UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Aplicación multiplataforma para la gestión de archivos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Francisco Juan Nicolás

Tutor: Ramón García Escrivá

2013-2014

Resumen

El proyecto a desarrollar trata de una aplicación multiplataforma para dispositivos móviles la cual permita gestionar tanto los archivos locales, como los almacenados en alguna de las plataformas de almacenamiento de archivos en la nube más conocidas. Para alcanzar nuestro objetivo hemos hecho un estudio de los *framework*s destinados al desarrollo de aplicaciones multiplataforma, seleccionando el que más se adapta a las necesidades que hemos determinado. Tras la elección, estudiamos más a fondo dicho *framework* y los componentes que lo acompañan. Una vez conocido el *framework*, hemos estudiado las tecnologías que utiliza, en su mayoría tecnologías web (HTML, CSS y JavaScript), y las hemos utilizado para conseguir nuestro objetivo. Para desarrollar el código que hace útil nuestra aplicación, nos hemos apoyado en diferentes herramientas. Estas herramientas han variado según la necesidad que cada sección de la aplicación requería. Para la primera sección, cuya finalidad es listar los archivos del dispositivo que ejecutara la aplicación, la herramienta utilizada ha sido Phonegap, y así poder abarcar casi todas las plataformas existentes en el mercado. En cuanto la segunda sección, para gestionar los archivos de la plataforma en la nube "DropBox", hemos utilizado la API propia que ofrece la plataforma. Por último, la tercera sección, cuya misión es solventar las dudas de los usuarios de la aplicación, se ha desarrollado con texto simple formateado con etiquetas HTML. Todas las secciones las hemos desarrollado sobre un editor de textos que nos facilita la lectura de las tecnologías utilizadas, y con ellos el desarrollo del código fuente.

Palabras clave: aplicaciones móviles, phonegap, multiplataforma , html, css, javascript.

Tabla de contenidos

1. Introducción

Vamos a desarrollar un proyecto cuya finalidad es la creación de una aplicación móvil para la gestión de documentos propios basado en plataformas de dispositivos móviles.

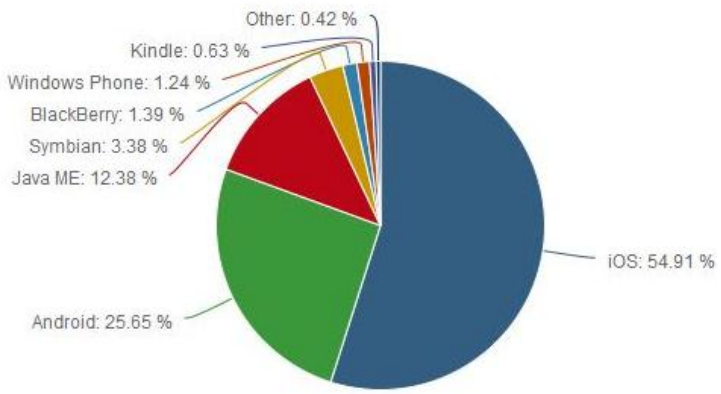
El objetivo del proyecto es tener aplicación en nuestro dispositivo móvil, cuya funcionalidad es la de un explorador de archivos del propio dispositivo, y que además, incluya la posibilidad de gestionar los archivos contenidos en algunas de las principales plataformas de almacenamiento de archivos en la nube, tales como DropBox, Google Drive o Box. Con ello conseguiremos tener una gestión total tanto de nuestros archivos almacenados en nuestro dispositivo, como los almacenados en un repositorio de internet.

Como además, queremos que la aplicación desarrollada sea una solución para el mayor número de plataformas posibles, o al menos para las de uso más extendido, vamos a emplear un *framework* que nos permita esta funcionalidad con el mayor ahorro de código posible. Para reducir en la medida de lo posible el coste de nuestro proyecto, vamos a basarnos, si las características lo permiten, en programas y tecnología de código libre, lo que nos ayudará a alcanzar este objetivo.

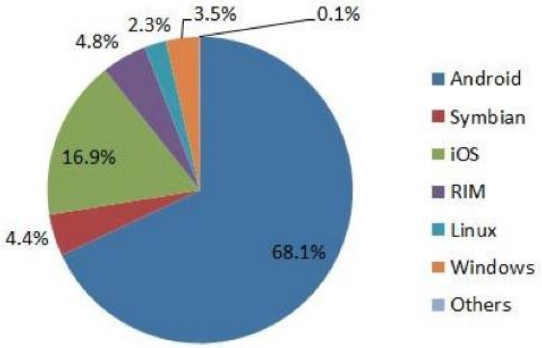
Para introducirnos y entender mejor el objetivo y desarrollo de nuestro proyecto, vamos a documentarnos sobre las aplicaciones móviles y sus plataformas. Vamos a empezar hablando de los Smartphone y el papel que desempeñan en la actualidad estos dispositivos. Un Smartphone, como define su palabra en inglés, es un teléfono móvil inteligente. Con inteligente, nos referimos a que es capaz de realizar funciones muy superiores a las de un teléfono móvil de los que teníamos hace una década. Estos han mejorado sus características en cuanto a capacidad, pantalla, usabilidad, y velocidad. Ahora no podemos imaginar un teléfono móvil con tan solo un 1GB de almacenamiento de datos, seguramente un porcentaje bastante alto de los usuarios no podrían almacenar todos los archivos, aplicaciones, fotos,... que tiene guardados actualmente en su móvil. Tampoco se sentiría cómodo utilizando un móvil cuya pantalla sea inferior a 3 pulgadas, incluso se sentiría extraño cualquier joven que tuviera que usar un teléfono móvil con teclas físicas, y no pudiera disfrutar de una pantalla táctil para su día a día

con el móvil. La característica por excelencia de los Smartphone es la accesibilidad a internet y la navegación por cualquier zona de la *World Wide Web* como de un ordenador se tratara. Según un estudio realizado por la empresa *Ipsos MediaCT* a petición de la empresa *Google* asegura que el uso de Internet móvil a diario alcanza el 93% en España, es decir, prácticamente todo aquel que posee un teléfono móvil, se conecta a internet para solicitar y/o intercambiar información.

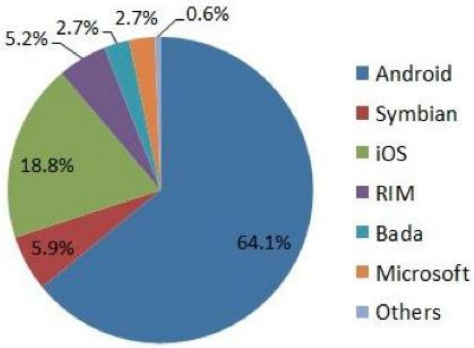
Otra estadística importante a tener en cuenta es el porcentaje de los sistemas operativo que utilizan dichos dispositivos. Como es de suponer, Android y iOS son los líderes en este aspecto, pero vamos a ver a continuación más detalladamente los porcentajes que ocupa cada uno de ellos. Este estudio ha sido realizado por dos grandes consultar y de investigación de IT, IDC y Gartner en el 2012.



Como podemos observar, Android tenía ganada la partida a iOS, su relación calidad/precio fue todo un éxito y muchos fabricantes apostaron por sumarse a la integración de este sistema operativo. Pero iOS se puso manos a la obra y convenció a los consumidores de que sus dispositivos, junto con su sistema operativos valían la pena, y en 2013 remontó las estadísticas superando a Android. Lo vemos en la siguiente gráfica:



IDC - Worldwide Mobile Phone Tracker



Gartner - IT research and advisory firm

2. Las aplicaciones móviles

Ahora vamos a centrarnos más en concreto en las aplicaciones que corren en cada uno de los dispositivos, que, en definitiva, es el objetivo de nuestro estudio. Por definición, una aplicación móvil es software desarrollado para ser ejecutado en un dispositivo móvil, teniendo en cuenta las características y limitaciones de los mismos. Hay tres tipos de aplicaciones móviles: Aplicaciones nativas, aplicaciones web y aplicaciones híbridas. Vamos a diferenciar cada una de ellas para entenderlas mejor.

1 – Aplicaciones Nativas.

Como podemos intuir por el nombre, este tipo de aplicaciones son aquellas que se desarrollan de forma específica para un sistema operativo, permitiéndonos acceder a elementos del propio dispositivo, tales como acelerómetro, actividad de la pantalla,... Cada una de las plataformas existentes tiene un sistema diferente, con lo cual, si queremos que una aplicación sea utilizada en diferentes plataformas, habrá que desarrollarla para cada una de esas, en principio. Decimos en principio, por que más adelante intentaremos solventar este problema, para conseguir que nuestra aplicación corra en todas las plataformas posibles con el menor desarrollo posible.

2 – Aplicaciones Web.

Una aplicación web es una aplicación desarrollada con leguajes tales como HTML, CSS y JavaScript. Este tipos de aplicaciones no tiene la restricción de plataforma como ocurría en las aplicaciones nativas, ya que podemos programar con estos lenguajes para cualquier plataforma, y ejecutarla en los diferentes dispositivos que la utilicen. Este tipo de aplicaciones se ejecutan dentro del navegador web y el contenido se adapta al dispositivo simulando una aplicación cualquiera.

