distintas plataformas.

Ionic v2 está basado en Angular 2, por lo que sigue una arquitectura MVC (Modelo Vista Controlador). Ionic se combina con Apache Cordova para acceder a las características del dispositivo, como la cámara, el GPS, bluetooth, etc. Ionic utiliza Gulp[38] como herramienta de construcción para compilar las aplicaciones. La Figura 18 muestra esta arquitectura.

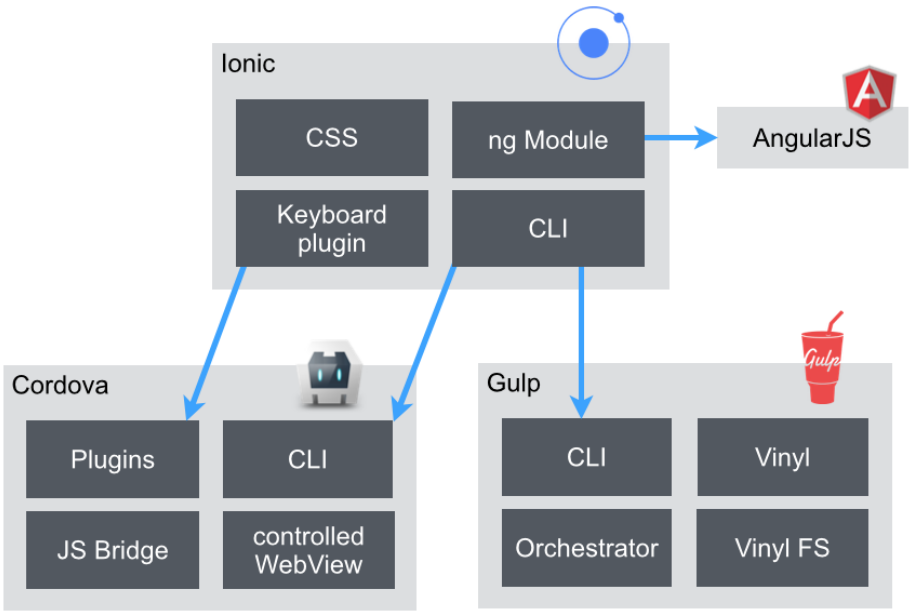


Figura 18: Arquitectura de Ionic v2

Fuente: *https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-rise*

[38] Gulp: *http://gulpjs.com*

5.2.3.1 Estructura del proyecto

Podemos crear una nueva aplicación de Ionic desde la terminal de nuestro equipo, haciendo uso de la utilidad de línea de comandos que proporciona Ionic. Al igual que Yeoman, nos permite indicar una serie de opciones y elegir una plantilla para generar la aplicación. La plantilla "sidemenu", que se utilizó durante la realización de este proyecto, genera una aplicación con un menú lateral desplegable. Una consideración importante a tener en cuenta es que en el momento de generar la aplicación podemos elegir si queremos que se genere para la versión 1 o la versión 2 de Ionic.

Tras generar la aplicación, el directorio del proyecto tendrá la estructura que muestra la Figura 19.

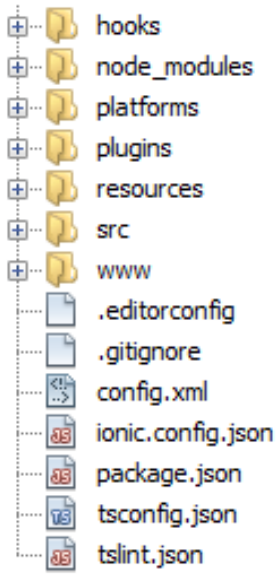


Figura 19: Estructura de un proyecto Ionic

Los elementos más importantes son:

- Archivo **src/index.html.** Es el punto de entrada principal de la aplicación. Su propósito es configurar los scripts y archivos CSS necesarios, y arrancar nuestra aplicación.
- Carpeta **src.** En esta carpeta se encuentra el código sin compilar, y es donde se realizará la mayor parte del trabajo para la aplicación.
- Carpeta **www.** Contiene el código compilado de la aplicación.
- Carpeta **hooks.** Contiene scripts que se ejecutarán durante el proceso de compilación.

- Carpeta **node_modules.** Contiene las dependencias de los módulos de Node.js
- Carpeta **platforms.** Contiene archivos de las plataformas para las que se ha compilado el proyecto.
- Archivo **config.xml.** Contiene las opciones de configuración relevantes para la compilación del proyecto para las distintas plataformas.
- Carpeta **resources.** Contiene las imágenes utilizadas por la aplicación, como los iconos y las imágenes de bienvenida.

5.2.4 Otros frameworks de desarrollo

En este apartado se habla sobre otros frameworks de desarrollo que, aunque no entran dentro de la comparativa realizada en este proyecto, resulta de interés tenerlos en cuenta.

5.2.4.1 Onsen UI

Onsen UI[39] es un framework de desarrollo de aplicaciones híbridas móviles basadas en Cordova y Phonegap[40].

Onsen UI se encuentra actualmente en la versión 2. Su principal ventaja es que tiene soporte para distintos frameworks JavaScript: Angular, Angular 2, React[41], Vue.js[42] y Meteor[43]. Las principales características de Onsen UI son:

- Desarrollo multiplataforma.
- Uso de plugins Cordova.
- Gran variedad de componentes.
- Compatibilidad con el diseño de la interfaz de usuario para Android e iOS.

5.2.4.2 Framework7

Framework7[44] es un framework open source para el desarrollo de aplicaciones web y aplicaciones móviles híbridas con soporte para las plataformas Android e iOS. Es compatible

[39] Onsen UI: *https://onsen.io*
[40] PhoneGap: *http://phonegap.com*
[41] React: *http://cobaltians.org*
[42] Vue.js: *https://vuejs.org*
[43] Meteor: *https://www.meteor.com*
[44] Framework7: *https://framework7.io*

con Cordova y puede combinarse junto con React y Angular. Las principales características de Framework7 son:

- Desarrollo multiplataforma.
- Uso de plugins Cordova.
- Uso de componentes.
- Independencia con librerías de terceros.
- Look & Feel compatible con Android e iOS.
- Animaciones de alto rendimiento.
- Uso de su propia librería DOM, llamada Dom7.
- Navegación flexible, que proporciona diferentes formas de gestionar las páginas.

5.2.4.3 Xamarin

Xamarin[45] es un framework de desarrollo de aplicaciones híbridas en Android, iOS y Windows. Xamarin utiliza C# como principal lenguaje de programación y puede integrarse con Visual Studio, ofreciendo un desarrollo más sencillo y rápido, funciones Drag & Drop y paletas de componentes, entre otras.

Sus principales características son:

- Desarrollo multiplataforma.
- Uso de plugins para acceder a los servicios del dispositivo.
- Interfaces de usuario nativas.
- Gran variedad de componentes.

5.2.4.4 jQuery Mobile

jQuery Mobile[46] es un framework de desarrollo basado en HTML5 para aplicaciones móviles multiplataforma. El propósito de este framework es ayudar a que las aplicaciones web funcionen correctamente en todos los navegadores móviles.

Sus principales características son:

[45] Xamarin: *https://www.xamarin.com*
[46] jQuery Mobile: *https://jquerymobile.com*

- Desarrollo multiplataforma, con soporte para Android, iOS, Windows Phone, Blackberry y Symbian.
- Uso de plugins Cordova para acceder a los servicios del dispositivo.
- Uso de componentes.

5.2.4.5 React Native

React Native[47] es un framework de desarrollo de aplicaciones mediante JavaScript y React, con soporte para Android e iOS. Su propósito es crear aplicaciones lo más similar posible a una aplicación nativa, en lugar de aplicaciones híbridas que se ejecutan en un navegador. A diferencia de otros frameworks, que utilizan HTML y CSS para su renderizado, React Native utiliza componentes nativos.

Las principales características de React Native son:

- Desarrollo multiplataforma.
- Uso de módulos nativos para acceder a las características del dispositivo.
- Uso de componentes nativos.

5.2.4.6 Native Script

Native Script[48] es un framework de desarrollo de aplicaciones móviles para Android e iOS. Es un framework open source que permite crear aplicaciones utilizando Angular, TypeScript o JavaScript. Al igual que React Native, utiliza componentes nativos para el renderizado de las aplicaciones.

Las principales características de Native Script son:

- Desarrollo multiplataforma.
- Interfaz de usuario nativa.
- Uso de plugins de npm, CocoaPods y Gradle. Además de plugins específicos de Native Script.
- Uso de módulos para acceder a las características del dispositivo.

[47] React Native: *http://facebook.github.io/react-native*
[48] Native Script: *https://www.nativescript.org*

5.2.4.7 Sencha Touch

Sencha Touch[49] es un framework JavaScript de desarrollo multiplataforma para aplicaciones móviles, que sigue el patrón MVC (Model-View-ViewModel). Con Sencha Touch se pueden desarrollar aplicaciones para las plataformas Android, iOS, Windows Phone y Blackberry.

Las características de este framework son:

- Desarrollo multiplataforma.
- Uso de plugins Cordova para acceder a las características del dispositivo.
- Más de 50 componentes para la creación de interfaces de usuario.
- Look & Feel nativo.
- Paquete gráfico, con distintos tipos de gráficas con soporte para las interacciones con gestos.

5.2.4.8 DevExtreme

DevExtreme[50] es un framework de desarrollo tanto de aplicaciones web como de aplicaciones móviles multiplataforma. Tiene soporte para Android, iOS, Windows Phone y Tizen. Este framework permite desarrollar las aplicaciones usando JavaScript o TypeScript.

Sus principales características son:

- Desarrollo multiplataforma.
- Puede combinarse con plugins Cordova para acceder a los servicios del dispositivo.
- Look & Feel automático para las distintas plataformas.
- Widgets con soporte para gestos.
- Widgets para visualización de datos.
- Patrón de diseño MVVC.
- Integración con Visual Studio.

[49] Sencha Touch: *https://www.sencha.com/products/touch*
[50] DevExtreme: *https://js.devexpress.com*

5.2.4.9 Cobalt

Cobalt[51] es un framework de desarrollo de aplicaciones multiplataforma para Android e iOS. Cobalt permite compartir entre un 60% y un 70% de código entre las plataformas, manteniendo algunas características nativas del sistema, haciendo posible que las aplicaciones mantengan el aspecto de la plataforma en la que se ejecutan.

Las principales características de Cobalt son:

- Desarrollo multiplataforma.
- Navegación nativa.
- Look & Feel adaptado a cada plataforma.
- Variedad de componentes.
- Uso de plugins Cobalt para acceder a los servicios del dispositivo.

5.2.4.10 PhoneGap

PhoneGap[52] es un framework para el desarrollo de aplicaciones móviles híbridas con JavaScript, HTML y CSS. Se trata de un framework open source distribuido por Cordova.

Las características de PhoneGap son:

- Desarrollo multiplataforma.
- Uso de plugins cordova.
- PhoneGap desktop app, una aplicación de escritorio que permite crear aplicaciones Phonegap sin necesidad de utilizar la línea de comandos.
- PhoneGap developer, una aplicación móvil que permite conectar el dispositivo móvil con el ordenador, permitiendo al desarrollador ver en el dispositivo móvil los cambios que se realizan en el proyecto.
- Phonegap build, un servicio en la nube ofrecido por PhoneGap que nos permite compilar nuestra aplicación.

[51] Cobalt: *http://cobaltians.org*
[52] PhoneGap: *http://phonegap.com*

5.3 Comparativa detallada de los frameworks

En este apartado se expone un estudio comparativo de las características que ofrecen Oracle JET, Oracle MAF e Ionic v2.

5.3.1 El sistema de enlace de datos

El enlace de datos (data binding, en inglés) es la sincronización de los datos entre el modelo y la vista. En el proceso de data binding, cada cambio en los datos se ve reflejado automáticamente en los elementos que están vinculados a esos datos. A continuación se expone como se realiza el proceso de data binding en cada uno de los tres frameworks.

