UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Informática

TRABAJO FIN DE GRADO

Aplicación móvil multiplataforma, nativa y de código único para la publicación de contenidos gestionados desde Web

Stanislav Mostovoy
Tutor: José Antonio Clavijo Blázquez
Ponente: Eloy Anguiano Rey

JUNIO 2015

Resumen

La creación de aplicaciones móviles es una necesidad en crecimiento. Estas aplicaciones suelen ser interfaces de usuario que gestionan información que se encuentra en servidores web de contenidos. Es por ello que el desarrollo de aplicaciones móviles necesita emplear nuevas tecnologías y herramientas que agilicen el tiempo de publicación de las aplicaciones a la vez que disminuyen el coste de desarrollo de las mismas, especialmente en un mundo en el que existe diversidad de plataformas móviles (iOS, Android, Windows Phone, Firefox OS, etc.) y se hace necesario gestionar el código fuente de aplicaciones desde un punto de vista único. Existe una amplia variedad de herramientas y frameworks que facilitan el desarrollo de código único y que cuentan con distintas peculiaridades, por lo que es necesario valorar todas las características que nos ofrecen para poder decidir cuál es la más idónea para el desarrollo concreto de cada aplicación.

En este proyecto se realiza un estudio de varias de las tecnologías existentes en la actualidad con el objeto de determinar cuáles son los puntos fuertes y débiles de cada una de ellas. En el proyecto se estudia la calidad con la que el framework representa visualmente sus componentes básicos, se analizan las posibilidades que ofrece el framework para acceder a las API del sistema operativo y se determina la facilidad de trabajo con este sistema: analizando el IDE disponible para trabajar con el framework, la cantidad y calidad de la documentación o la existencia de comunidades de desarrolladores para ofrecer ayuda o módulos adicionales. El coste de la licencia es otro factor que se tiene en cuenta en los estudios realizados.

Para los desarrollos de los prototipos, se emplea un ciclo de vida en espiral que permite afrontar los distintos problemas con adecuados niveles de riesgo en el proyecto, siendo el principal de ellos el desconocimiento previo de las tecnologías que se estudian. Se realizan tres iteraciones para obtener los resultados y conclusiones finales del proyecto.

Los resultados logrados son fundamentalmente un conjunto de errores y aciertos obtenidos bajo la hipótesis de selección de cada uno de los frameworks. Estos resultados se analizan posteriormente para poder realizar las recomendaciones de utilización de los distintos frameworks según las distintas situaciones de partida dadas.

También se ha realizado una aplicación de referencia con cada uno de los frameworks para poder apreciar la forma de empleo de éstos en un proyecto real.

Por último se concluye el trabajo con una serie de posibles trabajos futuros que se pueden realizar.

Palabras clave

Comparación, Android, iOS, Windows Phone, Ionic, Cordova, Qt, Appcelerator, Xamarin, ReactNative, NativeScript, TabrisJs

Abstract

The creation of mobile applications is a growing need. These applications usually are user interfaces which manage information stored in content Web servers. As a result, the development of mobile applications requires the employment of new technologies and tools which hasten the publication times of the applications as well as decrease the cost of their development, especially in a world with a variety of mobile platforms (iOS, Android, Windows Phone, Firefox OS, etc.), and so arises the need of managing the source code from a unique point of view. There is a wide range of tools and frameworks which facilitate the development of unique code and have their own peculiarities, thus the need to evaluate all their characteristics in order to decide which one is the most suitable for the development of each particular application.

In this project, a study of several current technologies is conducted to determine their strengths and weaknesses. This project analyses the quality of the frameworks' visual representation of their basic components as well as the possibilities presented by the frameworks to access the APIs of the operating system and determines the easiness of working with them — analysing the available IDEs to use with each framework, the extension and quality of the documentation or the existence of developer communities which may provide help or additional modules. The license cost is also taken into consideration in this study.

A spiral model life cycle is used to develop the prototypes. It allows facing the different problems posed during the project with the appropriate risk levels, being the main problem the initial lack of knowledge on the studied technologies. Three iterations are made to obtain the results and final conclusions of this project.

The obtained results are essentially a group of successes and failures obtained under the hypotheses of selecting each one of the frameworks. These results are later analysed in order to put forward recommendations for using the different frameworks according to the different given starting situations.

Furthermore, a reference application is developed with each one of the frameworks in order to evaluate their mode of use in a real project.

Finally, the project is concluded with a series of possible future work.

Keywords

Comparison, Android, iOS, Windows Phone, Ionic, Cordova, Qt, Appcelerator, Xamarin, ReactNative, NativeScript, TabrisJs

Glosario

Framework: Marco de programación y tecnológico que sirve de soporte para el desarrollo de una aplicación.

Smartphone: Teléfono móvil inteligente, con un sistema operativo capaz de ejecutar aplicaciones comerciales.

API: (*Application Programming Interface*) Interfaz de programación de aplicaciones

Picker: Elemento visual que se utiliza para seleccionar un dato de un conjunto de opciones.

IDE: (*Integrated development environment*) Entorno de desarrollo integrado.

SDK: (*software development kit*) Conjunto de herramientas para el desarrollo de software

HTML: (HyperText Markup Language) Lenguaje de marcado utilizado en las páginas web

Plugin: Aplicación que sirve como complemento a otra aplicación.

Índice de contenido

1 Introducción....1
1.1 Motivación....1
1.2 Objetivos y retos....1
1.3 Estructura de la memoria....2
2 Estado del arte....4
2.1 Crecimiento de las aplicaciones móviles....4
2.2 Sistemas Operativos móviles....6
2.2.1.iOS....6
2.2.2.Android....6
2.2.3.Windows....7
2.3 Aplicaciones híbridas....7
3 Metodología....9
4 Criterios de comparación....12
5 Resultados....15
5.1 Ionic....15
5.1.1.Introducción....15
5.1.2.Forma de trabajo....15
5.1.3.Análisis del framework....15
5.2 Qt....17
5.2.1.Introducción....17
5.2.2.Forma de Trabajo....17
5.2.3.Análisis del framework....17
5.3 Appcelerator....19
5.3.1.Introducción....19
5.3.2.Forma de Trabajo....19
5.3.3.Análisis del framework....19
5.4 Xamarin....21
5.4.1.Introducción 21
5.4.2.Forma de Trabajo....21
5.4.3.Análisis del framework....22
5.5 ReactNative....23
5.5.1.Introducción....23
5.5.2.Forma de Trabajo....24
5.5.3.Análisis del framework....24
5.6 NativeScript....25
5.6.1.Introducción....25
5.6.2.Forma de Trabajo....25
5.6.3.Análisis del framework....26
5.7 TabrisJs....27
5.7.1.Introducción 27
5.7.2.Forma de Trabajo....27
5.7.3.Análisis del framework....28
6 Valoración de resultados....30
6.1 Valoración final....31
7 Aplicación final....32
8 Trabajos futuros....35

9 Bibliografía....36
Apéndice A: Análisis detallado de los resultados....37
Ionic....37
Qt....42
Appcelerator....46
Xamarin....51
ReactNative....55
NativeScript....60
TabrisJs....63

Índice de ilustraciones

Figura 1: Gráfica de descargas de aplicaciones en distintas plataformas....4
Figura 2: Gráfica de usos de aplicaciones por diferentes categorías....5
Figura 3: Gráfica de porcentaje de venta de dispositivos por plataforma....6
Figura 4: Representación del ciclo de vida en espiral....10
Figura 5: Pantalla inicial de la aplicación....32
Figura 6: Pantalla con el listado de noticias....33
Figura 7: Pantalla con el menú lateral desplegado....33
Figura 8: Pantalla donde se pueden modificar los datos personales....33
Figura 9: Error del puntero de texto en Ionic en iOS....39
Figura 10: Aplicación con la cabecera en la posición correcta de iOS....39
Figura 11: Error de la posición de la cabecera de Ionic en iOS....39
Figura 12: Error de elemento Spinner de Qt en iOS....43
Figura 13: Diferencia de colores en Appcelerator. iOS (izquierda), Android (derecha)....48
Figura 14: Error en el texto en negrita en iOS....48
Figura 15: Picker no nativo en Android....48
Figura 16: Representación correcta de Xamarin en iOS....53
Figura 17: Error en el diseño del Picker de ReactNative para iOS....57
Figura 18: Elemento de la lista con imagen de TabrisJs en Android....65
Figura 19: Error en el elemento de la lista sin imagen de TabrisJs en Android....65

Índice de tablas

Tabla 1: API soportadas por Ionic....16
Tabla 2: API soportadas por Qt....18
Tabla 3: API soportadas por Appcelerator....20
Tabla 4: API soportadas por Xamarin....22
Tabla 5: API soportadas por ReactNative....24
Tabla 6: API soportadas por NativeScript....26
Tabla 7: API soportadas por TabrisJs....28

1 Introducción

1.1 Motivación

En la actualidad, el desarrollo de aplicaciones móviles esta en crecimiento, y por consiguiente también las distintas herramientas y frameworks que se utilizan para el desarrollo. La elección correcta del conjunto de tecnologías y de herramientas que se emplearán en los proyectos permite reducir la complejidad de las tareas que se necesita realizar y reducir los distintos costes del proyecto.

Todo el contexto de las aplicaciones móviles es relativamente nuevo y esta en constante avance, desarrollando nuevas formas de aproximación a la creación de proyectos. Para ello se suelen tomar como referente las tecnologías desarrolladas anteriormente para otras plataformas como PC o entorno web.

La empresa Delonia Software SL está realizando proyectos de aplicaciones móviles en la actualidad y desea perfeccionar la forma con la que son desarrollados. Para ello necesita un estudio de los framework que están disponibles en el mercado para la creación de aplicaciones. Posteriormente se emplea este estudio para el desarrollo de una aplicación final.

El estudio de los frameworks se centra en la capacidad que ofrecen para el desarrollo de proyectos reales, haciendo especial hincapié en los costes de desarrollo, por lo que se valora con mucho detenimiento las facilidades y dificultades que ofrecen a la hora de conseguir tener una aplicación funcionando en diferentes plataformas requiriendo el menor esfuerzo posible, sin que esto menoscabe la capacidad de modificar y evolucionar la aplicación en un futuro.

No se pretende buscar los frameworks que ofrezcan el mejor rendimiento de recursos o que se ejecuten más rápido, si estos no actúan como una herramienta completa que permita reducir los tiempos de desarrollo y mantenimiento de las aplicaciones móviles.

1.2 Objetivos y retos

Los objetivos del proyecto es valorar los distintos frameworks entre los que hay disponibles en este momento para el desarrollo de una aplicación móvil. Estos framework tienen que cumplir las características de ser de código único para las diferentes plataformas y crear aplicaciones no basadas en HTML, es decir, se desea poder crear un proyecto que posteriormente se compile para distintas plataformas móviles y se ejecute de forma nativa, sin ningún motor de HTML de por medio. No cumplen estos requisitos aquellos frameworks en los que se tenga que crear un proyecto con código distinto para cada plataforma o aquellos que utilicen motores HTML, como WebKit.

Las funcionalidades de la aplicación estarán limitadas. Se busca crear aplicaciones que

permitan gestionar contenido en un servidor externo, y facilitar la interacción del usuario con este, típicamente un sistema de información (interfaz a una base de datos). Este estudio no cubre otros tipos de aplicaciones como los videojuegos.

El reto a los que nos enfrentamos es principalmente el desconocimiento de los distintos frameworks y al resto del contexto de las aplicaciones móviles. Para poder evaluar correctamente las distintas cualidades de los frameworks es necesario poder identificar los problemas que presentan a los desarrolladores con distintos grados de conocimiento sobre sus estructuras.

Es decir, es importante poder separar errores que se producen por la falta de experiencia del programador de los errores que son causados por el mal funcionamiento del framework. También es necesario especificar cual es la cantidad de experiencia necesaria para poder utilizar esta herramienta.

Para ello es necesario adquirir experiencia trabajando con cada framework en diferentes tipos de componentes y buscar documentación de todos los errores y soluciones que ha causado a otros desarrolladores.

Por último hay que señalar que es imposible probar todas las funcionalidades en todos los entornos y situaciones posibles que se pueden presentar durante el desarrollo de los distintos proyecto reales, sobretodo en este contexto de constante avance, donde todos los framework continúan creciendo. Es necesario estimar en cierta medida la completitud del diseño del framework y hacer una predicción en como avanzará en el futuro.

1.3 Estructura de la memoria

Esta memoria comienza presentando el contexto de las aplicaciones móviles en el capítulo *2 Estado del arte*, esta parte del documento es muy útil para poder identificar cuales son los distintos factores que son necesarios valorar para posteriormente realizar un estudio correcto.

Este estudio sigue un ciclo de vida especifico determinado por las características de este proyecto. En el capítulo *3 Metodología* se determina como se llevará a cabo el estudio y las razones de elección de un ciclo de vida frente a otros.

En el capítulo *4 Criterios de comparación* se establecen cuales son los parámetros que se analizan de los distintos frameworks.

La sección *5 Resultados* es la parte principal del documento, en este capítulo se presentan los datos los resultados que se han obtenido durante los análisis de los frameworks.

El capítulo *6 Valoración de resultados* ofrece un breve resumen de los resultados obtenidos en el capítulo anterior y posteriormente se realizan las recomendaciones de uso de los distintos framework tomando en cuenta sus resultados.

En el capítulo *7 Aplicación final* se expone el estado de las aplicaciones finales realizadas

con los distintos frameworks.

Por último en el capítulo *8 Trabajos futuros* se analizan las posibles formas de completar este estudio y como es posible avanzar con el estudio en el futuro, reanalizando los cambios que sufren los frameworks en el futuro.

En el *Apéndice A: Análisis detallado de los resultados* se puede encontrar toda la documentación detallada sobre el estudio realizado. Contiene la versión extendida de los fallos y aciertos que se han encontrado en los distintos frameworks. Esta sección es muy útil para ampliar los datos del capítulo 6 si así se desea.

2 Estado del arte

Este estudio se debe empezar observando y analizando el contexto de las aplicaciones móviles. Se tiene que determinar si su demanda está en crecimiento. Analizar qué características tienen las aplicaciones móviles y cuál es la estructura que se sigue para desarrollar proyectos en estas plataformas.

2.1 Crecimiento de las aplicaciones móviles

Se comienza analizando las tendencias de las aplicaciones móviles. Esto es de vital importancia ya que determina si existirá un mercado para los proyectos en teléfonos móviles y qué tipo de aplicaciones son las más solicitadas. Para ello se emplea una gráfica que ofrece Mobile Statistics (www.mobilestatistics.com), una organización que permite ver una instantánea del estado del mercado de las aplicaciones móviles y como avanza con el tiempo.

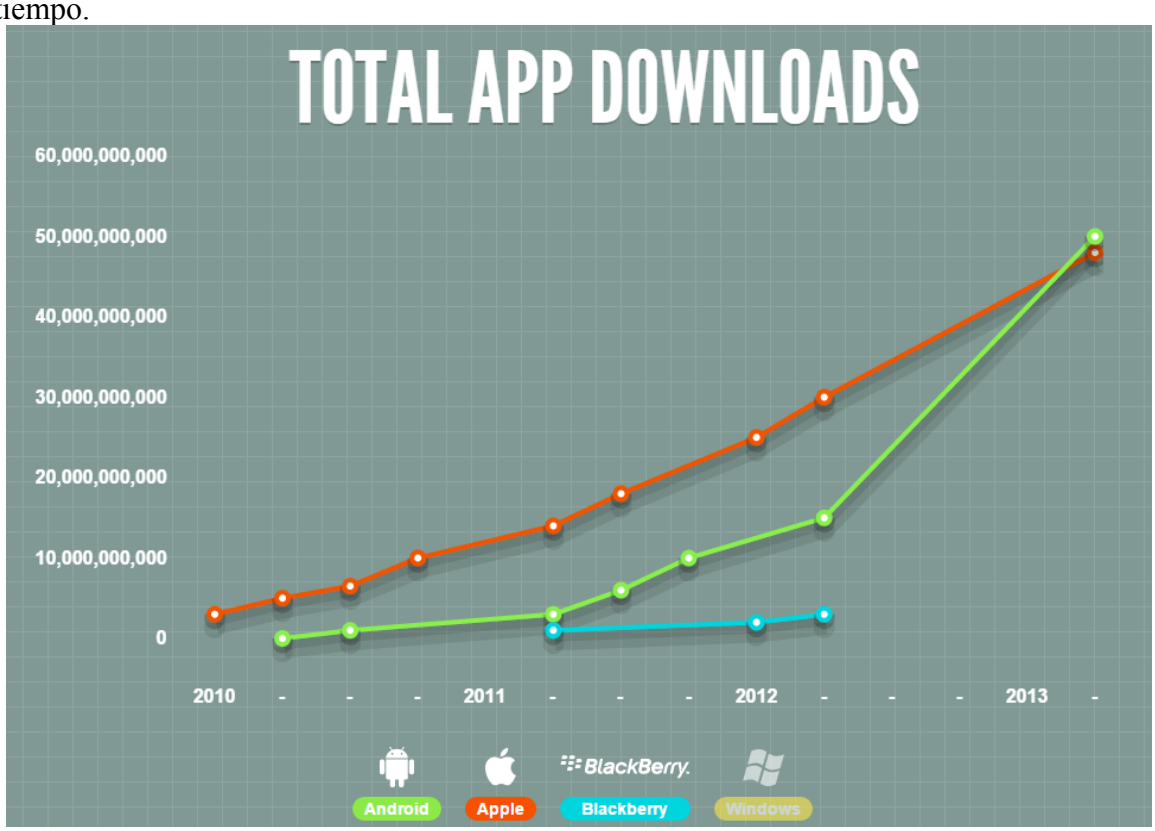

Figura 1: Gráfica de descargas de aplicaciones en distintas plataformas

A lo largo de la historia el desarrollo de aplicaciones móviles han crecido de una manera exponencial. Esto se debe a que existe un mayor público que posee un smartphone y sobretodo a que las distintas empresas ven las aplicaciones móviles como una manera de promocionarse y conectar con sus clientes. Esta tendencia a la popularización de aplicaciones recuerda mucho a lo que ocurría anteriormente con las páginas web.

