Herramientas para desarrollo multiplataforma

Pau Ballada

PID_00209909

UOC
Universitat Oberta de Catalunya

Índice

1. Introducción al lenguaje HTML5+Javascript+CSS3

Para desarrollar aplicaciones multiplataforma de navegador, se utiliza el mismo lenguaje empleado para programar las páginas web. Este ha ido evolucionando, de un tiempo a esta parte, y actualmente consta de tres pilares fundamentales: el lenguaje HTML5, el lenguaje Javascript y el lenguaje CSS.

A la hora de desarrollar una web, uno de los problemas más importantes que ha de encarar el programador es la diversidad de navegadores diferentes y la problemática para que nuestra web se vea correctamente en todos ellos (y en todo el abanico de versiones en cada uno de ellos), puesto que estos utilizan diferentes motores de renderizado para interpretar las páginas web.

El motor de renderizado es un componente de software que toma por un lado el contenido en HTML, XML, archivos de imágenes, etc.), y por el otro, la información de formato (en CSS, XSL, etc.) y muestra directamente el contenido con el formato ya aplicado en la pantalla.

Firefox utiliza el Gecko, mientras que Chrome, Safari y Opera, el WebKit, e Internet Explorer, el Trident. Esta diversificación representa una gran dificultad a la que se tienen que enfrentar los desarrolladores web, que tienen el reto de que sus productos puedan dar soporte a todos los navegadores, pues muchas veces una web no se visualiza del mismo modo en diferentes navegadores o directamente deja de funcionar.

Afortunadamente, en el mundo de las aplicaciones de los dispositivos móviles, hasta el día de hoy, las dos plataformas más importantes, iOS y Android, han utilizado el mismo motor de renderizado, el WebKit. Actualmente, el WebKit es el más extendido y ampliamente utilizado por la mayoría de navegadores en dispositivos móviles como Chrome, Safari, Opera, etc. No obstante, Google ha anunciado recientemente que dejará de emplearlo para utilizar su propio motor, denominado Blink. Por ahora son bastante similares, pero hay que prever que en el futuro puedan existir diferencias o tengan un funcionamiento distinto, provocando diferentes resultados.

La aparición de FireFox OS con el motor Gecko, y Windows Phone con el Trident acentúa todavía más la futura fragmentación de los navegadores en el mundo móvil.

Por suerte, las herramientas y librerías de desarrollo móvil también están experimentando un rápido crecimiento, sumado a la experiencia de fragmentación ya existente en los mundos de los navegadores de escritorio; cada vez las herramientas y librerías darán soporte a más plataformas móviles y motores de navegación.

1.1. HTML5

Esta es la última revisión del lenguaje HTML, que se encarga de definir los elementos y estructuras de las diferentes páginas o pantallas.

Esta última revisión del lenguaje HTML ha puesto al alcance de los programadores una gran cantidad de nuevas funciones y librerías que nos permitirán desarrollar aplicaciones para el mundo móvil, hasta ahora imposibles de realizar con las anteriores revisiones de HTML.

Algunas de estas nuevas características son la posibilidad de incorporar vídeo o audio a las páginas sin utilizar ningún plugin como el Flash. También ha incluido el soporte del canvas, de los gráficos SVG, drag&drop, Websockets, navegación offline, geolocalización y muchas más librerías, como las que se desglosan a continuación.

Enlace recomendado

Podemos ver el soporte de estas en relación con las diferentes plataformas móviles en el enlace: http://mobilehtml5.org/

Multimedia API

Esta librería nos permite reproducir archivos de vídeo y audio sin necesidad de ningún plugin, utilizando los tags de <video> y <audio>.

Geolocation API

Se trata de un nuevo estándar para pedir la posición del cliente, a través del navegador.

http://en.wikipedia.org/wiki/w3c_geolocation_api

Canvas API

Esta librería nos permite dibujar directamente elementos mediante el tag <canvas>; se utiliza Javascript para realizar las acciones de dibujo.

http://en.wikipedia.org/wiki/canvas_element

Web Storage API

Esta librería nos permite guardar información de forma transparente en el navegador; es una evolución de las cookies.

http://en.wikipedia.org/wiki/web_storage

WebSockets API

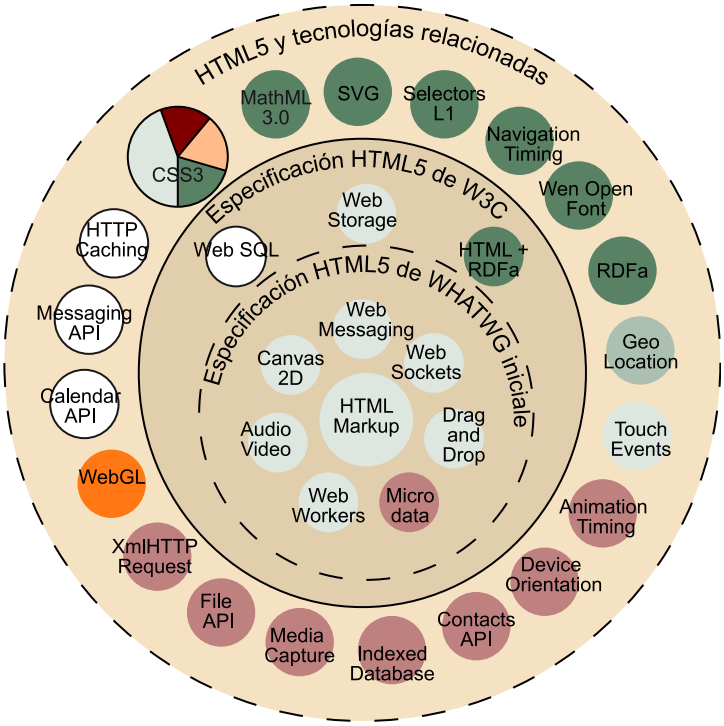
Los websockets son una nueva manera de comunicarse cliente-servidor de forma bidireccional en el mismo canal; esto nos permite un gran abanico de nuevas aplicaciones que quieran utilizar una comunicación más directa.

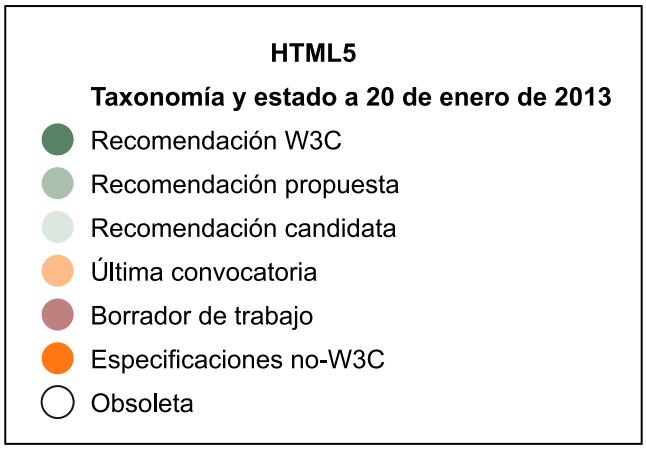
http://en.wikipedia.org/wiki/websockets

Enlace recomendado

Existen más librerías específicas. Para más información de cada una de ellas, podéis consultar: http://en.wikipedia.org/wiki/html5

Esquema con todas las librerías que incorpora HTML5





1.2. Javascript

El lenguaje Javascript es uno de los lenguajes más utilizados de programación, y se utiliza en muchos ámbitos diferentes; sobre todo es utilizado por el navegador para controlar cómo tienen que comportarse e interactuar los diferentes elementos de una página web o pantalla.

Es el lenguaje propio del navegador que podemos utilizar para acceder y manipular el DOM; el *DOM* o *document object model* es el árbol o lenguaje con el cual podemos acceder a los diferentes elementos de una página.

Debido a la fragmentación de navegadores que hemos comentado anteriormente, el DOM o el acceso a él puede ser diferente según el navegador, de modo que se han acabado utilizando librerías Javascript como jQuery para facilitar su acceso.

Hoy en día, hay una gran cantidad de librerías disponibles en lenguaje Javascript, algunas de ellas incluso de la parte de servidor, como NodeJS. Para desarrollo móvil propiamente también han aparecido muchas que nos pueden ser útiles, y seguirán apareciendo; como siempre, acabará dependiendo de nuestras preferencias y del tipo de proyecto por el que nos decantaremos para utilizar una u otra.