3 – Aplicaciones Híbridas.

Las aplicaciones híbridas, como podemos deducir, son aplicaciones que contienen características de los dos tipos anteriores. Este tipo de aplicaciones se desarrollan con el

lenguaje propio de una aplicación web (HTML, CSS y JavaScript), pero también tiene la ventaja de poder acceder a las características hardware de los dispositivos, como ocurría en las aplicaciones nativas. Además, este tipo de aplicaciones nos permite agrupar el código y distribuirla en las diferentes plataformas mediante su mercado de aplicaciones.

PhoneGap es uno de los *frameworks* más utilizados para el desarrollo de este tipo de aplicaciones, aunque existen otras como Enyo, Cordova o Rhodes. Más adelante haremos un estudio sobre estos *frameworks*.

Para reforzar el estudio que hemos hecho nosotros sobre los tres tipos de aplicaciones, vamos a apoyarnos en una tabla basada en el estudio de desarrollos para móviles por la empresa Dzone Research, en la cual resume las características de cada tipo:

NATIVE vs. WEB vs. HYBRID: 7 FACTORS OF COMPARISON KEY: CON | PRO | NEUTRAL

	NATIVE	HYBRID	WEB
COST	Commonly the highest of the three choices if developing for multiple platforms	Similar to pure web costs, but extra skills are required for hybrid tools	Lowest cost due to single codebase and common skillset
CODE REUSABILITY/ PORTABILITY	Code for one platform only works for that platform	Most hybrid tools will enable portability of a single codebase to the major mobile platforms	Browser compatibility and performance are the only concerns
DEVICE ACCESS	Platform SDK enables access to all device APIs	Many device APIs closed to web apps can be accessed, depending on the tool	Only a few device APIs like geolocation can be accessed, but the number is growing
UI CONSISTENCY	Platform comes with familiar, original UI components	UI frameworks can achieve a fairly native look	UI frameworks can achieve a fairly native look
DISTRIBUTION	App stores provide marketing benefits, but also have requirements and restrictions	App stores provide marketing benefits, but also have requirements and restrictions	No restrictions to launch, but there are no app store benefits
PERFORMANCE	Native code has direct access to platform functionality, resulting in better performance	For complex apps, the abstraction layers often prevent native-like performance	Performance is based on browser and network connection
MONETIZATION	More monetization opportunities, but stores take a percentage	More monetization opportunities, but stores take a percentage	No store commissions or setup costs, but there are few monetization methods

3. Herramientas

Antes de estudiar y elegir nuestra herramienta de trabajo, vamos a definir qué es y para qué sirve un *framework*, ya que conociendo este tipo de herramientas, nos será mucho más fácil desarrollar una aplicación como la nuestra.

Como podemos observar, la palabra *framework* viene del inglés, y la podemos dividir en dos para adecuar esta palabra a nuestro vocabulario. *Frame*, en español significa "Marco", y *work* significa "Trabajo", con lo que ya podemos hacernos una pequeña idea inicial de lo que es un *framework*, un marco de trabajo. Pero, ¿Para qué y cómo se utiliza este "marco de trabajo"?

Como definición simple y aplicada la informática, podríamos utilizar la siguiente que nos ofrece la wikipedia:

"En el desarrollo de *software*, un *framework*, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*.". Con palabras menos técnicas, un *framework* es un *software* que nos permite, mediante un esqueleto ya desarrollado, desarrollar nuestro *software* de manera más fácil y eficiente.

Como hemos nombrado al principio de la explicación, un *framework* es un marco de trabajo, y ahora podremos ver más claramente esta traducción, y así poder ver las ventajas de utilizar uno. El marco de trabajo, nos da una estructura ya hecha, donde nosotros solo tenemos que rellenarla con lo que creamos que tiene que contener, lo que nos ahorra mucho trabajo y esto se traduce en una reducción de coste temporal y económico. Otro motivo por el que es conveniente utilizar un framework, es que nos facilita mucho la colaboración dentro de un equipo de trabajo, ya que no tenemos que ir descifrando código de otros compañeros, sino que al tener una estructura ya definida, nos basta con conocer el *framework* para colaborar en el desarrollo con más personas. Por último, al trabajar con una herramienta común a muchos otros programadores, podemos beneficiarnos de utilidades ya desarrolladas por estos, y que ya están adaptadas al propio *framework*.

Una vez sabemos qué es y para qué nos va a servir el *framework*, vamos a indicar nuestras preferencias a la hora de elegir entre los posibles *framework*s existentes para el desarrollo de nuestra aplicación. Como hemos mencionado anteriormente, nuestras aplicación tiene tres características importantes en las que debemos basarnos a la hora de elegir nuestra mejor *framework* para el desarrollo:

- Permita el desarrollo de aplicaciones multiplataforma, con las menores modificaciones posibles de código.

- Permita el uso de tecnologías estándar, que son asumibles por todas las plataformas y SDKs.

- Sean programas cuya licencia está basada en la licencia GNU GPL, es decir, software libre de carácter público, garantizando al usuario la utilización del mismo sin coste alguno.

Con los anteriores puntos definidos, podemos pasar a comparar los diferentes *framework*s y sus características, para posteriormente seleccionar el que más conviene para nuestra solución.

Tras una intensa búsqueda, hemos seleccionado los siguientes *framework*s, los cuales cumplen las premisas mencionadas de las cuales vamos a ver sus características.

PhoneGap - Apache Cordova

Como ventajas, PhoneGap, basado en apache cordova (En el siguiente punto explicaremos más sobre este tema), cumple los tres requisitos citados previamente, como la licencia de código libre, la aceptación de la mayoría de las plataformas móviles actuales, y el uso de tecnologías estándar, en este caso HTML, CSS y JavaScript. También tiene como ventaja que la finalidad de este *framework*, es la construcción de aplicaciones empotradas, es decir, se ha diseñado la herramienta con el objetivo específico para el que vamos a utilizarlo nosotros. Esto nos permite poder acceder al hardware del dispositivo para el que estamos desarrollando la aplicación, es decir, recoger información del acelerómetro, cámara, almacenamiento, contactos, etc.

Otras características no tan importantes para nuestra solución, pero a tener en cuenta son: Que permite la implantación de bases de datos vía SQLite, y que no requiere de una conexión a internet permanente para su funcionamiento.

Como desventaja en este *framework*, tenemos que no permite *multi-thread*, es decir, no nos permita crear una aplicación concurrente.

Enyo

Al igual que pasaba con PhoneGap, Enyo cubre las necesidad básicas que requiere nuestra aplicación. Es de código libre, utiliza las mismas tecnologías para desarrollo que PhoneGap (HTML, CSS y JavaScript), y es soportado por las principales plataformas. También tiene en común con PhoneGap que este software ha sido diseñado para la creación de aplicaciones lo cual le da un valor añadido, aunque en este caso fue diseñado para aplicaciones web también permite aplicaciones de escritorio y dispositivos móviles. Una cosa cosa muy valorable de Enyo es que soporta Web Services con JSON.

Enyo como contrapartida, tiene algunas carencias. No soporta el sistema operativo Symbian, que aunque no sea de los más utilizados en la actualidad, fue si no el más utilizado, de lo más utilizados cuando salió al mercado. En cuanto al acceso al hardware del los dispositivos, solo permite acceder a la geolocalización. En cuanto al soporte de base de datos, no se ha encontrado nada en la documentación oficial, con lo cual descartamos dicha característica. Aunque a priori nuestra aplicación no lo va a requerir, es un punto a tener en cuenta.

Rhodes

El *framework* Rhodes, cumple, en parte, todos los criterios que queremos, aunque con algunas diferencias respecto a los otros dos. La licencia de Rhodes no es libre, pero esta desarrollado bajo la licencia MIT (Massachusetts Institute of Technology), esta licencia no tiene *copyright*, lo cual permite modificar su código, pero tiene ligeras diferencias con la licencia GNU GPL. En cuanto a tecnologías, permite el desarrollo de aplicaciones utilizando las tecnologías HTML y Ruby. Y su desarrollo es

posible para cualquiera de las plataformas disponibles. Además de todas estas características, Rhodes tiene una ventaja sobre Phonegap y Enyo, y es que tiene un servicio en la nube, que permite al desarrollador crear aplicaciones *online* con el *framework* Rhodes, sin necesidad de tener actualizada la última versión del SDK para cada plataforma. Lo que hace al *framework* mucho más "portable" que los otros dos.

4. Tecnologías

Como hemos avanzado en la introducción del proyecto, vamos a utilizar las tecnologías más estándar y conocidas posibles. Y como también hemos citado en la descripción de nuestro *framework* elegido, PhoneGap nos lo permite. Las tecnologías soportadas por nuestro *framework*, son las tecnologías web HTML, con su versión 5 incluida, CSS3 y Javascript, admitiendo también a raíz de esta, la librería JqueryMobile. Vamos a explicar cada una de estas tecnologías a continuación:

1 – HTML y HTML5.

HTML son las siglas de *HyperText Markup Language*, el estándar referencia para el desarrollo de páginas web. Con HTML definimos la estructura para situar el contenido en nuestra página web. HTML es un estándar creado por la W3C (*World Wide Web Consortium*), esta organización es la destinada a la estandarización de la mayoría de las tecnologías relacionadas con la web, sobre todo en referencia a la interpretación de los lenguajes.