Pero también es muy importante analizar que tipo de aplicaciones son las que más se crean y qué finalidad tienen. Para ello se utilizar el informe sobre el primer trimestre del año 2015 realizado por la compañía AppAnnie (www.appannie.com), especializada en el estudio del mercado de las aplicaciones móviles.

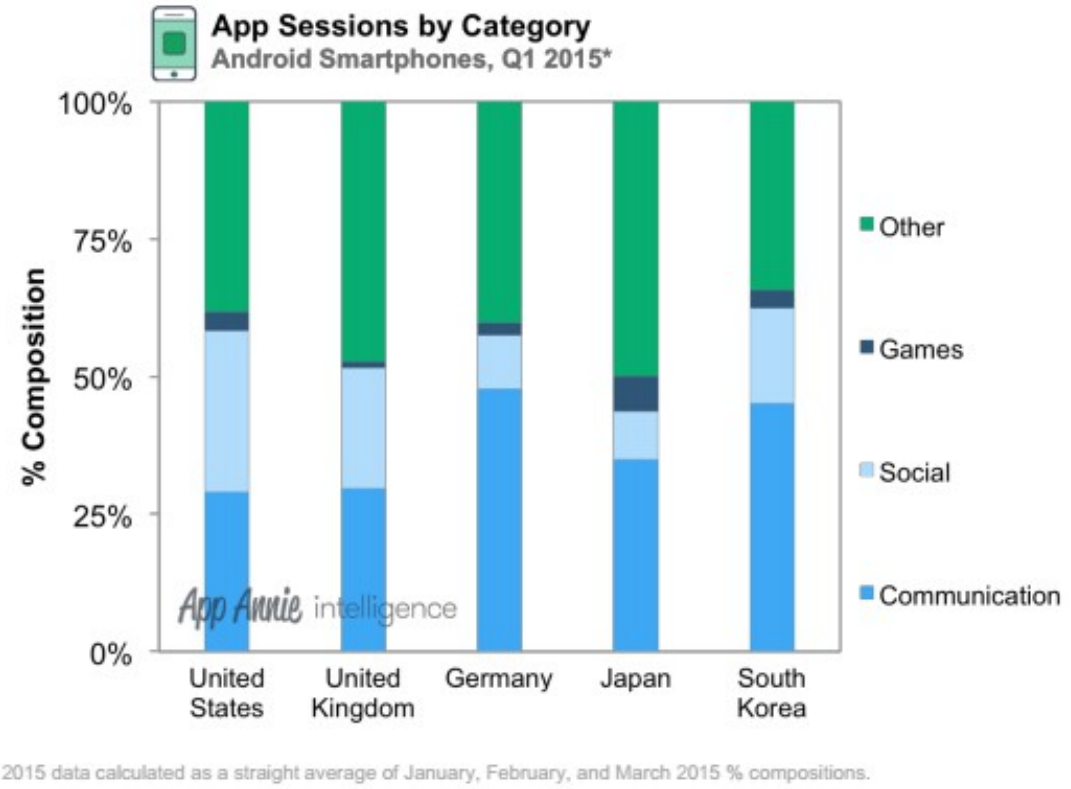

Figura 2: Gráfica de usos de aplicaciones por diferentes categorías

En este informe se hace visible la tendencia de uso de aplicaciones en la categoría social y la categoría de comunicaciones. Estos tipos de aplicaciones se basan en una interacción entre el cliente en el teléfono móvil y un servidor de contenidos. El cliente posee funcionalidades limitadas, como enviar y recibir mensajes, gestionar grupos de usuarios y modificar sus datos personales.

La aplicación móvil dispone normalmente de unas funcionalidades de gestión de información, información que se encuentra almacenada en un servidor externo. Es decir, la lógica principal de la aplicación se encuentra fuera del dispositivo móvil y el teléfono tan solo actúa como una forma de interfaz para acceder al servidor.

De este dato se puede estimar que la mayor parte de las aplicaciones móviles tiene como finalidad gestionar información que se encuentra en un servidor de contenidos externo. Por lo tanto no es necesario una gran carga computacional ni complejos sistemas de lógica, pero es muy necesario realizar una correcta interfaz gráfica para el usuario.

2.2 Sistemas Operativos móviles

El siguiente paso es detectar cuales son las plataformas móviles con las que se va a trabajar y cuales son sus características.

Acudimos a las estadísticas que proporciona IDC (www.idc.com), empresa que se dedica al estudio y análisis de mercado de las tecnologías. Esta compañía ofrecen una tabla donde se muestran los porcentajes de dispositivos de cada sistema operativo que fueron vendidos en el primer trimestre de los años 2015, 2014, 2013 y 2012.

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q1 2015	78.0%	18.3%	2.7%	0.3%	0.7%
Q1 2014	81.2%	15.2%	2.5%	0.5%	0.7%
Q1 2013	75.5%	16.9%	3.2%	2.9%	1.5%
Q1 2012	59.2%	22.9%	2.0%	6.3%	9.5%

Source: IDC, May 2015

Figura 3: Gráfica de porcentaje de venta de dispositivos por plataforma

En esta tabla se puede observar que mayoritariamente se compran dispositivos Android (78%) seguidos por iOS(18,3). Windows Phone tan solo ocupa el 2,7% y BlackBerry el 0,3%. También se debe tomar en cuenta como ha disminuido el mercado de BlackBerry y del conjunto de otros sistemas operativos no nombrados en la tabla.

Estos datos obligan a centrar nuestros esfuerzo en el mercado de aplicaciones Android e iOS, pero sin olvidar la existencia de Windows Phone.

2.2.1. iOS

iOS es un sistema operativo desarrollado por Apple. Esta compañía solo distribuye su software a los dispositivos físicos creados por ella. Apple mantiene una política de actualización constante de su sistema operativo en los dispositivos ya vendidos a los clientes.

La creación de nuevas aplicaciones para esta plataforma se realizan mediante código escrito en Objective C o en su recién lanzado lenguaje de programación Swift, a través de la plataforma Xcode. También es necesario mencionar que las aplicaciones deben ser firmadas por el desarrollador y ser aprobada tras una riguroso proceso de verificación por parte de Apple para poder ser instaladas en el dispositivo físico, pero es posible trabajar en el emulador sin esta restricción.

2.2.2. Android

Android fue desarrollado por Google. La estructura de los teléfonos Android cuentan con

tres botones característicos, incluso en los dispositivos que no tienen estos botones en el hardware, son emulados por el sistema operativo y colocados en la parte inferior de la pantalla, o incluidos de alguna otra forma en la interfaz.

Es importante mencionar la gran fragmentación de versiones de Android existentes en la actualidad. Esto se debe a que el sistema operativo no esta diseñado para actualizarse en los dispositivos, abandonando a los usuarios con los problemas y fallos de seguridad encontrados en versiones antiguas. Esta fragmentación de versiones dificulta la programación para Android, especialmente cuando se usa WebKit.

La forma de desarrollo para Android es usar Java para manejar la lógica de la aplicación y usar archivos XML para la representación de elementos gráficos. Las aplicaciones también cuentan con archivos de configuración para facilitar la creación de la aplicación. Los recursos gráficos como las imágenes se suelen almacenar con diferente resolución para que se utilice posteriormente la imagen correcta en cada dispositivo.

2.2.3. Windows

Windows es desarrollado por Microsoft. Comenzó siendo un sistema operativo orientado a un mercado empresarial, pero posteriormente su uso se amplio a un mercado de consumo. Microsoft realizó varios cambios al nombre de la plataforma, comenzó llamándose Windows Mobile, posteriormente Windows Phone y en las nuevas versiones se llamará Windows.

El sistema operativo de Microsoft no ha contado con gran popularidad entre los consumidores, al menos no el grado deseado por la empresa. Consciente de este problema, Microsoft anunció en la Build developer conference, realizada el día 29 de abril de 2015, que van a trabajar para poder ofrecer una estructura en su sistema operativo con la que se podrán ejecutar aplicaciones Android e IOS en los dispositivos Windows con unos cambios mínimos (Microsoft, 2015).

El desarrollo para las plataformas de Windows se realiza con los lenguaje de programación basados en .Net como es C# y Visual Basic, también se puede utilizar C++. Para el diseño de componentes gráficos se emplea XAML, que es un lenguaje descendiente de XML. Al igual que con otras tecnologías, Microsoft ofrece la posibilidad de extender nuestras aplicaciones a otras plataformas de Microsoft, como Xbox o PC.

2.3 Aplicaciones híbridas

En la actualidad es muy popular utilizar frameworks que crean aplicaciones híbridas. Se emplean técnicas y tecnologías web en el desarrollo de las aplicaciones. Las aplicaciones híbridas son, básicamente, paginas web que se ejecutan en el motor del navegador web del teléfono móvil. Este motor web se llama WebKit en los dispositivos Android e iOS y Trident para Windows Phone.

Utilizar WebKit facilita el desarrollo, pero provoca una serie de problemas. El motor web

es muy pesado en la ejecución, ralentizando las acciones en las aplicaciones. Este no es un problema muy grave, ya que en los teléfonos móviles suele haber solo una aplicación ejecutándose en primer plano, teniendo disponible todos los recursos del sistema.

Otro problema que presenta WebKit es la gran cantidad de versiones que existen en la actualidad. Hay que tener en cuenta que cada versión de Android tiene una versión distinta de WebKit, si a esto se le suma las versiones de los dispositivos iOS y las características muy particulares que tiene el motor Trident de Windows Phone. Se tiene muchos motores web distintos sobre los que se va a ejecutar la aplicación, además, las versiones antiguas del motor no tienen las mejoras y las actualizaciones que se han introducido en las siguientes versiones.

Es muy probable que alguna versión de WebKit no interprete correctamente nuestra aplicación y provoque cambios en el diseño visual. Es muy difícil identificar estos problemas durante las fases de desarrollo, al no poder probar con todas las versiones del motor web. Los clientes serán los que encuentren los errores, provocando quejas. Para arreglar los problemas, de los que informen los clientes, es necesario primero localizar en que versiones se produce el error y posteriormente modificar nuestro código para que se produzcan cambios solo para esas versiones de WebKit. Este proceso es muy complicado de ejecutar y añade complejidad para las siguientes fases del mantenimiento de la aplicación.

Si no se desean utilizar las aplicaciones híbridas, es necesario usar aplicaciones nativas que accedan a las API del sistema operativo de cada plataforma. También es posible emplear llamadas al motor gráfico (OpenGL) para poder pintar los elementos si no se desea emplear WebKit y evitar usar el sistema operativo.

3 Metodología

La creación de todo proyecto software necesita la elección correcta de un ciclo de vida que ayudará lograr los objetivos y resultados deseados. Mientras que, normalmente, estos objetivos suelen ser un producto software, en este proyecto, los objetivos son el análisis del proceso de desarrollo y el análisis del producto final.

Para la realización de los estudios de los frameworks, se va a utilizar un ciclo de vida en espiral. Este ciclo de vida tiene una base iterativa e incremental (Cockburn, A., 2008).

Iterativo, ya que se divide el proyecto en diversos bloques o módulos que buscan alcanzar una serie de logros en cada iteración. Se realiza el desarrollo completo del bloque y posteriormente se aplica una evaluación del resultado producido en la iteración y se analiza la capacidad de ejecución del siguiente módulo.

También sigue la política incremental, este proceso busca avanzar en la creación del producto final, por ello intenta hacer crecer las funcionalidades del producto con cada iteración. El crecimiento se aplica tanto a la aplicación software final resultante del proyecto, como al estudio del framework.

Pero, además, se enfatiza el análisis del riesgo mediante la creación de unos prototipos, mediante los cuales se decide como avanzará el proyecto, ya sea comenzando la siguiente iteración, corrigiendo la anterior o terminando el proyecto.

La elección de este ciclo de vida se basa en cuatro factores claves de este proyecto:

- **Alto desconocimiento de los frameworks**. La falta de experiencia de trabajo con cada uno de los frameworks y del entorno general de las aplicaciones móviles provocan la necesidad de realizar un gran número de análisis de riesgo. Se desconoce el estado exacto de los frameworks y la capacidad que poseen de poder cumplir los requisitos de la aplicación software.
- **Tomas de contacto graduales**. La falta de conocimiento concreto de las capacidades de los frameworks obligan a realizar una serie de iteraciones a través de los distintos frameworks, permitiendo ganar experiencia y realizar un estudio detallado de todos los elementos en cada momento del proyecto.
- **Equilibrio de evaluación**. Al ser una incógnita el entorno de desarrollo móvil, no se consideraría justo la evaluación completa de un frameworks. ya que los siguientes se encontrarían en desigualdad de condiciones. El evaluador poseería unos conocimientos más amplios y podría realizar un producto software mejor del que realizó en los primeros frameworks evaluados.
- **Capacidad de realización de cambios**. En el momento inicial del proyecto se tiene una ruta de estudio de los frameworks, pero es posible que a lo largo de las distintas fases del desarrollo sea necesario modificar el trayecto para poder realizar un estudio mas amplio a la luz de los nuevos conocimientos adquiridos y resultados obtenidos en fases anteriores.

Utilizando esta metodología se crearán tres iteraciones, estas iteraciones se usarán para realizar el estudio de los frameworks.

Siguiendo el ciclo de vida seleccionado, las iteraciones se separarán en cuatro fases:

- **Determinación de objetivos**. En esta fase se decidirá cuales son los hitos concretos que se desean alcanzar en esta iteración
- **Análisis de riesgos**. Se evalúan los posibles factores que pueden afectar a la ejecución de la iteración.
- **Desarrollo**. La fase principal donde se realizará la aplicación que intente alcanzar los objetivos decididos anteriormente . Esta fase terminará con la creación de un prototipo al que se aplicarán las pruebas de análisis del estudio del framework.
- **Planificación**. Se analizar el prototipo de la fase anterior para determinar como se continuará con el desarrollo del proyecto en las siguientes iteraciones.

Para poder representar de forma gráfica este ciclo de vida se emplea la ilustración publicada en el artículo *A Spiral Model of Software Development and Enhancement* de Barry W. Boehm en 1988

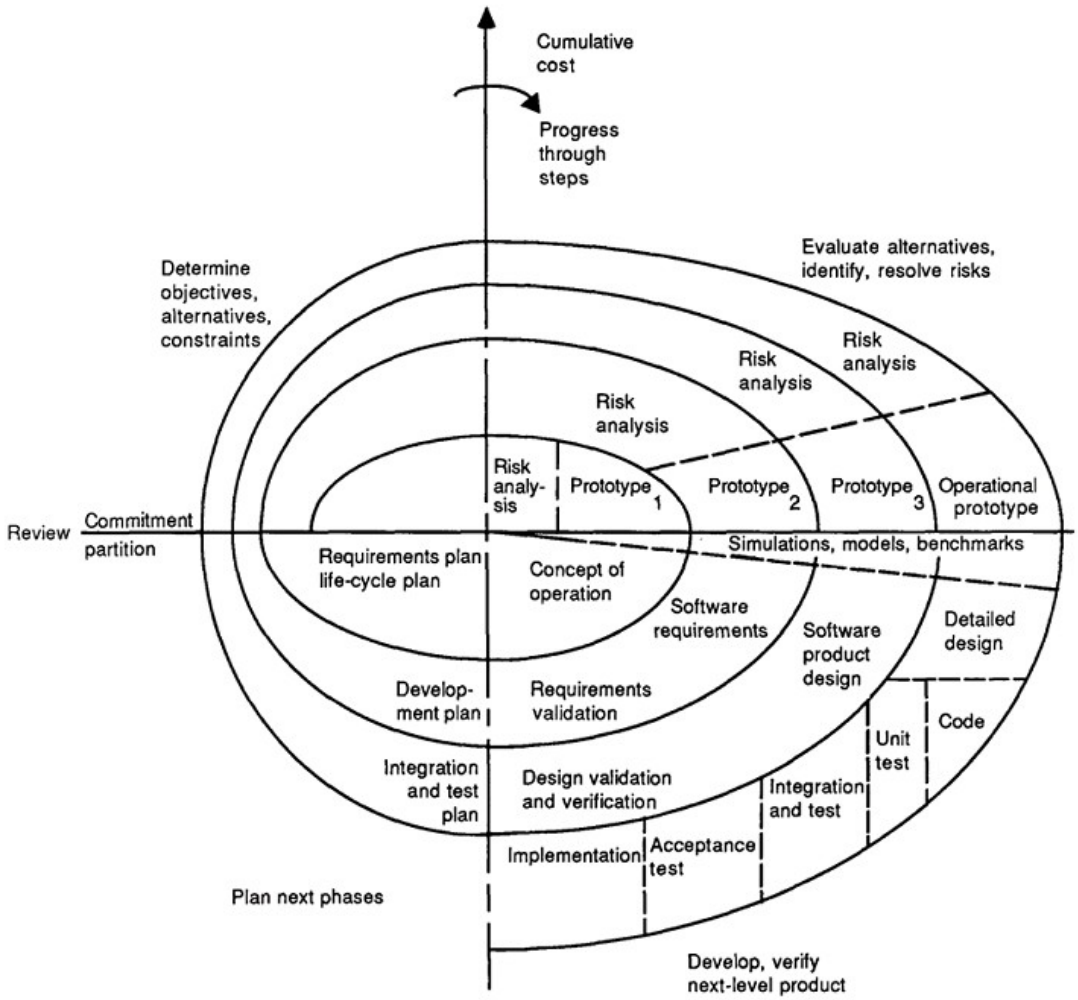

Figura 4: Representación del ciclo de vida en espiral

En el inicio del proyecto se estiman 3 iteraciones. Para cada una de las iteraciones se selecciona un objetivo general que se profundiza a medida que avance el desarrollo:

1. **Primera toma de contacto**. En esta iteración se busca realizar un prototipo simple con el que se pueda aprender la dinámica de cada uno de los frameworks. Se busca realizar una aplicación de una sola página que contendrá varios elementos simples (un texto, una entrada de texto y un botón) colocados de manera arbitraria.

2. **Creación de los componentes principales de la aplicación**. En esta fase se intentará completar el prototipo con elementos complejos que funcionen en todos los tipos de pantallas

3. **Ampliación de funcionalidades**. Se emplearán las distintas API de los frameworks para observar y analizar la capacidad que poseen de cumplir las distintas necesidades que se les pueda exigir posteriormente.

Se parte de un proyecto base de referencia en HTML creado con Ionic, este proyecto será el primero en ser implementado en cada iteración de esta forma se podrá comparar las funcionalidades de los demás framework con él.