1.2.1. Librerías móviles

Las principales librerías de desarrollo móvil son las siguientes:

jQuery Mobile

Seguramente es la librería Javascript más utilizada en el mundo móvil, pues se basa en jQuery. Muy adecuada para proyectos donde las plataformas son iOS y Android. Ocupa unos 20Kb.

http://jquerymobile.com/

xui.js

Es una librería muy pequeña, con solo las funciones de acceso al DOM básicas. Ocupa solo 4Kb.

http://xuijs.com/

zepto.js

Es una librería pensada para navegadores webkit; viene a ser una versión ultra reducida de jQuery, solo con las funcionalidades más importantes.

http://zeptojs.com/

EmbedJS

Esta librería está pensada para cubrir muchos dispositivos diferentes como teléfonos, tabletas, televisiones, etc. Dispone de una versión diferente para cada tipo de dispositivo; de este modo consigue que el dispositivo descargue solo el código que necesita.

http://uxebu.github.io/embedjs/

JQTouch

Librería pensada sobre todo para iOS y Android; es una alternativa al uso de jQuery Touch.

http://jqtjs.com/

IUI

Es un framework orientado a dar a las aplicaciones un aspecto lo más parecido posible al sistema iOS.

http://www.iui-js.org/

Sencha Touch

Se trata de un framework parecido a jQuery Mobile, que funciona con componentes, más orientado al mundo corporativo, y que requiere de una licencia para aplicaciones comerciales.

http://www.sencha.com/products/touch/

1.2.2. Otras librerías a destacar

NodeJS

Se trata de una implementación del motor Javascript de Chrome; viene a ser como un servidor en Javascript.

http://nodejs.org/

AngularJS

Se trata de una librería creada por Google, que nos ayuda a hacer más dinámico el HTML, desde un punto de vista MVC.

http://angularjs.org/

Bootstrap

Es una librería creada por Twitter que permite crear webs o webapps reponsive, es decir, que se adaptan al dispositivo de forma automática según las características de formato de este.

http://twitter.github.io/bootstrap/

1.3. CSS3

Es el lenguaje encargado de indicar cómo se tienen que visualizar los diferentes elementos del HTML. El lenguaje CSS ha ido evolucionando y, a estas alturas, ya se utiliza la versión 3 de este lenguaje.

El CSS3 nos ha traído muchas novedades, como las media queries, las transformaciones 3D, web fonts, esquinas redondeadas, sombras y otros efectos visuales.

La novedad más destacada en el marco del desarrollo móvil son las media queries, ya que nos permite discriminar los estilos a aplicar según una serie de parámetros, como por ejemplo la orientación, el tamaño de pantalla, etc. Esto nos ayuda a dar diferentes versiones de nuestra app según las características del dispositivo con que se está visualizando.

2. Herramientas de desarrollo multiplataforma

El mundo de las aplicaciones móviles no ha parado de evolucionar y crecer estos últimos años, y lo mismo ha pasado con sus herramientas.

El amplio abanico de posibilidades que se nos abre a la hora de desarrollar aplicaciones móviles requiere analizar cuidadosamente las necesidades del proyecto para elegir las herramientas más convenientes en cada caso.

Hay que tener presente que, en un entorno que avanza tan rápidamente como es el de las aplicaciones móviles, hay que mantenerse informado y actualizar nuestro conocimiento con las nuevas herramientas que van apareciendo y que modifican sustancialmente las ya existentes.

Tal como hemos comentado anteriormente, ahora mismo podríamos dividir el desarrollo de aplicaciones multiplataforma en dos tipos: un primero orientado a ejecutarse dentro del navegador, y un segundo donde el código se interpreta para generar aplicaciones nativas.

2.1. Aplicaciones multiplataforma "nativas"

El deseo de cualquier desarrollador móvil es el de, con un único desarrollo, conseguir aplicaciones que funcionen de forma nativa en cualquier plataforma; con este concepto han aparecido diferentes soluciones que intentan conseguir este objetivo con diferentes soluciones.

Algunas de las más utilizadas para hacer aplicaciones son: Appcelerator Titanium, Adobe Air, Mono, Corona, etc.

Algunos de estos frameworks utilizan lenguajes diferentes, dan soporte solo a ciertas plataformas o están orientados a un tipo de aplicación en concreto, por lo que dependiendo del proyecto y de nuestros conocimientos tendremos que decidir el adecuado para nosotros.

Incluso hay frameworks más especializados solo orientados al desarrollo de juegos multiplataforma, como por ejemplo Unity 3D, Starling, Cocos, etc. No entraremos en analizarlos.

A continuación, introduciremos algunos de los frameworks para realizar aplicaciones multiplataforma de ejecución nativa que hemos comentado:

1) Adobe Air

Se llama Adobe Integrated Runtime o AIR; es la solución de Adobe para poder desarrollar aplicaciones multiplataforma, para que se ejecuten de forma nativa.

Permite, utilizando las tecnologías Adobe Flash, Flex, Javascript, HTML y AJAX, programar interfaces complejas para multitud de dispositivos, incluyendo la posibilidad de generar aplicaciones para el escritorio.

Internamente, el contenido se ejecuta utilizando el reproductor Flash Player y el lenguaje de programación ActionScript 3.0, por lo que no se ejecuta de forma nativa realmente, pero tampoco utiliza el navegador sino que utiliza su propio reproductor e intérprete. Podemos llamar a funciones nativas utilizando el framework llamado AIR framework, pero no es posible utilizar los elementos de la UI nativos de cada plataforma, por lo que generalmente se suele utilizar una única interfaz similar para todas ellas.

Soporta las plataformas móviles más conocidas como iOS, Android y BlackBerry, incluyendo además soporte para aplicaciones de escritorio para Mac OS, Windows.

Si el proyecto requiere hacer aplicaciones de escritorio, el lenguaje ActionScript es una de las mejores opciones para ello.

Uno de los problemas de Adobe Air es que se trata de una tecnología propietaria, de modo que siempre dependeremos de Adobe en última instancia.

Enlace recomendado

Podéis encontrar más información de Adobe Air en: http://www.adobe.com/es/products/air.html

2) Mono

Es un framework opensource creado por la empresa Xamarin, que permite desarrollar aplicaciones para iOS, Android y aplicaciones de escritorio para Mac, Windows e incluso consolas como Wii, PlayStation 3 o XBOX 360 utilizando el lenguaje de programación C# y la tecnología .NET creada por Microsoft.

Permite acceder a las librerías nativas desde lenguaje C#, por lo que el soporte es el mismo que si utilizáramos Objective-C. También nos permite añadir librerías Objective-C para iOS y Javaper Android junto con código C# en un mismo proyecto, así como utilizar muchas de las tecnologías relacionadas con el desarrollo en C#.

En el ámbito IDE se puede integrar dentro del Visual Studio o, si no, dispone de una herramienta propia llamada Xamarin Studio.

Las aplicaciones generadas utilizan los elementos nativos de cada plataforma, por lo que es una alternativa muy válida si estamos familiarizados en desarrollar en C# y con las tecnologías de Microsoft.

Darse de alta es gratuito, pero existe un límite de peso a la hora de publicar aplicaciones; si lo sobrepasamos tendremos que adquirir una licencia.

Enlace recomendado

Podéis encontrar más información en http://xamarin.com/

3) Appcelerator Titanium

Titanium es la solución de la empresa Appcelerator, para desarrollar aplicaciones nativas multiplataforma para las plataformas iOS, Android y BlackBerry; el lenguaje de programación utilizado es Javascript, a pesar de que tenemos que hacer llamadas a una API propia. Titanium ha avanzado mucho estos últimos años; con las primeras versiones utilizaba el navegador para presentar la aplicación del mismo modo que lo hace PhoneGap, por lo que las aplicaciones no eran nativas y el rendimiento se veía afectado, pero con la versión 1.0 este funcionamiento se modificó y se buscó una solución diferente para permitir generar aplicaciones nativas y mejorar el rendimiento.

Actualmente, utiliza un intérprete Javascript y una serie de librerías internas para traducir el Javascript a nativo. Cuando cargamos una app hecha con Titanium, al iniciar tiene que cargar estas librerías y la intérprete Javascript; a partir de entonces, interpreta nuestro código y hace las llamadas de forma nativa utilizando las librerías previamente cargadas. Estas librerías es lo que se denomina Titanium API, y es la guía que tenemos que utilizar para hacer las llamadas nativas.

El desarrollo se hace con la herramienta Titanium Studio, que es una adaptación del programa Aptana, que Appcelerator compró hace unos años.

Una de las grandes ventajas de Titanium es que podemos utilizar los elementos de UI nativos de cada plataforma.

Es un muy buen framework pero presenta algunos inconvenientes, por ejemplo, el hecho de que al iniciar la app se tengan que cargar las librerías, cosa que hace que al iniciar la aplicación esta pueda tardar algo más, a pesar de que después se ejecute de forma nativa.