HTML ha tenido diferentes versiones desde su creación en 1991. Actualmente estamos en la versión 5, que su primera versión salió a la luz en 2008, pero no fue hasta 2011, que empezó a extenderse su desarrollo, y empezó a ser soportado por la mayoría de los navegadores. Actualmente, junto con CSS3, como veremos en el siguiente apartado, es el lenguaje preferido por los desarrolladores dada la flexibilidad y funcionalidad que nos ofrece. Aunque todavía la mayoría de las webs están creadas en HTML4, por diversos motivos, el principal, es conocido por todos los desarrolladores web del mundo, y es la versión que más tiempo ha estado durante el auge de las páginas web. HTML5, por su parte, está creciendo de manera vertiginosa por la cantidad de funciones y mejoras que implementa además de conservar la nomenclatura de etiquetas de HTML añadiendo algunas propias.

Las principales mejoras de HTML5 respecto HTML4 son la incorporación de etiquetas (Canvas 2D, 3D, audio y video) con *codecs* para mostrar el contenido multimedia, etiquetas para el manejo de paquetes grandes de datos, mejoras en formularios con nuevos tipos de etiqueta que permite valida contenido sin usar javascript , y visores para gráficos vectoriales.

2 – CSS y CCS3.

Ahora vamos a hablar de las CSS (*Cascading Style Sheets*), cuya traducción literal es "Hoja de Estilos en Cascada". Las hojas de estilos, o CSS, es el lenguaje utilizado como complemento de HTML que nos permite mejorar y personalizar el formato de los datos en una página web. Al igual que en HTML, el lenguaje CSS está estandarizado por la W3C y esta publicó la primera versión del CSS (CSS1) en 1996. Esta primera versión solo permitía personalizar ciertos estilos como el tamaño y tipo fuente, colores de los textos, márgenes, bordes, etc. En 1998 la W3C pasó a recomendar CSS2, la cual duró hasta el año 2008, donde abandonó esta versión para recomendar la actual CSS3. CSS2 ampliaba la personalización de estilos, permitiendo dar funcionalidad de posicionamiento de las etiquetas " ", sombreados, *media types*, etc.

Actualmente se utiliza la tercera versión de CSS, la cual está dividida en varios documentos a los que llaman módulos. Cada uno de estos módulos, tiene como base el anterior, y mantiene la compatibilidad añadiendo mejoras de funcionalidad y eficiencia. En esta versión de CSS tenemos la posibilidad de crear funciones, mover elementos de manera dinámica sin necesidad de JavaScript además de otras, lo que la convierte sin duda en la mejor versión de hojas de estilos.

3 – JavaScript y JqueryMobile.

JavaScript, cuya abreviatura común es "JS", es un lenguaje de programación estándar utilizado para la gestión dinámica de atributos de los objetos en páginas web. Esta tecnología apareció en 1995 de la mano de un trabajador de Netscape, cuando intentaba agilizar las aplicaciones web ya que en esa época no se disfrutaba de un ancho

de banda como el de ahora. Pero su finalidad de uso se ha ido modificando hasta que ha llegado al punto de utilizarse en su gran mayoría para crear efectos y acciones en las páginas web actuales, agregando una mayor funcionalidad a las mismas. Más enfocado al desarrollo de aplicaciones, y más en PhonaGap, podríamos decir que JavaScript nos permite disfrutar de funcionalidad añadida. Sin esta tecnología, nuestro desarrollo sería mucho más limitado tanto gráficamente hablando, porque permite tener objetos dinámicos, como funcionalmente, ya que nos permite integrar funciones y llamadas a servicios externos a la aplicación.

La librería más conocida de JavaScript es jQuery, la cual se ha convertido en complemento de la gran mayoría de webs. Esta librería ofrece funciones ya creadas vía JavaScript, lo que reduce en tiempo y desarrollo y aumenta en funcionalidad tu aplicación. Por ellos, y para el desarrollo de portales móviles, jQuery ha desarrollado su propio *framework*, llamado jQuery-Mobile. Este *framework* nos provee de herramientas que nos permiten crear nuestra página de manera más sencilla, además de generar interfaces con una alta usabilidad y accesibilidad.

4 – Node.js

Node.js es un programa que se ejecuta en la parte del servidor, y es una interfaz para ejecutar JavaScript del lado del servidor de manera rápida y escalable, y está basado en el motor Javascript V8. Al estar desarrollado en lenguaje JavaScript, no requiere el aprendizaje de un nuevo lenguaje para desarrollar servicios. La finalidad principal de Node, es proporcionar una manera fácil para desarrollar programas que contengan las propiedad esencial de escalabilidad.

Para evitar confusiones, diremos que Node es un programa de servidor, pero no tiene nada que ver con Apache o Tomcat. Estos últimos son programas del preparados para ser instalados y proporcionar un servicio de manera automática e instantánea. Node no tiene esta finalidad como hemos mencionado en los párrafos anteriores.

5. Elección del *framework*

Una vez expuestos todos los *framework*s y sus respectivas características, vamos a elegir el que más nos conviene para desarrollar la aplicación. A priori, cualquiera podría servirnos para nuestro propósito, pero ya que tenemos que elegir uno, nos quedaremos con PhoneGap por distintos motivos. PhoneGap, cumple al 100% los requisitos que requeríamos, cosa que no ocurría con Rhodes, que cumplía dichos requisitos pero con ligeras discrepancias. Otro de los motivos es la utilización de tecnología tan extendidas y conocidas como son HTML, CSS y JavaScript, lo que nos permite un desarrollo más eficiente por el conocimiento avanzado que tenemos de las diferentes tecnologías empleadas. En Rhodes se emplea Ruby, del que no tenemos apenas conocimiento ni experiencia previa. Hasta aquí tanto PhoneGap como Enyo cumplen la similitud de características, por ello, Rhodes ha sido descartada previamente, y ahora tenemos que decidir entre PhoneGap y Enyo. Ambos tiene alguna ventaja respecto al otro, PhoneGap soporta la integración de bases de datos con SQLite, y permite desarrollar para todos los sistemas operativos actuales, por el contrario, aunque no tenga una gran relevancia, Enyo no permite desarrollar para Symbian, ni soporta bases de datos. Por contrapartida, Enyo tiene como ventaja sobre PhoneGap integra servicios web vía JSON, aunque con JavaScript es fácil complementar esta carencia en PhoneGap.

En definitiva, creemos que para el desarrollo de la aplicación que tenemos como objetivo, el mejor de los *framework*s es PhoneGap, por ello vamos a profundizar en dicho *framework*, y conocer en la medida de lo posible sus características, ya que esto nos ayudará a ser más eficiente y más completos en nuestro desarrollo, y así podremos llevar a cabo de manera metódica nuestra aplicación y poder alcanzar nuestro objetivo.

6. PhoneGap

6.1 Introducción a PhoneGap

PhoneGap es un *framework* implementado por Adobe, para el desarrollo de aplicaciones móviles multiplataforma haciendo uso de tecnologías web. Este *framework* ha tenido gran impacto en el mundo de los desarrolladores por su característica de desarrollo multiplataforma. Para los desarrolladores era y es un problema el tener que desarrollar la misma aplicación tantas veces como plataformas querían que su aplicación abarcara. Es decir, tenían que desarrollar la aplicación en lenguaje nativo de Android para poder exportarlo a esta plataforma, luego volver a realizar el desarrollo para la plataforma iOS si querían entrar en el mercado de Apple, y así sucesivamente para cada una de las plataformas existentes. Con PhoneGap, esto no ocurre, basta con desarrollar la aplicación una vez, con las tecnologías HTML, CSS y JavaScript, y una vez terminada, mediante el SDK correspondiente a cada plataforma, exportar la aplicación para cada una de ellas. Es por este motivo principal, por el que ha tenido tanta repercusión en el mundo del desarrollo de aplicaciones, y también en el mundo del desarrollo web, ya que son las tecnologías web las que se utilizan para desarrollar las aplicaciones.

6.2 Historia

PhoneGap es una distribución de Apache Cordova, un software de código abierto, desarrollado por la comunidad Apache y que puede ser considerado como el motor que impulsa a PhoneGap. Apache Cordova nació con el mismo objetivo con el que se utiliza ahora PhoneGap, el desarrollo de aplicaciones móviles, pero Adobe creó PhoneGap con la intención de mejorar y añadir funcionalidad y herramientas propias de su compañía. De momento, Adobe no ha añadido herramientas ni complementos, con lo que podemos decir que Apache Cordova i PhoneGap es el mismo software, pero con diferente nombre.

6.3 SDKs soportados y tecnologías requeridas

Como hemos nombrado en la descripción del *framework*, PhoneGap soporta sino todas, la mayoría de los sistemas operativos existentes, y por supuesto, todos los que tienen un nivel de utilización tal como para ser relevante para los desarrolladores.

A continuación nombramos todos y cada uno de ellos:

- Android.

- iOS.

- Windows Phone.

- BlackBerry OS.

- Web OS.

- Symbian.

- Bada.