4 Criterios de comparación

Para realizar este estudio de los frameworks, es necesario definir que aspectos son relevantes para analizarlos, y cuales tienen un valor inferior.

Para ello inicialmente es necesario analizar el contexto y la finalidad de las aplicaciones móviles. Como ya se explica anteriormente, en el apartado *2 Estado del arte*, la mayoría de las aplicaciones tienen como objetivo la gestión de información y no se espera que tengan una gran carga computacional, ni exijan un gran uso de los recursos del dispositivo.

Además, la arquitectura de los teléfonos móviles permite al usuario mantener activa en primer plano tan solo una aplicación, por lo tanto nuestra aplicaciones mantienen casi en exclusiva los recursos del sistema operativo.

Estas características de las aplicaciones móviles permiten centrar nuestro análisis en los aspectos de coste económico, productivo y de tiempo que conlleva la producción y mantenimiento de una aplicación con cada uno de los frameworks.

Se pueden identificar las siguientes categorías:

- **Calidad de representación y pintado de la interfaz de usuario**.

En este apartado se valorará el estado de los elementos que el usuario final observará en nuestra aplicación. Se examinará la diversidad de componentes y la calidad de pintado de éstos. Se pretende que los componentes se dibujen igual que lo hacen los programas nativos.

- **Capacidad de acceso a las API nativas del sistema operativo**

Se analiza la facilidad que ofrece el framework para llegar a las funcionalidades nativas del dispositivo. En particular se analizarán los siguientes aspectos:

- **Mapa**: Hoy en día la mayor parte de las aplicaciones móviles utilizan un mapa con el que marcar localizaciones de distintos puntos. Por ejemplo, señalar donde se encuentran las tiendas más cercanas. Se analiza la capacidad de poder crear una vista dentro de la aplicación que llame a la interfaz del sistema operativo del dispositivo, Google Map en caso de Android y Apple Maps en el de IOS.

- **Calendario**: Se analizará la capacidad de acceder al calendario del dispositivo para poder observar, crear y eliminar los eventos guardados en el sistema operativo.

- **Almacenamiento de datos** en el dispositivo: Una gran cantidad de las aplicaciones móviles suelen necesitar almacenar datos en el cliente para poder mejorar las prestaciones de la aplicación. Dependiendo de los tipos de datos y de la estructura de la aplicación se suelen almacenar a través de un sistema de ficharos o en una base de datos local. Es necesario explorar las facultades de los framework

para almacenar los datos mediante ambas vías.

- **GPS**: Los teléfonos móviles ofrecen la capacidad de permitir al usuario localizar su posición mediante las redes WiFi o mediante el satélite GPS. El sistema operativo suele ofrecer la capacidad de decidir el grado de precisión que desea utilizar y registrar eventos al cambiar la posición del usuario. Al igual que los sensores, la capacidad de acceso a los datos de localización puede ser muy útil para poder diseñar una aplicación móvil que sea de gran provecho para el usuario. Por lo tanto es necesario analizar las capacidades de los frameworks relativos a este aspecto.

- **Sensores**: Una de las características principales de los dispositivos móviles son los diferentes sensores que poseen. Estas cualidades suelen ser explotadas por las aplicaciones móviles para mejorar la interacción con el usuario. Estos sensores dependen de la plataforma, por ejemplo, los dispositivos iOS cuentan con brújula mientras que los de Android tienen termómetro y sensor de presión.

- **Bluetooth**: La capacidad de conexión a través del canal de bluetooth está valorada positivamente ya que permite poder comunicarnos de forma remota con otros dispositivos, ya sea para transferir ficheros, comunicación de chat o manejar remotamente un aparato.

- **Integración de redes sociales**: Hoy en día las redes sociales proporcionan una gran cantidad de publicidad a diversos proyectos, incluidas las aplicaciones móviles. La capacidad de integrar distintas funcionalidades de acceso de las redes sociales a través de nuestra aplicación permitirán mantener a los antiguos consumidores de la aplicación y difundir nuestra información en busca de nuevos clientes.

- **Push Notification**: Este es un sistema de los servicios de Google y Apple con el que se pueden enviar mensajes a los dispositivos que contengan nuestra aplicación, aunque no este en ejecución. Se suele usar para mantener el contenido de la aplicación actualizado, por ejemplo, nuevos mensajes en una aplicación de comunicación, nuevas artículos para un gestor de noticias o alertar de un evento a los jugadores de un videojuego. Esta herramienta suele mantener un carácter fundamental para algunos tipos de aplicaciones, por lo que es necesario estudiar la forma de integración que poseen los frameworks con él.

- **NFC** (Near field communication): Esta es una tecnología inalámbrica de corto alcance (unos 15 cm). Se suele usar para comunicaciones cortas que no necesite una gran transmisión de datos. Esta tecnología esta principalmente enfocada en la identificación del usuario del dispositivo y como método de pago. Mientras que los dispositivos iOS han deshabilitado el poder usar esta capacidad en nuestras aplicaciones, los dispositivos Android permiten acceder a sus funcionalidades. Es necesario valorar la capacidad de los framework de emplear esta tecnología, ya que esta siendo usada por diversas aplicaciones, por ejemplo, existen proyectos enfocados en el uso de NFC en el mundo del videojuego.

- **Conexión Http**: Las conexiones a servidores web suelen ser muy comunes en las aplicaciones. Los servidores web suelen ser los que se encargan de almacenar y procesar toda la información que usará nuestra aplicación móvil, por lo tanto es necesario analizar la forma de enviar y recibir la información que ofrecen los frameworks.

– **Sistemas operativos con los que trabaja**

Se analiza cuales son las plataformas para las que se puede compilar nuestro código.

– **Facilidad de uso**: IDE, documentación y comunidad de desarrolladores

La complejidad y el esfuerzo que requieren la creación y mantenimiento de la aplicación es un factor clave en este estudio. En esta sección se estudiarán todos los elementos relativos al framework que faciliten el uso de su tecnología. La mayoría de los frameworks tienen un IDE que suele estar adaptado a trabajar con sus proyectos, se analiza su utilidad y el grado de reducción de trabajo que ofrece al desarrollador.
Este apartado se centra en la cantidad, calidad y claridad de la documentación que ofrece.
Toda tecnología suele tener una comunidad de gente que trabaja con ella, esta comunidad suele ser muy útil para buscar documentación, ejemplos de código o realizar consultas. Además, la comunidad de desarrolladores suele participar en el avance del framework o de librerías adicionales.

– **Licencias y costes**

Se analiza los tipo de licencias que ofrece el framework y sus distintos módulos adicionales. Además, en caso de poseen una licencia privada se tomará en cuanta su coste.

– **Previsiones de cambio**

En este apartado se intentará hacer una previsión de la forma en la que crecerá cada uno de los frameworks. Para ello se tomará en cuenta los avances que se han realizado en el pasado y los futuros cambios anunciados por sus desarrolladores.

– **Subjetivo**

Aquí se intentará valorar las sensaciones que ha dado el framework. En este apartado se dará una opinión subjetiva obtenidas al trabajar con el framework. Solo es para ayudar al lector a entender el estudio, este apartado no será valorado para decidir la capacidad del framework.

– **Otros**

Se incluye en esta sección todas las características analizadas del framework que no corresponden a las demás categorías.

5 Resultados

A continuación se exponen los datos de los resultados obtenidos del estudio realizado a los distintos frameworks. En el *Apéndice A* se puede encontrar la versión extendida de los resultados. Se recomienda que en caso de desear ampliar la información de un apartado particular, se acuda al *Apéndice A*.

5.1 Ionic

5.1.1. Introducción

Ionic es una tecnología muy popular que se basa en el uso de WebKit en los dispositivos móviles, la tecnología que se desea sustituir.

Drifty es una empresa que se fundó en el año 2012 por Ben Sperry y Max Lynch. La versión beta de Ionic se publico en noviembre de 2013 y se estima que se han creado un total de unas 500,000 aplicaciones con esta tecnología.

5.1.2. Forma de trabajo

Apache Cordova es una tecnología que se basa en un conjunto de API de JavaScript que permiten acceder a las funcionalidades nativas de nuestros teléfonos móviles.

Posteriormente Ionic facilitó las tareas de creación de aplicaciones, ofreciendo un conjunto de librerías de componentes HTML, CSS y JavaScript. Ionic intenta optimizar las operaciones propias de las páginas web a las plataformas móviles. Además, Ionic emplea Angular como base para todos sus componentes.

5.1.3. Análisis del framework

5.1.3.1 Calidad de representación y pintado de la interfaz de usuario.

El principal problema que tiene Ionic es que utiliza el WebKit, Trident en la plataforma Windows Phone. Por lo tanto existe un motor de pintado distinto para cada versión de WebKit o Trident, sobretodo en dispositivos Android que existe una versión de WebKit distinta para cada versión de Android, ya dada la excesiva fragmentación de Android esto supone un enorme problema.

5.1.3.2 Sistemas operativos con los que trabaja

Al ser ejecutado en el motor del navegador web, un proyecto de Ionic puede ser ejecutado en la mayoría de las plataformas. Ionic funciona en Android, iOS y Windows Phone.

5.1.3.3 Capacidad de acceso a las API nativas del sistema operativo

Como ya se ha comentado antes, Ionic utiliza Cordova para poder acceder a las API nativas y poder emplear las funcionalidades nativas desde el motor de WebKit.

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa	✓		
Calendario	✓		
Almacenamiento	✓		
GPS	✓		
Sensores	✓		
Bluetooth	✓		
Redes Sociales	✓		
Push Notification	✓		
NFC	✓		
Conexión HTTP	✓		

Tabla 1: API soportadas por Ionic

Todas estas funcionalidad son utilizando Plugins, pero estos son ofrecidos de manera gratuita por los desarrolladores de Cordova o la comunidad de desarrollo.

5.1.3.4 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Para el desarrollo de Ionic es posible emplear NetBeans, que tiene incorporados funcionalidades para proyectos de Cordova. Al ser Ionic un framework que emplea tecnología web, existe la posibilidad de trabajar directamente en el navegador web de la máquina de desarrollo.

La documentación que se encuentra de Ionic suele ser bastante amplia y organizada. La comunidad de desarrolladores es muy amplia y existe una gran cantidad de ejemplos fuera de la página oficial. También se participa en el desarrollo de Ionic ya sea con el framework principal o con la creación de Plugins adicionales.

5.1.3.5 Licencias y costes

Ionic es gratuito en la actualidad.

5.1.3.6 Previsiones de cambio

Se espera que Ionic avance en el desarrollo de las aplicaciones híbridas actualizando sus componente, creando nuevos elementos y permitiendo nuevas funcionalidades

Actualmente existen otro proyecto, llamado Crosswalk que intenta crear una versión especifica de WebKit lo suficientemente compacta para poder ser instalada como una aplicación auxiliar. Crosswalk añade unos 15 MB al tamaño del instalador (.apk) y unos 70

MB al tamaño de la aplicación instalada.

5.1.3.7 Subjetivo

El desarrollo con Ionic fue muy fácil y rápido, daba la sensación de completitud de funcionalidades que se desean de este framework. Pero existe un problema insalvable en la actualidad, este problema es el WebKit.

5.1.3.8 Otros

Al emplearse un WebKit, la aplicación necesita una gran cantidad de recursos del sistema operativo y las operaciones dentro de la aplicación son lentas.La aplicación compilada solo necesita guardar los archivos HTML, CSS y JavaScript, gracias a esto la aplicaciones tienen un tamaño muy reducido.

5.2 Qt

5.2.1. Introducción

Qt es un framework multiplataforma utilizado en una gran variedad de entornos de software y hardware muy variable.Actualmente Qt es desarrollada por la compañía Digia y por la comunidad de desarrolladores, por lo que existe actualmente dos versiones con una licencia pública y otra privada.

5.2.2. Forma de Trabajo

La librería de Qt esta compuesta de distintos módulos que ofrecen una API idéntica para las distintas plataformas aunque posteriormente se compilen de distinta manera. El acceso a esas API se puede realizar desde código escrito en C++ o desde QML. QML es un lenguaje de programación creado para QT que se basa en JavaScript, e incluso permite insertar código JavaScript dentro de QML. Para la realización de pruebas del framework de QT se va a emplear QML.

5.2.3. Análisis del framework

5.2.3.1 Calidad de representación y pintado de la interfaz de usuario.

El proyecto de Qt se basa en la creación de componentes nativos utilizando el motor gráfico, sin realizar llamadas al sistema operativo. Esto implica que dependiendo de la plataforma en la que se use Qt, se pintará cada elemento tal como considera el framework que se deba pintar. Esta característica hace que la interfaz gráfica no de una sensación nativa en alguno de sus componentes.

5.2.3.2 Capacidad de acceso a las API nativas del sistema operativo

Qt es un proyecto grande, que esta pensado para una gran cantidad de plataformas. Pero los avances y modificaciones están pensados para el proyecto entero. Los avances que se realizan en la parte de plataformas de teléfonos móviles de Qt suele ser enfocados a solucionar problemas y a implementar funcionalidades del proyecto general de Qt. Pero no

se realizan avances específicos para plataformas móviles.

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa			✓
Calendario			✓
Almacenamiento	✓		
GPS	✓		
Sensores	✓		
Bluetooth			✓
Redes Sociales		✓	
Push Notification		✓	
NFC			✓
Conexión HTTP	✓		

Tabla 2: API soportadas por Qt

Las comunicación mediante Bluetooth no es posible en dispositivos iOS, mientras que las funcionalidades de NFC están implementadas solo para BlackBerry 10.

Por último mencionar que Qt no cuenta con un sistema de módulos que facilite la creación de componentes básicos. Por ejemplo no existe un componente que represente la cabecera de la aplicación y facilite el uso de botones en ella. Esto provoca la necesidad de realizar librerías adicionales para poder realizar después nuestras aplicaciones.

5.2.3.3 Sistemas operativos con los que trabaja

Qt trabaja con Android, iOS, Windows Phone y a otros como PC y Mac.

5.2.3.4 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Qt posee un IDE propio llamado Qt Creator. Este IDE ofrece una gran cantidad de ayuda para la realización de nuestros proyectos en esta plataforma.

Qt cuenta con una documentación extensa sobre todos sus componentes con toda la información necesaria para su uso. Además, existen numerosos ejemplos para todos los elementos de esta tecnología.

Al ser Qt un proyecto con mucha historia, cuenta con una gran cantidad de desarrolladores que suelen avanzar en el proyecto creando sus propios elementos.

5.2.3.5 Licencias y costes

Qt cuenta con una licencia abierta de tipo LGPL.

5.2.3.6 Previsiones de cambio

Qt cuenta con una larga historia de desarrollo en la que ha avanzado hacia un proyecto muy completo. Pero no se espera que se realicen avances rápidos para plataformas móviles ya que el proyecto de Qt se centra en otros aspectos.

5.2.3.7 Subjetivo

Qt no da la sensación de poder ser un framework para los propósitos de creación de aplicaciones móviles de forma rápida y cómoda. Esto se debe a la poca implementación de los módulos de funcionalidades básicas para los teléfonos móviles.

Parece que Qt es un proyecto para plataformas de sobremesa que también se puede ejecutar en teléfonos móviles.

5.2.3.8 Otros

Los proyectos creados en esta plataforma suelen presentar un tamaño bastante grande.

5.3 Appcelerator

5.3.1. Introducción

La compañía Appcelerator surge en 2006 por Jeff Haynie y Nolan Wright. Desde entonces sigue un crecimiento avanzado recolectando una amplia financiación y sus productos son empleada por un gran número de usuarios y compañías.

El día 31 de marzo la compañía anunció que a partir de 1 de junio se realizará un cambio en la estructura de sus productos. Mantendrán el SDK como código abierto pero limitarán el resto de las herramientas relacionadas con él, incluido el framework compilado. Es decir a partir del día 1 de junio es necesario pagar una cuota para seguir utilizando estas herramientas.

5.3.2. Forma de Trabajo

Para manejar los controladores se emplea archivos JavaScript, la vista se compone de archivos XML y Tss (subconjunto de CSS adaptado al framework)

Appcelerator se basa en utilizar un motor de JavaScript en el dispositivo móvil para acceder a las API nativas desde él, sin la necesidad de utilizar un WebKit.

5.3.3. Análisis del framework

5.3.3.1 Calidad de representación y pintado de la interfaz de usuario.

Se han encontrado varias diferencias en la forma de representación gráfica de los elementos. Uno de los más preocupantes fue la diferencia de pintado de los colores, es decir, un mismo código de color se pinta de forma distinta en los distintos sistemas operativos.

5.3.3.2 Capacidad de acceso a las API nativas del sistema operativo

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa	✓		
Calendario	✓		
Almacenamiento	✓		
GPS	✓		
Sensores	✓		
Bluetooth		✓	
Redes Sociales	✓		
Push Notification		✓	
NFC	✓		
Conexión HTTP	✓		

Tabla 3: API soportadas por Appcelerator

Por último mencionar que, es difícil implementar componentes externos. Todos los componentes del framework reaccionan de manera inesperada fuera de las condiciones normales.

5.3.3.3 Sistemas operativos con los que trabaja

En la actualidad es posible utilizar Appcelerator para compilar a Android e iOS. Se esta desarrollando la versión para trabajar con Windows Phone.

5.3.3.4 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

La empresa de Appcelerator ofrece el uso de sus IDE Titanium Studio que se basa en Eclipse. Este entorno de desarrollo, esta preparado para poder compilar y ejecutar nuestro proyecto.

Appcelerator usa un sistema de documentación en el que existe una gran cantidad de contenidos pero se encuentra de forma desorganizada.

Existe una gran comunidad de desarrolladores que crean sus propias librerías e intentan ayudar con las consultas de los usuarios.

5.3.3.5 Licencias y costes

Actualmente Appcelerator es de código abierto, pero a partir del día 1 de junio es necesario el pago mínimo de $39 al mes por cada desarrollador.

5.3.3.6 Previsiones de cambio

Como se ha mencionado anteriormente, este framework se encuentra actualmente en un

proceso de modificación. Estos pueden traer avances a la tecnología empleada en este framework. Pero es vital arreglar todos los fallo expuestos anteriormente, para que las aplicaciones creadas con Appcelerator posean unos acabados correctos.