También, debido a la velocidad en que aparecen nuevas versiones de los SDK nativos, es muy difícil mantener soportadas todas las funcionalidades, por lo que no está el SDK nativo soportado al 100%, y para algunas características concretas nos podemos encontrar con que no está soportado.

No es un framework propietario, sino que la mayor parte del código es open-source, así que es más fácil que se resuelvan bugs y evolucione el framework.

Es un framework muy utilizado en parte debido a que el lenguaje Javascript es uno de los más conocidos por los desarrolladores.

Enlace recomendado

Más información de Titanium en http://www.appcelerator.com/

Conclusiones

Este tipo de frameworks pueden ser adecuados para algunos proyectos donde necesitamos dar soporte a más de una plataforma y a los conocimientos de programación de que disponemos.

Uno de los problemas más grandes para conseguir un desarrollo multiplataforma nativo es que los elementos de las interfaces de las diferentes plataformas no son los mismos, por lo que los tendremos que desarrollar por separado para cada plataforma. Por el contrario, obtendremos un rendimiento bastante más grande.

Por lo tanto, los desarrollos más cercanos a la "multiplataforma" ideal son los que utilizan Adobe Air y PhoneGap, pues son los únicos que utilizan los mismos elementos para construir la interfaz. Con PhoneGap, por eso, sacrificamos rendimiento respecto de las otras opciones al funcionar en navegador.

Con Adobe Air tenemos el problema añadido de que se trata de una plataforma cerrada, por lo que dependemos de la tecnología de una empresa y, por lo tanto, es una apuesta arriesgada, ya que en cualquier momento, si se abandona el proyecto, podemos perder el soporte futuro.

Para Adobe Air y Titanium encontramos también el problema de que los intérpretes se tienen que actualizar constantemente para dar soporte a las nuevas versiones nativas que van apareciendo; en estas soluciones dependemos completamente de la capa que traduce nuestro código a código nativo, por lo que si hay alguna función no soportada o algún bug en esta implementación, es complicado de solucionar, ya que no disponemos de acceso a esta capa propietaria, por lo que tendremos que esperar a que se solucione el bug.

Por ahora, Appcelerator Titanium seguramente es la opción de las comentadas con un futuro más prometedor, sobre todo debido a la gran cantidad de desarrolladores con conocimientos de Javascript existentes.

2.2. Aplicaciones multiplataforma ejecutadas en el navegador

Antes de la aparición de las herramientas para hacer aplicaciones multiplataforma nativas, la única alternativa era utilizar el navegador para ejecutar la aplicación.

En este campo, la novedad más importante fue la aparición de PhoneGap, sin la cual no sería posible generar apps multiplataforma de manera sencilla y utilizando las funciones de diferentes plataformas como GPS, cámara, acelerómetro, brújula, etc., de una manera transparente. Además, utiliza lenguajes muy conocidos para llevar a cabo el desarrollo como el HTML5, Javascript y CSS.

Actualmente, existen otras alternativas en línea en este proceso; muchas de ellas optan por utilizar soluciones en la nube donde el desarrollo, la gestión y la publicación se gestionan desde su web, haciendo mucho más sencillo el proceso de desarrollo, y donde en muchos casos no son necesarios conocimientos de programación.

La principal ventaja de estas soluciones son su practicidad, pues permite crear aplicaciones de forma muy sencilla, poder controlar y customizar nuestras aplicaciones o añadir servicios avanzados como notificación PUSH, etc. de forma sencilla. Pueden ser soluciones adecuadas para algunos proyectos sencillos o corporativos donde el cliente pida tener acceso directo a los contenidos.

Estas herramientas van ampliando paulatinamente y mejorando los servicios que ofrecen.

Antes de decantarse por estos servicios en línea, hay que tener presentes sus desventajas, que suelen ser que, generalmente, es necesaria una suscripción y que, por otro lado, se pierde el control de la aplicación, ya que el desarrollador no dispone de acceso al código de la aplicación.

Muchas de estas soluciones seguramente utilizan PhoneGap de forma interna, o una estructura similar donde la aplicación se ejecuta en el navegador y permite hacer llamadas a funciones nativas utilizando una serie de librerías Javascript.

En este apartado, por eso, nos centraremos en PhoneGap, la cual es el framework que soporta más plataformas móviles; es opensource, gratuita, y utiliza los lenguajes de programación más habituales, como pueden ser HTML, Javascript y CSS.

Enlaces recomendados

Podéis encontrar información adicional en los siguientes enlaces:

trigger.io: https://trigger.io/

appmobi.com: http://www.appmobi.com/

stackmob: https://www.stackmob.com/

brightcove: http://www.brightcove.com/es/

viziapps: http://www.viziapps.com/

3. Instalación y preparación de herramientas de desarrollo multiplataforma (PhoneGap)

PhoneGap es una librería creada por la empresa Nitobi que nos permite tratar con diferentes plataformas móviles como si fuera solo una. Una vez hayamos desarrollado nuestra aplicación en un lenguaje de programación común, PhoneGap se encargará de encapsularlo después en la plataforma que deseemos.

Para hacerlo, PhoneGap utiliza el navegador del dispositivo para visualizar una página web que actúa como si se tratara de una aplicación; de este modo, podemos utilizar PhoneGap en cualquier dispositivo que disponga de navegador.

Esta librería se encarga de comunicar las acciones que recibe del usuario a las herramientas nativas del dispositivo.

Nos permite acceder a las funciones particulares de cada dispositivo, como el acelerómetro, la cámara, la geolocalización, etc. mediante plugins.

Estos plugins están hechos con lenguaje nativo, es decir, disponen de acceso total al dispositivo y se comunican con PhoneGap a través de Javascript.

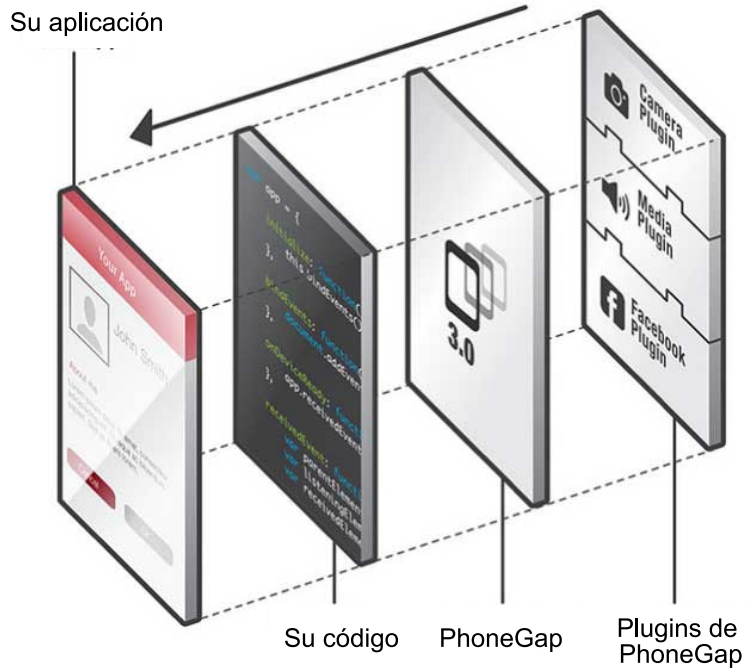
Es importante tener en cuenta, en la estructura interna de un proyecto PhoneGap, que estos suelen tener varios directorios. Generalmente solo nos interesará el directorio llamado www, en el cual introduciremos nuestro código (como mínimo una página llamada index.html).

El otro archivo importante es el archivo config.xml, que tiene que estar ubicado en la raíz de nuestro proyecto. En este archivo indicamos el nombre, la descripción, los iconos, las pantallas de inicio, los plugins que utilizamos, etc. Podemos crearlo nosotros mismos o utilizar la herramienta del PhoneGap Build Service que se nos cree de forma automática.

Una vez terminado el desarrollo de la app, la librería también nos permite generar la aplicación en el formato final para cada una de las plataformas.

A continuación, podéis ver un pequeño esquema de la estructura de una aplicación PhoneGap:

Estructura interna de una aplicación PhoneGap



Fuente: http://phonegap.com/blog/2013/06/20/coming-soon-phonegap30/

PhoneGap empezó siendo un proyecto independiente, pero la compra del framework por parte de Adobe hace unos años supuso algunos cambios.

Para empezar, se creó el servicio PhoneGap Build Service, una web que permite subir el código de nuestra aplicación y compilar las plataformas que queramos de forma gratuita.