El sistema de data binding en Ionic 2 es proporcionado por Angular 2, mientras que en Oracle JET esta funcionalidad nos la da Knockout.js. La diferencia fundamental es que Angular gestiona la aplicación completamente y define directrices sobre como la aplicación debe estructurarse, mientras que con Knockout la estructura depende completamente del desarrollador. En el caso de Oracle MAF, el enlace de datos nos lo proporciona su capa de bindings. Esta capa es la encargada de simplificar la conexión de los servicios de datos con la interfaz de usuario.

Actualización de los datos

Para mantener sincronizados los datos mostrados en la interfaz de usuario con los datos a los que está vinculada en la capa lógica es necesario conocer cuando se produce un cambio en estos datos.

Knockout (Oracle JET) utiliza el patrón observador para chequear cambios y notificarlos a los suscriptores. Es necesario declarar las propiedades del modelo como ***observables***, que son objetos especiales JavaScript que pueden notificar a los suscriptores sobre sus cambios y detectar dependencias automáticamente.

Angular (Ionic), al trabajar con objetos planos, y no con observables, chequea los cambios en las variables de un modo diferente. Cada vez que Angular evalúa una expresión, se comparan los valores actuales de los objetos con los valores anteriores.

Oracle MAF realiza este proceso de una forma bastante diferente a la de JET e Ionic. Los enlaces que soportan los componentes en una página de la interfaz de usuario se describen en un archivo XML específico para la página, llamado **archivo de definición de página.** La capa de modelo utiliza este archivo en tiempo de ejecución para instanciar los enlaces de la página. Estos enlaces se mantienen mapeados en lo que se llama *binding container*, que es accesible durante cada solicitud de página mediante la expresión *#{bindings}*. Esta expresión siempre evalúa el *binding container* para la página actual. Cuando se utilizan los *Data Controls* para crear un componente en la interfaz de usuario, los IDEs JDeveloper y Eclipse (integrado con *Oracle Enterprise Pack*) crean automáticamente el código y los objetos necesarios para crear el enlace entre el componente de la interfaz de usuario y el *Data Control*.

5.3.2 El sistema de navegación

Una de las principales características a tener en cuenta a la hora de desarrollar una aplicación es la manera de gestionar la navegación entre páginas y módulos. Cada uno de los tres frameworks tiene su propio sistema de navegación.

Sistema de navegación en Oracle JET

El sistema de navegación de Oracle JET proporciona soporte para el historial de navegadores HTML5 y la gestión de estados. Para ello utiliza la clase **Router,** que está diseñada para simplificar la navegación en aplicaciones de página única (SPA, por sus siglas en inglés). Un router siempre se encuentra en uno de los posibles estados, y cuando se realiza una acción en la interfaz de usuario se ejecuta una transición entre estados.

Dado que el Router realiza transiciones entre los estados de la aplicación, lo primero que debemos hacer es definir todos los estados posibles en los que la aplicación puede encontrarse. La clase Router nos proporciona la función *configure* para definir estos estados. El router se puede configurar de dos formas:

- Describiendo todos los posibles estados que pueden ser usados por el router.
- Proporcionando una función callback que devuelva un objeto de tipo RouterState (clase que representa un estado del router).

Una vez que hemos configurado el router, se debe utilizar la función *sync()* para sincronizar la URL con el estado del router. Si se ha definido un estado por defecto, el router

realizará la transición hacia ese estado, si no se ha definido, no se realizará ninguna transición y el router quedará en un estado indefinido.

Entre las funciones y campos que podemos encontrar en la clase Router destacan las siguientes:

- **getState.** Devuelve el objeto de tipo RouterState que coincide con el ID que se pasa como parámetro.
- **go.** Se usa para hacer una transición a un estado nuevo. Requiere como parámetro el ID del estado al que se quiere realizar la transición.
- **currentState.** Es un observable de un objeto del tipo RouterState correspondiente al estado en el que se encuentra el router actualmente.
- **moduleConfig.** Es un objeto que simplifica la integración entre el router y el *ojModule*. Permite crear un enlace en Knockout que gestiona el reemplazo de contenido de una región particular de la página. Cuando el router cambia de estado, *ojModule* cargará y renderizará automáticamente el contenido del nuevo módulo en la región de la página.

Sistema de navegación en Ionic 2

En la primera versión de Ionic el concepto de navegación era el mismo que en Oracle JET, es decir, estaba basado en el uso de estados y de un router. Sin embargo, a partir de la versión 2 de Ionic, se introduce un nuevo concepto, la **pila de navegación.** Cada página que se encuentra en la pila de navegación es una página que el usuario ha visitado previamente. Se puede pensar en la pila de navegación como en el historial de un navegador: cada vez que se visita una página, esta se añade al historial. Cuando se pulsa el botón atrás, se vuelve a la página anterior del historial.

La pila de navegación de Ionic se controla con la clase NavController. Para añadir ítems a la pila se usa el método *push*, y para eliminarlos el método *pop*. Por lo tanto, para ir de una página a otra utilizaremos *push*, y usaremos *pop* cuando queramos volver a la página anterior. Además de estos dos métodos, la clase NavController nos da la posibilidad de establecer una página como raíz con el método *setRoot*. Utilizando este método vaciamos la pila de navegación e insertamos en ella una nueva página, de modo que ya no podremos volver a las páginas anteriores. La Figura 20 muestra el concepto de pila de navegación.

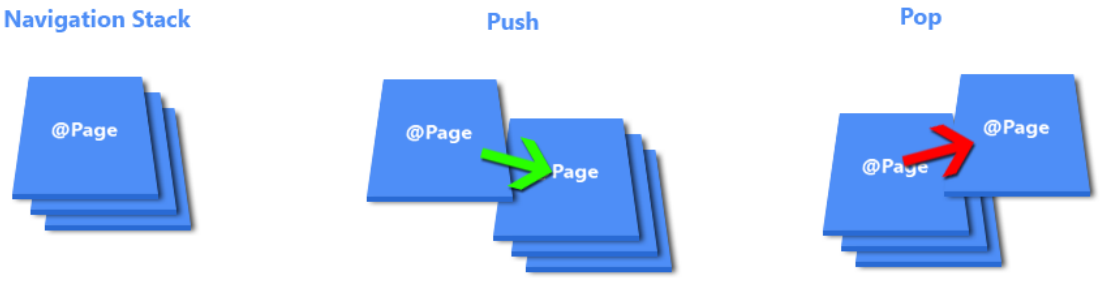


Figura 20: Pila de navegación de Ionic 2

Además de la clase NavController, Ionic proporciona dos directivas que pueden usarse directamente en la vista para realizar la misma funcionalidad que los métodos *push* y *pop* de la clase NavController:

- *navPush*. Directiva para añadir una nueva página en la pila de navegación. Junto a esta directiva se puede usar navParams para el envío de parámetros a la nueva página.
- *navPop*. Directiva para eliminar la última página de la pila de navegación.

Sistema de navegación en Oracle MAF

Una aplicación MAF está compuesta de una o varias *features*. Una *feature* representa un módulo dentro de la aplicación, y puede contener una o varias páginas.

El *Data Control* ***ApplicationFeatures*** proporcionado por MAF nos ofrece una serie de métodos que facilitan la navegación de la aplicación. Los métodos proporcionados por este *Data Control* para la navegación son los siguientes:

- **gotoDefaultFeature.** Navega a la *feature* por defecto de la aplicación.
- **gotoFeature.** Navega a una *feature* específica designada por el parámetro que se pasa al método.
- **gotoPreferences.** Navega a la página de preferencias.
- **gotoSpringboard.** Navega al springboard.
- **hideNavigationbar.** Oculta la barra de navegación.
- **showNavigationbar.** Muestra la barra de navegación
- **resetFeature.** Resetea la *feature* de la aplicación designada por el parámetro que se pasa al método.

- **hideSpringboard.** Oculta el springboard.
- **showSpringboard.** Muestra el springboard.

Además de la navegación mediante los métodos proporcionados por el *ApplicationFeatures,* MAF proporciona un sistema basado en ***task flows.*** Un *task flow* es una representación visual del flujo de la aplicación. Un *task flow* puede estar comprendido por las páginas de la interfaz de usuario y actividades no visuales, como llamadas a métodos. Los elementos no visuales pueden ser utilizados para evaluar expresiones EL (Expression Language) o llamar a otro *task flow*. MAF proporciona dos tipos de *task flow*: *bounded task flow*, que tiene un solo punto de entrada, y *unbounded task flow*, que puede tener múltiples puntos de entrada. Un *task flow* puede contener los siguientes elementos:

- **Method Call.** Permite hacer una llamada a un método que invoca la lógica de la aplicación desde cualquier lugar dentro del flujo de control de la aplicación.
- **Router.** Se utiliza para controlar la ruta a otras actividades basándose en una expresión EL. Por ejemplo, se puede seguir una u otra ruta del *task flow* dependiendo del valor de una variable.
- **Task Flow Call.** Permite hacer llamadas a otros *task flows.*
- **Task Flow Return.** Identifica cuando un *task flow* se ha completado y devuelve el flujo de control al elemento que hizo la llamada.
- **View.** Muestra una página o un fragmento de página. Múltiples elementos *View* pueden representar la misma página o fragmento de página.
- **Control Flow Case.** Identifica cómo controlar el paso de una actividad a otra en la aplicación.
- **Wildcard Control Flow rule.** Representa un caso de flujo de control que puede originarse desde cualquier actividad cuyo ID coincida con la expresión indicada en el *Wildcard*.

Estos elementos se muestran en la Figura 21.

La Figura 22 muestra un ejemplo de *task flow* para la aplicación Try&Win. Este task flow se compone de los Method Call *loadUserFromDatabase* y *resetFeature*, el Router *tieneUsuarioEnBBDD* y la View *welcome*.

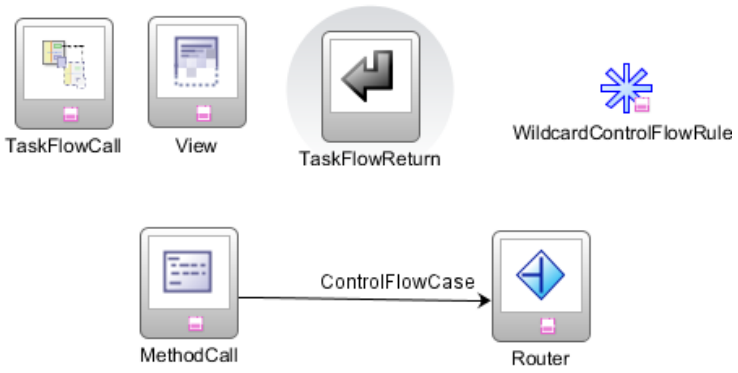


Figura 21: Elementos de un task flow

El círculo verde que rodea a *loadUserFromDatabase* indica que es la actividad por defecto que se ejecutará al entrar en el *task flow*. El flujo es el siguiente:

1. Al entrar al *task flow* se ejecuta el Method Call *loadUserFromDatabase.* Este *Method Call* invoca a un método que comprueba si existe un usuario en la base de datos SQLite de la aplicación y almacena el ID de usuario en una variable.
2. Tras la ejecución del *Method Call* el *Router* comprueba el valor de la variable donde se almacena el ID de usuario.
3. Si el valor de la variable es 0, es decir, no había un usuario en base de datos, quiere decir que el usuario aún no se ha registrado, y por lo tanto el flujo pasa a la *View welcome*. En caso de que si hubiese un usuario en base de datos, quiere decir que el usuario ya se había registrado anteriormente en la aplicación y el flujo pasa al *Method Call resetFeature*. Esta última actividad hará que el flujo de control pase a una nueva *feature*.