5.3.3.7 Subjetivo

Este framework no se encuentra completo, le falta terminar los componentes y corregir errores que se encuentran actualmente en él. Pero la compañía ha optado por intentar ampliar sus funcionalidades añadiendo mas módulos.

Esto ha provocado que en la actualidad es solo posible emplear aquellas rutas de desarrollo que estén creados por los desarrolladores del framework, ya que el resto de la tecnología no esta completa ni documentada, además, esta sujeta a cambios que se pueden realizar en versiones futuras del framework.

5.3.3.8 Otros

Este framework utiliza Node para su funcionamiento. Pero las últimas versiones de Node no funcionan correctamente con Appcelerator, por lo tanto es necesario mantener una versión antigua de Node.

La compañía Appcelerator ha creado un lugar donde se puede comprar y vender módulos para su framework. Pero esta zona se encuentra con un número bajo de productos. Y estos suelen ser defectuosos, incluso algunos productos son de otros framework, no funcionando con Appcelerator.

5.4 Xamarin

5.4.1. Introducción

Esta empresa se creo en 2011 por los creadores de Mono. Esta tecnología se uso como base en los proyectos realizados por Xamarin.

5.4.2. Forma de Trabajo

Para trabajar con esta tecnología es necesario programar en C# tanto la interfaz gráfica como la lógica de la aplicación. También existe la posibilidad de usar XAML, un lenguaje descendiente de XML, para el diseño gráfico.

Xamarin es capaz de acceder a las API nativas desde Mono, es decir es posible ejecutar código escrito en C# que será traducido a operaciones en Java y Objective C.

Se empleará el producto Xamarin.Forms, que es una interfaz que une todas las implementaciones de la librería Mono en distintas plataformas, permitiendo realizar código que funciona en varios sistemas operativos.

5.4.3. Análisis del framework

5.4.3.1 Calidad de representación y pintado de la interfaz de usuario.

Xamrin presenta un framework con una alta calidad de pintado, pero existen algunos errores en algunos componentes. Pero estos errores son muy escasos.

En caso de ser necesario, es posible definir como serán pintados los elementos, para ello es necesario volver a definir la función de pintado de cada plataforma.

5.4.3.2 Capacidad de acceso a las API nativas del sistema operativo

Aunque Xamarin.Forms está diseñado para funcionar con varias plataformas al mismo tiempo, sigue exigiendo la posibilidad de implementar funcionalidades de forma distinta para cada una de las plataformas, utilizando los objetos nativos.

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa	✓		
Calendario			✓
Almacenamiento	✓		
GPS	✓		
Sensores	✓		
Bluetooth	✓		
Redes Sociales	✓		
Push Notification			✓
NFC	✓		
Conexión HTTP	✓		

Tabla 4: API soportadas por Xamarin

Las funcionalidades que están marcadas como NO en la tabla, son aquellas que no están implementadas con una interfaz única para distintas plataformas, pero es posible utilizar código particular de cada plataforma para acceder a estas funcionalidades.

5.4.3.3 Sistemas operativos con los que trabaja

Este framework crea aplicaciones para las plataformas Android, iOS y Windows Phone.

5.4.3.4 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Xamarin ofrece utilizar su IDE, llamado Xamarin Studio. También ofrecen un Plugin para la plataforma de Visual Studio de Microsoft. Se han encontrado ciertos problemas puntuales con el IDE Xamarin Studio. Ignora nuestros comandos en algunas situaciones.

Xamarin ofrece amplia documentación sobre sus productos, además, esta información esta bien estructurada y cuenta con una amplia cantidad de ejemplos.

Xamarin posee con una gran comunidad de desarrolladores, que trabajan con Xamarin.Forms u otros productos de esta empresa.

5.4.3.5 Licencias y costes

El coste de utilizar Xamarin.Forms es de $25 por cada desarrollador por cada plataforma, iOS y Android, cada mes. No es necesaria la compra de la licencia para Windows Phone.

5.4.3.6 Previsiones de cambio

Si se observa el avance de la empresa se puede ver que la mayoría de los errores que se encuentran, son solucionados en un corto espacio de tiempo. Pero también existen errores muy antiguos que no se han arreglado

5.4.3.7 Subjetivo

El planteamiento de utilizar Mono en las distintas plataformas ofrece un acercamiento al desarrollo móvil desde una dirección distinta a la del resto. Mientras que los demás framework se basan en tecnologías web, Xamarin utiliza Mono, .Net y librerías mas propios de desarrollos en programas para plataformas como PC.

Xamarin.Forms es un producto bastante completo, pero tiene un coste de $50 al mes por cada licencia de desarrollador que se desea comprar.

5.4.3.8 Otros

Xamarin, a diferencia del resto de los frameworks estudiados, funciona con C# y tecnología .Net. Esto puede ser muy útil para desarrolladores que vengan desde estas tecnologías, pero puede causar dificultados a otros desarrolladores que hayan especializado con tecnologías web.

5.5 ReactNative

5.5.1. Introducción

ReactNative es un framework realizado por Facebook primero de forma privada para sus productos. Pero posteriormente, decidió publicarlo de forma pública bajo una licencia *BSD License*. La base de la política de la arquitectura de este proyecto provienen de otro proyecto realizado por Facebook llamado React.

Aunque ReactNative se ha utilizado anteriormente en los productos de Facebook. Se publicó de forma abierta el día 27 de abril de 2015. El framework todavía está en una etapa de crecimiento. Actualmente se dispone solo de la versión iOS, la versión de Android esta prevista para el mes de septiembre.

5.5.2. Forma de Trabajo

Tanto React como ReactNative siguen un planteamiento de la creación de componentes de

interfaces de usuario separados, que se relacionan entre si para componer la totalidad de la página web, o en nuestro caso de las páginas de la aplicación móvil. El código se escribe en Javascript y JSX, que es una estructura parecida al XML.

5.5.3. Análisis del framework

5.5.3.1 Calidad de representación y pintado de la interfaz de usuario.

En la actualidad existen ciertos errores de pintado, estos errores son propios de la poca edad del framework. A medida que pasa el tiempo, muchos de estos errores son corregidos.

ReactNative ofrece la posibilidad de usar FlexBox que facilita la forma de colocar los elementos gráficos dentro de la página de la aplicación.

Capacidad de acceso a las API nativas del sistema operativo

Facebook implementa la posibilidad de realizar llamadas a librerías de código nativo desde la plataforma de ReactNative. Esta facilita la implementación de funcionalidades nativas para el framework.

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa	✓		
Calendario			✓
Almacenamiento	✓		
GPS	✓		
Sensores	✓		
Bluetooth	✓		
Redes Sociales	✓		
Push Notification	✓		
NFC			✓
Conexión HTTP	✓		

Tabla 5: API soportadas por ReactNative

Las funcionalidades del mapa están implementados, pero no están completas. No es posible poner marcadores en el mapa.

Al estar, actualmente, disponible el framework sólo para los dispositivos iOS, los cuales no tienen permitido el uso de NFC por los desarrolladores, no se han creado los módulos de NFC para ReactNative.

5.5.3.2 Sistemas operativos con los que trabaja

Actualmente ReactNative solo esta disponible para iOS.

5.5.3.3 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

ReactNative no cuenta con un IDE específico para sus proyectos. Sin embargo sí existen varias herramientas que facilitan el desarrollo. Existe una conexión con el navegador Chrome de nuestro ordenador de desarrollo, permitiendo utilizar las herramientas de desarrollo del navegador, como inspector de elementos o la consola de JavaScript.

Al ser ReactNative una tecnología joven, no cuenta con una documentación muy extensa y la que existe está sujeta a cambios con cada versión.

ReactNative fue bien acogido por la comunidad de desarrolladores tanto de aplicaciones móviles como por los antiguos usuarios de React para aplicaciones web.

5.5.3.4 Licencias y costes

Facebook registro ReactNative bajo la licencia BSD, permitiendo su uso de forma libre para los desarrolladores.

5.5.3.5 Previsiones de cambio

ReactNative es una tecnología muy joven, aunque ya se ha demostrado su funcionamiento en las páginas web, en los teléfonos móviles todavía no tiene una historia. Actualmente se desarrolla activamente, ofreciendo nuevas versiones de forma muy frecuente.

Pero no existe una versión todavía para Android y es muy importante ver como se acomodará la arquitectura de este framework.

5.5.3.6 Subjetivo

Las sensaciones que ofrece este framework son de que se encuentra todavía en sus primeras fases, pero no va a tardar mucho en convertirse en una opción para el desarrollo de aplicaciones móviles.

5.6 NativeScript

5.6.1. Introducción

NativeScript es un producto de la compañía Telerik. Desde hace un tiempo esta empresa está desarrollando un producto que sirva para crear aplicaciones móviles nativas sin el uso de WebKit. Este producto se llama NativeScript y la primera versión pública salió el día 5 de marzo de 2015.

5.6.2. Forma de Trabajo

NativeScript utiliza el lenguaje de JavaScript para implementar la lógica de nuestra aplicación y XML y CSS para el diseño de la interfaz gráfica.

Desde el punto de vista mas técnico, NativeScript es un sistema que aplica empaquetamiento automático a los objetos nativos ya sean Java en Android o Objective C en iOS.

5.6.3. Análisis del framework

5.6.3.1 Calidad de representación y pintado de la interfaz de usuario.

No están implementados una gran cantidad de elementos gráficos necesarios para la realización de una aplicación. Hay que recordar que este framework es muy nuevo y no está todavía terminado. La mayoría de los elementos que faltan están planteados para las siguientes versiones.
Los elementos que están implementados contienen algunos errores comunes a otros frameworks.

5.6.3.2 Capacidad de acceso a las API nativas del sistema operativo

Mientras que NativeScript permite acceder a todos los objetos nativos, esto es muy poco útil si no existe una interfaz que facilite el trabajo con los objetos, juntando las funcionalidades de la plataforma Android con la de iOS.

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa			✓
Calendario			✓
Almacenamiento	✓		
GPS	✓		
Sensores			✓
Bluetooth			✓
Redes Sociales			✓
Push Notification			✓
NFC			✓
Conexión HTTP	✓		

Tabla 6: API soportadas por NativeScript

Es posible guardar los datos en el sistema de ficheros. Pero no existe un módulo para manejar bases de datos, aunque esto se está creando por la comunidad.

Algunas de estas funcionalidades están previstas para las siguientes versiones del framework.

5.6.3.3 Sistemas operativos con los que trabaja

NativeScript puede crear aplicaciones Android e iOS.

5.6.3.4 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Existe un IDE para NativeScript, pero es privado. En el caso de no querer adquirir productos adicionales es necesario utilizar este framework desde la línea de comandos.

La documentación es muy escasa y está mal organizada. La escasez de la documentación se debe a que es un framework muy joven. La documentación que existe actualmente esta muy fraccionada en varias páginas por cada componente.

Este framework cuenta con un gran número de desarrolladores externos, que ayudan con la creación de nuevos módulos, ampliando la documentación y solventando dudas de otros usuarios.

5.6.3.5 Licencias y costes

NativeScript tiene una licencia del tipo Apache versión 2

5.6.3.6 Previsiones de cambio

Al igual que ReactNative, este framework es muy joven y no está completo actualmente. Los desarrolladores anunciaban un sistema mucho más completo que el que se publicó, pero algunos componentes del framework se han retrasado para las siguientes versiones.

5.6.3.7 Subjetivo

A este framework le falta mucho para poder ser utilizado en un proyecto real. Actualmente tan solo es una forma de acceder a los objetos nativos desde JavaScript, pero esto no es suficiente para un framework de este tipo. Si se desea trabajar con los objetos nativos es mas fácil utilizar las formas de desarrollo nativas en Java u Objective C.

5.6.3.8 Otros

Las aplicaciones móviles necesitan un grupo de permisos del sistema operativo para poder realizar ciertas acciones. Pero es necesario compilar la aplicación y posteriormente cambiar manualmente los archivos de configuración

5.7 TabrisJs

5.7.1. Introducción

TabrisJs es uno de los productos desarrollados por la compañía EclipseSource. Uno de sus productos mas conocidos es Yoxos, una distribución de Eclipse.

5.7.2. Forma de Trabajo

Para trabajar con este framework es necesario utilizar JavaScript para desarrollar todos los componentes de la aplicación.

TabrisJs utiliza la librería de Cordova para acceder a los elementos nativos, igual que hace Ionic. Pero TabrisJs se ejecuta en un motor JavaScript y no necesita emplear WebKit.

5.7.3. Análisis del framework

5.7.3.1 Calidad de representación y pintado de la interfaz de usuario.

Existen numerosos errores en la forma de representar los elementos visuales. También

faltan atributos de componentes con los que se podrá modificar su estilo.

5.7.3.2 Capacidad de acceso a las API nativas del sistema operativo

Una de las ventajas que ofrece este framework es la posibilidad de utilizar casi todos los Plugins de Cordova, todos los que no realicen cambios en el DOM.

Esto es posible en teoría, pero en la practica se tiene unas limitaciones que impiden utilizar la mayoría de los Plugins. Esta limitación surge del hecho de que no se puede realizar compilaciones de nuestro proyecto en local y es necesario compilar utilizando sus servidores. La falta de documentación y la nula ayuda por parte de los desarrolladores del framework hace imposible la utilización de la mayoría de Plugins.

API\Implementación	SÍ	Módulo externo de pago	NO
Mapa			✓
Calendario			✓
Almacenamiento			✓
GPS			✓
Sensores	✓		
Bluetooth			✓
Redes Sociales			✓
Push Notification			✓
NFC			✓
Conexión HTTP			✓

Tabla 7: API soportadas por TabrisJs

El único sistema que se ha conseguido utilizar fue el de los sensores por el módulo Device Motion, un Plugin ya incrustado en el proyecto de TabrisJs.

5.7.3.3 Sistemas operativos con los que trabaja

TabrisJs permite crear aplicaciones Android e iOS.

5.7.3.4 Facilidad de uso: IDE, documentación y comunidad de desarrolladores

No existe un IDE particular para este framework, para ejecutar las diferentes acciones es necesario utilizar herramientas adicionales.

Para realizar las pruebas es posible utilizar un servidor en la máquina de desarrollo y una aplicación móvil auxiliar que actúa de cliente y reproduce nuestro proyecto en el dispositivo.

Esta forma de realizar pruebas está limitada, en que no se puede modificar ningún archivo

de la estructura de la aplicación. Es decir si se desea añadir un Plugin a nuestra aplicación se tiene que compilar nuestro proyecto. Si se desea utilizar la versión gratuita del framework, tan solo se puede compilar utilizando sus servidores y almacenando nuestro código en un repositorio publico de GitHub. Esta compilación tarda mucho en realizarse por la necesidad de mover archivos y la tardanza del servidor, para los desarrolladores de proyectos móviles este no es un proceso óptimo.

5.7.3.5 Licencias y costes

TabrisJs es de código abierto, pero el problema está en la forma de compilar los proyectos creados con este framework. Existen tres posibilidades:

a) Gratuito: Solo se puede compilar en sus servidores y empleando un repositorio GitHub publico.
b) $5/mes: Solo se puede compilar en sus servidores pero se pueden emplear repositorios privados de GitHub.
c) $50/mes: Es posible compilar en tu propia máquina local.

5.7.3.6 Previsiones de cambio

Si se analiza el avance de este proyecto, se puede esperar que se realicen avances, pero no existen previsiones de grandes mejoras en el framework.

5.7.3.7 Subjetivo

La opción de utilizar la versión gratuita de este framework es imposible para proyectos privados. Y el framework no muestra suficiente capacidad para invertir dinero en una licencia privada.

Lo mas preocupante de este framework es la ausencia de una comunidad de desarrolladores que lo utilicen, es imposible que los creadores de TabrisJs sean capaces de realizar solos todo el proyecto.

Podrían crearse serios problemas si se hubiera empezado un proyecto real y nos encontráramos en la misma situación de intención de añadir nuevos Plugins y la falta de ayuda por parte del equipo de TabrisJs.

6 Valoración de resultados

En esta sección se intentará resumir todos los datos que se han presentado en el capítulo anterior. Hay que separar los frameworks en los que están ya consolidados de los que son futuras promesas. Las plataformas de Ionic, Qt, Appcelerator y Xamarin se consideran framework que están ya terminados y probados en distintos proyectos reales, mientras que ReactNative, NativeScript y TabrisJs son nuevos proyectos que todavía no están suficientemente completos y contrastados para poder ser utilizados en proyectos destinados a los clientes.

Ionic se caracteriza por la facilidad de uso que ofrece. Esto es gracias a la capacidad de utilizar los avances en las tecnologías web. La principal debilidad de este framework es la necesidad de utilizar el motor de WebKit para funcionar. Este motor puede representar visualmente de forma distinta nuestra aplicación en distintos dispositivos móviles.

Qt es gratuito y cuenta con una gran comunidad de desarrolladores. Existen graves fallos en algunos elementos visuales. No existen las API necesarias para el desarrollo de aplicaciones móviles. No existen perspectivas de crecimiento o de cambio de política de desarrollo.

Appcelerator está en un proceso de cambio que no obligará a pagar una cuota mensual. Tiene algunos fallos visuales. Este framework no da una sensación de seguridad de completitud de sus componentes. Un proyecto se puede complicar si se intenta crear un nuevo elemento.

Xamarin funciona muy bien visualmente. Ofrece un conjunto de API muy completas. Mucha documentación y es posible encontrar ayuda en caso de necesitarlo. Es necesario pagar $50 al mes por cada desarrollador.

ReactNative es muy joven y solo funciona para iOS. No esta completado y contiene varios fallos visuales, además, faltan algunas API todavía. Es desarrollado por Facebook y sigue los pasos de React, un framework consolidado. Tiene una amplia comunidad desarrollándolo.

NativeScript es todavía muy joven. Contiene pocos elementos visuales. Este framework, actualmente, es un empaquetador de API nativas.

En TabrisJs existen fallos visuales. En la versión gratuita es necesario compilar en sus servidores. No existe ninguna comunidad.

6.1 Valoración final

La elección de utilización de un framework depende del tipo de proyecto que se vaya a realizar.
Los framework promesa no están terminados suficientemente para poder se empleados en un proyecto real.

El framework que ofrece un mejor conjunto de propiedades es Xamarin, pero este es un producto de pago, por lo tanto es necesario pagar una cuota mensualmente. Esto provoca que se recomiende el empleo de Xamarin para proyectos grandes o para un conjunto de muchos proyectos pequeños que se encuentren temporalmente seguidos. De esta forma se compensa los inconvenientes del precio y de la necesidad de aprendizaje de C# y tecnologías .Net en caso de no conocerlos.