Para generar las aplicaciones en nuestro ordenador en diferentes plataformas, es necesario instalar las herramientas de desarrollo de cada una de ellas. Hasta ahora, para compilar una app para iOS era necesario disponer de un Mac, y para compilar una app para Windows Mobile también era necesario utilizar un ordenador con Windows. Con este nuevo servicio de Adobe en la nube, ya podemos generar aplicaciones en cualquier plataforma sin tenernos que preocupar de instalar nada.

La compra de PhoneGap por parte de Adobe también ha significado un cambio de nomenclatura importante; ahora la versión opensource de PhoneGap se llama Cordova y PhoneGap ha pasado a pertenecer a Adobe.

Ahora PhoneGap es una distribución que saca Adobe de la versión opensource Cordova. Prácticamente no hay diferencias, la única importante es que la distribución PhoneGap, además de las funciones de Cordova, añade servicios de Adobe, como acceso al PhoneGap Build Service, etc.

3.1. El entorno de trabajo con PhoneGap

Para desarrollar aplicaciones multiplataforma dentro del navegador, lo más habitual es, antes o después, acabar utilizando PhoneGap para generar los ejecutables finales ya que es la herramienta más utilizada para este propósito y la que dispone de más recursos y documentación.

A continuación, desglosamos tres maneras de trabajar diferenciadas; las tres utilizan PhoneGap ya sea de manera local o remota.

3.1.1. Entorno remoto PhoneGap

Este es el entorno de trabajo de desarrollo web tradicional. Es adecuado también para desarrollar webs para dispositivos móviles, webapps o incluso apps multiplataforma sencillas.

Recordemos que una app multiplataforma ejecutada en el navegador no deja de ser HTML, de forma que podemos utilizar las herramientas de desarrollo HTML habituales.

Es una solución muy adecuada cuando nuestra aplicación no utilice muchas funciones avanzadas del dispositivo, como por ejemplo GPS, cámara, etc. También es el lenguaje más sencillo si tenemos conocimientos previos de desarrollo web convencional.

Una vez desarrollada nuestra aplicación con lenguaje HTML, utilizaremos el servicio en la nube de Adobe PhoneGap Build Service para generar las diferentes aplicaciones multiplataforma.

Hay que tener en cuenta que necesitamos tres elementos para desarrollar utilizando un entorno de programación web tradicional:

1) Editor de código o IDE

El primer elemento necesario es un programa donde podamos escribir todo nuestro código HTML, Javascript y CSS.

Existen tantas opciones como gustos, y habrá que tener en cuenta también si trabajamos en un entorno Windows o Mac.

Los programas más habituales son:

- **Multiplataforma**
  - Aptana: http://www.aptana.com/
  - Dreamweaver: http://www.adobe.com/es/products/dreamweaver.html

  - Eclipse: http://www.eclipse.org/

- **Windows**
  - UltraEdit: http://www.ultraedit.com/

- **Mac**
  - Coda: http://panic.com/coda/
  - SublimeText: http://www.sublimetext.com/
  - TextMate: http://macromates.com/

2) Navegador web

En segundo lugar, necesitaremos también disponer de un navegador donde poder ir probando nuestra webapp. Este habrá de tener el mismo motor de renderizado que el dispositivo al cual esté destinada la webapp, normalmente Webkit. Por lo tanto, se suele trabajar con Chrome o Safari.

En el caso del navegador Chrome, además, contamos con una extensión muy potente desarrollada por BlackBerry para poder testear una gran cantidad de dispositivos diferentes desde nuestro navegador.

Esta herramienta se llama Ripple, y permite emular a Android, BlackBerry, iPhone, etc.

Podemos utilizarla de forma local o utilizando el servicio en la red en http://emulate.phonegap.com/.

Se trata de una herramienta muy completa que nos permite simular elementos como el GPS, movimientos del dispositivo, el acelerómetro, enviar eventos compartidos, etc.

3) Entorno de depuración local y remoto

El tercer elemento necesario es un entorno donde poder depurar de forma local y también remota nuestro dispositivo, y poder resolver así los errores más complicados.

En cuanto a la solución local, podemos utilizar la Web Inspector que viene instalada en Chrome, puesto que algunas soluciones remotas están basadas en este mismo componente.

Otras soluciones donde podemos trabajar son:

- **JSConsole**: http://jsconsole.com/
- **Weinre**: http://people.apache.org/~pmuellr/weinre/docs/latest/Hombre.html
- **Socketbug**: http://socketbug.com/

- **iWebInspector**: http://www.iwebinspector.com/
- **Mobile Safari Remote Debugger**

3.1.2. Entorno local PhoneGap

En el caso de trabajar en el entorno PhoneGap de forma local, la opción recomendada es seguir el proceso de instalación recomendado por la misma librería; en este momento se trata de utilizar la línea de pedidos para generar nuestro proyecto PhoneGap, y después utilizar la herramienta de desarrollo nativo que deseamos para desarrollar nuestra app multiplataforma. En las situaciones más habituales, se trata del Xcode para iOS y del Eclipse para Android.

Tendremos que utilizar el programa de desarrollo nativo para programar nuestras apps en HTML5, pero esto nos aporta una serie de ventajas.

- Probar de manera más sencilla nuestro proyecto utilizando el simulador de la plataforma.

- Podremos instalar nuestra app en el dispositivo más rápido y sin depender del servicio de Adobe.

- Aprenderemos a utilizar la herramienta nativa, conocimiento que nos puede ser útil por si tenemos que hacer algún plugin.

Esta dinámica de trabajo nos ayudará a desarrollar y testear más rápidamente nuestras aplicaciones, sobre todo si estas utilizan funcionalidades avanzadas del dispositivo. Siempre podemos recurrir a un editor externo que nos sea más cómodo para trabajar, y utilizar el IDE nativo solo para probar la aplicación con las funcionalidades del dispositivo.

Sin embargo, con este entorno seguiremos necesitando de otro entorno para depurar nuestro código HTML5.

Si preferimos no tener que utilizar las herramientas nativas, también podemos utilizar la línea de pedidos de PhoneGap (CLI) para compilar, abrir el simulador, etc.

Más adelante analizaremos exactamente el proceso a seguir.

3.2. Instalación de PhoneGap

Actualmente podemos trabajar con la herramienta de PhoneGap de varias maneras, desde la más sencilla, que pasa por utilizar el servicio de Adobe Phone Gap Build Service en la nube, a las más complejas, con las cuales podemos generar en nuestro propio ordenador las compilaciones para las diferentes plataformas.

A continuación, analizaremos las diferentes opciones de trabajo con PhoneGap:

3.2.1. Opción A - PhoneGap en la nube

Esta es la opción más sencilla para desarrollar aplicaciones multiplataforma.

Recordemos que la posibilidad de trabajar con PhoneGap de manera online existe desde que Adobe compró la herramienta, y crearon a continuación el servicio PhoneGap Build Service en la nube, que nos permite compilar en todas las plataformas sin tener que instalar el programa.

De este modo, utilizando herramientas de desarrollo web para crear nuestra aplicación, y usando el servicio PhoneGap Build Service de Adobe, no sería necesario ningún tipo de instalación de PhoneGap en nuestro ordenador.

Para trabajar en este entorno en la nube, necesitaremos los siguientes elementos:

- **Servidor web Apache: Windows**: http://www.wampserver.com/
- **Mac**: http://www.mamp.info/
- **Aptana studio**: http://www.aptana.com
- **Navegador Chrome**: https://www.google.com/intl/es/chrome/browser/?hl=es
- **Ripple Emulator**: https://chrome.google.com/webstore/detail/ripple-emulator-beta/geelfhphabnejjhdalkjhgipohgpdnoc

Es recomendable tener un servidor web instalado en nuestra máquina para poder probar las aplicaciones HTML5 que vamos desarrollando, a pesar de que podemos utilizar otras soluciones, como subir los cambios a un FTP y probarlos a través de internet, aunque teniéndolo en local siempre da un acceso más rápido. También es recomendable que sea un servidor web Apache puesto que es más compatible con el Aptana. Lo más sencillo es descargarse un paquete donde venga integrado, por ejemplo, el WAMP para Windows o el MAMP para Mac u otras distribuciones como XAMPP, ya que estos paquetes integran varios servicios como un servidor web, un servidor de base de datos MySQL y la posibilidad de ejecutar PHP, XAMPP, además de incluir PERL.

Hay muchas alternativas diferentes, pero estas son seguramente las más fáciles a la hora de instalar todo el paquete necesario para trabajar.

Una vez tengamos instalado y funcionando el servidor web, continuaremos descargándonos el Aptana y el navegador Chrome e instalándolos en nuestro ordenador.