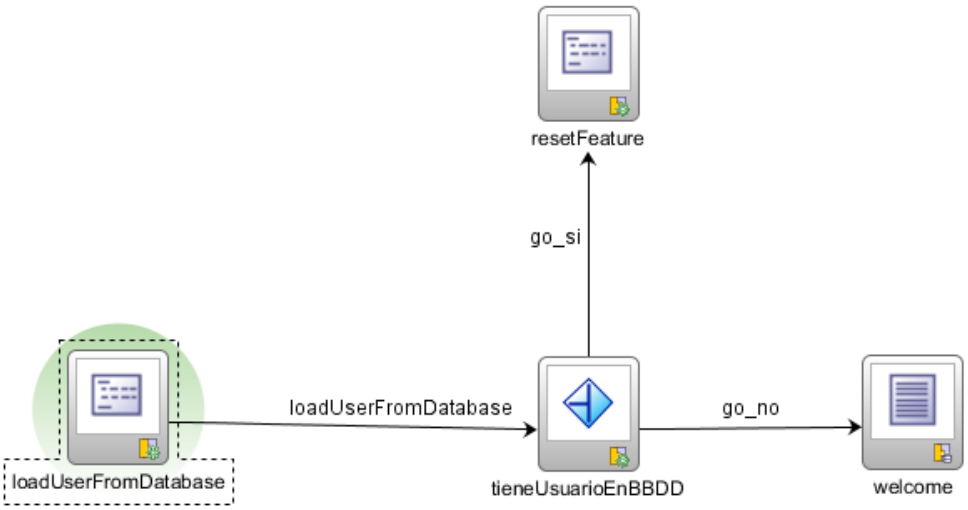


Figura 22: Ejemplo de task flow de la aplicación Try&Win

La navegación en la aplicación #smact Try&Win

En este apartado se exponen los requisitos de navegación que exigía la aplicación Try&Win para su correcto funcionamiento y como se llevó a cabo con cada uno de los tres frameworks. El flujo de la aplicación es el siguiente:

1. La primera pantalla es la de bienvenida, donde el usuario puede elegir un avatar que le represente.
2. La siguiente pantalla permite al usuario elegir uno de los premios que se sortearán. El usuario puede volver hacia atrás a la pantalla de bienvenida.
3. Tras la pantalla anterior, el usuario queda registrado y pasa a una pantalla que muestra sus logros (retos conseguidos, si ha entrado en el sorteo, el premio que tiene elegido, etc.). Desde esta pantalla puede acceder a un menú lateral para navegar al resto de módulos de la aplicación: *Mis logros, cambiar premio, Retos, Notificaciones, Twitter, Equipo avanttic* y *Acerca de*. Una vez llegados a este punto, el usuario no puede dar marcha atrás, por lo que ya no se puede acceder a las pantallas anteriores mediante el botón *atrás*. Sin embargo, sí que puede acceder a la pantalla de selección de premio desde el menú.
4. Las pantallas *Retos, Notificaciones* y *Equipo avanttic* muestran una lista de retos, notificaciones y personas, respectivamente. Haciendo click en un elemento de la lista podemos pasar a una pantalla que muestra el detalle de dicho elemento.

La Figura 23 muestra la navegación en la aplicación *#smact Try&Win*.

A la hora de desarrollar el sistema de navegación de la aplicación con Oracle JET nos encontramos un problema, y es que el sistema de navegación que proporciona almacena en el historial todas las pantallas por las que el usuario ha pasado. Para solucionarlo, fue necesario añadir un *event listener* que capturarse el evento del botón *atrás* en las pantallas en las que el usuario no debería tener la opción de volver a la pantalla anterior, y eliminar el *event listener en el resto* de pantallas. De este modo, fue necesario capturar el evento en las pantallas *Mis logros, Retos, Notificaciones, Twitter*, *Equipo avanttic* y *Acerca de*.

Con Oracle MAF esto no fue necesario, ya que cuando navegas de una *feature* a otra, por defecto no puedes volver a la *feature* anterior. Sin embargo, el usuario si debe tener la posibilidad de volver de la *feature* de selección de premio a la *feature* de bienvenida. Para implementar este comportamiento se ha utilizado un componente llamado *systemActionBehavior* con su propiedad type con el valor *back*.

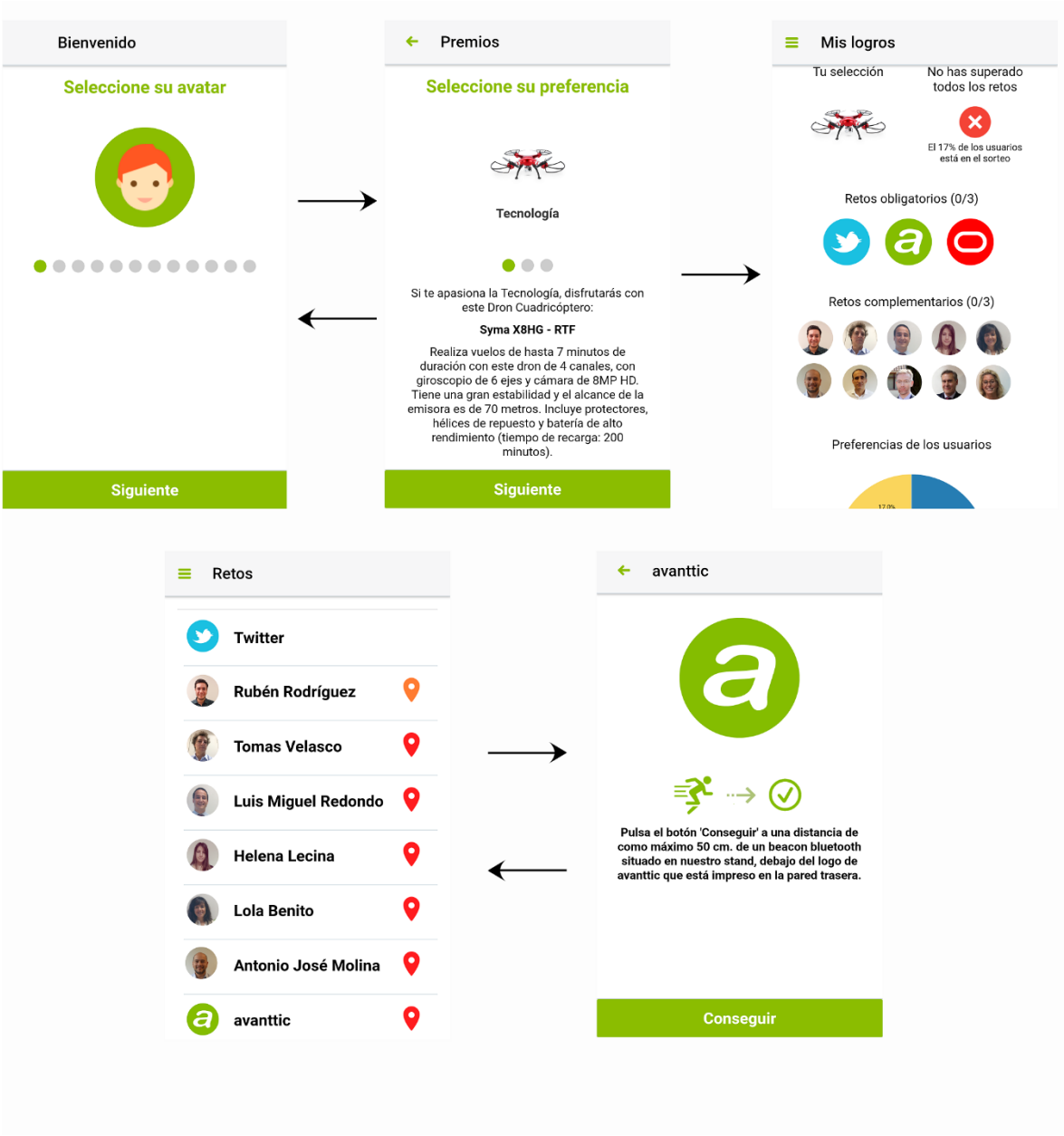


Figura 23: Navegación en la aplicación Try&Win

Con este componente podemos detectar cuando se pulsa el botón *atrás* en la pantalla de selección de premios e indicar que queremos volver a la *feature* de bienvenida. En el caso de las *features Retos*, *Notificaciones* y *Equipo avanttic*, que contienen una pantalla con una lista de elementos y se debe poder navegar hacia el detalle de un elemento y, posteriormente volver hacia atrás, lo que se ha hecho es utilizar un *task flow* que contiene dos pantallas (*View*), permitiendo directamente la navegación hacia atrás desde el detalle de un elemento a la lista completa. La Figura 24 muestra el *task flow* de la *feature* de retos.

Figura 24: Task Flow de la feature Retos

El caso de Ionic fue el más sencillo de desarrollar, ya que el sistema de navegación de Ionic nos permite insertar pantallas en la pila de navegación de dos formas: añadir una pantalla a la pila (función *push*) y añadir una pantalla vaciando previamente la pila (función *setRoot*). De este modo, si queremos que el usuario no pueda volver hacia atrás cuando llegue a una determinada pantalla, simplemente debemos acceder a ella utilizando el método *setRoot*. En otro caso, utilizamos el método *push*. Por tanto, en este caso, se utiliza *setRoot* al acceder a la pantalla *Mis logros, Retos, Notificaciones, Twitter, Equipo avanttic* y *Acerca de,* y el método *push* para las pantallas *Premios, Detalle de Reto, Detalle de Notificación y Detalle de Equipo avanttic*.

5.3.3 Acceso a los servicios del dispositivo

Los tres frameworks de desarrollo permiten el acceso a los servicios del dispositivo, como la cámara, el GPS, SMS, los contactos, etc. En este caso, el acceso a estos servicios se realiza a través de la plataforma de Apache Cordova mediante el uso de plugins.

Cada uno de los frameworks incluye por defecto algunos plugins en el proyecto. Además, el acceso a la base de datos SQLite en Oracle MAF se realiza de manera diferente, ya que no hace uso de un plugin Cordova. El acceso a SQLite en MAF se codifica a través de JDBC (Java Database Connectivity), una API que permite ejecutar operaciones sobre bases de datos desde Java, independientemente del sistema operativo donde se ejecute.

La Tabla 2 muestra los accesos a las características del dispositivo que por defecto proporciona cada framework.

Plugin	Descripción	Oracle JET	Oracle MAF	Ionic 2
Whitelist	Implementa una política de lista blanca que define los dominios a los que la aplicación puede acceder	Si	Si	Si
Contacts	Proporciona acceso a la base de datos de los contactos	No	Si	No
SMS	Permite el envío de SMS	No	Si	No
Email	Proporciona acceso a la interfaz que gestiona el email	No	Si	No
Device	Define un objeto que describe el software y hardware del dispositivo	No	No	Si
Splashscreen	Muestra y oculta una pantalla de bienvenida al lanzar la aplicación	No	No	Si
Statusbar	Proporciona funciones para personalizar la barra de estado	No	No	Si
Keyboard	Proporciona funciones para personalizar y controlar el teclado	No	No	Si
Geolocation	Proporciona información sobre la localización del dispositivo, como la latitud y la longitud	No	Si	No
PushPlugin	Permite registrar y recibir notificaciones push	No	Si	No
Camera	Proporciona una API para hacer fotos y elegir imágenes de la librería de imágenes del sistema	No	Si	No

Tabla 2: Servicios del dispositivo por defecto en cada framework

Otra diferencia es la forma en la que cada framework hace uso de los plugins. Mientras que Oracle JET hace un uso directo de los plugins de cordova, Oracle MAF e Ionic 2 tienen sus propios *wrappers* para hacer uso de algunos de estos plugins de forma más sencilla. El *wrapper* de Ionic 2 se llama *Ionic Native*, y está pensado para facilitar el acceso a los plugins Cordova mediante los lenguajes ECMAScript y TypeScript. Proporciona una interfaz común para todos los plugins y se asegura de lanzar los eventos necesarios cuando se detecta algún cambio en Angular 2. Oracle MAF permite el acceso a los servicios de manera muy sencilla. En lugar de escribir múltiples líneas de código, MAF permite agregar a la aplicación los servicios del dispositivo mediante *Drag&Drop*.

5.3.4 Reconocimiento de gestos en pantalla

Actualmente es común encontrar aplicaciones que ofrecen la posibilidad al usuario de realizar ciertas funciones mediante el reconocimiento de gestos en la pantalla del dispositivo, como el doble click, la rotación, movimientos con varios dedos, etc. JET, MAF e Ionic ofrecen soporte para el reconocimiento de este tipo de gestos.