Para proyectos pequeños separados en el tiempo, la mejor opción es utilizar Ionic por la facilidad de desarrollo, empleando tecnologías web. Los posibles defectos que pueden surgir del motor WebKit se pueden localizar y solventar en aplicaciones no complejas.

No hay que olvidar que este estudio es una instantánea en el tiempo y a medida que pase el tiempo los distintos framework se desarrollaran y cambiarán. Por ello hay que observar periódicamente su situaciones.

Los frameworks promesa se irán desarrollando y dependiendo como avancen pueden ser una posible alternativa a las elecciones realizadas anteriormente. El framework que a día de hoy presenta la mayor posibilidad de avance es ReactNative.

Además, los framework que están completados también avanzarán, Appcelerator tiene planeado añadir nuevas características que pueden resultar muy interesantes en el desarrollo de aplicaciones móviles.

Por último hay que mencionar el proyecto de Crosswalk, que intenta empaquetar un motor especifico de WebKit en nuestras aplicaciones de Ionic. Ahora mismo no es una opción factible por la cantidad de memoria extra que añade a nuestras aplicaciones. Pero si en el futuro, los desarrolladores son capaces de reducir esta cantidad de memoria añadida, esta opción puede ser la mejor forma de desarrollo de aplicaciones móviles.

7 Aplicación final

Por último se desea acabar este estudio realizando una aplicación final. Esta aplicación será una publicación de avisos y noticias de comunidades cerradas.

La aplicación final se ha realizado en todas las plataformas hasta donde las capacidades de estas lo permitían.

Se ha aplicado un mayor esfuerzo en el framework de Ionic, ya que es nuestro framework de referencia y, por consiguiente, la aplicación realizada con Ionic es la aplicación de referencia.

La aplicación constará de una ventana de entrada donde se le pedirá los datos al usuario.

Figura 5: Pantalla inicial de la aplicación

Después de autenticarse, se verá un conjunto de noticias en una lista "infinita", se le llama así por que la lista empieza con un grupo de elementos finitos y cuando el usuario hace *scroll* hacia abajo, se le pide más objetos al servidor y se añaden a la lista. De esta forma parece que la lista no tiene fin. Desde el punto de vista de rendimiento, la aplicación no hace una petición de todos los datos, sino que los va pidiendo cuando los necesita. Al presionar una noticia, se abre una ventana con la noticia extendida.

Figura 6: Pantalla con el listado de noticias

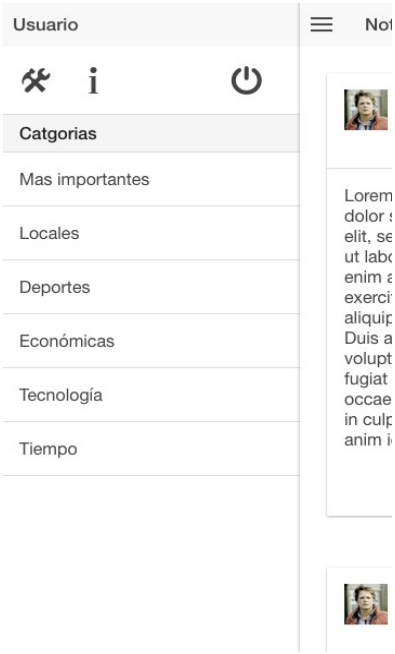

Figura 7: Pantalla con el menú lateral desplegado

La aplicación utiliza un menú lateral para permitirle al usuario navegar entre diferentes páginas.
También se ha añadido una página para poder cambiar los datos personales generales y otra para cambiar los datos de la contraseña.

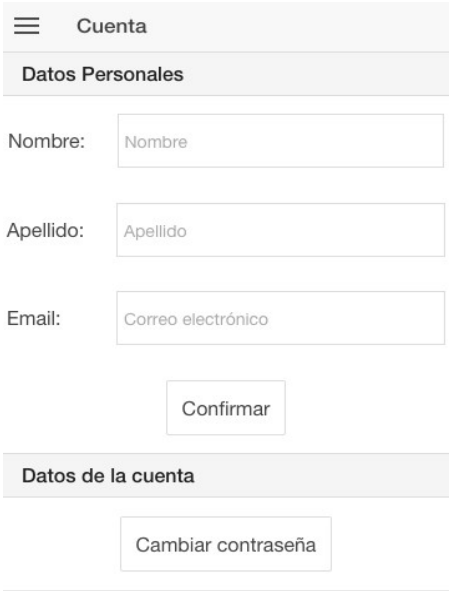

Figura 8: Pantalla donde se pueden modificar los datos personales

Todos los datos necesarios, como el listado de noticias, los artículos y los datos de usuario, han sido recibidos desde un servidor. Por lo que los datos de la aplicación se actualizan correctamente.
Posteriormente, se han utilizado los distintos frameworks para poder apreciar las diferencias que surgen frente a la aplicación de referencia de Ionic.

Qt presentaba un conjunto de dificultades para la realización de la aplicación. Esto es debido a que Qt no ofrece componentes propias de las aplicaciones móviles como la cabecera con el título y conjunto de botones que varían al avanzar la página. También existen problemas con los tipos de teclado que aparecen al presionar un elemento de entrada de datos en la aplicación.

Appcelerator nos dificultaba mucho la creación del menú lateral. Este menú no lo tiene implementado como un elemento del framework, por lo tanto es necesario emplear librerías y código adicional. Los elementos visuales no reaccionaban correctamente al abrir y cerrar el menú. También se intentó emplear una librería externa, pero todas los módulos existentes no funcionan correctamente con las nuevas versiones del framework.

Xamarin tenía una dificultad de entrada, que es el uso de la tecnología .Net para la realización de aplicaciones móviles que usaban datos JSON.
Pero la principal dificultad es la necesidad de crear noticias con tamaño variable. Algunas noticias tienen varias imágenes mientras que otras pueden tener un fragmento corto de texto. Xamarin tenía un problema en iOS para poder dar valor de altura de forma particular a cada elemento. Tan sólo permite dar un único valor de altura a todos los elementos.

ReactNative ofrecía una nueva arquitectura de realización de aplicaciones, a través de componentes que utilizan otros componentes. Después de aprender esta tecnología, es relativamente fácil implementar la aplicación deseada. Pero existe un error en la actualidad, al salir el teclado en la pantalla, oculta elementos visuales que están situados en la zona donde aparece el teclado.

NativeScript no está suficientemente desarrollado para poder implementar la totalidad de la aplicación de prueba. Pero se ha podido realizar una parte de esta.

En TabrisJs se ha podido implementar algunas partes de la aplicación. Pero se han encontrado serias dificultades al intentar incluir los Plugins deseados.

8 Trabajos futuros

El análisis realizado, se ha llevado a cabo con unos frameworks determinados, de los que se ha esperado que ofrezcan los resultados más óptimos, pero es posible ampliar este estudio a otros frameworks.

Este proyecto presenta un estudio de los distintos frameworks que están disponibles en la actualidad para el desarrollo de las aplicaciones móviles. Este mismo estudio es posible volver a realizarlo en el futuro obteniendo resultados distintos. Esta reiteración sería muy conveniente para poder observar como han avanzado los distintos framework y posiblemente cambiar las recomendaciones que se han realizado en este documento.

9 Bibliografía

Microsoft News Center. (2015). Microsoft empowers Windows, iOS, Android, Mac and Linux developers to reach billions of new customers. Recuperado 31 de mayo de 2015 de http://news.microsoft.com/2015/04/29/microsoft-empowers-windows-ios-android-mac-and-linux-developers-to-reach-billions-of-new-customers/.

Cockburn, A. (2008 mayo). Using Both Incremental and Iterative Development. Crosstalk, Vol 21. No 5. pp. 27-30

Jacobson, I., Booch, G., Rumbaugh, J. (1999). The Unified Software Development Process. Addyson-Wesley

Sommerville, I., (1989). *Ingeniería del software*. Addison-Wesley

Bagnardi, F., Beebe, J., Feldman, R., Hallett, T., Højberg, S., Mikkelsen, K. Developing a React Edge. Bleeding Edge Press

Apéndice A: Análisis detallado de los resultados

A continuación se mostrarán los resultados detallados obtenidos del estudio. Los datos están separados por framework y categoría analizada.

Ionic

Introducción

Se tiene que empezar nuestro estudio analizando este framework. Ionic es una tecnología muy popular que se basa en el uso de WebKit en los dispositivos móviles. Se puede decir que, esta es la tecnología que deseamos sustituir y por ello tenemos que empezar analizando cuales son sus características positivas y debilidades.

Drifty es una empresa que se fundó en el año 2012 por Ben Sperry y Max Lynch. Realizaron un grupo de herramientas que funcionaban con jQuerry y posteriormente decidieron comenzar el proyecto de Ionic basado en Angular. La versión beta de Ionic se publico en noviembre de 2013 y desde ese momento fue bien recibida por la comunidad de desarrolladores que buscaban realizar aplicaciones móviles híbridas. Se estima que se han creado un total de unas 500,000 aplicaciones con esta tecnología.

Durante el último año la compañía ha recaudado dinero de sus inversores, ya que la compañía ofrece Ionic como código abierto y no vende actualmente ningún producto o servicio, pero la compañía tiene pensado la creación de herramientas y productos privados con los que planea ganar dinero.

Forma de Trabajo

Apache Cordova es una tecnología que se basa en un conjunto de API de JavaScript que permiten acceder a las funcionalidades nativas de nuestros teléfonos móviles. Esto da la posibilidad de crear aplicaciones web que utilicen HTML, CSS y JavaScript, ejecutar la aplicación en el WebKit de nuestro dispositivo y acceder a las funcionalidades nativas a través de Cordova.

Por lo tanto tenemos la posibilidad de crear páginas web con las capacidades que tienen las aplicaciones nativas, además, se ejecutan en el WebKit, que es el motor de nuestro navegador web del móvil.

Posteriormente Ionic facilitó las tareas de creación de aplicaciones, ofreciendo un conjunto de librerías de componentes HTML, CSS y JavaScript. Ionic intenta optimizar las operaciones propias de las páginas web a las plataformas móviles. Además, Ionic emplea Angular como base para todos sus componentes.

```
      <ion-tab title="Home" icon="ion-home"
href="#/tab/home">
        <ion-nav-view name="home-tab"></ion-nav-view>
      </ion-tab>

      <ion-tab title="About" icon="ion-ios-football"
href="#/tab/about">
        <ion-nav-view name="about-tab"></ion-nav-view>
      </ion-tab>
```

Por lo tanto para trabajar con este framework tenemos que crear plantillas en HTML, estas plantillas serán la representación visual de las páginas. Se emplea CSS de la misma forma que en las páginas web, para dar el formato deseado a los elementos HTML. Los controladores se realizan en JavaScript y contienen la lógica de la aplicación.

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

El principal problema que tiene Ionic es que utiliza el WebKit, Trident en la plataforma Windows Phone. Por lo tanto existe un motor de pintado distinto para cada versión de WebKit o Trident, sobretodo en dispositivos Android que existe una versión de WebKit distinta para cada versión de Android. Esto provoca que existan diferencias de pintado entre distintos dispositivos.

Este error tiene una alta importancia por dos factores. Es muy difícil encontrar fallos de pintado ya que es imposible probar con todos los dispositivos existentes en el mercado y suele ser el usuario final el que encuentra el error. Por otra parte, después de encontrar el error, es muy difícil corregirlo, esto se debe a que es necesario aplicar unos cambios muy específicos al código de nuestra aplicación que modifiquen el comportamiento de una versión muy particular del motor de pintado, pero no afecte al resto de las versiones, donde ya funcionaba correctamente la aplicación. Además, se complica el mantenimiento posterior.

Al darse nuevas situaciones en las aplicaciones móviles, a las que las páginas web no estaban diseñadas, surgen un nuevo grupo de errores. Uno de ellos es el hecho de que la cabecera de la aplicación tan solo es un elemento gráfico en este framework y por ello no tiene las mismas cualidades que el mismo componente nativo.

Esto provoca que no se oculte correctamente el cursor que espera la introducción de datos. Al seleccionar un campo de introducción de datos aparece una barra vertical que parpadea, pero este cursor no debería de tener mas prioridad de pintado que la cabecera.

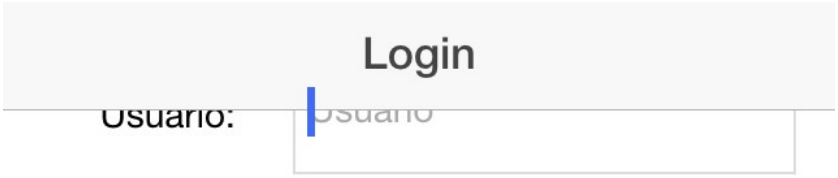

Figura 9: Error del puntero de texto en Ionic en iOS

El framework de Ionic también incorpora en el teclado unas flechas para poder navegar entre los distintos elementos de introducción de datos. Al usar estas flechas la pantalla se mueve para centrarse en el siguiente elemento. Pero la cabecera no debería realizar ningún cambio, ya que no se espera que cambie de posición y se mantenga constante en la parte superior de la página. Los cambios de posición también se aplican a la cabecera que aumenta o disminuye su tamaño.

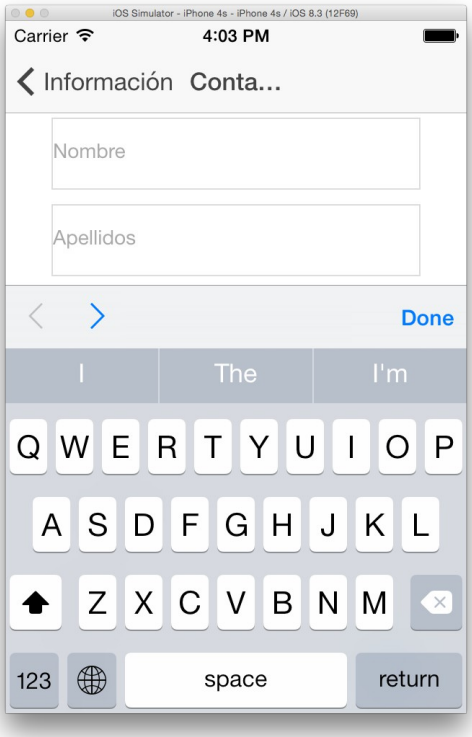

Figura 10: Aplicación con la cabecera en la posición correcta de iOS

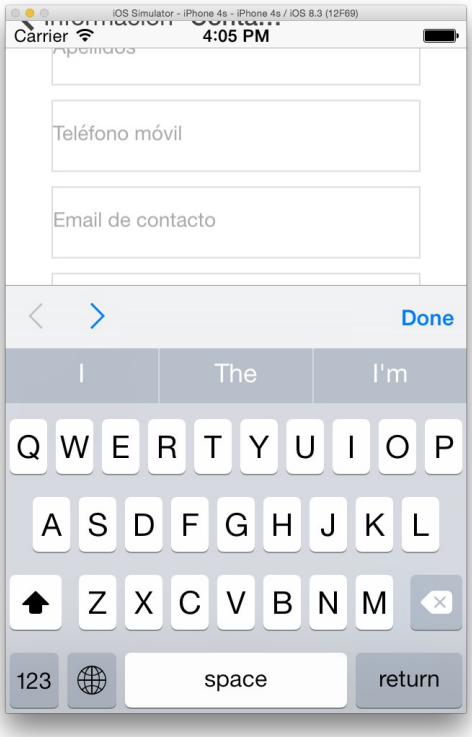

Figura 11: Error de la posición de la cabecera de Ionic en iOS

Una de las mejores facultades que ofrece Ionic es la capacidad de emplear los elementos de las páginas web. En la actualidad existe un gran número de estilos CSS predefinidos que podemos emplear directamente en nuestras aplicaciones. Además, es posible utilizar Angular para realizar cualquier componente que necesitemos.

Sistemas operativos con los que trabaja

Al ser ejecutado en el motor del navegador web, un proyecto de Ionic puede ser ejecutado en la mayoría de las plataformas que existen, tanto móviles como otras mas clásicas como PC o Mac. Ionic funciona en Android, iOS y Windows Phone.

Capacidad de acceso a las API nativas del sistema operativo

Como ya se ha comentado antes, Ionic utiliza Cordova para poder acceder a las API nativas y poder emplear las funcionalidades nativas desde el motor de WebKit.

Para la creación de mapas, no se suele utilizar los componentes nativos, ya que es mas fácil emplear el WebKit para ejecutar la librería JavaScript de Google Maps. Esto pintará un mapa de la misma forma que lo pintaría en el navegador web, este mapa ofrece soporte para las acciones propias de las pantallas táctiles de los teléfonos móviles

Existen Plugins para acceder a las funcionalidades de calendario. De la misma forma es posible utilizar el sistema de almacenamiento de datos en fichero y almacenamiento en bases de datos.

Es posible utilizar el sistema de GPS, pero no ofrece la posibilidad de determinar la precisión que queremos emplear para la localización. Podemos utilizar el módulo de Cordova Motion para acceder a los sensores del teléfono móvil.

También existen Plugins para utilizar las comunicaciones a través de Bluetooth y emplear los servicios de Push Notification. Existe una librería para emplear las API nativas de las redes sociales.

NFC está implementado para los dispositivos Android.

Ionic da la posibilidad de realizar peticiones a un servidor web para recibir datos desde este.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Ionic no ofrece un IDE propio pero es normal el uso de IDE para el desarrollo web como NetBeans, esto se debe a que NetBeans ofrece un conjunto de funcionalidades para proyectos de Cordova, como la opción de compilar y ejecutar aplicaciones. Las acciones propias de Ionic como la creación de proyectos, gestión de Plugins y la compilación se realizan desde la consola de comandos.

Al ser Ionic un framework que emplea tecnología web, es posible ejecutar sus proyectos en cualquier navegador web, incluidos los de la máquina de desarrollo. Esto ofrece la posibilidad de trabajar directamente en nuestro navegador sin la necesidad de tener un emulador o un dispositivo físico. Además, Chrome tiene la opción de emular el tamaño de la ventana de varios dispositivos móviles.