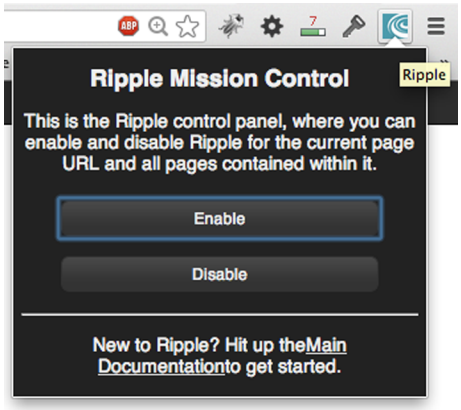
A continuación, abriremos el Aptana y añadiremos el navegador Chrome como un acceso directo cuando apretemos la tecla de Build; lo podemos hacer con diferentes navegadores si lo deseamos. En nuestro caso, solo necesitamos el Chrome para poder utilizar el Ripple Emulator.

Para hacerlo, clicaremos la flecha junto al icono de Build y seleccionaremos la opción Run configurations; aquí añadiremos una nueva entrada en Web Browser seleccionando el ejecutable de Chrome. Apuntemos que aquí también podemos definir si el servidor web que utilizamos es remoto.

Después abriremos el navegador Chrome e instalaremos la extensión Ripple Emulator mediante la URL indicada anteriormente. Reiniciaremos el navegador y tendremos que comprobar que vemos un nuevo icono en la barra superior del navegador.

Desde cualquier página donde estemos, clicando encima de este icono accederemos al emulador, donde podremos testear la página como si estuviéramos en un dispositivo móvil.

Captura donde vemos la ejecución del plugin Ripple



Podemos hacer pruebas abriendo la URL http://view.jquerymobile.com/1.3.1/dist/demos/ y ejecutando el plugin Ripple para hacernos una idea de las posibilidades de las apps en HTML5.

Ripple tiene una limitación que hay que tener presente, y es que no trabaja con la última versión de PhoneGap, hecho que representa que no nos podemos fiar al 100% del resultado hasta que no lo probemos en un dispositivo. Además, en la última versión de Ripple, no se inicializa PhoneGap cuando se tiene seleccionada la versión 2.0, mientras que con la 1.0 funciona correctamente.

Por ahora, utilizaremos la plataforma 1.0, y más adelante veremos cómo solucionar este problema.

También aprovecharemos para darnos de alta en el servicio de PhoneGap Build, en la dirección https://build.phonegap.com/. Allí nos registraremos creando un Adobe ID o con una cuenta de github. Solo podremos crear una única aplicación de forma gratuita.

Siguiendo este proceso, ya tendremos las herramientas listas para empezar el desarrollo.

3.2.2. Opción B - Instalación local

Esta es la opción de instalación más complicada, pero a la vez nos permitirá trabajar de manera independiente del servicio de Adobe.

Para hacerlo, nos descargaremos el archivo de PhoneGap desde la web http://phonegap.com.

En este paquete encontraremos las instrucciones para instalar PhoneGap en la plataforma que escojamos, iOS, Android, etc., así como ejemplos para cada plataforma.

Tendremos que instalar el IDE nativo de cada plataforma donde queramos probar de forma directa nuestra aplicación.

Como ejemplo, veremos el proceso de instalación de las herramientas de Android. Para iOS es necesario disponer de un Mac, y también estar dado de alta como desarrollador para poder instalar aplicaciones en el dispositivo.

Para esta opción de instalación en Android necesitaremos los siguientes elementos.

- **Última versión de PhoneGap**: http://phonegap.com
- **Programa Eclipse**: http://www.eclipse.org/downloads/
- **Aptana Plugin**: http://download.aptana.com/studio3/plugin/install
- **ADT Plugin**: https://dl-ssl.google.com/android/eclipse/
- **Android SDK**: http://developer.android.com/sdk/index.html

Para desarrollar aplicaciones nativas para Android, podríamos descargar el paquete completo, es decir, Eclipse con el plugin ADT incluido, pero como nosotros lo que queremos es trabajar con aplicaciones HTML5, y el paquete integrado de Android Eclipse+ADT da algunos problemas al instalar el plugin del Aptana posteriormente, es preferible instalarlo todo por separado. Es decir, primero Eclipse, después Aptana y, por último, el plugin ADT.

Empezaremos descargando la aplicación Eclipse desde su web; una vez la hayamos instalado, la abriremos.

A continuación, el Eclipse nos pedirá la ruta de nuestro espacio de trabajo (workspace); hará falta, pues, indicarle la ruta donde queramos dejar nuestros proyectos PhoneGap a partir de ahora.

Seguidamente, instalaremos el plugin Aptana, que nos ayudará a la hora de editar los archivos HTML, Javascript y CSS desde el Eclipse. Recordemos que Aptana está enfocado en el desarrollo web.

Para ayudarnos a editar los archivos html, css, etc., es recomendable instalar el plugin Aptana en el Eclipse.

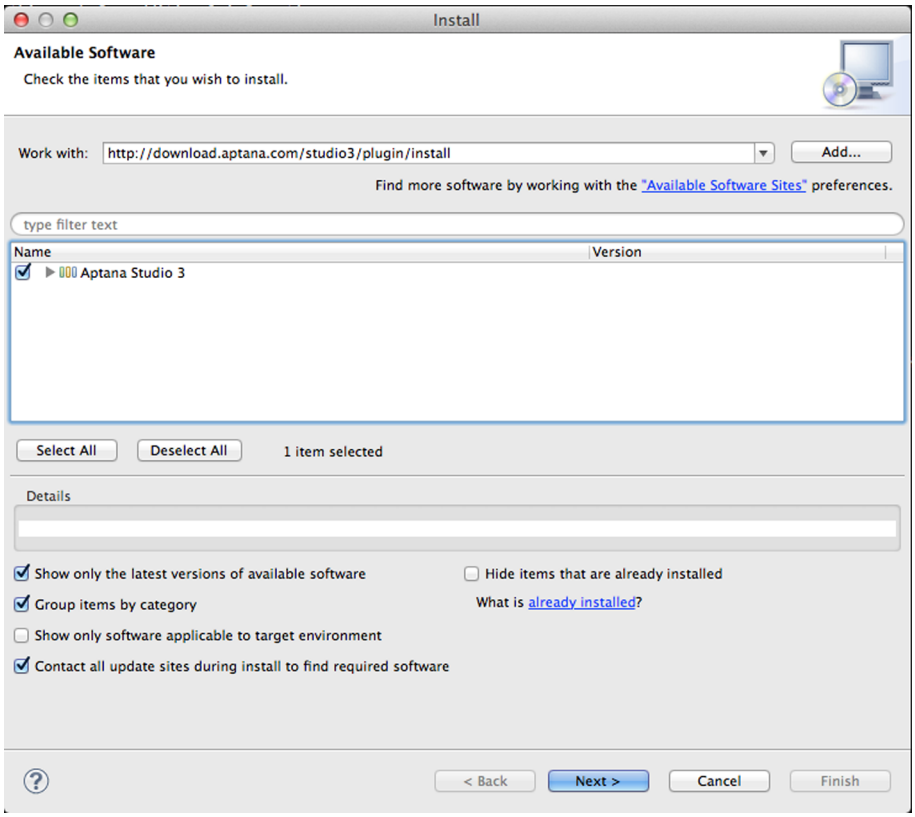
Para hacerlo, tenemos que ir al menú Help>Install New Software.

En el campo Work with añadiremos la URL y haremos clic en Add.

```
http://download.aptana.com/studio3/plugin/install
```

Le ponemos el nombre de Aptana y esperamos a que aparezca el plugin Aptana Studio 3, y entonces pulsamos el botón de Next para proceder a la instalación.