En el caso de Oracle JET, se hace uso de la librería **Hammer.js**, una librería creada para reconocer gestos por el tacto (aplicaciones móviles) y mediante el ratón (aplicaciones web). Oracle MAF utiliza **Oracle Alta UI**, un sistema para la creación de interfaces de usuario desarrollado por Oracle que incluye reconocimiento de gestos. Por último, Ionic ha creado su propia clase, llamada **Gesture**. Esta clase es un *wrapper* de la librería Hammer.js para TypeScript. Por lo tanto, Gesture tiene soporte para el reconocimiento de todos los gestos soportados por Hammer.js.

La Tabla 3 muestra una comparativa de los gestos soportados por cada una de estas librerías.

5.3.5 Seguridad

La seguridad es una de las principales prioridades para el desarrollo de aplicaciones móviles, ya que los dispositivos móviles tienen un mayor riesgo de pérdida o robo. En este apartado se exponen las características de seguridad proporcionados por cada framework.

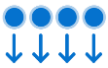
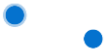
Nombre	Gesto	Hammer.js	Alta UI	Gesture
Tap		Si	Si	Si
3D Touch	-	No	Si	No
Swipe		Si	Si	Si
Two-Finger Drag		Si	Si	Si
Pinch Close		Si	Si	Si
Double Tap		Si	Si	Si
Rotate		Si	Si	Si
Multi-Finger Drag		Si	Si	Si
One-Finger Press, Other-Finger Tap		No	Si	No
Press and Release		No	Si	No
Rub Out		No	Si	No
Encircle		No	Si	No

Tabla 3: Gestos de pantalla soportados por cada librería

Oracle JET proporciona seguridad tanto a **nivel de componentes** como a **nivel de framework**. A nivel de componente, JET sigue las siguientes prácticas:

- Todo el código JavaScript se ejecuta en modo *strict*. El modo *strict* pasa las advertencias como la sintaxis pobre y las variables no declaradas a errores que se deben corregir.

- No utiliza elementos de script en línea.
- No genera números aleatorios.
- Los componentes de JET han sido desarrollados siguiendo una guía de buenas prácticas sobre el desarrollo de aplicaciones seguras en JavaScript.

A nivel de framework, Oracle JET proporciona un plugin llamado *oj.OAuth* que soporta el protocolo OAuth 2.0. OAuth proporciona un mecanismo para que los usuarios concedan acceso a datos privados sin compartir sus credenciales privadas como el nombre de usuario y la contraseña. La Figura 25 muestra el proceso de autenticación OAuth.

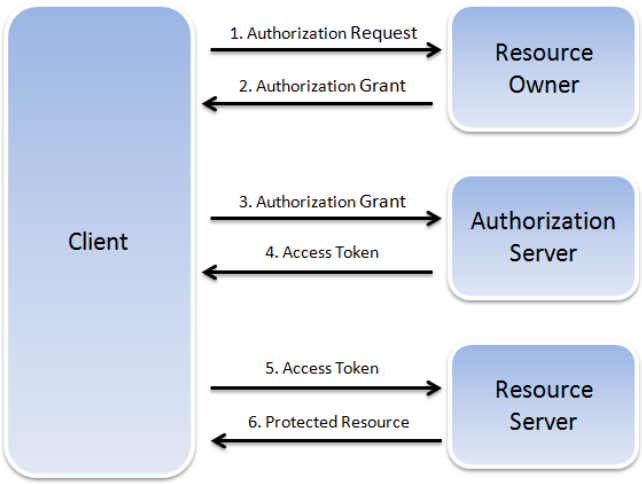


Figura 25: Proceso de autenticación OAuth

Oracle MAF viene con seguridad integrada que puede limitar el acceso a sus aplicaciones y garantizar el cifrado de datos confidenciales. Proporciona métodos de autenticación y control de acceso a **nivel de *feature*** en la aplicación. Las aplicaciones MAF pueden autenticarse contra cualquier servidor que proporcione autenticación básica sobre HTTP o HTTPS. También soporta autenticación contra Oracle Identity Management[53]. En tiempo de ejecución, se muestra a los usuarios una pantalla de login, como la que muestra la Figura 26 y los tokens apropiados son accesibles para otras llamadas a servicios. Los desarrolladores pueden crear interfaces de usuario que satisfagan las necesidades de usuarios

[53] Oracle Identity Management: *http://www.oracle.com/technetwork/middleware/id-mgmt/overview/index.html*

con diferentes privilegios, permitiendo que la misma aplicación sirva para múltiples roles en una organización.

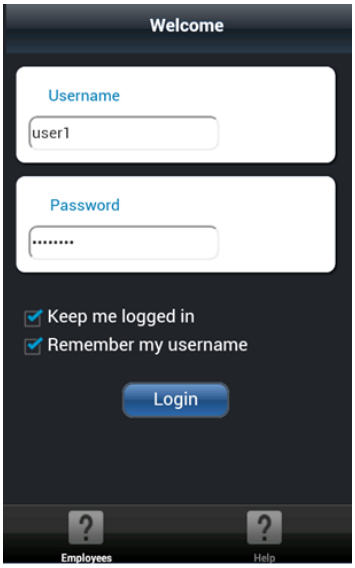


Figura 26: Ejemplo de pantalla de login en MAF

Oracle MAF aplica el cifrado en las siguientes áreas:

- **Cifrado de comunicaciones:** mediante SSL y TLS (HTTPS).
- **Cifrado en el dispositivo:** Las credenciales se pueden guardar en un almacén de claves cifrado y se utilizan para la validación cuando se admite la autenticación offline.
- **Cifrado de base de datos SQLite:** MAF incluye la extensión de cifrado de SQLite.

MAF está certificado con varias soluciones empresariales de gestión móvil para proporcionar más funciones de seguridad, como el *tunneling*, políticas de seguridad y control de acceso, y autenticación única.

En el caso de Ionic, es posible implementar cualquier método de autenticación Web en nuestra aplicación. Sin embargo, el framework no proporciona ninguna API, clase o librería que facilite esta tarea, por lo que todo el trabajo de securizar al aplicación recae en el desarrollador. Aun así, existen librerías externas y plugins de Cordova para facilitar esta tarea, como *ng2-cordova-oauth*[54], un plugin Cordova que se integra fácilmente con aplicaciones basadas en Angular 2 e Ionic 2 y que facilita la obtención del acceso basado en tokens.

[54] Angular 2 Cordova Oauth: *https://github.com/nraboy/ng2-cordova-oauth*

La Tabla 4 muestra los protocolos de seguridad soportados por cada uno de los frameworks.

Protocolo	Oracle JET	Oracle MAF	Ionic
HTTP Basic	No	Si	No
Mobile-Social	No	Si	No
OAuth	Si	Si	No
Web SSO	No	Si	No

Tabla 4: Protocolos de autenticación soportados por cada framework

5.3.6 Rendimiento

En este apartado se exponen una serie de resultados, tablas y gráficas que muestran distintas mediciones realizadas para **comparar el rendimiento** de las aplicaciones en determinados aspectos, como el tiempo requerido para aplicar bindings en una pantalla, el acceso a los datos de SQLite, la transición entre vistas, etc.

Los resultados obtenidos se han medido ejecutando la aplicación desarrollada por cada framework en un dispositivo Android. En concreto, se trata de un BQ Aquaris M5.5 con las siguientes características principales:

- Versión de Android 6.0.1.
- Memoria interna de 32 GB.
- 3 GB de RAM.
- CPU Qualcomm Snapdragon 615 Octa Core A53 hasta 1,5 GHz.
- GPU Qualcomm Adreno 405 hasta 550 MHz.

Aplicación de bindings

En primer lugar se quiso comprobar cuanto tiempo tardaba cada framework en **realizar el *binding*** (enlazado de datos con la interfaz de usuario) y mostrar los datos en pantalla. Para obtener esta medida, se necesitaba realizar la medición sobre una vista que tuviese un número significativo de elementos. Por este motivo, se creó un módulo que generase un *array* de 1000 elementos y los mostrase en pantalla. Oracle JET e Ionic proporcionan

directivas que permiten ejecutar código cuando ocurre un determinado evento, como la entrada a una nueva pantalla, su salida o cuando los *bindings* han sido aplicados. En este caso, nos interesa este último evento. El proceso para realizar la medición de tiempo en JET e Ionic es el siguiente:

1. Generamos y almacenamos un *array* de 1000 elementos.
2. Guardamos la fecha actual (que será la de inicio) en una variable.
3. Asignamos el *array* de elementos a la variable que tiene el *binding* con la vista.
4. En el callback que se ejecuta cuando JET e Ionic terminan de aplicar los *bindings* se calcula una nueva fecha y restamos la de inicio para obtener el resultado.

La Figura 27 muestra el pseudocódigo para hacer el cálculo en JET e Ionic.

```
función init
    data ← generar lista de 1000 elementos
    startDate ← fecha actual
    dataBind ← data

función bindingsApplied
    time ← fecha actual menos startDate
```

Figura 27: Cálculo del tiempo en aplicar *bindings* en JET e Ionic

El proceso para hacer el cálculo con MAF es algo más complicado. Debido a que MAF no proporciona ningún método para ejecutar un callback cuando la página ha sido cargada completamente, fue necesario crear un archivo JavaScript asociado a la *feature*. Este archivo se ejecutará automáticamente al entrar en la *feature* en la que se muestra la lista de elementos. Para saber cuándo la página se ha cargado completamente, se añadió un *eventListener* al archivo JavaScript. El proceso de ejecución es el siguiente:

1. Al entrar en la feature se genera la lista de mil elementos.
2. Calculamos el tiempo actual.
3. Se asigna el *array* de elementos a la variable que tiene el *binding* con la vista.
4. Cuando la página se ha cargado completamente, se invoca a una función de Java desde el archivo JavaScript, que calcula el tiempo actual y le resta el tiempo de inicio.

La Figura 28 muestra el resultado obtenido en cada uno de los frameworks para una lista de 1000 elementos. Se han realizado diez mediciones con cada framework.

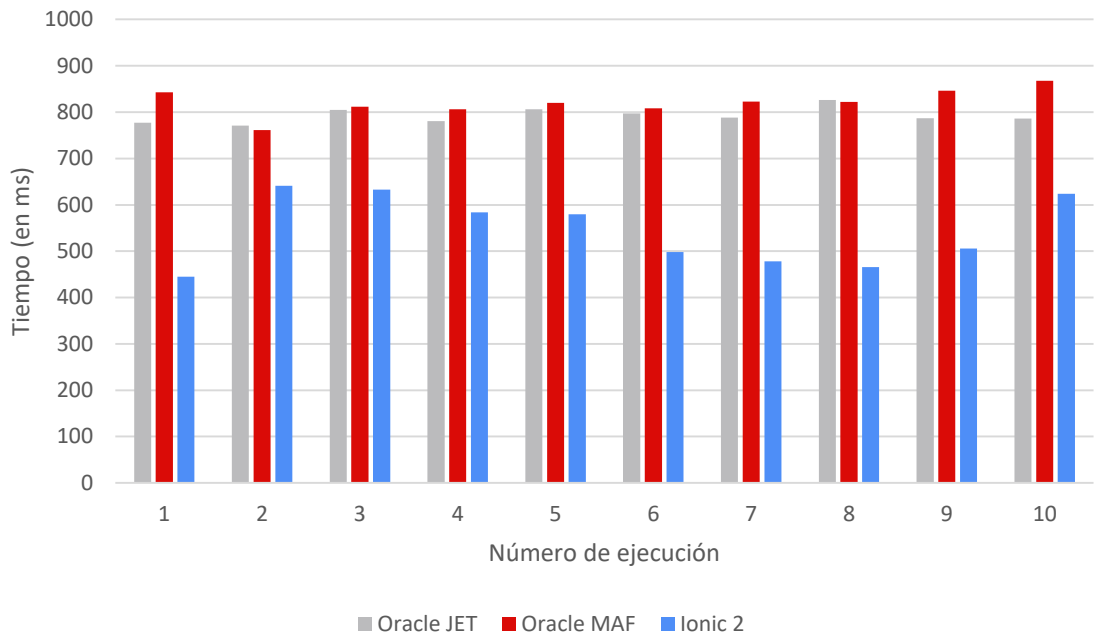


Figura 28: Tiempo en mostrar una lista de 1000 elementos

Posteriormente, para ver cómo se comporta cada framework con un número más elevado de elementos, se realizó el mismo proceso con un *array* de 10 mil elementos. Los datos se muestran en la Figura 29.