Para poder desarrollar en cualquiera de las plataformas citadas, necesitamos el SDK *(Softare Developer Kit)* correspondiente, lo que nos proporciona, como podemos deducir de su notación en inglés, el *kit* de software de desarrollo correspondiente a la plataforma.

A partir de este momento, vamos a tener que tomar la decisión de para qué plataforma queremos desarrollar nuestra aplicación inicialmente. Aunque no variará demasiado con respecto a otras plataformas, pero tenemos que centrarnos en alguno en concreto para mostrar los pasos que vamos a seguir. En nuestro caso, vamos a desarrollar para Android, por motivos de popularidad, aunque como hemos comentado, no existen grandes diferencias si desarrollamos para otra plataforma, solamente las que nos exija el SDK correspondiente.

6.4 Instalación de PhoneGap

Antes de todo, vamos a instalaros en *framework* principal, para lo cual nos adentramos en la página web oficial del mismo, y en el apartado "Install" tenemos una breve explicación de cómo hacerlo.

Primero, nos indica que debemos tener instalado "Node.js" antes de instalar PhoneGap. Node.js es un intérprete JavaScript del lado del servidor cuyo objetivo es permitir al programador construir aplicaciones altamente escalables y cuyo código pueda manejar un alto número de conexiones.

Una vez instalado Node, pasamos a instalar PhoneGap. Para ello, basta con utilizar la línea de comandos siguiendo los comandos que nos indican si vamos a desarrollar desde un sistema operativo Linux o MacOS, si por el contrario vamos a desarrollar desde un sistema Windows, tendremos que descargar el paquete desde la misma interfaz para instalarlo.

Nosotros al desarrollar desde un sistema MacOS, introducimos el siguiente comando:

sudo npm install -g phonegap

Como podemos ver en las capturas siguientes, automáticamente se descarga el paquete y se instala en nuestro ordenador:

```
npm http 200 https://registry.npmjs.org/hoek/0.8.5
npm http GET https://registry.npmjs.org/hoek/-/hoek-0.8.5.tgz
npm http 200 https://registry.npmjs.org/async/-/async-0.2.10.tgz
npm http 200 https://registry.npmjs.org/hoek/-/hoek-0.8.5.tgz
npm http GET https://registry.npmjs.org/hoek/0.9.1
npm http 200 https://registry.npmjs.org/hoek/0.9.1
npm http GET https://registry.npmjs.org/hoek/-/hoek-0.9.1.tgz
npm http 200 https://registry.npmjs.org/hoek/-/hoek-0.9.1.tgz
/usr/local/bin/phonegap -> /usr/local/lib/node_modules/phonegap/bin/phonegap.js
phonegap@3.5.0-0.20.4 /usr/local/lib/node_modules/phonegap
├── pluralize@0.0.4
├── colors@0.6.0-1
├── semver@1.1.0
├── qrcode-terminal@0.9.4
├── shelljs@0.1.4
├── optimist@0.6.0 (wordwrap@0.0.2, minimist@0.0.10)
├── prompt@0.2.11 (revalidator@0.1.8, pkginfo@0.3.0, read@1.0.5, winston@0.6.2, utile@0.2.1)
├── phonegap-build@0.8.4 (qrcode-terminal@0.8.0, optimist@0.3.7, shelljs@0.0.9, phonegap-build-api@0.3.3)
├── connect-phonegap@0.11.0 (home-dir@0.1.2, connect-inject@0.3.2, shelljs@0.2.6, request-progress@0.3.1, node-static@0.7.0, useragent@2.0.8, tar@0.1.19, gaze@0
.4.3, request@2.33.0, connect@2.12.0)
├── cordova-lib@0.21.4-dev (osenv@0.0.3, properties-parser@0.2.3, bplist-parser@0.0.5, mime@1.2.11, q@0.9.7, semver@2.0.11, underscore@1.4.4, dep-graph@1.1.0, p
list-with-patches@0.5.1, glob@3.2.11, npmconf@0.1.16, tar@0.1.19, rc@0.3.0, elementtree@0.1.5, xcode@0.6.6, request@2.22.0, npm@1.3.4)
└── cordova@3.5.0-0.2.4 (q@0.9.7, underscore@1.4.4, cordova-lib@0.21.3)
```

Ahora que ya tenemos instalado el *framework* en nuestro ordenador, tenemos la funcionalidad para poder crear proyectos iniciales de aplicaciones. Nuestro proyecto se va a llamar "app-tfg", así que crearemos un proyecto para desarrollar nuestra aplicación con ese nombre mediante el siguiente comando:

phonegap create app-tfg

Y así nos creará una nueva carpeta con ese nombre a partir de la ruta en la que nos encontremos.

Una vez completados estos pasos, ya tenemos un proyecto creado con el contenido necesario para desarrollar una aplicación para cualquiera de las plataformas citadas en la descripción del *framework*. Así que vamos a pasar a instalar el SDK de la plataforma, y para ellos nos vamos a guiar por la documentación que nos ofrece el propio *framework* en una de las secciones de su página web (http://docs.phonegap.com/en/edge/guide_platforms_index.md.html). En esta sección de la página web de PhoneGap encontramos guías para todas y cada una de las plataformas disponibles sobre las que podemos desarrollar aplicaciones.

Así pues, vemos que para completar el desarrollo y que se pueda utilizar en un dispositivo con sistema operativo Android necesitamos estos dos complementos:

- El software Eclipse.

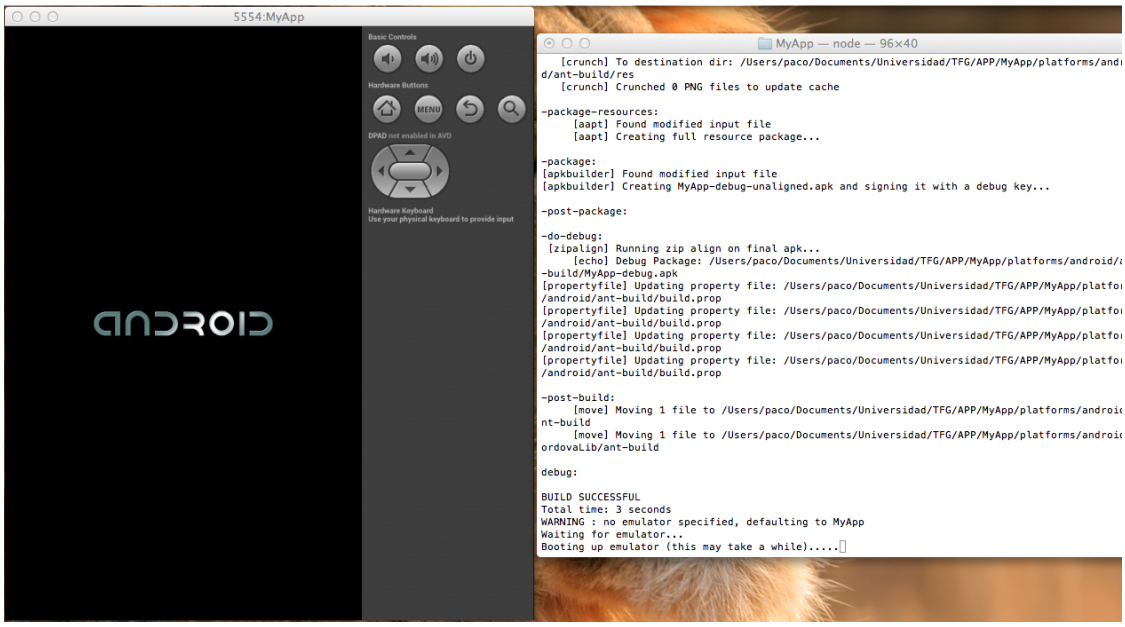
- El *plugin* ADT (Android Developer Tools) para Eclipse.

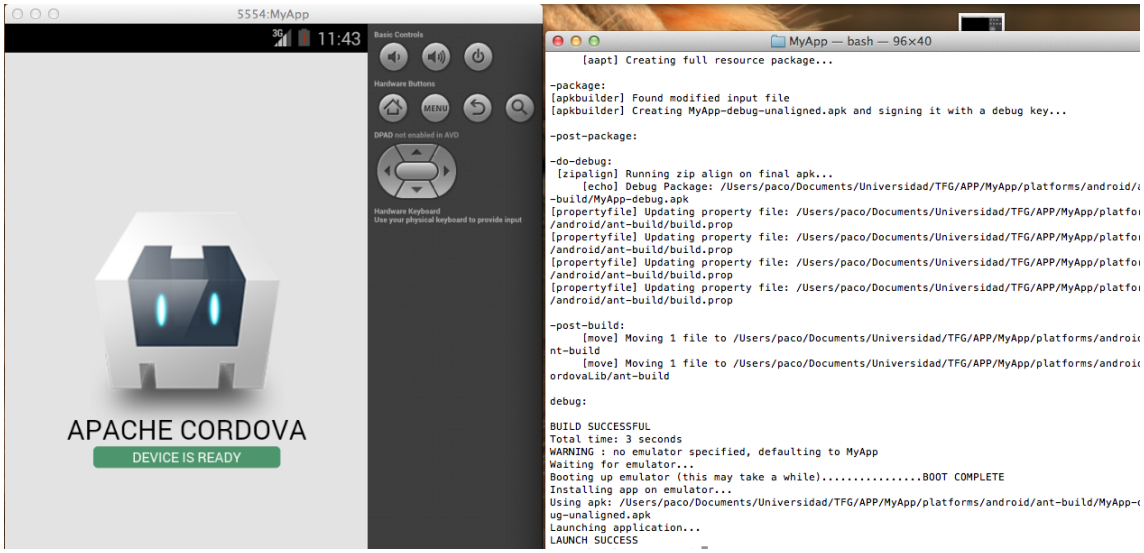
Una vez tenemos instalados este software, entramos en la carpeta creada por la aplicación PhoneGap cuyo nombre es “www”, donde se crea una aplicación de ejemplo, y tratamos de ejecutar con el siguiente comando:

phonegap emulate android

Este comando lanza la aplicación sobre el emulador de Android que hemos instalado, y podremos ver la aplicación de ejemplo que se genera de manera automática al crear inicialmente la aplicación.