Diálogo para la instalación del plugin Aptana



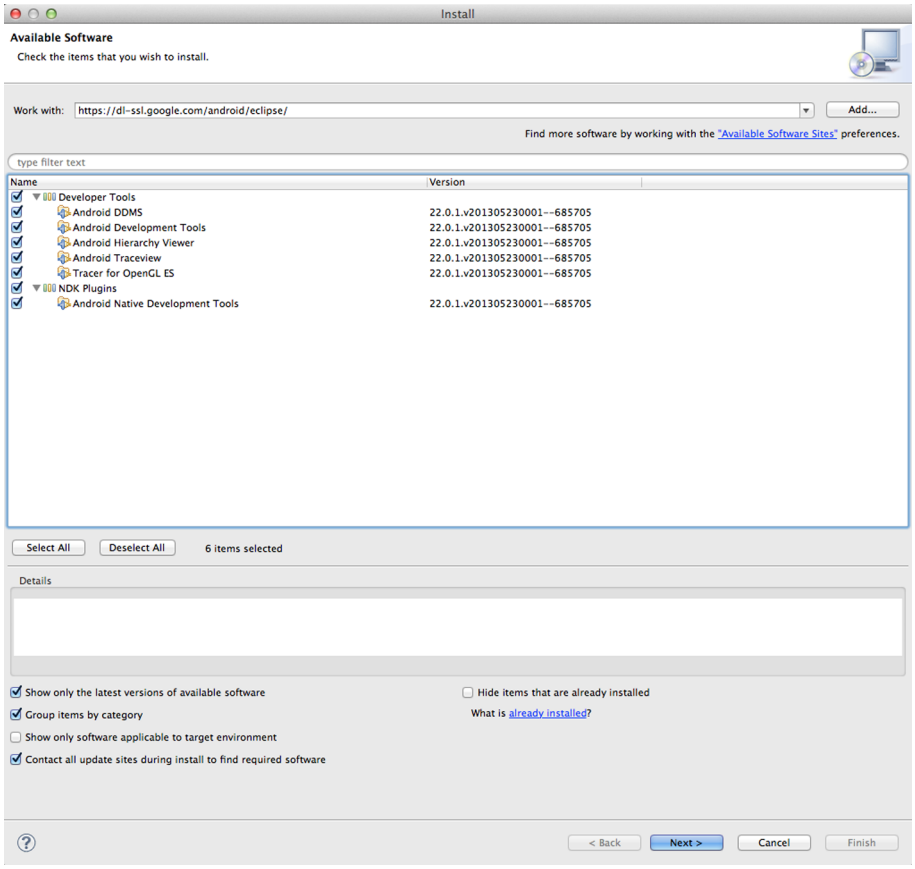
Una vez esté instalado el plugin de Aptana, nos aparecerá el mensaje de reiniciar el Eclipse; le confirmamos la orden de reinicio y, a continuación, procederemos a instalar la ADT plugin para desarrollar apps para Android.

Para hacerlo, seguiremos el mismo proceso que el indicado anteriormente, es decir, iremos a Help>Install New Software. Y añadiremos la siguiente URL:

```
https://dl-ssl.google.com/android/eclipse/
```

Seleccionaremos la opción de instalar todas las opciones y pulsaremos el botón de Next.

Ventana de diálogo de instalación de las herramientas de desarrollo de Android (ADT)



Después de este proceso, ya tendremos el Eclipse y los plugins necesarios instalados y configurados.

A continuación, necesitaremos descargarnos e instalar como mínimo una versión del SDK de Android. Para hacerlo, iremos a la web http://developer.android.com/sdk/index.html y escogeremos la opción "Use an existing IDE" para descargarnos solo el SDK y no el programa APT, similar al Eclipse.

Una vez lo hemos descargado para Windows, ejecutaremos el instalador para que guarde el SDK en la ubicación por defecto, mientras que para Mac crearemos la carpeta sdk dentro de la carpeta Eclipse y arrastraremos todos los archivos.

El siguiente paso será abrir el Eclipse y seleccionar la opción Window> Android SDK Manager; aquí comprobaremos que tenemos como mínimo una versión instalada, en caso contrario habrá que instalarla.

Para trabajar con proyectos PhoneGap desde Eclipse, tendremos que conocer el proceso para crear uno nuevo e importarlo a Eclipse.

Por lo tanto, habrá que volver a la carpeta de PhoneGap que nos hemos descargado anteriormente, donde encontraremos dos carpetas, una primera llamada doc donde hay una copia de toda la documentación, y una segunda llamada lib donde aparece el código de PhoneGap.

Dentro de lib veremos una serie de carpetas, una para cada plataforma, y tendremos que abrir la de Android.

Seguidamente, hará falta que accedamos a una ventana de línea de pedidos si utilizamos Windows o una ventana de Terminal si utilizamos Mac o Linux.

Y a continuación accederemos a la carpeta Android/bin.

Una manera de acceder a ella rápidamente es escribir cd y arrastrar la carpeta dentro de la ventana de la línea de pedidos.

Una vez hemos accedido, escribiremos el pedido:

```
.create <path> <package> <activity>
```

Donde

<path> es la ruta donde queremos guardar nuestro proyecto

<package> es el identificador interno de la app, suele ser algún pedido similar a com.example.nomapp, es decir, como una dirección web pero al revés.

<activity> es el nombre de la ventana inicial de nuestra app

Por ejemplo:

Para Windows:

```
.create %USERPROFILE%\Escritorio\UOC edu.uoc.PhoneGap PhoneGap
```

Para Mac:

```
.create ~/Desktop/UOC edu.uoc.PhoneGap PhoneGap
```

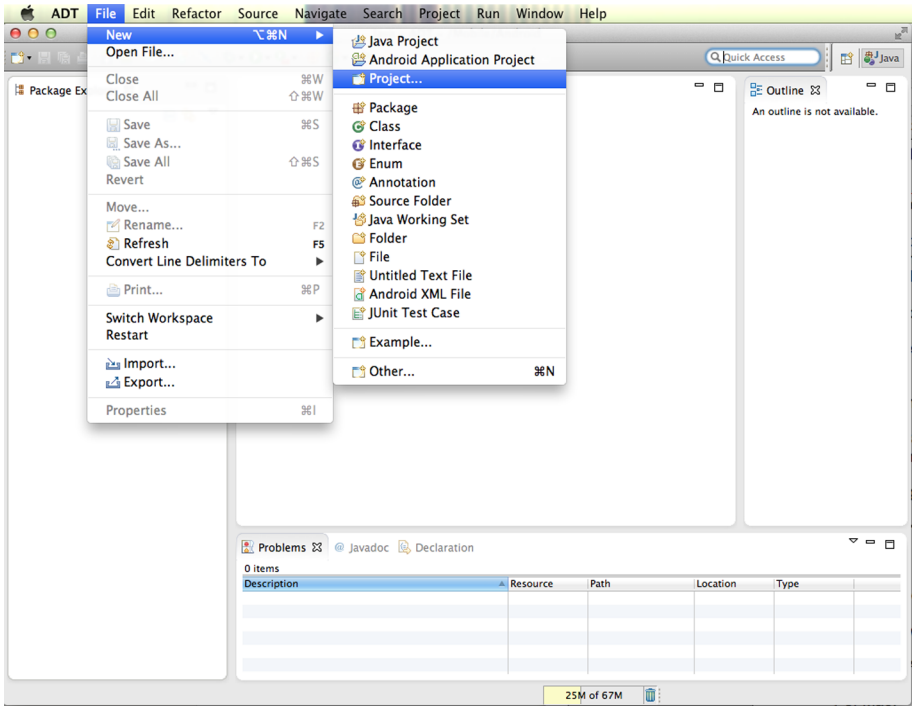
Nota

Cómo estamos generando el proyecto sin utilizar el Eclipse, es preferible trabajar fuera del workspace, puesto que al Eclipse le gusta crear los directorios de los proyectos él mismo.

Una vez generado el proyecto, lo abriremos desde el Eclipse y seleccionaremos la opción:

File > New Project

Menú de Eclipse para la creación de un nuevo proyecto

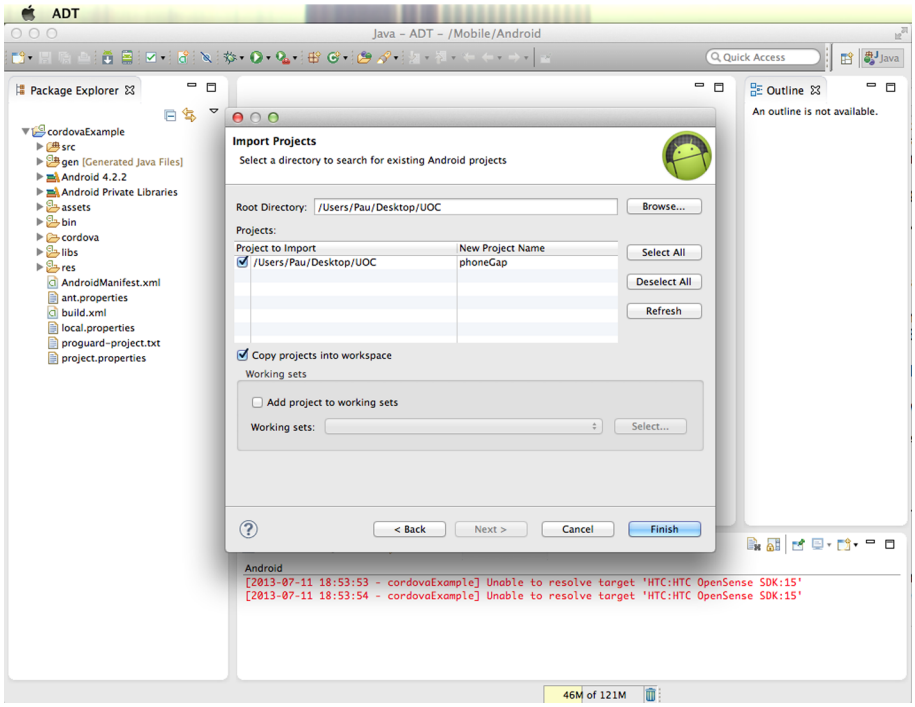


De entre las opciones elegiremos:

Android Project from Existing Code, y seleccionamos Next.