La documentación que se encuentra de Ionic suele ser bastante amplia y organizada por tipos de elementos. Además, existen numerosos ejemplos visuales, al ser Ionic un conjunto

de operaciones web, estos ejemplos visuales suelen ser pequeñas aplicaciones que se muestran en nuestro navegador. Esto es muy útil para poder observar como funcionan los componentes.

Ionic ha sido muy bien recibido por los desarrolladores, sobretodo por los desarrolladores web, que han visto la posibilidad de crear aplicaciones móviles empleando la misma tecnología con la que ya trabajan. La comunidad de desarrolladores es muy amplia y existe una gran cantidad de ejemplos fuera de la página oficial. También se participa en el desarrollo de Ionic ya sea con el framework principal o con la creación de Plugins adicionales.

Licencias y costes

Ionic es gratuito en la actualidad. La empresa Drifty no ofrece ningún otro producto. Pero están desarrollando un sistema de actualización de aplicaciones que no implique la necesidad de ser revisado por Apple o Google. Esto es posible por el hecho de que los desarrolladores tan solo suelen modificar archivos HTML o JavaScript, que se podrían actualizar en la aplicación sin la necesidad de realizar cambios en el motor de la aplicación.

Previsiones de cambio

Se espera que Ionic avance en el desarrollo de las aplicaciones híbridas actualizando sus componente, creando nuevos elementos y permitiendo nuevas funcionalidades que permitan a Ionic comportarse como una aplicación nativa y no como una aplicación externa.
Pero es muy probable que Ionic siga utilizando un WebKit que provocará los errores que se han comentado anteriormente.
Actualmente existen otro proyecto, llamado Crosswalk que intenta crear una versión especifica de WebKit lo suficientemente compacta para poder ser instalada como una aplicación auxiliar en los dispositivos móviles. Este WebKit se podrá actualizar posteriormente las siguientes versiones para unificar todos los dispositivos a un solo motor web. Pero esto es imposible en los dispositivos iOS y Windows por las políticas que imponen las compañías propietarias de los sistemas operativos. Crosswalk añade unos 15 MB al tamaño del instalador (.apk) y unos 70 MB al tamaño de la aplicación instalada.

Subjetivo

El desarrollo con Ionic fue muy fácil y rápido, daba la sensación de completad de funcionalidades que se desean de este framework. Pero existe un problema insalvable en la actualidad, este problema es el WebKit, sobre todo causado por la excesiva fragmentación de Android. Por lo tanto se desea encontrar un framework que ofrezca las mismas cualidades sin usar un motor WebKit.

Otros

Al emplearse un WebKit, la aplicación necesita una gran cantidad de recursos del sistema operativo y las operaciones dentro de la aplicación son lentas. Estos problemas no son muy

importantes ya que las aplicaciones móviles suelen tener la totalidad de los recursos disponibles, solo hay una aplicación ejecutándose en el teléfono móvil.
La aplicación compilada solo necesita guardar los archivos HTML, CSS y JavaScript, gracias a esto la aplicaciones tienen un tamaño muy reducido.

Qt

Introducción

Qt es un framework multiplataforma utilizado en una gran variedad de entornos de software y hardware muy variable. Fue utilizado para la realización de una gran cantidad de aplicaciones como KDE, libLastFm, Ubuntu Touch y Wireshark.

Inicialmente Qt fue creado por Trolltech y adquirido posteriormente por Nokia para ser empleado en Symbian, el sistema operativo para dispositivos móviles. Pero tras la decisión de terminar el desarrollo de Symbian, Nokia decide vender parte de los derechos de la plataforma QT a Digia.
Actualmente Qt es desarrollada por la compañía Digia y por la comunidad de desarrolladores, por lo que existe actualmente dos versiones con una licencia pública y otra privada. Aunque la versión privada se centra en ofrecer servicio técnico y Qt Cloud Services (Almacenamiento de datos de la aplicación en la nube).

Forma de Trabajo

La librería de Qt esta compuesta de distintos módulos que ofrecen una API idéntica para las distintas plataformas aunque posteriormente se compilen de distinta manera. El acceso a esas API se puede realizar desde código escrito en C++ o desde QML. La posibilidad del uso de QML es ofrecida por el módulo QT Quick, permitiendo además, enlazar código de QML con C++.

```
    Image {
        anchors.left: parent.left
        anchors.bottom: parent.bottom
        anchors.margins: 10
        source: "content/arrow.png"
        rotation: -90
        opacity: clockview.atXBeginning ? 0 : 0.5
        Behavior on opacity { NumberAnimation
{ duration: 500 } }
    }
```

QML es un lenguaje de programación creado para QT que se basa en JavaScript, e incluso permite insertar código JavaScript dentro de QML. Para la realización de pruebas del framework de QT se va a emplear QML ya que ofrece una forma rápida de creación de

interfaces que es la parte principal de aplicaciones móviles.

A diferencia de otros frameworks, QT no utiliza llamadas al sistema operativo para hacer la funcionalidad de pintado en pantalla, en vez de ellos emplea directamente el motor gráfico para dibujar los elementos de la interfaz de usuario. Esto da garantías de que sea la misma interfaz gráfica en los diferentes dispositivos, pero si no se integra suficientemente, es un problema

Qt, además, cuenta con un IDE, llamado QT Creator, diseñado para facilitar el desarrollo de aplicaciones con su framework.

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

El proyecto de Qt se basa en la creación de componentes nativos utilizando el motor gráfico, sin realizar llamadas al sistema operativo. Esto implica que dependiendo de la plataforma en la que se use Qt, se pintará cada elemento tal como considera el framework que se deba pintar, el problema surge cuando estos no se integran adecuadamente con el entorno donde se ejecuta, cosa que se pudo apreciar en las pruebas con los dispositivos móviles, por lo que existen componentes que tienen un diseño incorrecto ya que el sistema operativo lo pintaría de manera distinta.

Este problema hace que la interfaz gráfica no de una sensación nativa en alguno de sus componentes. Algunos componentes están correctamente implementados como el Picker. Pero otros no se sienten nativos como el Spinner.

Figura 12: Error de elemento Spinner de Qt en iOS

Existe el componente gráfico de calendario con el que se puede seleccionar un día o un conjunto de días para posteriormente realizar un grupo de acciones.

Capacidad de acceso a las API nativas del sistema operativo

Qt es un proyecto grande, que esta pensado para una gran cantidad de plataformas. Pero los avances y modificaciones están pensados para el proyecto entero. Esto implica que no se desarrollen funcionalidades especificas para cada plataforma, ya que no son necesitados en todas las plataformas. Los avances que se realizan en la parte de plataformas de teléfonos móviles de Qt suele ser enfocados a solucionar problemas y a implementar funcionalidades del proyecto general de Qt. Pero no se realizan avances específicos para nuestra

plataforma.

Este hecho implica que las API implementadas son las mismas que se implementa en los ordenadores de sobremesa como PC o Mac. No existe las funcionalidades de creación de mapas nativas, ya que no existe esta necesidad en las plataformas tradicionales. Para poder implementar un mapa de Google Map es necesario utilizar un WebView que necesita iniciar todo el complejo de sistemas para interpretar funcionalidades web. Este proceso es el que se desea evitar acudiendo a los frameworks que trabajan con componentes nativas como Qt.

Tampoco existe la facilidad de interactuar con el calendario del sistema operativo. El sistema GPS está implementado, pero no nos ofrece todas las funcionalidades como definir la precisión de localización. Existe la posibilidad de acceder a los sensores del dispositivo.

El sistema de almacenamiento se puede implementar con la interfaz de base de datos. Pero surgen problemas al intentar utilizar un sistema de ficheros. Recordemos que estamos utilizando QML que se basa en el lenguaje de JavaScript, este lenguaje no contiene funciones para manejar archivos ya que no es necesario para entornos web. Los desarrolladores de Qt decidieron mantener esta restricción y dejar QML sin funcionalidades para manejar ficheros. Para poder realizar estas operaciones es necesario programar un conjunto de funciones en C++ que realicen las gestiones de los ficheros y posteriormente llamar a estas funciones desde QML.

Las comunicación mediante Bluetooth no es posible en dispositivos iOS, mientras que las funcionalidades de NFC están implementadas solo para BlackBerry 10.

No es posible utilizar Push Notification en ninguna de las plataformas, aunque se pueden adquirir productos realizados por otros desarrolladores que cumplen estas funcionalidades. También es necesario comprar productos adicionales si se desea utilizar las API de las redes sociales en nuestras aplicaciones. Qt ha implementado las directivas aplicadas a las páginas web para las funcionalidades de emisión y recepción de datos a través de la red. Estas directrices principalmente, limitan el acceso que se tiene a las *cookies*

Por último mencionar que Qt no cuenta con un sistema de módulos que nos facilite la creación de componentes básicos. Por ejemplo no existe un componente que represente la cabecera de la aplicación y nos facilite el uso de botones en ella. Esto provoca la necesidad de realizar librerías adicionales para poder realizar después nuestras aplicaciones

Sistemas operativos con los que trabaja

Qt trabaja con una larga lista de plataformas y podemos compilar nuestro código a los dispositivos Android, iOS, Windows Phone y a otros como PC y Mac.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Qt posee un IDE propio llamado Qt Creator. Este IDE nos ofrece una gran cantidad de ayuda para la realización de nuestros proyectos en esta plataforma. Este entorno nos ofrece una gran cantidad de ejemplos y nos facilita la tarea de compilar nuestro código para

distintas plataformas.

Una de las funcionalidades que nos ofrece es la creación de nuestra interfaz mediante una representación visual en la que se puede crear, mover y modificar los componentes. Este diseñador no esta completamente implementado y suele dar errores con componentes relativamente complejos.

Tenemos hay mencionar que surgieron problemas al compilar para la plataforma iOS ya que no se eliminaba archivos de la compilación anterior, provocando errores. Para ello era necesario limpiar los ficheros antes de compilar.

Qt Creator se cerraba con un mensaje de error en algunas ocasiones, pero dado el desarrollo de este IDE, se considera un error puntual relativo a la versión que se espera que se solucione pronto.

Qt cuenta con una documentación extensa sobre todos sus componentes con toda la información necesaria para su uso. Además, existen numerosos ejemplos para todos los elementos de esta tecnología. Todos las paginas de información de los componentes suelen contener imágenes que muestran como funcionan. Esta documentación, además, esta bien estructurada permitiendo reducir el tiempo de búsqueda.

Al ser Qt un proyecto con mucha historia, cuenta con una gran cantidad de desarrolladores que suelen avanzar en el proyecto creando sus propios elementos. Además, es posible encontrar información y ejemplos sobre este proyecto en numerosos sitios web.

Licencias y costes

Qt cuenta con una licencia abierta de tipo LGPL. Esto nos permite crear libremente aplicaciones con esta tecnología. Adicionalmente la empresa ofrece soporte técnico y módulos privados en el caso de desear adquirir productos adicionales.

Previsiones de cambio

Qt cuenta con una larga historia de desarrollo en la que ha avanzado hacia un proyecto muy completo. Pero Qt no es exactamente el proyecto que nos preocupa, a nosotros nos interesa la parte móvil de Qt. Para estas plataformas están previstas un grupo de corrección de errores en las siguientes versiones de Qt.
A partir de estas correcciones de errores no se espera que se realicen avances rápidos para nuestras plataformas ya que el proyecto de Qt se centra en otros aspectos.

Subjetivo

Qt no da la sensación de poder ser un framework para nuestros propósitos de creación de aplicaciones móviles de forma rápida y cómoda. Esto se debe a la poca implementación de los módulos de funcionalidades básicas para los teléfonos móviles como son los mapas, Push Notification o las redes sociales.
Parece que Qt es un proyecto para plataformas de sobremesa que también se puede ejecutar

en teléfonos móviles. Pero este planteamiento no es el que deseamos para nuestro framework.

Otros

Los proyectos creados en esta plataforma suelen presentar un tamaño bastante grande. Este es un dato importante ya que los dispositivos móviles suelen contar con poco espacio libre disponible.

Appcelerator

Introducción

La compañía Appcelerator surge en 2006 por Jeff Haynie y Nolan Wright. Desde entonces sigue un crecimiento avanzado recolectando una amplia financiación y sus productos son empleada por un gran número de usuarios y compañías.

En 2012 se conoció una noticia publicada por uno de los desarrolladores que usaba este framework, en el que informaba que la empresa de Appcelerator le exigía incorrectamente el pago de la licencia a él y a sus clientes. Aunque posteriormente al hacerse este problema público, se cambiaron las políticas de la empresa y se reestructuraron las licencias de los diferentes productos. Siguió existiendo una preocupación en la comunidad de desarrolladores por no encontrarse en una situación similar.

El día 31 de marzo la compañía anunció que a partir de 1 de junio se realizará un cambio en la estructura de sus productos. Mantendrán el SDK como código abierto pero limitarán el resto de las herramientas relacionadas con él, como las versiones compiladas, el IDE (Titanium Studio) y algunos módulos que amplían las funcionalidades del framework. Es decir a partir del día 1 de junio es necesario pagar una cuota para seguir utilizando estas herramientas.

Forma de Trabajo

La compañía de Appcelerator se ha centrado en mantener un patrón de arquitectura de Modelo-Vista-Controlador como la estructura básica de las aplicaciones que se realizan mediante este framework.
Para manejar los controladores se emplea archivos JavaScript, la vista se compone de archivos XML y Tss, y no existe una limitación para manejar los modelos.

```
</Alloy>
  <Window>
     <Button id='loginButton'
        onClick="clickedLogin"
```

```
        title="Entrar"
        top="120" width="130"
        class="mainButton" ></Button>
  </Window>
</Alloy>
```

Los archivos XML contienen una representación la estructura de los objetos que verá el usuario y con los que podrá interaccionar, estas interacciones son manejadas en el controlador.

```
".mainButton": {
  backgroundColor: '#c3c3c3',
  width: Ti.UI.SIZE
}
```

Appcelerator permite el uso de hojas de estilo, llamadas TSS, parecido a CSS de las aplicaciones web, pero con varias diferencias. TSS es un archivo de tipo JSON en el que se definen para un conjunto de objetos (seleccionados por ID o por clase de objeto) del documento los valores de los atributos que tomarán. Estos cambios no se propagan a los hijos de los objetos.

Tanto los archivos XML como TSS, son posteriormente compilados a código JavaScript para posteriormente ser ejecutado en el dispositivo móvil.

Appcelerator se basa en utilizar un motor de JavaScript en el dispositivo móvil para acceder a las API nativas desde él. Ofreciendo acceso a las funcionalidades nativas del sistema operativo.
Por lo tanto los desarrolladores de aplicaciones crean productos que son ejecutados en el motor Javascript sin la necesidad de utilizar un WebKit. También es necesario mencionar que las funciones de pintado se realizan mediante peticiones nativas al sistema operativo, sin implicar el WebKit y sin realizar llamadas al motor gráfico.

Tal como se ha mencionado anteriormente, Appcelerator posee un IDE llamado Titanium Studio que desciende del IDE de Eclipse.

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

Nos hemos encontrado con varias diferencias en la forma de representación gráfica de los elementos. Ya que mientras que para la versión Android se dejaba un margen después del texto, en la versión iOS no existía este margen. Esto puede ocasionar serios problemas, ya que el cliente puede preferir homogeneizar ambas versiones, causando la necesidad de modificar el código para uno de los dispositivos dificultando el manejo y el mantenimiento de este código.

Existe una diferencia de pintado de los colores, es decir, un mismo código de color se pinta de forma distinta en los distintos sistemas operativos. Este error es crítico en determinadas situaciones, por ejemplo si nuestro cliente desea utilizar un color especifico, el color asociado a su marca. Este fallo nos impediría poder especificar con precisión el color con el que se pintará la interfaz de la aplicación

Figura 13: Diferencia de colores en Appcelerator. iOS (izquierda), Android (derecha)

Otro problema con el que nos encontramos es el hecho de que normalmente deseamos darle estilo a nuestro texto, como por ejemplo ponerlo en negrita, cursiva o subrayarlo. El framework de Appcelerator nos permite realizar estas operaciones utilizando un subconjunto de formato HTML en la plataforma Android. Pero no existe la misma facilidad en la versión iOS, que ofrece unas opciones limitadas utilizando una técnica de concatenación. Existe una librería externa, creada por un grupo de usuarios, que ofrece una solución conjunta a este problema. Pero esta librería esta limitada en sus funcionalidades y comete algunos errores, como por ejemplo pinta en un tamaño inferior el texto en negrita

Figura 14: Error en el texto en negrita en iOS

También existe un problema con la forma que tiene el framework de representar el elemento Picker. Este framework utiliza una representación gráfica diferente a la que usa el sistema operativo nativo.

Figura 15: Picker no nativo en Android

No existe un elemento gráfico que represente un calendario. Sin embargo existe un reproductor de vídeo.

Capacidad de acceso a las API nativas del sistema operativo

Appcelerator implementa las API de Mapas, pintando de forma nativa los mapas con Google Map en Android y Apple Map en iOS. También es posible gestionar los eventos del calendario del dispositivo con este framework.
Se puede guardar los archivos en el dispositivo utilizando el sistema de archivos y la base de datos. Funciona correctamente el acceso al GPS y a otros sensores como el acelerómetro. También es posible utilizar las funcionalidades de Bluetooth si se adquiere un módulo adicional. Esto mismo pasa con Push Notification, que requiere la compra de un módulo.
Es posible implementar funcionalidades de las redes sociales ya que permite interaccionar con el sistema operativo para hacer estas funciones. El sistema de NFC funciona correctamente en Android, esta funcionalidad no es posible usar en iOS por limitaciones de sus dispositivos.

Appcelerator nos permite enviar y recibir datos a través de la red, incluso existe un modulo privado para trabajar con autocertificados.

Por último mencionar que, es difícil implementar componentes externos. Todos los componentes del framework reaccionan de manera inesperada fuera de las condiciones normales. Esto se debe a que los desarrolladores del framework solo nos aseguran la forma con la que reaccionaran los componentes en condiciones normales, pero la creación de módulos externos suele implicar la necesidad de sobrepasar las condiciones normales.