Primero se lanzará el simulador de Android, y luego cargará la aplicación, tal como vemos a continuación:





Si quisiéramos desarrollar nuestra aplicación para iOS también, en la misma URL podemos encontrar los requisitos que Apple nos solicita para desarrollar aplicaciones que funcionen en su plataforma, que son los siguientes:

- El SDK de Apple, Xcode.

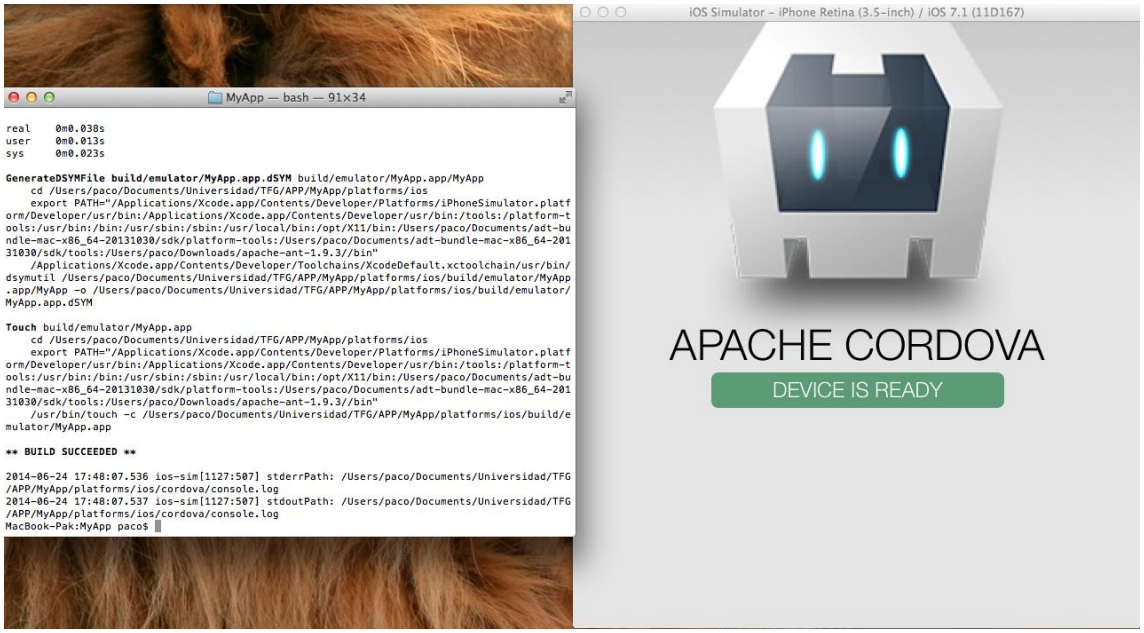
- Portátil/Ordenador con sistema operativo MacOS.

En la primera captura vemos en la parte izquierda el simulador iniciándose y en la parte derecha la ejecución dentro de la consola. Si nos fijamos en la primera captura, podemos ver mientras como carga el simulador, aparece el texto “BUILD

SUCCESSFUL", que indica que el simulador ha arrancado sin errores y va a iniciarse sin problemas de compilación.

En la segunda captura, vemos como ha terminado de ejecutar el simulador, y arrancado la aplicación de pruebas sin problemas.

Para comprobar que PhoneGap permite el desarrollo para varias plataformas, vamos a mostrar el mismo proceso para mostrar la ejecución en iOS, que junto con Android son las dos plataformas que ocupan el 80% del mercado. Con el fin de lanzar la aplicación de prueba en iOS, necesitamos, al igual que ocurría con la plataforma Android, necesitamos el software de desarrollo para aplicaciones iOS. De acuerdo con la documentación oficial, es software requerido es el programa xCode. Y al lanzar la sentencia (En este caso cambiamos la palabra "android" por la palabra "ios"), nos mostrará lo siguiente:



6.5 Build PhoneGap

Como hemos comentado en "Introducción al *framework* Phonegap", Phonegap ha sido desarrollado por Adobe Systems, lo cual lo dota de una importancia mayor para todos los que conocen las herramientas de Adobe. Para respaldar dicho *framework*, Adobe ha desarrollado una herramienta muy interesante con el nombre de "Phonagap Build". "Phonegap Build" es una servicio basado en la nube, construido sobre Phonegap y que permite desarrollar aplicaciones sobre este mismo *framework* a través de internet. Esta herramienta, simplemente requiere un fichero con la extensión ".zip" que contenga los ficheros con el código HTML, JavaScript y CSS necesarios para el desarrollo de la aplicación. Una vez consigamos empaquetar esos archivos, solo tenemos que entrar en https://build.phonegap.com y subir nuestro paquete.

No obstante, Phonegap Build nos ofrece otro modo de subir nuestros archivos para construir nuestra aplicación. Para ello requiere tener un repositorio de tipo "git", ya sea GitHub o cualquier otro similar. Tanto para un método como para otro, se requiere tener una cuenta en Adobe, la cual se puede obtener accediendo a la propia página de Adobe. Con esta herramienta, evitamos tener que instalar todos los SDKs de las plataformas para las que queremos desarrollar en nuestro ordenador, ya que Phonegap Build nos generará todos y cada uno de los instaladores. Pero por contrapartida, para probar estas aplicaciones y ver que funcionan tal y como queremos en los dispositivos finales, necesitaremos tener un terminal que incorpore dichas plataforma.

Como cabía esperar, una empresa tan potente como es Adobe, no iba a desarrollar una herramienta sin poder obtener algún beneficio económico de ello. Para sacar el mayor rendimiento posible, Adobe ha creado dos tipos de cuentas para acceder a esta herramienta. La primera, es tal como hemos mencionado, teniendo una cuenta Adobe básica, la cual es gratuita, y con ella ya tendríamos acceso a la herramienta, pero esta cuenta tiene una cantidad de aplicaciones máxima, de las cuales tan solo una de la aplicaciones puede ser privada, el resto serán aplicaciones *open-source*, es decir, su código puede ser visto por cualquiera. Eso sí, puedes tener tantas como quieras.

Si estas condiciones no nos benefician, y queremos poder tener hasta un total de veinticinco aplicaciones privadas, podemos conseguirlo por 9,99 dólares mensuales.

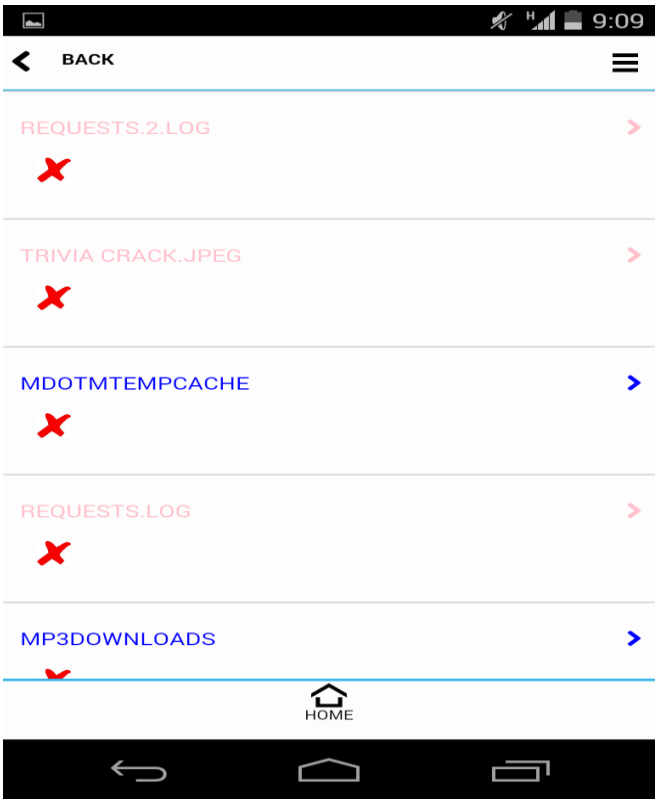
7. Esquema de la aplicación

Como hemos descrito al inicio de la memoria, nuestra aplicación va a tener distintas funcionalidades. La primera y principal, será la gestión de los archivos almacenados en nuestro dispositivo, y como ampliación para gestionar la gran mayoría de nuestros documentos, acceso a nuestras plataformas de almacenamiento en la nube tales como DropBox. Para hacer esta aplicación más agradable y funcional para el usuario, la vamos a distribuir de manera que al abrir la aplicación, nos aparezca el menú principal, donde tengamos separados por ítems de menú, todas y cada una de las funcionalidades, como se muestra a continuación:

Ahora vamos a describir la funcionalidad extendida de cada uno de los ítems del menú.