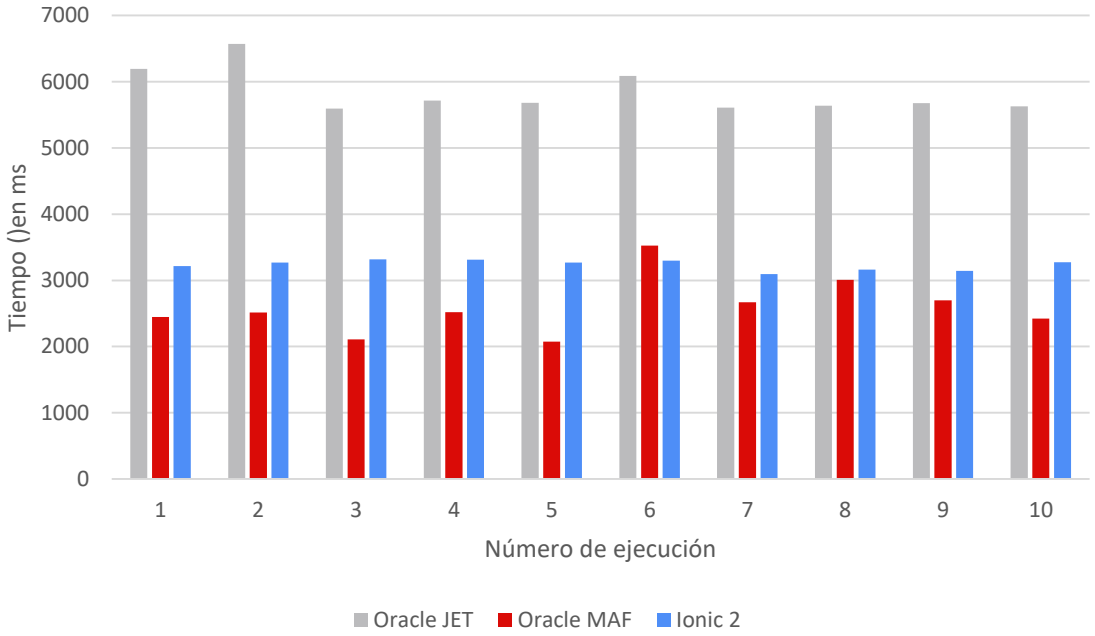


Figura 29: Tiempo en mostrar una lista de 10000 elementos

La Tabla 5 muestra la media del tiempo requerido por cada framework para cada uno de los casos.

	Tiempo (en ms)		
Nº de elementos	**Oracle JET**	**Oracle MAF**	**Ionic 2**
Mil elementos	792,4	820,9	545,5
10 Mil elementos	5839,3	2597,1	3235

Tabla 5: Media de tiempo en mostrar una lista

Como podemos observar, cuando se trata de una lista de mil elementos, Ionic es el más rápido, ocupando MAF el último puesto. Sin embargo, cuando el tamaño de la lista crece hasta 10 mil elementos, MAF sufre una mejora considerable con respecto a los demás, siendo el más rápido de los tres. Esto se debe a la forma que tiene MAF de gestionar las listas, que permite establecer el modo en el que se retienen las filas, mostrando solo en cada momento un número determinado de filas dependiendo del tamaño de la pantalla del dispositivo. Esto reduce el tamaño de memoria usada por la aplicación.

Acceso a SQLite

Otra comparativa que resulta de interés es el tiempo requerido para conectarse a una base de datos SQLite, ejecutar una consulta y obtener el resultado. El proceso para realizar medir el tiempo requerido es el siguiente:

1. Obtenemos y almacenamos el tiempo de inicio.
2. Obtenemos la conexión a la base de datos.
3. Ejecutamos la consulta y guardamos el resultado.
4. Calculamos el tiempo final y le restamos el de inicio.

La Figura 30 muestra los tiempos obtenidos por cada framework en diez ejecuciones distintas.

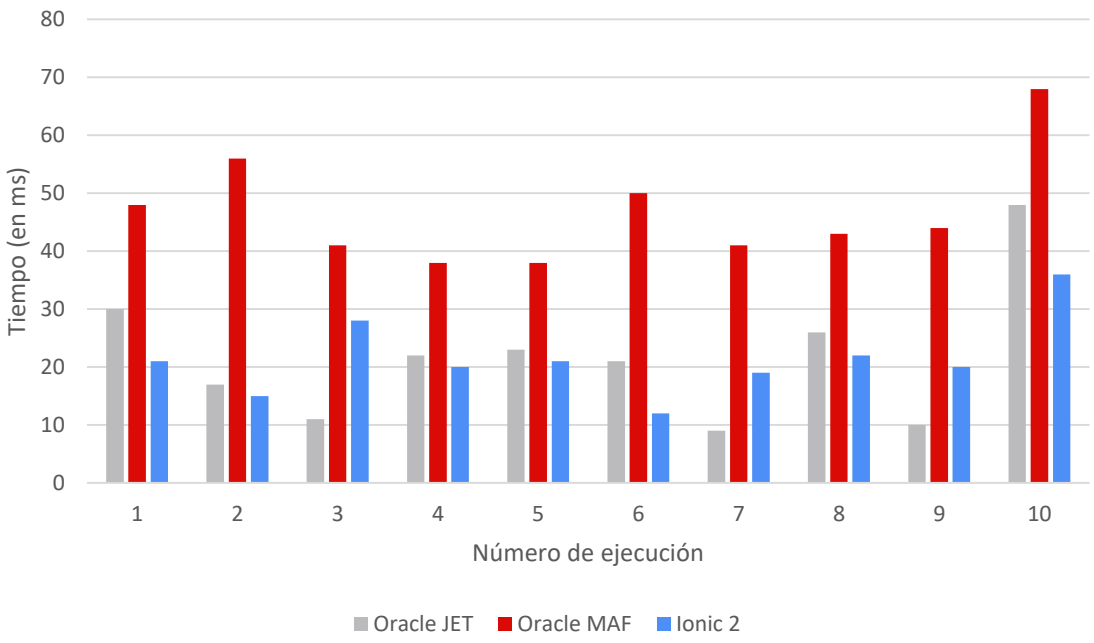


Figura 30: Tiempo de acceso a la base de datos SQLite

El tiempo medio requerido por cada framework es de:

- Oracle JET: 21,7 ms.
- Oracle MAF: 46,7 ms.
- Ionic 2: 21,4 ms.

Se puede observar que Ionic y Oracle JET tardan básicamente el mismo tiempo, siendo más rápido Ionic por tan solo 0,3 ms. En cambio, Oracle MAF tiene una media de tiempo de más del doble en comparación con Ionic y JET. Esto puede deberse a la diferencia que existe en el modo de acceder a la base de datos SQLite en JET e Ionic con respecto a MAF. Mientras que en JET e Ionic se hace uso de un plugin Cordova para el acceso a la base de datos, MAF utiliza JDBC (Java Database Connectivity), una API para la ejecución de operaciones en bases de datos desde el lenguaje Java. La primera vez que se hace una consulta, se realiza la conexión con la base de datos y se cargan los drivers requeridos por JDBC, lo que no es necesario para posteriores consultas a la base de datos. Debido a esto, resulta interesante medir cuánto tarda cada framework en realizar una consulta cuando la conexión a la base de datos ya ha sido establecida previamente. La Figura 31 muestra esta comparativa.

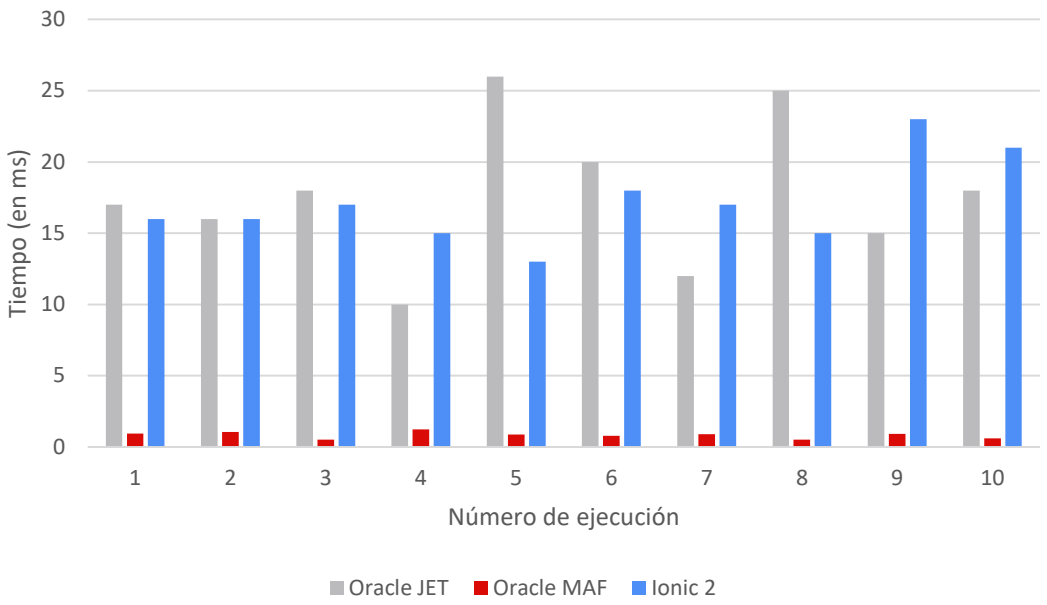


Figura 31: Tiempo de acceso a la base de datos SQLite tras hacer la conexión

EL tiempo medio requerido por cada framework una vez hecha la conexión es:

- Oracle JET: 17,7 ms.
- Oracle MAF: 0,8274 ms.
- Ionic 2: 17,1 ms.

Podemos observar que mientras JET e Ionic no tienen una reducción significativa en el tiempo de acceso, la media de tiempo en MAF se ve reducido enormemente, no llegando a superar el milisegundo. Esto es debido a que a que solo requiere cargar los drivers de JDBC una sola vez.

Transición entre vistas

A continuación se realizó una medición del tiempo requerido por cada framework en realizar una transición entre una vista y otra. De este modo, podemos hacernos una idea de que framework proporciona una **navegación más fluida** entre las pantallas. La transición que se va a medir es la realizada entre la pantalla *Mis logros*, que muestra los retos conseguidos por el usuario, entre otras cosas, y la pantalla *Acerca de*, que muestra información sobre la aplicación. Se ha elegido realizar la transición a la pantalla *Acerca de* porque es una pantalla estática y no necesita obtener datos de un servicio externo, de modo que esto no influya en los resultados.

Como ya se ha comentado anteriormente, JET e Ionic nos proporcionan *callbacks* que permiten conocer cuándo se realiza una transición entre una vista y otra. En este caso, nos interesa ejecutar un *callback* que nos permite conocer cuando la vista a la que navegamos ha sido insertada. El proceso que se ha seguido para realizar la medición en estos dos frameworks es el siguiente:

1. Obtenemos la fecha de inicio.
2. Ejecutamos el comando que realiza la transición a la siguiente vista.
3. En el código en el que se ejecuta el *callback* obtenemos la fecha y restamos la inicial.

En el caso de MAF, que no proporciona este tipo de *callback*, el proceso es diferente. MAF permite crear un *binding* a un método e invocarlo automáticamente cuando accedemos a una vista. El proceso es el siguiente:

1. Obtenemos el tiempo inicial.
2. Ejecutamos la instrucción que realiza la transición a la siguiente vista.
3. Con el *binding* creado, automáticamente se ejecuta una función que obtiene el tiempo actual y realiza la resta con el tiempo inicial.

La Figura 32 muestra el resultado obtenido en cada uno de los frameworks. Al igual que en los casos anteriores, se han realizado diez ejecuciones.