Sistemas operativos con los que trabaja

En la actualidad es posible utilizar Appcelerator para compilar a Android e iOS. Se esta desarrollando la versión para trabajar con Windows Phone, en este momento existen versiones en fase de prueba.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

La empresa de Appcelerator nos ofrece el uso de sus IDE Titanium Studio que se basa en Eclipse. Este entorno de desarrollo, esta preparado para poder compilar y ejecutar nuestro proyecto en el emulador o en un dispositivo físico desde el IDE. Ofrece un conjunto de teclas para simplificar tareas de gestión en los proyectos de este framework. Existe un error en el que la compilación iniciada desde Titanium Studio falla, pero tan solo es necesario volver a iniciar la compilación.

Appcelerator usa un sistema de documentación en el que existe una gran cantidad de contenidos pero se encuentra de forma desorganizada. También se desearía tener mas imágenes que muestren los elementos gráficos que se mencionan en la documentación.

Existe una gran comunidad de desarrolladores que crean sus propias librerías e intentan ayudar con las consultas de los usuarios. También es fácil encontrar documentación adicional fuera de la página oficial.

Licencias y costes

Actualmente Appcelerator es de código abierto con unos complementos, como módulos adicionales o análisis estadístico de pago. Pero la nueva política de la empresa pretende mantener el código principal abierto, mientras que la versión compilada se convierte en privada. Además, el IDE pasa de tener una licencia privada pero gratuita a ser un elemento de pago.

A partir del día 1 de junio es necesario el pago mínimo de $39 al mes por cada desarrollador. Esta licencia también permite el uso del IDE y de análisis estadísticos de uso de la aplicación.

Previsiones de cambio

Como se ha mencionado anteriormente, este framework se encuentra actualmente en un proceso de modificación, con el que pretende aumentar el coste del producto, ofreciendo una mayor cantidad de funcionalidades.
Estos cambios pueden traer avances a la tecnología empleada en este framework. Pero es vital arreglar todos los fallo expuestos anteriormente, para que las aplicaciones creadas con Appcelerator posean unos acabados correctos y actualmente no se han visto avances ni se ha anunciado la voluntad de los desarrolladores de centrarse en estos problemas.

Subjetivo

Este framework no se encuentra completo, le falta terminar los componentes y corregir errores que se encuentran actualmente en él. Pero la compañía ha optado por intentar ampliar sus funcionalidades añadiendo mas módulos. Esto ha provocado que en la actualidad es solo posible emplear aquellas rutas de desarrollo que estén creados por los desarrolladores del framework, ya que el resto de la tecnología no esta completa ni documentada, además, esta sujeta a cambios que se pueden realizar en versiones futuras del framework.

Otros

Este framework utiliza Node para su funcionamiento. Pero las últimas versiones de Node no funcionan correctamente con Appcelerator, por lo tanto es necesario mantener una versión antigua de Node.

La compañía Appcelerator ha creado un lugar donde se puede comprar y vender módulos para su framework. Pero esta zona se encuentra con un número bajo de productos. Algunos productos no son compatibles con el framework, estos productos fueron publicados por empresas que crean aplicaciones web o trabajan con otras tecnologías y encuentran esta zona como una forma de buscar clientes. Otro gran número de productos se encuentran

desactualizados o no funcionan correctamente.

Xamarin

Introducción

Esta empresa se creo en 2011 por los creadores de Mono. Esta tecnología se uso como base en los proyectos realizados por Xamarin.

Desde sus inicios, Xamarin se especializó en la creación de aplicaciones móviles, creando herramientas que faciliten esta tarea.
Xamarin creó un conjunto de productos que permitían crear aplicaciones para Android, iOS y Windows Phone separando la parte lógica de la aplicación de la parte mas específica del programa como es la interfaz gráfica. Para ello, la lógica se organizaba en una librería que se incluía en los diferentes proyectos de cada plataforma.

Pero todos los proyectos se realizaban en tecnología .Net y contaban con una librería de Mono en cada dispositivo que interpretaba el código de la aplicación.

Esta arquitectura seguía un estándar definido por la compañía que exigía la necesidad de realizar un diseño visual especifico para cada plataforma. Esto se contrapone con la otra tendencia de crear una vez y ejecutar en todos los sitios.

Posteriormente se creo un nuevo proyecto llamado Xamarin.Forms. Este proyecto unifica todas las API nativas, ofreciendo la posibilidad de crear un solo proyecto para todas las interfaces. Xamarin.Forms es el producto que se analiza en este estudio.

Forma de Trabajo

Para trabajar con esta tecnología es necesario programar en C# tanto la interfaz gráfica como la lógica de la aplicación. También existe la posibilidad de usar XAML para el diseño gráfico. XAML es un lenguaje parecido a XML, con el que se puede definir los elementos visuales y sus atributos. XAML es empleado junto a las tecnologías .Net y es muy común en desarrollos de Microsoft Silverlight.

```
<Label Text="Hello, XAML!"
         VerticalOptions="Start"
         XAlign="Center"
         Rotation="-15"
         IsVisible="true"
         FontSize="Large"
         FontAttributes="Bold"
         TextColor="Aqua" />
```

Xamarin fue capaz de acceder a las API nativas desde Mono, es decir es posible ejecutar código escrito en C# que será traducido a operaciones en Java y Objective C. Este logro permitió a Xamarin crear una estructura en las distintas plataformas con las que se ejecutaban aplicaciones creadas en .Net.

Posteriormente unifico las distintas API de los sistemas operativos, introduciendo una interfaz capaz de juntar las funcionalidades de los distintas plataformas. Esta interfaz es el producto Xamarin.Forms, con el que es posible ejecutar la misma aplicación creada en C# en las distintas plataformas móviles.

Al utilizarse la tecnología de Mono, este framework es muy fácil de integrar en Windows Phone.

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

Xamrin nos presenta un framework con una alta calidad de pintado, pero existen algunos errores en algunos componentes.

Uno de los errores mas importantes se encuentra en el elemento de lista para la plataforma iOS. Este error se basa en que todos los elementos de la lista tienen que tener el mismo tamaño, establecido al crear la lista. Esto nos provocó un serio problema ya que nosotros queríamos crear una lista de noticias, cada elemento de la lista contendría un texto y un grupo de imágenes. El tamaño de los noticias puede variar mucho, dependiendo de la cantidad de texto y sobretodo de las imágenes que se quieran incluir.
Se encontró una solución que sobrecargaba al framework, realizando un calculo aproximado del tamaño que se necesitaba y exigiendo que se recargase la lista cada poco tiempo. Esto bajaba el rendimiento de la aplicación, además, en algunos momento no se aumentaba el tamaño lo suficientemente rápido y se podían ver elementos gráficos pintados encima de otros.

Otro de los problemas con el que hemos encontrado es el hecho de que en algunas situaciones puntuales el framework ignoraba la alineación de texto dentro de los botones.
Mas concretamente este problema ocurre en la plataforma Android al navegar entre páginas con pestañas a páginas simples, en este caso todos los textos de los botones ignoran sus atributos establecidos por el desarrollador y siempre se alinean a la izquierda.
Este fallo se cumple solo en determinadas ocasiones, pero si en nuestra aplicación se cumplen las condiciones para ello, puede ser un error que aprecia fácilmente el usuario final.

En Xamarin.Forms no es posible acceder a las cabeceras de las páginas que se gestionan de forma automática por el framework. Esto nos dificulta la posibilidad de modificar los atributos de los botones que se colocan en la cabecera junto al título de la página.

No existe el elemento de reproductor de vídeo. Sin embargo existe el componente Picker, correctamente implementado en todas las plataformas.

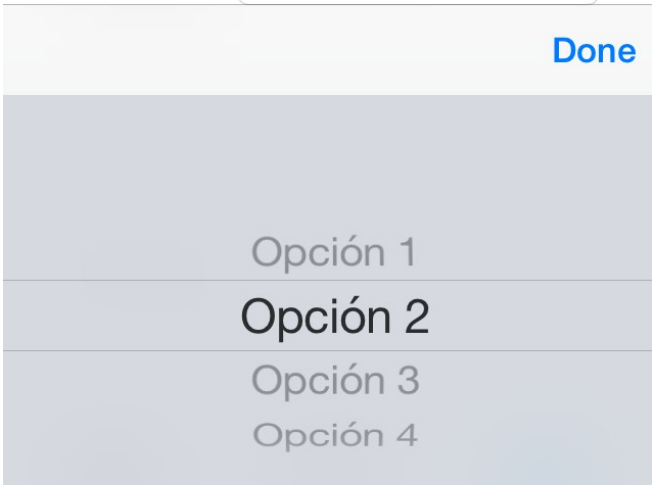

Figura 16: Representación correcta de Xamarin en iOS

En caso de ser necesario, es posible definir como serán pintados los elementos, para ello es necesario volver a definir la función de pintado de cada plataforma que utiliza los objetos nativos. Este proceso es difícil de realizar, pero puede ser muy útil en situaciones complicadas.

Capacidad de acceso a las API nativas del sistema operativo

Aunque Xamarin.Forms está diseñado para funcionar con varias plataformas al mismo tiempo, sigue exigiendo la posibilidad de implementar funcionalidades de forma distinta para cada una de las plataformas. Esto nos permite ampliar las funcionalidades del framework, utilizando los objetos nativos en caso de ser necesario.

Existe la posibilidad de utilizar los mapas nativos con este framework, incluso se puede usar Google Map en iOS de forma nativa. Para el uso del calendario es necesario emplear código particular de cada sistema operativo.

Xamarin.Forms nos permite almacenar los datos de la aplicación en ficheros dentro del dispositivo móvil y también a través de un gestor de base de datos.

Podemos acceder a los datos de GPS y a los sensores del dispositivo empleando un grupo de librerías creadas por un grupo de usuarios del framework, llamado Xlabs.

También es necesario utilizar una librería externa que nos permita emplear las funcionalidades de Bluetooth del teléfono móvil.

Xamarin también ofrece un grupo de librerías llamadas Xamarin.Auth y Xamarin.Social, que nos permiten interactuar con las API de las redes sociales de nuestro teléfono. Estas librerías son gratuitas.
Existe la posibilidad de utilizar Push Notification, pero para ello es necesario implementar estas funcionalidades de forma particular para cada plataforma.

Xamarin.Forms nos permite acceder al sistema de NFC para la plataforma Android. Los dispositivos iOS no permiten esto por limitaciones de sus dispositivos.

Tenemos la posibilidad de utilizar las funciones de C# para realizar peticiones web.

Sistemas operativos con los que trabaja

Este framework crea aplicaciones para las plataformas Android, iOS y Windows Phone.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Xamarin nos ofrece utilizar su IDE, llamado Xamarin Studio. También ofrecen un Plugin para la plataforma de Visual Studio de Microsoft. Nosotros hemos trabajado mayoritariamente con el IDE de Xamarin Studio.
Esta herramienta nos ofrece la posibilidad de crear nuestro código, gestionar las librerías que usaran, modificar los atributos de configuración de los proyectos y por último compilar y ejecutar nuestra aplicación en distintas plataformas.
Xamarin Studio ofrece la posibilidad de crear visualmente nuestra interfaz gráfica, pero esta opción tan solo está disponible para Xamarin.IOS y Xamarin.Android, no para Xamarin.Forms que es nuestro framework.

Nos hemos encontrado con ciertos problemas puntuales con este IDE. Ignora nuestros comandos en algunas situaciones. Cuando se especifica cual es el dispositivo emulado Android en el que se desea ejecutar la aplicación, el IDE ejecuta el proyecto en el dispositivo que este ya abierto aunque no sea el que hemos seleccionado nosotros.

Tampoco es posible ejecutar la aplicación en un dispositivo Android e iOS al mismo tiempo, en este caso, el IDE ignora nuestra acción sin mostrar ningún mensaje de error.

Para buscar e instalar distintas librerías es posible utilizar el manejador de paquetes llamado NuGet

Xamarin nos ofrece amplia documentación sobre sus productos, además, esta información esta bien estructurada y cuenta con una amplia cantidad de ejemplos.

Xamarin posee con una gran comunidad de desarrolladores, que trabajan con Xamarin.Forms u otros productos de esta empresa. Esta comunidad se suele implicar en el desarrollo de este framework, ayudando a otros usuario con la creación de aplicaciones y sobretodo ampliando las funcionalidades de Xamarin con nuevas librerías.

Licencias y costes

El coste de utilizar Xamarin.Forms es de $25 por cada desarrollador por cada plataforma, iOS y Android, cada mes. Ofrecen Windows Phone de manera gratuita.

Xamarin también ofrece un servicio con el que se puede probar las aplicaciones en mas de

1000 dispositivos reales conectados a sus servidores. Con este servicio es posible encontrar posibles errores en algún tipo de dispositivos antes de publicar nuestra aplicación. El coste de este servicio comienza en $12.000 anuales.

También existe un servicio con el que es posible analizar los eventos de nuestra aplicación que se producen en los dispositivos de nuestros clientes. Este producto actualmente no está terminado y está en versión de pruebas.

Previsiones de cambio

Si nos fijamos en el avance de la empresa en este ámbito, se puede observar que se intenta analizar y aplicar los cambios que se están realizando en las plataformas Android e iOS.
También se puede ver que la mayoría de los errores que se encuentran, son solucionados en un corto espacio de tiempo. Pero también existen errores muy antiguos que no se han arreglado

Esto nos hace pensar que Xamarin.Forms avanzara en su desarrollo a una velocidad constante, no se esperan grandes cambios ni estancamientos en el desarrollo.

Subjetivo

El planteamiento de utilizar Mono en las distintas plataformas ofrece un acercamiento al desarrollo móvil desde una dirección distinta a la del resto. Mientras que los demás framework se basan en tecnologías web, Xamarin utiliza Mono, .Net y librerías mas propios de desarrollos en programas para plataformas como PC.

Xamarin.Forms es un producto bastante completo, pero tiene un coste de $50 al mes por cada licencia que se desea comprar.

Otros

Xamarin, a diferencia del resto de los frameworks estudiados, funciona con C# y tecnología .Net. Esto puede ser muy útil para desarrolladores que vengan desde estas tecnologías, pero puede causar dificultados a otros desarrolladores que hayan especializado con tecnologías web.

ReactNative

Introducción

ReactNative es un framework realizado por Facebook primero de forma privada para sus

productos. Pero posteriormente, decidió publicarlo de forma pública bajo una licencia *BSD License*. Lo que permite a los usuarios crear aplicaciones de forma gratuita e incluso modificar el framework. Además, se espera que la comunidad de desarrolladores, junto a Facebook, contribuya en el avance de este proyecto

La base de la política de la arquitectura de este proyecto provienen de otro proyecto realizado por Facebook llamado React. React es una tecnología usada en la creación de páginas web dinámicas, sobretodo en Single-Page Applications.

Aunque ReactNative se ha utilizado anteriormente en los productos de Facebook. Se publicó de forma abierta el día 27 de abril de 2015. El framework todavía está en una etapa de crecimiento, es decir, se encuentran un gran número de errores que se corrigen rápido y se avanza en la creación de partes fundamentales de la librería.

Actualmente se dispone solo de la versión iOS, la versión de Android esta prevista para el mes de septiembre.

Forma de Trabajo

Tanto React como ReactNative siguen un planteamiento de la creación de componentes de interfaces de usuario separados, que se relacionan entre si para componer la totalidad de la página web, o en nuestro caso de las páginas de la aplicación móvil.

ReactNative crea internamente los componentes de la interfaz y posteriormente utiliza los métodos nativos para pintarlos en la pantalla, sin el uso de WebKit. Además, se guarda una copia del estado de cada componente para poder determinar que componentes han cambiado para repintar solo los que hayan tenido cambios en su estado.

```
var AttributeToggler = React.createClass({
  getInitialState: function() {
    return {fontWeight: '500', fontSize: 15};
  },
  increaseSize: function() {
    this.setState({
      fontSize: this.state.fontSize + 1
    });
  },
  render: function() {
    var curStyle = {fontSize: this.state.fontSize};
    return (
      <Text>
        <Text style={curStyle}>
          Tap the controls below to change attributes.
        </Text>
        <Text>
          See how it will even work on{' '}
          <Text style={curStyle}>
            this nested text
          </Text>
```

```
          <Text onPress={this.increaseSize}>
            {'>> Increase Size <<'}
          </Text>
        </Text>
      </Text>
    );
  }
});
```

El código se escribe en Javascript y JSX, que es una estructura parecida al XML. Como se puede apreciar en el fragmento de código, cada componente tiene una función *render* que realiza la representación visual del componente, esta función retorna un código JSX. Cada componente posee dos atributos *state* que representa el estado del componente y *props* que son los parámetros con los que se ha instanciado el componente.

Por lo tanto un proyecto creado en ReactNative es una jerarquía de componentes que engloban otros componentes, hasta llegar a los componentes básicos, que en React eran componentes HTML (div, a, ...), mientras que en ReactNative son componentes creados por el framework (View, Text, …)

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

El principal problema que existe en la actualidad es el hecho de que cuando aparece el teclado en la pantalla, esta debería reducir sus dimensiones de forma que el teclado no tape ningún elemento visual. Esto no está implementado, por lo tanto al aparecer el teclado, existen componentes que no se pueden ver en la pantalla. Por ejemplo si deseamos escribir texto en un campo, es posible que el teclado nos tape la zona de la pantalla en la que está el texto y no podamos ver como introducimos las letras.

En la actualidad existe problemas en el pintado de algunos componentes como el *Picker*, que no esta correctamente diseñado. La representación visual es correcta, pero el elemento debería mostrarse en la posición donde suele aparecer el teclado, pero este framework lo añade a la página como un elemento de texto o imagen.

Figura 17: Error en el diseño del Picker de ReactNative para iOS

No existe un componente que sea el botón, para ello se emplea un texto que implementa un manejado que se ejecuta al saltar el evento de pulsado sobre el texto. Este no es un problema visual en este momento, pero preocupa este diseño cuando se implemente en dispositivos Android o si iOS cambia sus elementos de botón, que en la actualidad es un texto, a un componente con una forma mas típica de botón.

También existen problemas puntuales de actualización del valor del texto al usar el corrector lingüístico automático del sistema operativo.

ReactNative nos ofrece la posibilidad de usar FlexBox que facilita la forma de colocar los elementos gráficos dentro de la página de la aplicación.

Capacidad de acceso a las API nativas del sistema operativo

Facebook implementa la posibilidad de realizar llamadas a librerías de código nativo desde la plataforma de ReactNative. Esto facilita la implementación de funcionalidades nativas para el framework.