Seguidamente, en Root directory señalamos la ruta donde tenemos la aplicación que acabamos de generar, y seleccionaremos la opción de Copy Projects into Workspace.

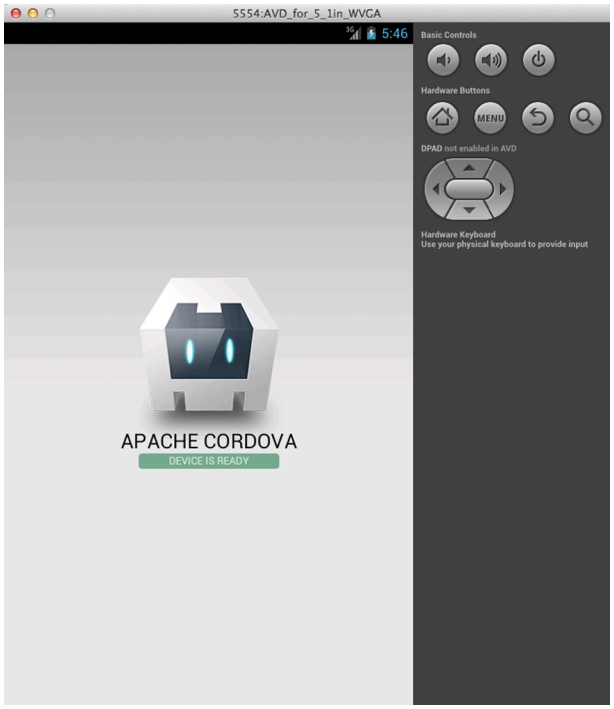
Ventana de diálogo para la importación de un proyecto ya existente



Con esto ya habremos creado nuestro proyecto en el wokspace del Eclipse, y ya podremos eliminar el proyecto que hemos generado anteriormente, comprobar el buen funcionamiento pulsando el botón de Run de color verde o la opción de Run; se tendría que ejecutar el emulador y tendríamos que visualizar la pantalla de nuestro proyecto PhoneGap.

En definitiva, siguiendo estas pasos, tendremos ya instaladas las herramientas para desarrollar para Android utilizando el Eclipse con el plugin Aptana, con lo que podremos empezar nuestro desarrollo.

Pantalla del emulador Android con la aplicación de prueba funcionando



Instalación avanzada CLI (Client Line Interface)

Desde el inicio PhoneGap ha dispuesto de una herramienta CLI o acceso con línea de pedidos, con la cual podemos obtener un control extra. Con las últimas versiones, este soporte ha mejorado y nos permite generar proyectos para varias plataformas de una manera sencilla, ejecutar y compilar nuestras apps en los diferentes emuladores de las plataformas, etc.

En la versión 3.0, además del Cordova CLI, también tendremos el PhoneGap CLI, que comparte algunas de las funcionalidades de Cordova CLI sumando la posibilidad de interactuar de forma remota con el PhoneGap Build Service de Adobe.

La generación de apps utilizando CLI sigue necesitando instalar las diferentes plataformas, pero nos ahorra el hecho de tener que abrir una a una y averiguar cómo funciona cada una de ellas para poder generar nuestra aplicación PhoneGap.

Esta instalación avanzada es la mejor solución en los casos que queramos compilar para diferentes plataformas desde nuestro ordenador, sin tener que depender del servicio en la nube de PhoneGap.

Se trata de una parte opcional en la instalación local, que además nos instala una serie de herramientas en la línea de pedidos, pero de una gran utilidad.

Son una serie de ejecutables que nos permiten compilar, abrir el emulador, abrir el proyecto en un navegador, etc. desde la misma línea de pedidos.

En este sentido, supone una gran ventaja ya que no tenemos que utilizar ninguna herramienta nativa, y podemos generar aplicaciones compatibles con muchas plataformas de una manera muy sencilla.

Los pasos de instalación son más complicados, pero a cambio nos aporta un control total a la hora de gestionar nuestras aplicaciones. Nos permite crear proyectos multiplataforma de una manera muy sencilla, así como añadir plugins al proyecto, etc. Todo desde la propia línea de pedidos.

Para proceder a la instalación, empezaremos por descargar e instalar el paquete NodeJS.

Este nos instalará en la línea de pedidos los archivos binarios node y npm. De estos, el que nos interesa ahora es el npm, que es un gestor de paquetes con el que efectuaremos la instalación de PhoneGap.

Una vez instalado, probaremos introduciendo npm en la línea de pedidos, y a continuación tendríamos que ver un listado con las opciones que permite este pedido.

A continuación, ejecutaremos la orden:

```
sudo npm install -g cordova
```

Tendremos que escribir la contraseña de Administrador, pues es necesaria para instalar nuevos binarios en /usr/local/bin

Ahora ya podremos utilizar el pedido Cordova para crear nuevos proyectos, instalar diferentes plataformas, compilar y ejecutar apps en diferentes plataformas, todo de forma local.

Si queremos, además, la posibilidad de subir apps al PhoneGap Build Service, ejecutaremos también la instrucción:

```
sudo npm install -g phonegap
```

Con esto ya podremos utilizar las ventajas de PhoneGap desde la línea de pedidos.

Algunos de los pedidos que podremos utilizar con Cordova CLI son:

```
cordova create <path>
```

Nos permite crear un nuevo proyecto phonegap desde la línea de pedidos.

```
cordova platform(s) [{add|remove|rm} <PLATFORM>
```

Podemos añadir o eliminar plataformas a este proyecto; esto nos permite crear proyectos phonegap que soporten diferentes plataformas a la vez de forma fácil.

```
cordova build <PLATFORM>
```

Para preparar los archivos necesarios y compilar el proyecto en la plataforma que deseamos.

```
emulate <PLATFORM>
```

Para ejecutar el emulador de la plataforma que hayamos escogido.

```
serve <PLATFORM> [PORT]
```

Esta opción nos ejecuta la aplicación directamente en un servidor web local, para poderla visualizar desde un navegador.

Como vemos, podemos acceder perfectamente a todas las funciones básicas solo utilizando la línea de pedidos, sin tener que utilizar las herramientas de desarrollo nativas.

4. Publicación en stores

Existen muchas tiendas de aplicaciones diferentes, y su diferencia principal radica en la plataforma móvil que utilizan. Hay que tener en cuenta que actualmente solo la plataforma Android dispone de la posibilidad de utilizar otras tiendas de aplicaciones diferentes de la oficial.

Hoy en día, las dos tiendas de aplicaciones a destacar son la App Store de Apple para dispositivos iOS, y la tienda de aplicaciones de Android. Por ahora, el resto de tiendas todavía se encuentran un paso atrás en cuanto a usuarios y número de aplicaciones.

Android dispone de casi el 75% del mercado de smartphones, mientras que iOS, alrededor del 18%. No obstante, la tienda de Apple genera cerca de cinco veces más beneficios que la de Android.

Estos datos nos indican que, a pesar de que Android dispone de muchos más usuarios activos en su tienda de aplicaciones, Apple tiene más dispuestos a gastar dinero.

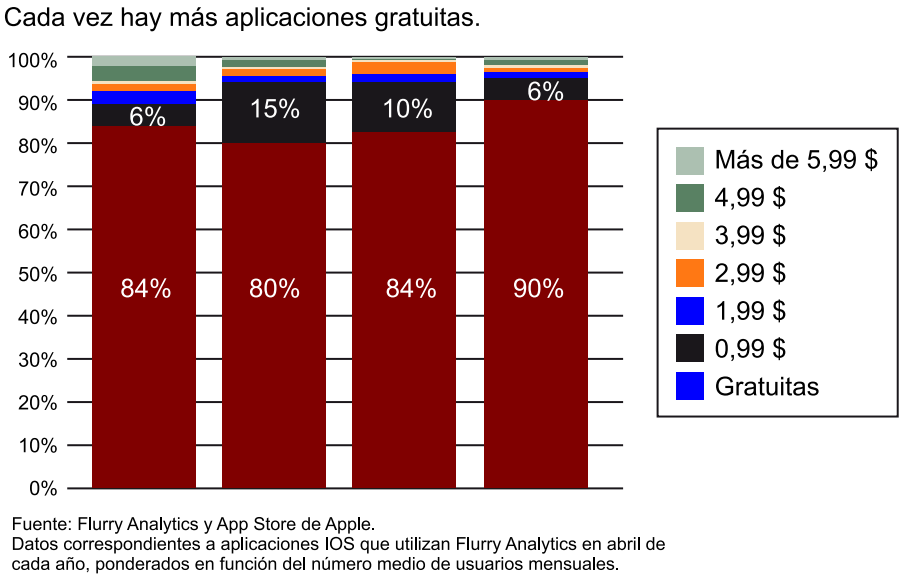
Respecto a las diferentes categorías de las aplicaciones, la que tiene más éxito es sin duda la de juegos, que en iOS supone el 25% de todas las aplicaciones, y en Android el 14%. También es la categoría que da más beneficios.