Mis Archivos

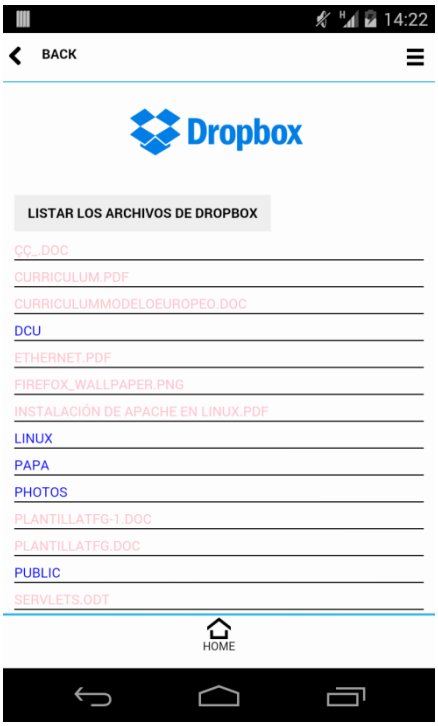
Cuando pulsemos sobre este ítem, el dispositivo nos va a mostrar todos los archivos y carpetas que tengamos en el directorio raíz del mismo. Una vez situados en el directorio raíz, el dispositivo nos permitirá navegar a través de nuestras carpetas y archivos, accediendo a los contenidos si pulsamos sobre una carpeta, o abriéndolo si hemos pulsado sobre un archivo.



Mi DropBox

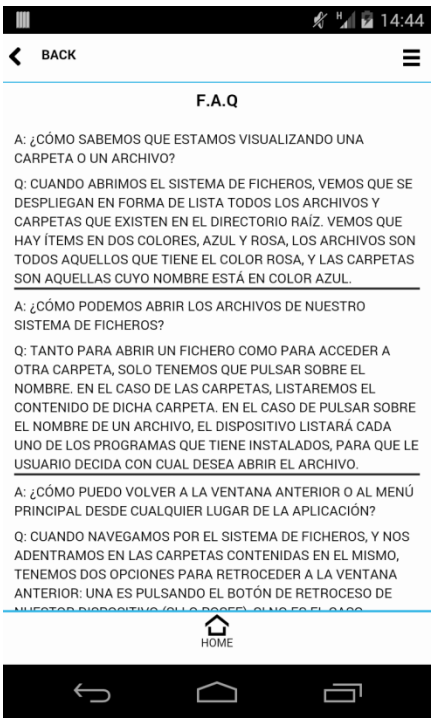
En este ítem de menú, accedemos a nuestra plataformas de almacenamiento en la nube, donde se listarán todos los archivos y carpetas de nuestra cuenta de Dropbox. Además de listar podemos abrir y borrar todos aquellos archivos que queramos.

Esta acción requerirá una conexión a internet previa, por el contrario la aplicación mostrará un mensaje de error donde solicitará una conexión.



F.A.Q

En este ítem (Totalmente exento de funcionalidad técnica), tiene un carácter informativo del uso y funcionalidad de la aplicación. La finalidad de este ítem es conectar con el usuario que ha obtenido la aplicación y solventarle cualquier tipo de duda que pueda surgir durante su uso.



8. Funcionalidad

Menú inicio

El menú de inicio estará formado por cuatro enlaces (uno por cada sección). Cada uno de estos ítems, nos llevará a las funcionalidades definidas a en esta sección. El conjunto de enlaces está implementado con un código simple de HTML y CSS, no requiere conocimientos de JavaScript, Phonegap u otras APIs de desarrollo externas. Los ítems, serán elementos de tipo botón, utilizamos este tipo de elemento por la usabilidad que tiene. Incita al usuario a pulsar sobre ellos de manera automática, sin necesidad de indicarle que ese elemento tiene la utilidad de interactuar con el dispositivo.

Explorador de archivos

En esta sección, vamos a tener que adentrarnos en las funciones propias del móvil, ya que queremos ver los archivos y carpetas almacenados en el dispositivo. Para ello, necesitamos tener conocimientos de phonegap, y de los *plugins* que implementa. Para conocer las funciones y partes del dispositivo que hay disponibles para su acceso, tenemos que adentrarnos en la documentación oficial del *framework* phonegap, que se encuentra en la página oficial del mismo. En la documentación, podemos ver, por una parte, los *plugins* oficiales desarrollados por Apache. Estos *plugins*, son una especie de librerías que ofrece el *framework* para acceder de manera nativa a las diferentes funciones del dispositivo para el que desarrollamos, como puede ser el sistema de ficheros, la cámara, las características del dispositivo, etc.

El plugin desarrollado por apache cordova para poder alcanzar nuestro objetivo , es el plugin “File”. Este plugin es nativo del *framework*, y como se puede deducir de su nombre, tiene como finalidad dar acceso en cualquiera de las plataformas soportadas por el *framework* al sistema de ficheros raíz. Dentro de este plugin, encontramos diferentes objetos, entre ellos, vamos a utilizar los siguientes para gestionar los archivos del dispositivo:

- DirectoryReader: Este objeto permite listar los archivos y directorio, dentro de un directorio. Solo contiene un método, llamado “readEntries”, que lista las entradas de un directorio.

- DirectoryEntry: Representa un directorio en el sistema de ficheros. Este objeto tiene diferentes propiedades para trabajar con el contenido del directorio: “isFile”, para saber si el ítem listado es un archivo; “isDirectory”, para saber si el ítem listado es un directorio; “name”, para conocer el nombre del directorio en el que nos encontramos, exceptuando el resto de la ruta; “fullPath”, que devolverá la ruta completa del directorio.

- File: El objeto “File” contiene los atributos de un archivo. Y con las propiedades “name”, “fullPath”, “type”, “size” y “lastModifiedDate” podemos trabajar con cualquiera de los archivos a los que tengamos acceso.

-FileError: “FileError” es un objeto que se ejecuta cuando ocurre un error en la ejecución de los métodos de un archivo.

- LocalFileSystem: Este objeto proporciona el modo de obtener el archivo raíz del sistema.

Añadiendo el *plugin* “File”, y utilizando sus objetos y funciones, ya podemos acceder al sistema de ficheros del dispositivo, y a partir del fichero raíz, navegar a través del sistema. Al implementar las funciones necesarias, conseguimos entrar en las carpetas existentes, retroceder a través de las mismas, abrir los archivos contenidos en el sistema o eliminar los archivos que no queramos contener en nuestro dispositivo.

Acceso a la plataforma de almacenamiento en la nube

La gestión de los archivos almacenados en la nube, es totalmente diferente al modo de gestionar los archivos del propio dispositivo. Así como en la sección anterior utilizábamos el *framework* Phonegap para acceder al sistema de archivos del dispositivo, en la nube, tenemos que acceder a cada plataforma haciendo uso de la API de cada una de las plataformas. Como estamos hablando de una aplicación

experimental, vamos a desarrollar un ejemplo para acceder de manera sencilla a la plataforma más conocida que existe, aunque no diferirá la metodología para otras similares. En nuestro caso, al utilizar un *framework* basado en tecnologías web, tenemos que buscar la API de Javascript, que es la que mejor se adapta al desarrollo de nuestra aplicación.

Como queremos desarrollar los accesos para la plataforma Dropbox, vamos a la página oficial de la plataforma (http://www.dropbox.com), y buscamos los métodos de acceso remoto a la plataforma para poder implementar la mejor opción en nuestra aplicación. En el caso de Dropbox, hemos encontrado en la siguiente URL la documentación necesaria para implementar las funciones de nuestra aplicación:

https://www.dropbox.com/developers/datastore/docs/js

En esta URL nos proporciona la documentación, métodos y parámetros suficientes para implementar las funciones que necesitamos en nuestra aplicación. Antes de todo, nos indica que necesitaremos una librería desarrollada por la propia plataforma, la cual nos abstraerá de algunas funciones más complejas. Para ello, solo es necesario incluir la siguiente línea de código en el apartado "<head>" de nuestra aplicación.

```
<script src="https://www.dropbox.com/static/api/dropbox-datastores-1.1-latest.js" type="text/javascript"></script>
```

Para poder trabajar con las funciones facilitadas, primero tenemos que crear un objeto del tipo "Client", que nos permitirá implementar las funciones como cliente. Una vez instanciado el objeto, podemos utilizar todas las funciones necesarias para listar nuestros archivos, navegar por cada una de nuestras carpetas, abrir los archivos que nos interesen, borrar los que no queramos tener almacenados, etc.

Haciendo hincapié en que es una aplicación experimental, y vamos a implementar las funciones básicas, a continuación vamos a indicar brevemente que métodos tiene definidos la API:

"***readdir***()": Esta función nos permite listar la carpeta que le pasamos en los parámetros de entrada.

"***remove***()": Con esta función eliminamos el archivo cuya ruta corresponde a la pasada por parámetros.