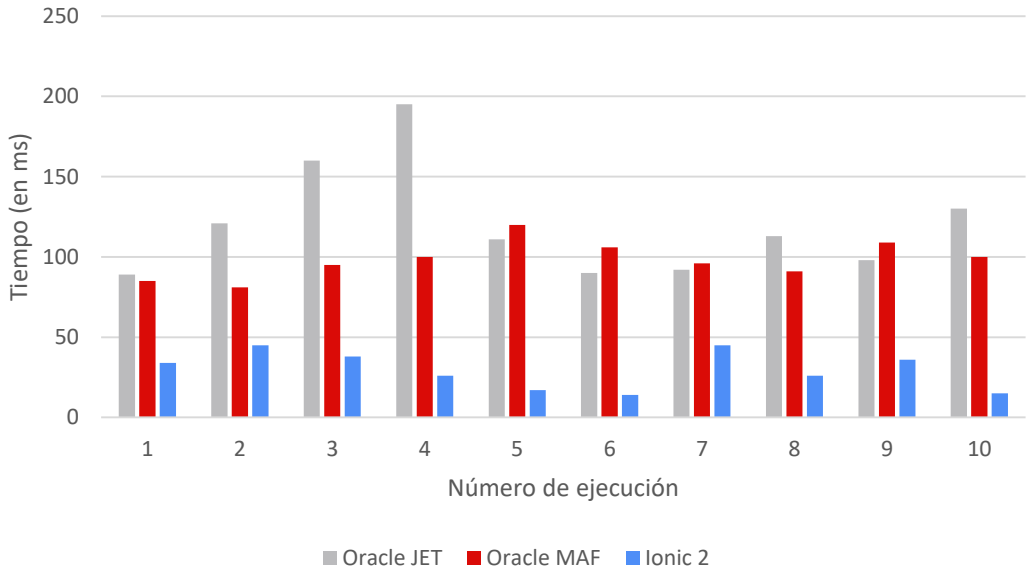


Figura 32: Tiempo en realizar una transición de pantalla

El tiempo medio requerido por cada framework es de:

- Oracle JET: 119,9 ms.
- Oracle MAF: 98,3 ms.
- Ionic 2: 29,6 ms.

Se puede ver claramente que Ionic es mucho más rápido en este aspecto que los otros frameworks. En concreto, es 3,32 veces más rápido que Oracle MAF y 4,05 más rápido que Oracle JET. Esto le da al usuario una sensación de mayor fluidez a la hora de ejecutar la aplicación realizada con Ionic.

Pintado y renderizado

Otro aspecto que se ha tenido en cuenta para determinar el rendimiento de la aplicación es el tiempo que se necesita para **renderizar** y **pintar** una vista. Para medir estas características se ha utilizado la herramienta *Timeline Tool*. Esta herramienta forma parte del kit de herramientas para desarrolladores que proporciona Google Chrome, llamado *DevTools*. Este kit de herramientas permite depurar aplicaciones desde el navegador, de modo que podemos conectar un dispositivo Android al ordenador y depurar nuestra aplicación. La principal característica de *Timeline Tool* es que permite grabar las acciones que se llevan a cabo en una aplicación y posteriormente mostrar una serie de resultados, como el tiempo de scripting, renderizado y pintado, entre otros. En este apartado nos centraremos en el tiempo requerido para renderizar y pintar la pantalla de la aplicación que muestra la lista de retos.

A continuación se define qué es lo que realmente mide la herramienta *Timeline Tool* cuando hablamos de renderizado y pintado de pantalla:

- **Renderizado:** se refiere al tiempo asociado con los cálculos de los estilos aplicados en cada nodo del DOM (Document Object Model) y de la posición de los elementos en la pantalla.
- **Pintado:** se refiere al tiempo relacionado con pintar los píxeles en la pantalla, e incluye eventos de decodificación y redimensionado de imágenes.

Cuando una aplicación emplea mucho tiempo en el renderizado se debe a la estructura del DOM y de los ficheros CSS. Cuando una aplicación emplea mucho tiempo en el pintado de la pantalla se debe a la apariencia de la vista, por ejemplo, estilos que sean muy costosos de pintar o imágenes que sean demasiado grandes.

Se han realizado las pruebas diez veces con cada framework. La Figura 33 y la Figura 34 muestran los resultados del renderizado y el pintado de pantalla, respectivamente

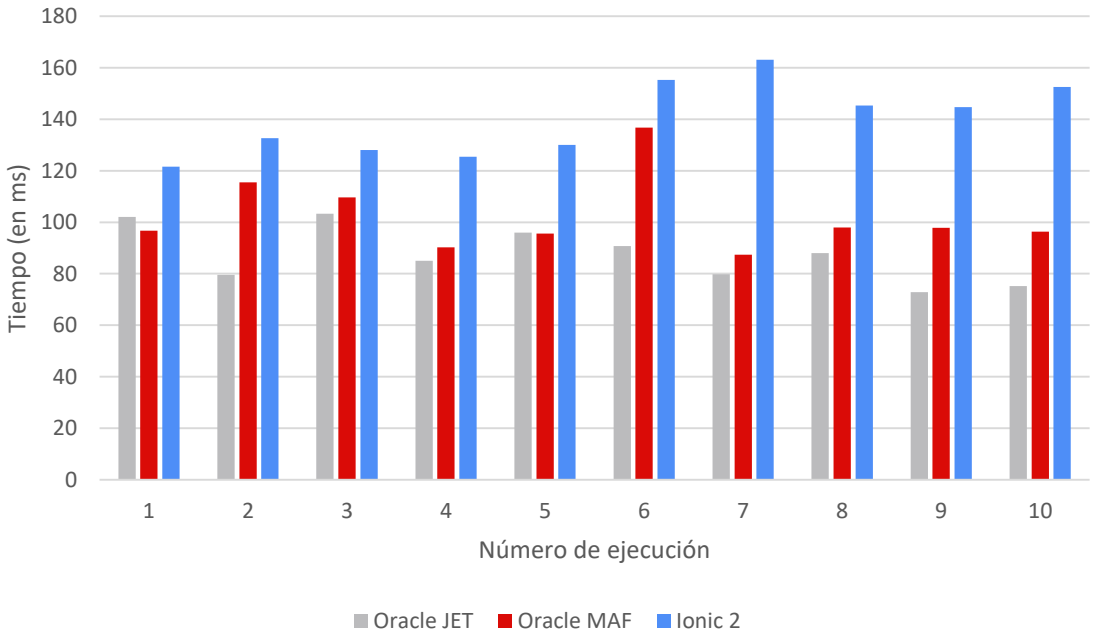


Figura 33: Tiempo de renderizado de la pantalla de retos

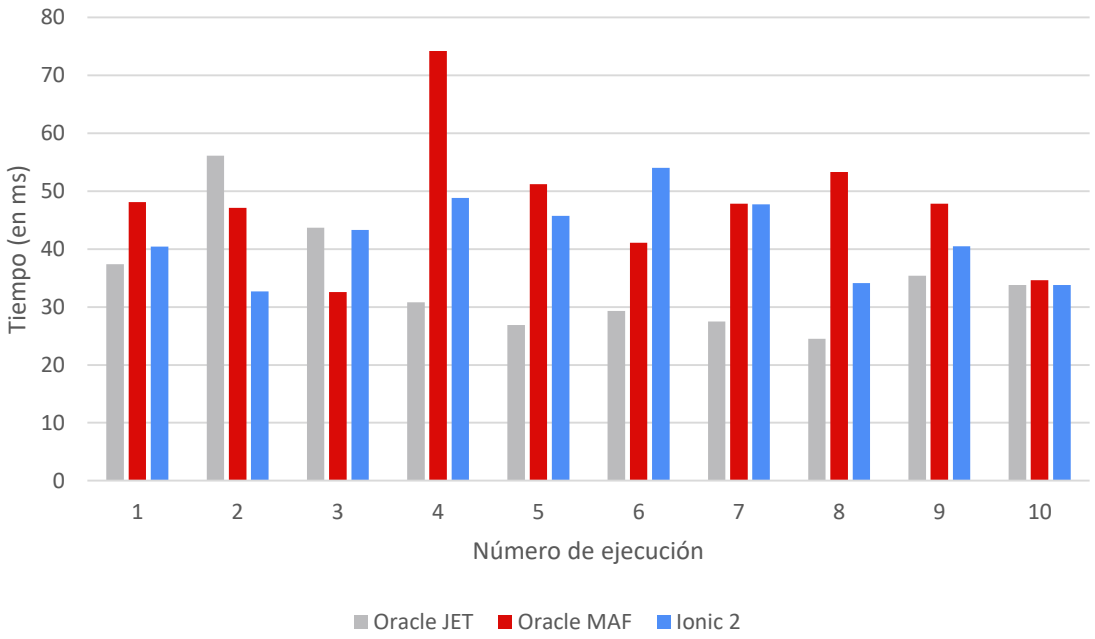


Figura 34: Tiempo de pintado de la pantalla de retos

La Tabla 6 muestra la media de tiempo requerido por cada framework para el renderizado y el pintado de la pantalla *Retos*.

	Tiempo (en ms)		
Característica	**Oracle JET**	**Oracle MAF**	**Ionic 2**
Renderizado	87,23	102,41	139,87
Pintado	34,54	47,78	42,1

Tabla 6: Tiempo medio de renderizado y pintado de la pantalla de retos

Como puede observarse, el framework que obtiene unos mejores resultados es JET en ambos casos. La mayor diferencia podemos verla en los resultados de renderizado, donde se observa claramente que el tiempo de la aplicación desarrollada con Ionic es mucho más elevado.

Uso de los recursos del dispositivo

En este apartado se muestra como las aplicaciones desarrolladas por cada framework hace uso de los **recursos del dispositivo móvil.** Para analizar estos resultados se ha usado *Trepn Profiler*[55]. Se trata de una aplicación que permite gestionar el rendimiento y potencia del dispositivo móvil en Android. Entre sus características, *Trepn Profiler* permite la monitorización de la CPU y GPU, y la vista en tiempo real de los núcleos de la CPU individualmente. Además, permite obtener datos de cada aplicación individualmente. Estos últimos datos son los que realmente nos interesan, ya que queremos ver los resultados de cada aplicación por separado. Para cada una de las aplicaciones se han obtenido los siguientes datos:

- **Uso de la CPU.**
- **Tamaño de la memoria virtual.** Memoria virtual que el proceso está utilizando.
- **Resident Set Size (RSS).** Es la cantidad de memoria física (RAM) utilizada por el proceso.

En cuanto al uso la CPU, se han obtenido datos por cada aplicación durante un intervalo de 10 segundos. La Figura 35 muestra el resultado.

[55] Aplicación Trepn Profile: *https://play.google.com/store/apps/details?id=com.quicinc.trepn&hl=es*

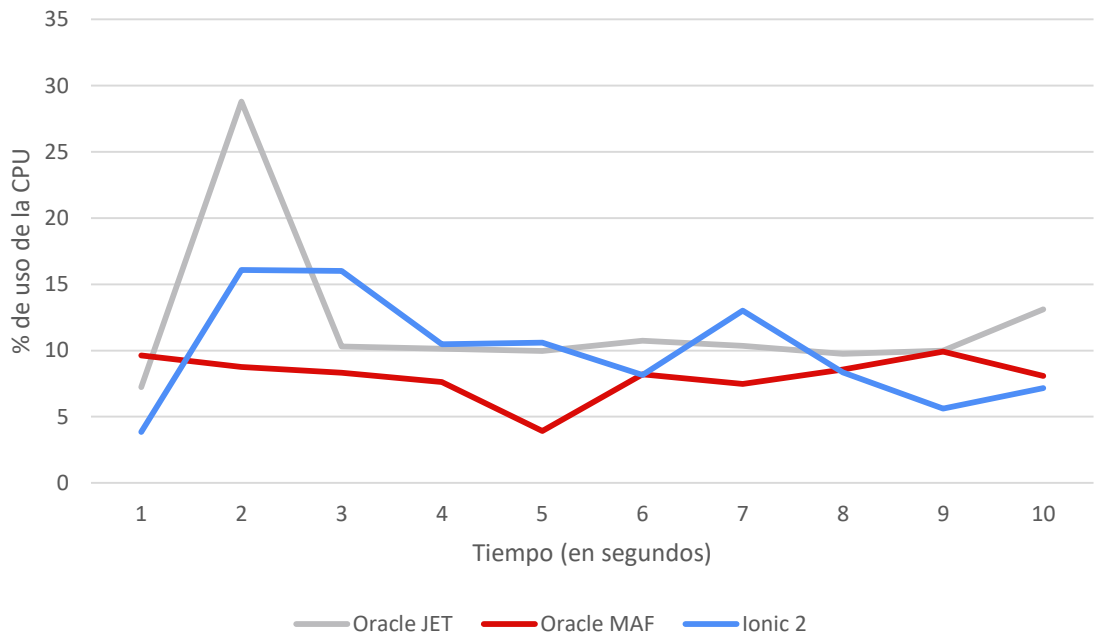


Figura 35: Uso de la CPU por cada aplicación

La media del uso de la CPU en cada framework es:

- Oracle JET: 12,034 %.
- Oracle MAF: 8,048 %.
- Ionic 2: 9,928 %.