Respecto a los diferentes dispositivos, el que más beneficios genera es el iPad, en parte porque el precio de sus aplicaciones acostumbra a ser ligeramente superior a las aplicaciones para móvil. También porque es un dispositivo que suelen utilizar diferentes miembros de una misma unidad familiar o de un entorno laboral compartido, y por lo tanto, suelen descargarse en un mismo dispositivo un número más elevado de aplicaciones, las cuales también suelen ser para un público más heterogéneo.

Otro elemento importante que ha tener en cuenta el desarrollador es la tendencia de los precios: antes era bastante común ofrecer aplicaciones de pago, pero ahora, con tanta competencia, la tendencia ha cambiado y en la tienda se suelen ofrecer las aplicaciones de forma gratuita, para sacar beneficio económico mediante compras dentro de la aplicación o a través de la publicidad.

Actualmente, las aplicaciones de pago suponen un porcentaje muy pequeño de las tiendas de aplicaciones. En iOS, por ejemplo, el 90% de las aplicaciones son ya gratuitas.

Gráfico con los diferentes segmentos de precios de las aplicaciones

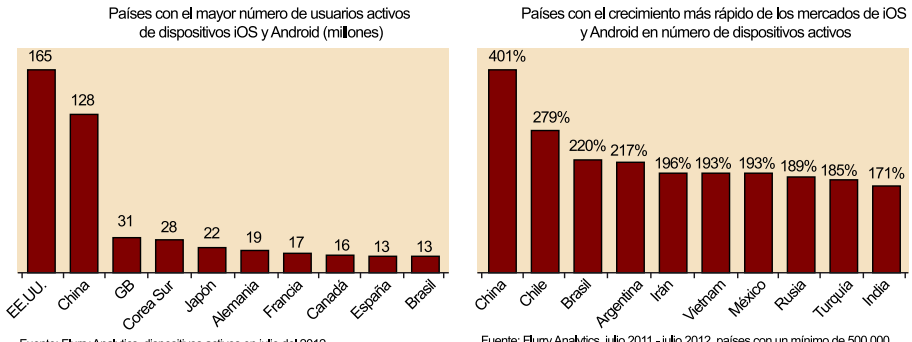


Fuente: Flurry Analytics y App Store de Apple.
Datos correspondientes a aplicaciones IOS que utilizan Flurry Analytics en abril de cada año, ponderados en función del número medio de usuarios mensuales.

En este gráfico podemos ver el incremento de aplicaciones gratuitas respecto de las de pago.

Desde el punto de vista geográfico, las tiendas de aplicaciones han ido creciendo y ampliándose en diferentes países (actualmente todavía no están cubiertos todos los países), y cada una dispone de su top ten particular, recomendaciones, etc.

El país con más descargas sigue siendo Estados Unidos; sin embargo, los países emergentes están registrando un rápido crecimiento en las descargas, encabezados por China, Chile o Brasil.



Fuente: Flurry, TechCrunch, Distimo. **A)** Gráfico donde podemos ver la cantidad de smartphones iOS y Android por países. **B)** Gráfico donde podemos ver el porcentaje de smartphones nuevos por países en un año.

A continuación, analizaremos las principales características específicas de cada tienda de aplicaciones:

iOS AppStore Apple

http://en.wikipedia.org/wiki/app_store_(iOS)

Fue la primera tienda de aplicaciones; existe desde el año 2008.

Actualmente, hay casi un millón de aplicaciones disponibles y los usuarios ya se han descargado alrededor de 50 millones de apps.

La plataforma iOS solo permite la instalación de aplicaciones desde la AppStore oficial de Apple.

Las aplicaciones que se publican están controladas por Apple y tienen que pasar una validación por parte de la compañía, que comprueba que se ajustan a su normativa.

Enlace recomendado

Apple ha ido actualizado su normativa a lo largo del tiempo en cuanto al desarrollo de apps; se recomienda leer la última actualización en:

https://developer.apple.com/appstore/guidelines.html

Hay que tener presente que, para publicar en la appStore, es necesario darse de alta como desarrollador de Apple; el precio de la inscripción es de 99 dólares anuales.

Además, las aplicaciones de la appStore van firmadas y encriptadas; es por este motivo por lo que para poder publicar o instalar en un dispositivo es necesario firmarlas con un certificado de Apple. De este modo, sin ser desarrollador se puede descargar y utilizar el xCode y el simulador, pero para instalar una app en un dispositivo se requiere estar dado de alta y haber pagado la cuota.

Por lo tanto, para poder instalar una aplicación desarrollada por nosotros en un dispositivo iOS, será indispensable darnos de alta como desarrolladores y pasar por caja.

Preparar el entorno de firmar aplicaciones es un proceso complejo, que incluye certificados, claves privadas, números de serie de dispositivos y ficheros. Por ahora, no entraremos en esta cuestión en profundidad.

En cuanto al beneficio de la venta de aplicaciones, este se reparte entre Apple y el desarrollador: un 70% es para el desarrollador y un 30% para Apple.

Con tantas aplicaciones disponibles, se hace cada vez más necesario disponer de un buen marketing, conocimientos de SEO y herramientas para aumentar la visibilidad de la aplicación, en paralelo a la publicación de una app.

Es la principal tienda a tener en cuenta, pues la AppStore, recordemos, sigue siendo la tienda con el porcentaje de aplicaciones de pago más elevado, y la que sigue produciendo más dinero en ventas.

Google Play - Android Market

http://en.wikipedia.org/wiki/google_play

Es la tienda de aplicaciones oficial de Android, fue creada en el año 2008, dispone de casi un millón de aplicaciones disponibles y 50 millones de descargas.

Como desarrolladores hay que saber que existen diferentes tiendas de aplicaciones que podemos utilizar con Android.

Por temas de seguridad es recomendable utilizar stores fiables, puesto que disponen de sistemas para evitar aplicaciones maliciosas.

Las condiciones de aprobación de Android son bastante más flexibles que las de Apple, y prácticamente no hay validación, a pesar de que sí hay un testeo para comprobar que sea una aplicación segura.

El proceso de publicación también es bastante más sencillo, pero continúa siendo necesario darse de alta como desarrollador y pagar 25 dólares.

Un elemento importante es que, con las últimas versiones de Android, es posible encriptar las apps que desarrollamos para reducir la piratería de nuestras aplicaciones.

Esta es la tienda de aplicaciones con más dispositivos, debido a la gran cantidad de terminales que disponen del sistema Android.

BlackBerry World

http://en.wikipedia.org/wiki/blackberry_world

Es la tienda oficial de la plataforma BlackBerry; apareció en el 2009.

Dispone de más de 100.000 aplicaciones y lleva registradas más de 3 millones de descargas.

Con la versión Blackberry 10, se abre la puerta a la compatibilidad de aplicaciones Android (solo para versiones antiguas por ahora), hecho que facilita mucho el traslado de una aplicación Android a BlackBerry.

Es gratuito tanto darse de alta como desarrollador como la publicación de apps.

Windows Phone Store

Windows Phone es el sistema operativo que estuvo inicialmente disponible en los dispositivos Nokia de gama alta, consiguiendo aumentar de forma significativa su presencia.

Windows Phone Store es la tienda oficial de app para Windows Phone, se lanzó en el 2010 y dispone de más de 130.000 aplicaciones.

Para publicar una aplicación es necesario darse de alta como desarrollador y pagar una cuota de 99 dólares.

Otras tiendas de aplicaciones

Existen otras alternativas de tiendas de aplicaciones presentes y futuras como Amazon appStore, Firefox Marketplace, Ubuntu, Bada Samsung Apps, en las que no entraremos, aunque hay que tenerlas presentes a la hora de decidir dónde distribuir las aplicaciones que podamos desarrollar.