"***readFile***()": Recupera la información de un archivo que indicamos por parámetro.

Para realizar la conexión con la plataforma, hemos intentado implementarlo de diferentes modos. Inicialmente utilizamos los llamados "dropins", que se utilizan para insertar la interfaz de dropbox en tu propia aplicación web. Para implementar dicho código, hemos desplegado un servidor de pruebas local, donde hemos creado un archivo ".html" con contenido javascript, el cual muestra la pantalla de *login* similar a la que encontramos en la página oficial de la plataforma. Con este desarrollo, nuestra funcionalidad estaba más que cubierta, pero nos encontramos con un inconveniente a la hora de desplegarlo en un dispositivo, como un móvil o una tablet, y es que al no ser lanzado desde un servidor, no podíamos tratar la respuesta a la petición sobre el servicio, ya que dicho servicio está preparado para ser ejecutado desde el lado del servidor.

Como esta primera opción, que parecía la más simple de implementar, no servía para nuestra solución, buscamos que otras opciones nos ofrecía la plataforma para acceder desde dispositivos simples, que no incorporan un servidor. La otra opción que encontramos para solventar la cuestión era crear una *app* desde nuestra interfaz de dropbox y utilizando la "APP key" que se genera dentro de nuestro Dropbox, y con ello acceder a nuestras carpetas con las funciones que nos facilita el API de javascript de Dropbox descritas a lo largo de esta sección. El único inconveniente que tiene este tipo de acceso, es que para cada usuario que quiera acceder a su Dropbox, tenemos que generar una "APP key" en su dropbox y luego incluirla en la aplicación.

F.A.Q

Por último, tenemos la sección de preguntas frecuentes de la aplicación. Esta sección tiene la simple finalidad de facilitar al usuario la utilización de la aplicación, y responder a las preguntas más habituales sobre el funcionamiento de las secciones implementadas.

Esta sección, al igual que sucedía con la primera sección, está desarrollada con código HTML y CSS. No ha sido necesaria la utilización de código JavaScript, lo que indica que su desarrollo ha sido mucho más sencillo y rápido.

9. Estética de la aplicación

Una vez conseguida la funcionalidad que deseábamos, vamos a utilizar **Intel App Framework** para dar un formato visual a la aplicación. Intel App Framework es una librería que está basada en JavaScript, esta librería está diseñada para aplicaciones móviles basadas en HTML5. Los estilos de esta librería, están diseñados para adaptarse a los estilos de las principales plataformas móviles como son Android, iOS, Windows Phone,etc. Lo que te permite desarrollar una aplicación adaptada a los estilos de la plataforma que más te convenga. Además, esta librería incluye una librería especial de selectores, que se integra con las características de HTML5 y CSS3 para ser soportado por los navegadores.

App *Framework* está compuesta por tres partes diferenciadas:

- Librería de consulta de selectores.

- Librería UI/UX para tablet y smartphone.

- *Plugins* desarrollados sobre App *Framework*

App *Framework* fue construida desde la base para ser uno de los *framework*s más destacados dado que es muy ligera i rápida, esto hace que tome ventaja sobre otros *framework*s por su utilización sobre navegadores de internet.

App *Framework* UI, también conocido como AFUI, es el único *framework* para móviles que soporta Android, iOS, Blackberry y Win8 del mismo modo. Es decir, AFUI no afecta su rendimiento cuando resuelve conflictos entre plataformas. Por ejemplo, la mayoría de los *framework*s consideran que Android tiene bugs en algunas de sus versiones , e ignora las versiones específicas de este sistema operativo o aplican métodos alternativos que no ofrecen un rendimiento adecuado. App *Framework* resuelve esta cuestión para proporcionar una interfaz limpia y *responsive* de HTML5.

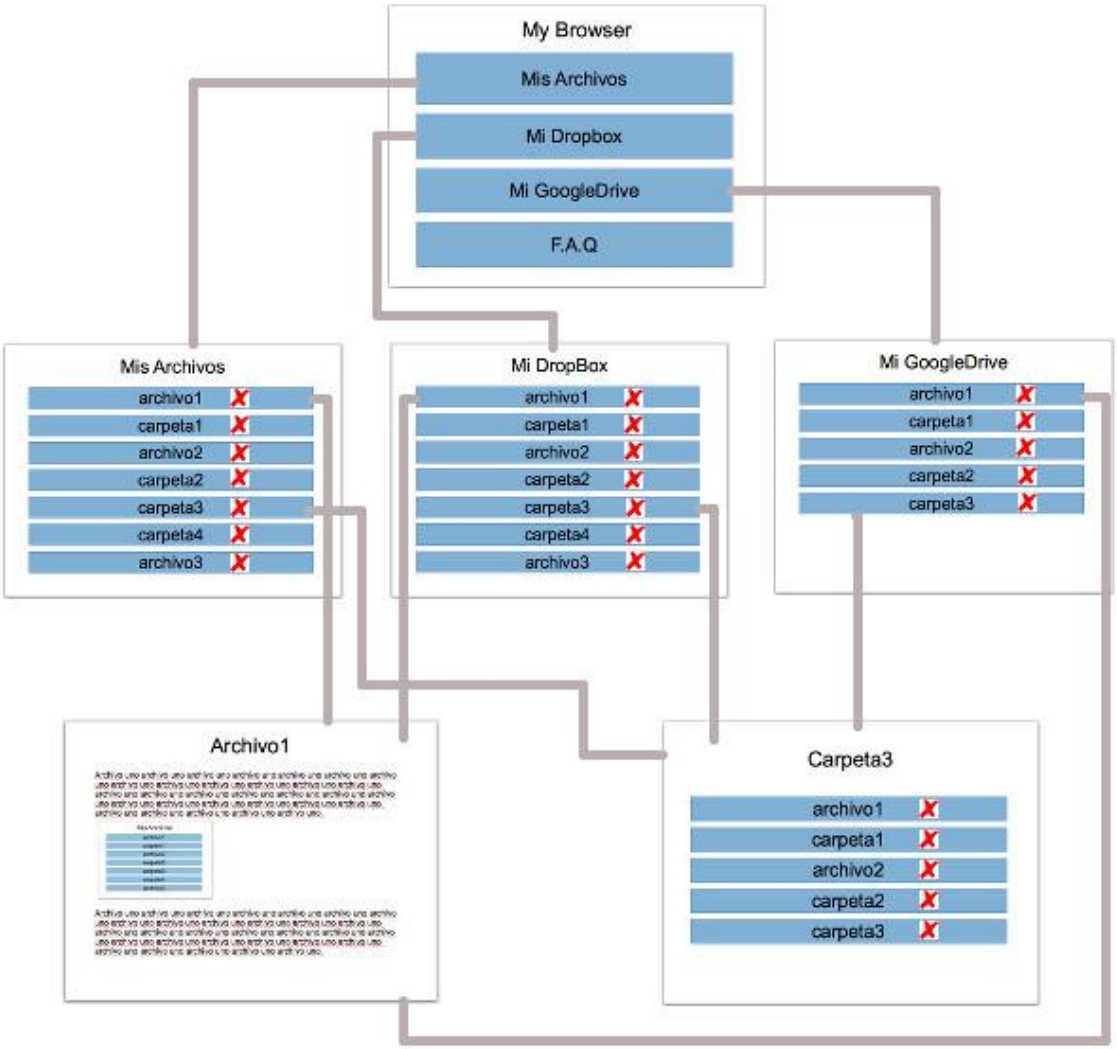
Los *Plugins* están separados de las librerías de JavaScript, creadas para facilitar el uso de App *Framework* y conseguir una mayor cantidad de aplicaciones desarrolladas

con este *framework*. Los *plugins* proporcionan unas características y/o funcionalidad que están fuera del alcance de otras aplicaciones más básicas.

Además de aprovechar los estilos que incluye esta librería, haremos uso de CSS3 para agregar algunos estilos que creamos convenientes con el fin de hacer la aplicación más intuitiva y útil para el usuario.

10. Modelado UML

10.1 Primera aproximación del diseño de la aplicación



10.2 Casos de uso

ID	1
Caso de Uso	Acceso los archivos del dispositivo.
Actores	Usuario.
Propósito	Acceder a los documentos almacenados en el sistema.
Resumen	El sistema le muestra al usuario los documentos y carpetas de la raíz del sistema, y a través de esa interfaz el usuario accede a los documentos del dispositivo.
Precondición	-
Postcondición	-

ID	2
Caso de Uso	Acceso a una plataforma de almacenamiento en la nube.
Actores	Usuario.
Propósito	El usuario consulta sus archivos en la nube.
Resumen	El usuario, a través de su usuario y contraseña de una de las plataformas sugeridas, accede a su cuenta y sus archivos para poder consultarlos directamente a través de la aplicación.
Precondición	Introducir el usuario y la contraseña de la plataforma.
Postcondición	-

ID	3
Caso de Uso	Abrir una carpeta.
Actores	Usuario.
Propósito	Acceder al contenido de una carpeta.
Resumen	El usuario selecciona una carpeta para que el sistema liste el contenido de esa carpeta.
Precondición	-
Postcondición	El usuario cambia de ruta, para situarse en la carpeta seleccionada.