Los datos obtenidos sobre la memoria virtual utilizada y el *resident set size* se muestran en la Tabla 7.

	Tamaño (en MB)		
	Oracle JET	**Oracle MAF**	**Ionic 2**
Media Memoria Virtual	944,926	1119,992	987,618
Máximo Memoria Virtual	980,576	1145,472	1010,356
Media RSS	28,98	39,826	34,493
Máximo RSS	31,464	41,267	35,764

Tabla 7: Uso de la memoria virtual y RSS

Se puede observar que la aplicación desarrollada con Oracle MAF es la que menos uso de CPU requiere, siendo la desarrollada con Ionic la que más uso de la CPU hace. Si nos fijamos en el uso de la memoria virtual y en el RSS, la aplicación desarrollada con Oracle JET es la que hace un uso más eficiente de la memoria. Por el contrario, Oracle MAF es el que presenta un uso menos eficiente de este recurso.

Peso de las aplicaciones

Por último, se realiza una comparativa entre el tamaño de los archivos *apk* generados para la plataforma Android. Esta comparativa se ha realizado tanto para las aplicaciones generadas en modo *debug* como en modo *release*. La principal diferencia entre una y otra es que en modo *debug* la aplicación incluye información en los archivos compilados que permiten que la depuración de la aplicación sea más sencilla. Por otro lado, las aplicaciones en modo *release* no incluyen esta información y generalmente optimizan el código durante la compilación.

	Tamaño (en KB)		
Modo	**Oracle JET**	**Oracle MAF**	**Ionic 2**
Debug	23771	43614	14749
Release	20080	41997	14750

Tabla 8: Peso de la aplicación en cada framework

Como muestra la Tabla 8, la aplicación generada por Oracle MAF es considerablemente mayor que en el caso de los otros dos frameworks. El tamaño de la aplicación en MAF se debe a que las aplicaciones generadas con este framework incluyen la máquina virtual de Java para poder ejecutarse. El framework que produce una aplicación de menor tamaño es Ionic. Sin embargo, en este caso, el tamaño de la aplicación no se ve reducido cuando pasa de modo *debug* a modo *release*.

5.3.7 Documentación y comunidad de soporte

Ionic 2 presenta una documentación bastante extensa, que cubre desde la instalación del framework y la creación de aplicaciones hasta su despliegue. En su documentación podemos

encontrar información sobre la API y sobre todos los componentes que presenta Ionic, incluyendo ejemplos de su integración en la aplicación. Incluye además documentación sobre *Ionic Native* para el uso de plugins Cordova con TypeScript e información sobre como personalizar el Look&Feel de la aplicación. El problema que presenta Ionic es que en algunos casos la documentación sobre el uso de componentes, plugins y API es escasa. También hay que tener en cuenta que Ionic 2 está en versión Beta, y que por lo tanto varios de sus componentes han ido sufriendo cambios con el lanzamiento de nuevas versiones, siendo la documentación distinta en cada una de ellas.

Oracle JET cuenta con una documentación muy competa y una amplia comunidad de soporte. En la web de Oracle podemos encontrar información muy completa sobre todas las características ofrecidas por el framework, incluyendo guías para desarrolladores, tutoriales y ejemplos. JET incluye además un *cookbook*[56] en el que se explica y se muestra el uso de cada uno de sus componentes. Este *cookbook* permite modificar el código de ejemplo de cada componente y ver cómo afecta a su comportamiento.

Oracle MAF es posiblemente el framework que cuenta con una documentación más completa. Al igual que en JET, podemos encontrar una gran cantidad de información sobre MAF en el sitio web de Oracle, incluyendo guías, ejemplos y tutoriales. La guía de MAF incluye descripciones paso a paso sobre la aplicación de distintas funcionalidades y configuraciones que ofrece el framework, con imágenes que muestran el uso de MAF en JDeveloper. Se pueden encontrar una gran cantidad de aplicaciones de ejemplo realizadas con MAF que pueden ser de gran utilidad para los desarrolladores. Entre estas aplicaciones se incluye una aplicación de galería que muestra el uso de los distintos componentes de MAF y cómo pueden modificarse.

5.3.8 Velocidad de desarrollo

La definición de “velocidad de desarrollo” es relativa, ya que se debe tener en cuenta que la velocidad con la que desarrollamos con un determinado framework depende en gran medida de la experiencia que tengamos con dicho framework. Sin embargo, cada framework presenta una serie de **herramientas** que pueden ayudar a desarrollar distintas funcionalidades de una forma más rápida.

[56] Cookbook de Oracle JET: *http://www.oracle.com/webfolder/technetwork/jet/jetCookbook.html*

La gran fortaleza de Oracle MAF reside en que está enfocado en ofrecer un desarrollo **visual** y **declarativo,** incrementando la productividad de los desarrolladores. La funcionalidad Drag&Drop proporcionada por MAF hace que la creación de interfaces de usuario sea muy rápida. Por ejemplo, podemos crear una lista de elementos simplemente pinchando y arrastrando sobre una colección de elementos que tengamos almacenada en un *array*. Además, el uso de *Data Controls* simplifica el acceso a los servicios, la lógica del backend y las características del dispositivo.

En cuanto a la navegación entre vistas, Ionic 2, con el concepto de **pila de navegación,** hace que sea relativamente sencillo de desarrollar, ya que en los otros frameworks podemos encontrar problemas que ralenticen el desarrollo si queremos una navegación entre vistas con un comportamiento muy concreto.

Tanto JET como MAF incluyen librerías que facilitan **su integración con los servicios y productos de Oracle,** sobre todo los servicios en la nube, como Oracle Mobile Cloud Service (MCS), que se utilizó en este proyecto para la creación del backend de la aplicación *#smact Try&Win*.

5.3.9 Interfaz de usuario

Cuando desarrollamos una aplicación móvil híbrida es importante que la interfaz de usuario se **adapte al estilo de la plataforma** para la que la aplicación va a utilizarse. En el caso de los frameworks que estamos comparando, todos permiten adaptar la interfaz para las plataformas Android, iOS y Windows Phone.

Ionic usa modos para personalizar el aspecto de los componentes. Cada una de las plataformas tiene un modo por defecto, aunque este se puede sobrescribir. Por ejemplo, una aplicación para la plataforma Android usará el modo md (Material Design). En este caso, la directiva <*ion-app*>, que define el componente raíz de la aplicación de Ionic, tendrá la clase md añadida por defecto y todos sus componentes utilizarán el estilo Material Design. La Tabla 9 muestra el modo por defecto añadido a cada plataforma.

MAF y JET utilizan Oracle Alta UI para el desarrollo de interfaces de usuario. Oracle Alta UI es un sistema de diseño para aplicaciones móviles y aplicaciones web que se ha utilizado para desarrollar los productos Cloud más recientes de Oracle y un gran número de aplicaciones móviles.

Plataforma	Modo	Detalle
iOS	ios	Las aplicaciones que se ejecuten en iPhone, iPad o iPod usan el estilo iOS
Android	md	Las aplicaciones ejecutadas en un dispositivo Android usan el estilo Material Design
Windows Phone	wp	En los dispositivos Windows, las aplicaciones usan el estilo Windows
Core	md	Cualquier plataforma distinta a las anteriores utilizará el estilo Material Design

Tabla 9: Estilos en Ionic 2 para cada plataforma

Al igual que Ionic, Alta UI incluye una serie de clases CSS por defecto que se aplicarán dependiendo de la plataforma en la que se ejecute nuestra aplicación, para dotarla de un aspecto nativo.

5.4 Ponderación de los resultados

En este apartado se detalla cómo se ha realizado el sistema de ponderación de los resultados obtenidos en el apartado anterior. En la Tabla 10 se muestra el resultado.

Tanto las categorías de *sistema de bindings* como la de *interfaz de usuario nativa* no influyen en el resultado, ya que los tres frameworks proporcionan dichas funcionalidades. En cuanto al resto de categorías hay que decir que no en todas ellas se ha podido asignar una puntuación en base a unos datos objetivos, sino que se ha asignado en función de la experiencia obtenida al desarrollar la aplicación *#smact Try&Win* con cada uno de ellos. Estas categorías son:

- El sistema de navegación.
- La seguridad.
- Documentación y comunidad.
- Velocidad de desarrollo.

Categoría	JET	MAF	Ionic	Ponderación
Sistema de binding	Si	Si	Si	-
Interfaz de usuario nativa	Si	Si	Si	-
Sistema de navegación	6	8	10	10%
Acceso a servicios del dispositivo	1,4	10	5,7	10%
Reconocimiento de gestos de pantalla	6	10	6	5%
Seguridad	7	10	5	15%
Documentación y comunidad	8	10	7	10%
Velocidad de desarrollo	7	10	8	20%
Aplicación de bindings	5,6	9	8,3	8%
Acceso a SQLite	2,3	8,9	2,4	4%
Transición de vistas	2,5	3	10	8%
Renderizado	10	8,5	6,2	2%
Pintado	10	7,2	8,2	2%
Uso de recursos del dispositivo	8,9	8,7	8,8	4%
Tamaño del APK	7	3,5	10	2%
Total	**5,9**	**8,8**	**7,3**	

Tabla 10: Ponderación de los resultados

Para la asignación de puntuaciones, en cada categoría se han asignado 10 puntos al framework que mejor resultado tiene, y se ha determinado la puntuación de los otros frameworks mediante una regla de tres (salvo en las características sin resultados objetivos).

Para determinar el framework que mejor resultado presentaba en las características *acceso a los servicios del dispositivo* y *reconocimiento de gestos de pantalla* se ha tenido en cuenta el número de plugins y funcionalidades que cada framework ofrece para el acceso a los servicios y el número de gestos de pantalla que permite reconocer por defecto, respectivamente.

En cuanto a las categorías que conforman el apartado de rendimiento hay que tener en cuenta las siguientes consideraciones:

- El conjunto de estas categorías suponen el 30% del total.
- La puntuación obtenida en la categoría de *aplicación de bindings* es la media de la nota obtenida en las mediciones realizadas con listas de mil y 10 mil elementos.
- Para la categoría acceso a SQLite, se ha considerado que el 80% de la puntuación corresponde a las medidas tomadas cuando ya se ha obtenido la conexión a la base de datos, y el 20% a las medidas tomadas cuando se hace una consulta inicial. Se le ha dado este peso debido a que el tiempo de acceso a base de datos cobrará importancia cuando la aplicación requiera de muchas peticiones a SQLite.
- En el uso de los recursos del dispositivo, se ha considerado el mismo peso para el uso de la CPU, la media de memoria virtual usada y la media de RSS.
- Para la puntuación en el tamaño de las APKs se ha asignado el 80% del peso al resultado obtenido para una APK en modo *release*, correspondiendo al 20% del peso en modo *debug*, debido a que las APKs en modo *release* son las que generalmente llegan al usuario final, quedando el uso del modo *debug* para los desarrolladores.