En la actualidad existen las API de mapas, pero no están completas ya que no permite crear marcadores. No existen la posibilidad de acceder al sistema de calendario y eventos del dispositivo móvil, pero está registrado como uno de los módulos que faltan y los desarrolladores del framework tienen pensado crear este módulo o avanzar con su desarrollo si algún usuario decide empezar antes con él.

El almacenamiento mediante el sistema de archivos funciona correctamente de forma asíncrona. Para guardar datos en la base de datos es necesario utilizar un módulo externo. También es necesario utilizar un módulo externo si se desea acceder a los sensores del teléfono móvil.

El sistema GPS está correctamente integrado junto al sistema de Push Notification. El acceso a las API de las redes sociales utilizan una librería creada por un usuario del framework.

Al estar actualmente disponible el framework sólo para los dispositivos iOS, los cuales no tienen permitido el uso de NFC por los desarrolladores, no se han creado los módulos de NFC para ReactNative.

ReactNative nos ofrece la posibilidad de utilizar las funciones básicas de trabajo con las peticiones web o emplear librerías en JavaScript mas completas.

Sistemas operativos con los que trabaja

Actualmente ReactNative solo esta disponible para iOS.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

ReactNative no cuenta con un IDE específico para sus proyectos. Sin embargo sí existen varias herramientas que nos facilitan el desarrollo.

El framework actúa como un servidor, permitiéndonos ver los cambios que realizamos en nuestro código sin la necesidad de recompilar el proyecto, tan solo es necesario pulsar una combinación de teclas para apreciar la nueva versión en nuestro dispositivo.

También existe una conexión con el navegador Chrome de nuestro ordenador de desarrollo, permitiendo utilizar las herramientas de desarrollo del navegador. Con este sistema podemos ver la consola de ejecución, inspeccionar los elementos y depurar los errores desde el navegador.

Al ser ReactNative una tecnología joven, no cuenta con una documentación muy extensa y la que existe está sujeta a cambios con cada versión, pero se va complementando la documentación a lo largo que pasa el tiempo. Existen una gran cantidad de ejemplos y los desarrolladores suelen emplear la plataforma de Github, donde tienen almacenado el proyecto, para responder a las dudas y sugerencias de los usuarios.

ReactNative fue bien acogido por la comunidad de desarrolladores tanto de aplicaciones móviles como por los antiguos usuarios de React para aplicaciones web. Por lo tanto es posible encontrar muchos recursos fuera de la página oficial, sobretodo opiniones, ejemplos, guías de uso y módulos adicionales.

Licencias y costes

Facebook registro ReactNative bajo la licencia BSD, permitiendo su uso de forma libre para los desarrolladores. Actualmente la empresa no ofrece ningún producto adicional de pago que pueda servir de complemento.

Previsiones de cambio

ReactNative es una tecnología muy joven, aunque ya se ha demostrado su funcionamiento en las páginas web, en los teléfonos móviles todavía no tiene una historia. Actualmente se desarrolla activamente, ofreciendo nuevas versiones de forma muy frecuente.
Pero no existe una versión todavía para Android y es muy importante ver como se acomodará la arquitectura de este framework par poder ofrecer un producto adecuado en dos plataformas simultáneamente.
También hay que señalar que Facebook, la empresa que desarrolla esta tecnología, muestra mucho interés en sacar este proyecto adelante y cuenta con una gran cantidad de recursos para él.

Subjetivo

Las sensaciones que nos ofrece este framework son de que se encuentra todavía en sus primeras fases, pero no va a tardar mucho en convertirse en una opción para el desarrollo de aplicaciones móviles. Existe una gran comunidad de desarrolladores tanto de la empresa

como externos que trabajan en este proyecto y se ve reflejado en los rápidos avances que se realizan. Aunque todavía existen fallos serios que es necesario corregir.

NativeScript

Introducción

NativeScript es un producto de la compañía Telerik, conocida por otros productos relacionadas con el mundo de aplicaciones móviles como es Kendo, framework HTML 5 para la web y dispositivos móviles, también realizaron módulos para PhoneGap y Xamarin.

Telerik surge en el año 2002 como una pequeña empresa creada por 4 amigos. Esta empresa comenzó con la creación de herramientas para el desarrollo web, centrándose en la tecnología ASP.Net. Creció ampliamente continuando con el desarrollo de herramientas web y en diciembre de 2014 es adquirido por Progress Software por unos $262 millones.

Después de la finalización de la compra, los fundadores de la compañía Telerik entraron en Progress Software y la compañía siguió desarrollando las herramientas por las que se hizo famosa.

Desde hace un tiempo Telerik está desarrollando un producto que sirva para crear aplicaciones móviles nativas sin el uso de WebKit. Este producto se llama NativeScript y la primera versión pública salió el día 5 de marzo de 2015.

Forma de Trabajo

NativeScript utiliza el lenguaje de JavaScript para implementar la lógica de nuestra aplicación y XML y CSS para el diseño de la interfaz gráfica. El framework se centra en el entrelazado de las variables de JavaScript con elementos en XML

```
<Page loaded="loginLoad">
    <StackLayout>
        <Label text="Name:" />
        <TextField id="name" width="220"/>

        <Label text="Password:" />
        <TextField id="pwd" width="220"/>

        <Button id="loginButton" text="Log In"
width="120"/>

    </StackLayout>
</Page>
```

Desde el punto de vista mas técnico, NativeScript es un sistema que aplica

empaquetamiento automático a los objetos nativos ya sean Java en Android o Objective C en iOS. De esta forma se consigue que se pueda utilizar objetos nativos de la plataforma desde JavaScript. Este sistema es muy parecido al aplicado por Appcelerator.

Posteriormente se aplica una capa de abstracción en la que se unen ambos objetos para unificar las funcionalidades de las dos plataformas.

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

Al igual que con otros frameworks, existe un problema con los dispositivos iOS al aparecer el teclado en la pantalla. La pantalla debería reducir sus dimensiones lógicas, permitiendo la posibilidad de seguir viendo todos los elementos de la página. Pero en este framework, el teclado tapa parte de la pantalla, pudiendo ocultar incluso el componente en el que se está introduciendo el texto.

No están implementados una gran cantidad de elementos gráficos necesarios para la realización de una aplicación. Hay que recordar que este framework es muy nuevo y no está todavía terminado. La mayoría de los elementos que faltan están planteados para las siguientes versiones.

Capacidad de acceso a las API nativas del sistema operativo

Mientras que NativeScript permite acceder a todos los objetos nativos, esto es muy poco útil si no existe una interfaz que nos facilite el trabajo con los objetos, juntando las funcionalidades de la plataforma Android con la de iOS. Si se desea trabajar con los objetos nativos es mas fácil utilizar las formas de desarrollo nativas en Java u Objective C.

No existe la capacidad de crear mapas nativas, los desarrolladores del framework crearon una demo para demostrar la creación de módulos externos, pero es un prototipo muy limitado y funciona tan solo en iOS.

Es posible guardar los datos en el sistema de ficheros. Pero no existe un módulo para manejar bases de datos, aunque esto se está creando por la comunidad.

Es posible acceder a los datos del GPS. No es posible utilizar los sensores de los dispositivos ni tampoco la comunicación con Bluetooth.

No están implementados las funcionalidades de las redes sociales ni Push Notification, este está planteado para la versión 1,1 en junio de 2015. Tampoco existe una forma de controlar el componente de NFC del teléfono móvil. Tenemos la posibilidad de realizar peticiones HTTP con este framework.

Sistemas operativos con los que trabaja

NativeScript puede crear aplicaciones Android e iOS.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

Existe un IDE para NativeScript, pero es privado, también ofrecen existe la posibilidad de contratar un servicio de compilar las aplicaciones en sus servidores y actualizar los dispositivos que usen la aplicación automáticamente. En el caso de no querer adquirir productos adicionales es necesario utilizar este framework desde la línea de comandos.

La documentación es muy escasa y está mal organizada. La escasez de la documentación se debe a que es un framework muy joven. La documentación que existe actualmente esta muy fraccionada en varias páginas por cada componente. También, faltan ejemplos de uso de los componentes. Incluso es necesario buscar en la documentación de Google y Apple para encontrar los nombres de las propiedades de los objetos nativos.

Este framework cuenta con un gran número de desarrolladores externos, que ayudan con la creación de nuevos módulos, ampliando la documentación y solventando dudas de otros usuarios. Hay que mencionar que el hecho de que NativeScript se haya publicado en las mismas fechas que ReactNative ha reducido el tamaño de la comunidad ya que el proyecto de Facebook, ReactNative, es mucho mas popular.

Licencias y costes

NativeScript tiene una licencia del tipo Apache versión 2. Este framework es gratuito, pero la empresa Telerik ofrece un gran abanico de herramientas privadas que pueden facilitar el desarrollo de proyectos con NativeScript.

Previsiones de cambio

Al igual que ReactNative, este framework es muy joven y no está completo actualmente. Los desarrolladores anunciaban un sistema mucho más completo que el que se publicó, también se establecieron avances en las siguientes versiones muy ambiciosas. Los desarrolladores, parece que se han dado cuenta que van a necesitar más tiempo para completar NativeScript y han modificado sus previsiones con unos resultados mucho más realistas, como la corrección de errores y las implementaciones de módulos que no estás disponibles actualmente.

Subjetivo

A este framework le falta mucho para poder ser utilizado en un proyecto real. Actualmente tan solo es una forma de acceder a los objetos nativos desde JavaScript, pero esto no es suficiente para un framework de este tipo.
Se deberían de centrar los esfuerzos en aumentar la utilidad de esta tecnología, depurando la interfaz gráfica y creando nuevos módulos que sirvan como una capa intermedia para los objetos nativos. Se espera que al usar este tipo de framework no sean necesarios conocimientos de los métodos y atributos de los objetos nativos. En caso de tener estos

conocimientos, es más optimo utilizarlos para programar en proyectos nativos.

Otros

NativeScript todavía se encuentra en un estado muy joven y por ello existen cambios en la estructura del framework como la eliminación o renombre de atributos de algunos elementos. Esto nos obliga a revisar todo nuestro código al cambiar de versión, pero al no estar terminado todavía el framework, son aceptables estos cambios.

Existe un error al iniciar la aplicación por primera vez, la aplicación tarda mucho en reaccionar. Este error está reportado por los usuarios y los desarrolladores han comunicado que van a centrarse en corregirlo.

Las aplicaciones móviles necesitan un grupo de permisos del sistema operativo para poder realizar ciertas acciones. Estos permisos están detallados en los archivos de configuración de cada plataforma. En la actualidad no existe ninguna forma de registrar estos permisos o acceder a los archivos de configuración durante el desarrollo. Es necesario compilar la aplicación y posteriormente cambiar manualmente los archivos de configuración dentro de los archivos compilados.

TabrisJs

Introducción

TabrisJs es uno de los productos desarrollados por la compañía EclipseSource. Uno de sus productos mas conocidos es Yoxos, una distribución de Eclipse. Pero la compañía también ofrece otros productos y servicios

En el ámbito de las aplicaciones móviles, EclipseScource ofrece dos productos, Tabris y TabrisJs. TabrisJs es un proyecto que comienza en mayo de 2014 y no ha contado con mucha popularidad.

Tabris es un sistema cliente-servidor, en el que el cliente esta instalado en el teléfono móvil y tan solo pinta los elementos que le manda el servidor. El servidor es el que contiene toda la lógica de la aplicación y decide cual es la forma que se representarán los elementos en la pantalla del cliente. Este sistema imposibilita tener aplicaciones sin conexión a Internet y puede causar tener un gran tiempo de reacción a las acciones del usuario.

No se analiza el producto Tabris, pero se decide realizar un estudio a la plataforma de TabrisJs

Forma de Trabajo

Para trabajar con este framework es necesario utilizar JavaScript para desarrollar todos los componentes de la aplicación.

```
tabris.create("Action", {
  title: "Settings",
  image: {src: "images/action_settings.png", scale: 3}
}).on("select", function() {
  createSettingsPage().open();
});
```

Las características mas singulares de este framework es el método de desarrollo que se sigue. Para desarrollar con TabrisJS es necesario descargar una aplicación de la compañía e instalarla en nuestro dispositivo en el que deseamos probar nuestra aplicación. Montamos un servidor en nuestra máquina de desarrollo y desde la aplicación móvil, que hemos instalado antes, accedemos a nuestro servidor, donde están los archivos JavaScript de nuestro proyecto. La aplicación de desarrollo cuenta con 500-1000 instalaciones en Google Play.

También hay que mencionar que para compilar posteriormente nuestro proyecto, es necesario utilizar sus servidores. Enviamos nuestro código, se procesa en sus servidores y se crea un ejecutable para la plataforma Android o para iOS. Si se desea compilar de forma local, es necesario adquirir una licencia adicional.

TabrisJs utiliza la librería de Cordova para acceder a los elementos nativos, igual que hace Ionic. Pero TabrisJs se ejecuta en un motor JavaScript y no necesita emplear WebKit.

Análisis del framework

Calidad de representación y pintado de la interfaz de usuario.

Tabris tiene una forma singular de implementar las pestañas, ya que no siguen los estándares marcados por los sistemas operativos y tienen una forma parecida a las pestañas dentro de un navegador web.

Faltan muchos atributos de formato de los elementos visuales. No existe la posibilidad de crear bordes ni *padding*[1].

El componente de la lista que se utiliza en este framework no permite tener elementos con distintos tamaños. Al modificar el tamaño de una fila de la lista, todas las demás filas cambian también su tamaño.

1 Espacio invisible de relleno que se deja entre el borde y el contenido de un elemento

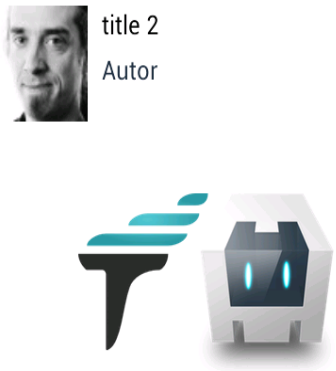

Figura 18: Elemento de la lista con imagen de TabrisJs en Android

Figura 19: Error en el elemento de la lista sin imagen de TabrisJs en Android

Capacidad de acceso a las API nativas del sistema operativo

Una de las ventajas que nos ofrece este framework es la posibilidad de utilizar casi todos los Plugins de Cordova, todos los que no realicen cambios en el DOM.

Esto es posible en teoría, pero en la practica tenemos unas limitaciones que nos impiden utilizar la mayoría de los Plugins. Esta limitación surge del hecho de que no podemos realizar compilaciones de nuestro proyecto en local y es necesario compilar utilizando sus servidores.
Tenemos que especificar en los archivos de configuración del proyecto los Plugins que vamos a emplear para que el servidor instale esos módulos y los añada a nuestro proyecto. El problema está en la nula documentación de estos archivos de configuración y la nula ayuda por parte de los desarrolladores del framework. En este caso el desconocimiento del funcionamiento de los servidores de compilación nos impiden utilizar una gran cantidad de Plugins. Tan solo podemos utilizar los Plugins que ya estén incluidos en el proyecto por defecto o aquellos que este registrados en un indice determinado (http://registry.cordova.io/) pero es imposible utilizar otros que estén en otro indice o en un repositorio de GitHub.

El único sistema que hemos conseguido utilizar fue el de los sensores por el módulo Device Motion, un Plugin ya incrustado en el proyecto de TabrisJs.

Sistemas operativos con los que trabaja

Tabris nos permite crear aplicaciones Android e iOS.

Facilidad de uso: IDE, documentación y comunidad de desarrolladores

No existe un IDE particular para este framework, para ejecutar las diferentes acciones es necesario utilizar herramientas adicionales.
Para realizar las pruebas es posible utilizar un servidor en la máquina de desarrollo y una aplicación móvil auxiliar que actúa de cliente y reproduce nuestro proyecto en el dispositivo. Esto reduce el tiempo que se emplea en compilar y mover nuestra aplicación al dispositivo. Pero nos obliga a trabajar con un dispositivo físico ya que los emuladores no disponen de Google Market para descargar e instalar la aplicación auxiliar.

Esta forma de realizar pruebas está limitada, en que no podemos modificar ningún archivo de la estructura de la aplicación. Es decir si deseamos añadir un Plugin a nuestra aplicación tenemos que compilar nuestro proyecto.

Si deseamos utilizar la versión gratuita del framework, tan solo podemos compilar utilizando sus servidores y almacenando nuestro código en un repositorio publico de GitHub. Esto nos obliga a guardar de forma público nuestro código que, además, posteriormente es enviado a sus servidores. Además, de los problemas de privacidad, existe un problema de usabilidad. Si deseamos realizar cualquier cambio en nuestro proyecto, tenemos que modificar los archivos en local, subir los cambios al repositorio GitHub, empezar la compilación en los servidores de TabrisJs, esperar a que termine de compilar, descargar el ejecutable generado y instalarlo en nuestro dispositivo usando las herramientas del SDK de Android. Este procedimiento tarda mucho en realizarse.

La documentación es muy limitada, sobretodo en lo referente a los archivos de configuración.

No existe muchos desarrolladores trabajando con este framework. Es imposible encontrar documentación, ejemplos o referencias fuera de la pagina oficial. Son los propios desarrolladores de TabrisJs los que reportan los errores y son los únicos que trabajan en este proyecto.

Licencias y costes

TabrisJs es de código abierto, pero el problema está en la forma de compilar los proyectos creados con este framework. Existen tres posibilidades:

a) Gratuito: Solo se puede compilar en sus servidores y empleando un repositorio GitHub publico.
b) $5/mes: Solo se puede compilar en sus servidores pero se pueden emplear repositorios privados de GitHub.
c) $50/mes: Es posible compilar en tu propia máquina local.

Previsiones de cambio

Si analizamos el avance de este proyecto, podemos esperar que se realicen avances, pero no existen previsiones de grandes mejoras en el framework.

Subjetivo

La opción de utilizar la versión gratuita de este framework es imposible para proyectos privados. Y el framework no muestra suficiente capacidad para invertir dinero en una licencia privada.
Lo mas preocupante de este framework es la ausencia de una comunidad de desarrolladores que lo utilicen, es imposible que los creadores de TabrisJs sean capaces de realizar solos todo el proyecto.
Podrían crearse serios problemas si hubiéramos empezado un proyecto real y nos encontráramos en la misma situación de intención de añadir nuevos Plugins y la falta de ayuda por parte del equipo de Tabris.