ID	4
Caso de Uso	Abrir un documento.
Actores	Usuario.
Propósito	Leer la información de un documento.
Resumen	El usuarios selecciona un documento y el sistema le muestra la información contenida en el documento seleccionado.
Precondición	- Seleccionar un documento.
Postcondición	El usuario se encuentra dentro de un documento, cambia su ruta.

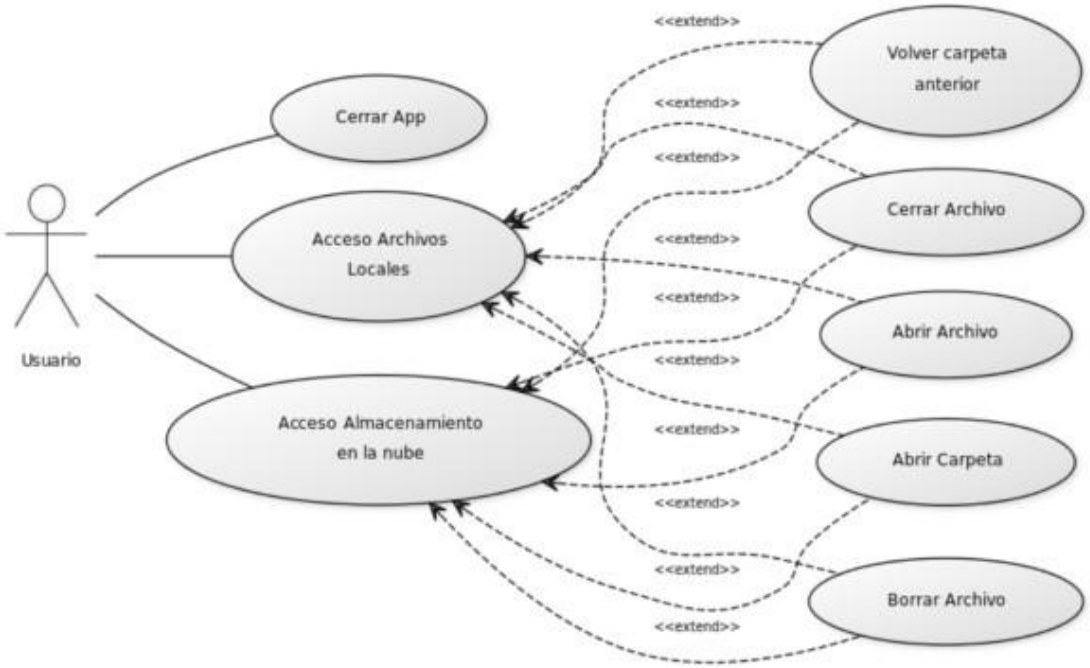
ID	5
Caso de Uso Actores Propósito	Borrar un documento. Usuario El usuario elimina un documento del sistema.
Resumen Precondición	El usuario usuario selecciona un documento y pulsa la opción de "borrar documento", y el sistema elimina ese documento, haciéndolo desaparecer de la lista. - Seleccionar el documento.
Postcondición	El documento ya no está disponible.

ID	6
Caso de Uso	Cerrar un documento
Actores	Usuario.
Propósito	Cerrar un documento.
Resumen	El usuario ya no desea más información del documento que tiene abierto, y quiere cerrarlo.
Precondición	Abrir un documento.
Postcondición	Vuelve a la ruta donde se encontraba antes de abrir el documento.

ID	7
Caso de Uso	Mover un documento.
Actores	Usuario.
Propósito	Cambiar un documento de directorio.
Resumen	El usuario desea cambiar de directorio un archivo/documento. El usuario selecciona el documento e indica la nueva ruta del documento seleccionado.
Precondición	Seleccionar el documento.
Postcondición	El documento cambia su ruta.

ID	8
Caso de Uso	Seleccionar documento
Actores	Usuario.
Propósito	Actuar sobre un documento.
Resumen	El usuario selecciona un documento, con la posibilidad de realizar sobre él algunas de las acciones ofrecidas (“Borrar”, “Abrir”, “Mover”).
Precondición	-
Postcondición	-

10.1 Diagrama de casos de uso



11. Conclusiones

Como conclusiones de este trabajo de final de grado, cabe destacar, entre otras cosas que indicaremos a continuación, la experiencia y aprendizaje tanto en la investigación de nuevas tecnologías y herramientas, como en el desarrollo de aplicaciones móviles multiplataforma y la gestión de archivos en las plataformas utilizadas. Ha sido un trabajo muy fructífero para aumentar mis conocimientos en un campo que no se profundiza demasiado durante los cuatro cursos del grado, y que ha cogido mucha fuerza durante los últimos años al aumentar la potencia tanto en *smartphones* como en *tablets*.

En cuanto al alcance del trabajo, diremos que hemos conseguido prácticamente todos los objetivos planteados, ya que la aplicación realiza todas las acciones detalladas inicialmente. Cierto es que hemos dejado la estética un poco de lado ya que no era lo más relevante en nuestro trabajo, hemos decidido hacer más hincapié en la funcionalidad ya que estamos trabajando en una aplicación experimental cuyo objetivo es gestionar archivos. También nos encontramos en un punto, que a pesar de tener una funcionalidad que hemos considerado básica, tiene por delante multitud de funciones y posibilidades a explotar, tanto en la gestión de archivos almacenados en el dispositivo, como en los almacenados en cualquier plataforma en la nube.

Como hemos comentado en algunos puntos anteriores, hemos desarrollado una aplicación experimental y de pruebas, lo que ha requerido mucho tiempo de investigación de todas las herramientas utilizadas, y eso ha perjudicado en otros aspectos como puede ser la estética. También hemos necesitado una gran parte del tiempo para entender la documentación de la plataforma Dropbox, que en mi opinión podría ser mucho más sencilla y menos costosa de utilizar. Y no hemos tenido tiempo suficiente para implementar la gestión de archivos almacenados en otra de las grandes plataformas que existe, Google Drive.

12. Posibles mejoras

Este proyecto, por su finalidad, y por el tiempo que exige toda la documentación, podemos hablar de que se encuentra en una fase experimental, es decir, que la mayoría de las características y diseños de la aplicación son mejorables. En este apartado vamos a comentar las cosas más importantes en las que podemos hacer hincapié para conseguir que sea una aplicación profesional y alcanzar la mayor cantidad de usuarios.

Una de las cosas más importantes para crear una aplicación móvil profesional, es el estudio de usabilidad de este tipo de aplicaciones. Para ello buscaríamos libros o artículos dedicados a ese fin, que estudian cómo distribuir los elementos de manera que sean más intuitivos y fáciles de usar para el usuario.

Algo imprescindible para conseguir que la aplicación tenga una buena valoración, sería aumentar la cantidad de acciones sobre los archivos que mostramos. Al ser una aplicación experimental, nos hemos limitado a desarrollar las acciones básicas, pero sería muy interesante introducir otras muchas para poder gestionar los archivos sin ninguna restricción al respecto.

También sería una buena práctica no mostrar archivos catalogados como “ocultos” o archivos correspondientes al sistema, y que una alteración de los mismos podría afectar al funcionamiento del sistema.

13. Bibliografía y referencias

http://en.wikipedia.org/wiki/Multiple_phone_web-based_application_*framework*

16 Junio,

Posibles *framework*s y características

http://www.phonegapspain.com/acerca-de/

17 Junio,

Información acerca del *framework* Phonegap.

http://enyojs.com/

17 Junio

Información acerca del *framework* Enyo.

http://www.motorolasolutions.com/US-EN/RhoMobile+Suite/Rhodes

17 Junio,

Información acerca *framework* Rhodes.

http://jordisan.net/blog/2006/que-es-un-*framework*/

18 Junio,

Información sobre qué es un *framework.*

http://blog.etips.cl/diferencia-entre-phonegap-apache-cordova/

18 Junio,

Información sobre Phonegap.

http://www.desarrolloweb.com/manuales/aplicaciones-moviles-phonegap.html

20 Junio,

Ejemplos y aplicaciones en Phonegap

http://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

18 Junio,

Instalación de PhoneGap, tutorial.

http://nodejs.org/

18 Junio,

Información sobre NodeJS

http://es.wikipedia.org/wiki/HTML

19 Junio,

Información sobre la tecnologías web, en concreto HTML.

http://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

19 Junio,

Información sobre la tecnologías web, en concreto CSS.

http://www.desarrolloweb.com/javascript/

19 Junio,

Información sobre la tecnologías web, en concreto JavaScript.

http://librosweb.es/javascript/capitulo_1/breve_historia.html

19 Junio,

Historia de JavaScript como tecnología.

http://sergioglez.webcindario.com/cargarArticulo.php?id=47

19 Junio,

Información sobre la tecnologías web, en concreto jQueryMobile.

http://www.movilwe.com/estadisticas-del-consumidor-movil/

25 Junio,

Información sobre los *smartphones*.

http://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/

25 Junio,

Información sobre los *smartphones*.

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EmpezandoPhoneGap

30 Junio,

Desarrollando de aplicaciones con Phonegap)

https://github.com/markeeftb/FileOpener

2 Julio,

Documentación *plugin* para la apertura de archivos en dispositivos móviles.