Conclusión

Viendo los resultados obtenidos podemos concluir que Oracle MAF es el framework más completo. Sin embargo, hay que destacar que la ponderación se ha realizado según qué categorías han sido consideradas de mayor o menor importancia según el autor. Por lo tanto, esta ponderación puede cambiar dependiendo de lo que cada desarrollador considere más importante, de modo que se ajuste a los requisitos de la aplicación que se quiera desarrollar. Por ejemplo, si se quiere desarrollar una aplicación en la cual la transición entre pantallas sea muy fluida, o que tenga una navegación muy concreta entre los módulos, estas categorías tendrán un mayor peso.

Además, dependiendo del tamaño y complejidad del desarrollo puede resultar más adecuado uno u otro framework. Para el desarrollo de proyectos pequeños y poco complejos posiblemente sea más adecuado utilizar Oracle JET o Ionic. Ionic 2, al utilizar TypeScript como lenguaje principal de programación, hace que el código desarrollado generalmente sea más claro, mediante el uso de objetos, que el código desarrollado en JavaScript. Para

proyectos grandes y complejos, Oracle MAF nos ofrece un desarrollo más estructurado, y como resultado obtendremos un software más robusto.

CAPÍTULO 6
CONCLUSIONES

Con el desarrollo de este Trabajo Fin de Master se han conseguido satisfacer los objetivos y requisitos propuestos en el Capítulo 2. En este capítulo se detallan las conclusiones obtenidas tras la realización de este TFM, y se propone una serie de propuestas futuras para continuar con la línea de este trabajo. Finalmente se exponen las conclusiones personales obtenidas con la realización del trabajo.

6.1 Consecución de objetivos

El objetivo principal de este TFM era la realización de una comparativa entre distintos frameworks del mercado con el fin de obtener un conocimiento que permitiese establecer el posicionamiento de los frameworks de desarrollo Oracle con respecto al resto de frameworks existentes en el mercado. Se considera que el objetivo principal ha sido cumplido gracias a la consecución de todos los objetivos parciales.

El objetivo de la realización de una **revisión exhaustiva del estado del arte de los frameworks** se ha conseguido gracias al estudio realizado sobre diferentes frameworks existentes, centrándose dicho estudio en los tres frameworks sobre los que se realizó la comparativa: MAF, JET e Ionic. Se completó con la preparación de los entornos de desarrollo de estos frameworks y la realización de ejemplos iniciales. Estos ejemplos se centraron en el uso de servicios del dispositivo, como la cámara, localización, bluetooth, etc.

Tras una primera toma de contacto con los frameworks, se llevó a cabo el **desarrollo de la aplicación** *#smact Try&Win* con los tres frameworks. La aplicación se presentó en el Oracle Digital Day organizado por Oracle en Madrid el 27 de Octubre de 2016, y posteriormente se presentó en el avanttic Day, que tuvo lugar en la Escuela Superior de Informática de Ciudad Real, el 16 de Diciembre de 2016.

Finalmente, se realizó satisfactoriamente el desarrollo de la **comparativa entre los frameworks,** mediante la realización de las comparativas de cada una de las características detalladas en el Capítulo 2 y la posterior ponderización de los resultados.

6.2 Propuestas futuras

Debido a la gran cantidad de frameworks y tecnologías de desarrollo existentes y su continua evolución podemos definir una serie de líneas de trabajo futuro que el proyecto podría seguir:

- Debido a la falta de recursos, las pruebas realizadas para obtener resultados de rendimiento solo se ha llevado a cabo en un dispositivo con la plataforma Android. Sería interesante realizar dichas pruebas en las **plataformas iOS** y **Windows Phone** y ver cómo se comporta la aplicación en estas plataformas.
- Extender el trabajo a **otros frameworks de desarrollo.** En concreto sería interesante introducir Xamarin, ya que este framework proporciona unas características y un enfoque de desarrollo muy parecidas a las de Oracle MAF, siendo uno de sus principales competidores.
- Desarrollo de una **aplicación más grande** y **compleja**, que permita experimentar y conocer los problemas de usar estos frameworks en un proyecto complejo.

En cuanto a la aplicación desarrollada durante este proyecto, esta sigue en continuo crecimiento, añadiendo mejoras continuas, ya que se tiene la intención de presentar la aplicación en futuros eventos. Una de las mejoras que se desea realizar a corto plazo es que los recursos que utiliza la aplicación y que cambian para cada evento, como pueden ser las fotos de los premios, se almacenen en el servidor, de modo que la aplicación las descargue automáticamente cada vez que haya un nuevo sorteo para un evento. La idea detrás de esta mejora es que no sea necesario cambiar la aplicación y subirla de nuevo a los Markets cada vez que surja un evento, simplemente bastará con modificar los recursos almacenados en la parte del servidor.

6.3 Conclusión personal

Como conclusión final, considero que la realización de este proyecto me ha servido para crecer mucho como profesional.

El hecho de que este trabajo se enmarque dentro de la compañía avanttic Consultoría Tecnológica me ha permitido adquirir unas competencias y conocimientos sobre el modo en que se trabaja en una compañía real, y las necesidades y dificultades que surgen durante el desarrollo de los proyectos, lo que me ha permitido afianzar los conocimientos obtenidos durante la realización del Máster.

Además de la adquisición de unos conocimientos técnicos mediante el desarrollo de la aplicación *#smact Try&Win*, considero que he mejorado mi capacidad de análisis de tecnologías, lo cual es importante a la hora de tomar decisiones sobre qué tecnología utilizar cuando se analiza un determinado proyecto, todo esto gracias al estudio y la comparativa realizados durante este trabajo.

Para finalizar, quisiera destacar la satisfacción por la realización de este proyecto, con el que he aprendido mucho, y que da comienzo a una nueva etapa.

BIBLIOGRAFÍA

1. *Informe ditrendia 2016: Mobile en España y en el Mundo*. 2016. p. 87.

6. Gironés, J.T., *El gran libro de Android*. 2013.

7. Morata, J.Q., *Desarrollo de una aplicación distribuida para dispositivos iOS*. 2011, Universidad Politécnica de Valencia.

8. Castellanos, F.J., *Desarrollo de aplicaciones para Windows Phone*. 2013, Universidad Carlos III de Madrid.

12. Andrade, P.R.M.d. and A.B. Albuquerque, *Cross platform App. A comparative study*. International Journal of Computer Science & Information Technology, 2015.

14. Schwaber, K. and J. Sutherland, *The Scrum Guide*. 2013.

15. Whittaker, M., *Oracle JavaScript Extension Toolkit (JET). Developing Applications with Oracle JET*. 2015.

16. Rekadze, L., C. Hall, and W. Egan, *Oracle Mobile Application Framework. Developing Mobile Applications with Oracle Mobile Application Framework*. 2015.

WEBGRAFÍA

2. *Gartner Recommends a Hybrid Approach for Business-to-Employee Mobile Apps*. Available from: http://www.gartner.com/newsroom/id/2429815.

3. *Net Market Share*. Available from: www.netmarketshare.com.

4. *Native Apps vs. Web Apps – What is the Better Choice?* Available from: https://www.lifewire.com/native-apps-vs-web-apps-2373133.

5. *The Pros and Cons of Native Apps and Mobile Web Apps*. Available from: https://www.lifewire.com/pros-and-cons-of-native-apps-and-mobile-web-apps-2373173.

9. *Mobile Website vs. Mobile App: Which is best for your organization?* Available from: https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/.

10. *The benefits of web-based applications*. Available from: http://www.magicwebsolutions.co.uk/blog/the-benefits-of-web-based-applications.htm.

11. *Por qué y cómo adaptar tu web a dispositivos móviles*. Available from: http://infoautonomos.eleconomista.es/blog/adaptacion-web-dispositivos-moviles/.

13. *Hybrid apps: The future of mobile development?* Available from: https://gcn.com/Articles/2013/02/12/Hybrid-apps-future-mobile-development.aspx.

17. *Ionic application framework*. Available from: http://ionicframework.com/.

ANEXO A
EVENTO ORACLE DIGITAL DAY

El día 27 de Octubre se celebró el Oracle Digital Day 2016, que tuvo lugar en el Palacio Cibeles de Madrid. Se trata del evento más importante celebrado por Oracle al año, sobre transformación digital.

La compañía avanttic, al ser patrocinador del evento, disponía de un stand en el que los asistentes podían disfrutar de una experiencia smact (social, mobile, analytics, cloud, things) a través de un sorteo digital. Para participar en el sorteo, los asistentes debían descargarse la aplicación *#smact Try&Win*, desarrollada durante la realización de este trabajo, y superar una serie de retos relacionados con IoT y redes sociales. En el stand, se podía visualizar en directo un resumen del estado actual del sorteo, mediante una aplicación web construida con Oracle JET. Además, podían visualizarse analíticas de eventos del sorteo, como el porcentaje de participantes que habían superado su primer reto, y de la plataforma cloud.

Durante el evento se mantuvieron conversaciones con diversos asistentes sobre la arquitectura de la aplicación, además de proporcionar soporte a los usuarios que participaban en el sorteo.

Tras observar el comportamiento de la aplicación durante un sorteo en directo se acordaron posibles mejoras que podían aplicarse tanto en la aplicación como en el lado del servidor. La realización de estas mejoras y la corrección de *bugs* permitieron que la aplicación tuviese un aspecto más robusto, y volvió a presentarse en la Escuela Superior de Informática el 16 de Diciembre de 2016 durante el avanttic Day, donde se pudo mantener conversaciones con algunos alumnos y profesores de la Universidad.

Figura 36: Imagen promocional para el Oracle Digital Day

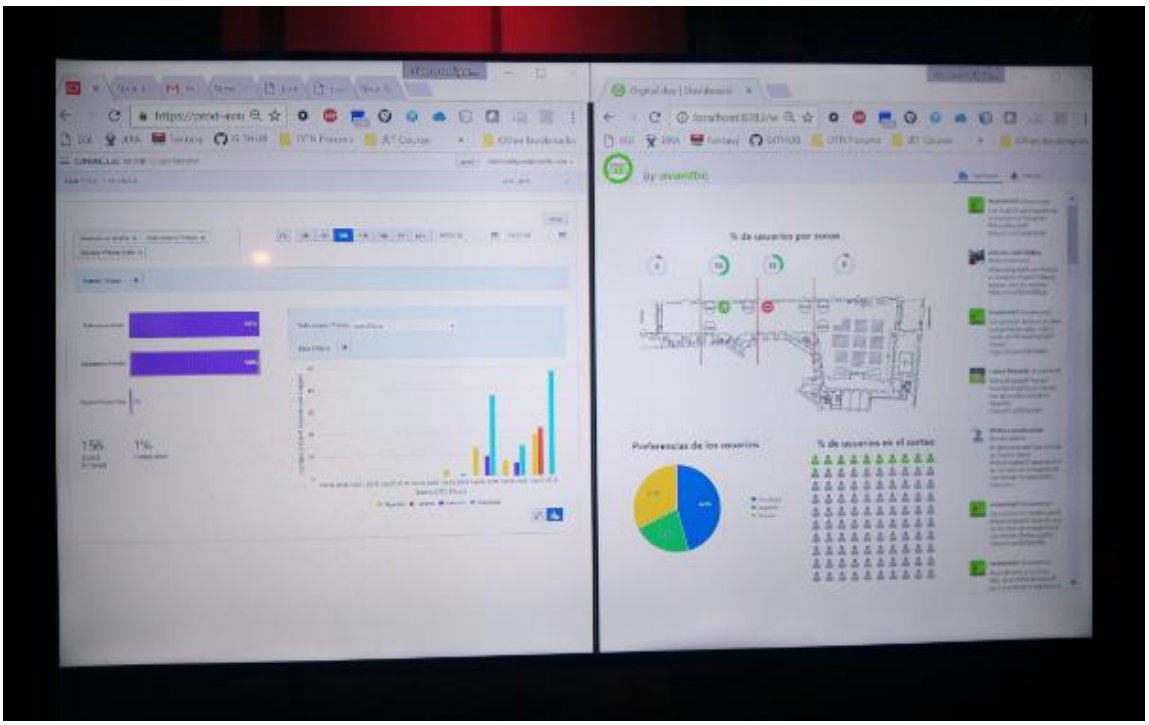


Figura 37: Analíticas de la aplicación (izq.) y resumen del sorteo (der.)

Figura 38: Stand de avanttic para el avanttic Day en la ESI