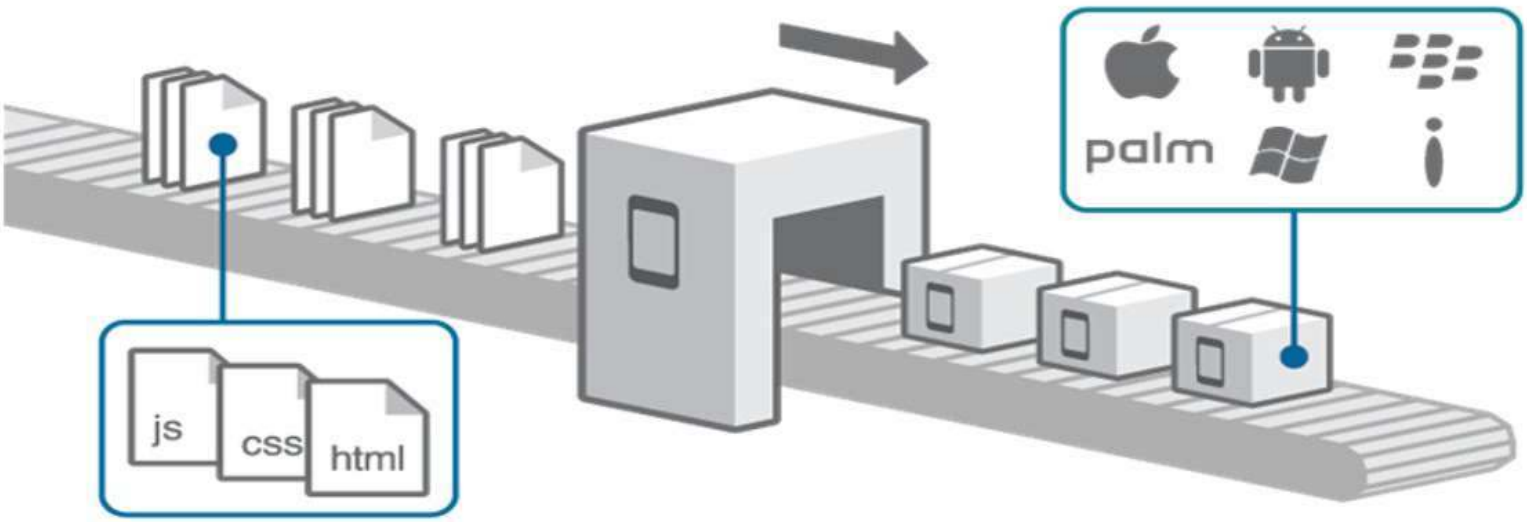
Introducción a la programación con...

PhoneGap



Antonio Javier Gallego Sánchez

Tabla de contenido

Contenidos

La tecnología *PhoneGap* o *Apache Cordova* permite compilar un código Web (HTML, CSS, JavaScript etc.) como si fuera una aplicación nativa para móvil. Es decir, con un solo desarrollo web es posible obtener aplicaciones nativas para diferentes plataformas destino como Android, iOS, Windows Phone, etc.

Esta herramienta además nos da acceso a los sensores y características nativas de los dispositivos móviles, como por ejemplo el GPS, acelerómetros, brújula, cámara, etc.

La librería *PhoneGap* se gestiona de forma muy sencilla mediante línea de comandos, dispone además de un fichero centralizado de configuración de nuestra aplicación, y la posibilidad de instalar multitud de *plugins* para el acceso a sensores o la ampliación de funcionalidad.

Los contenidos principales del libro son:

Introducción

PhoneGap es un framework de código abierto para el desarrollo de aplicaciones para móviles. Su principal característica es ser multiplataforma, es decir, con un solo código de aplicación podemos utilizarlo en multitud de plataformas móviles, como Android, iOS o Windows Phone.

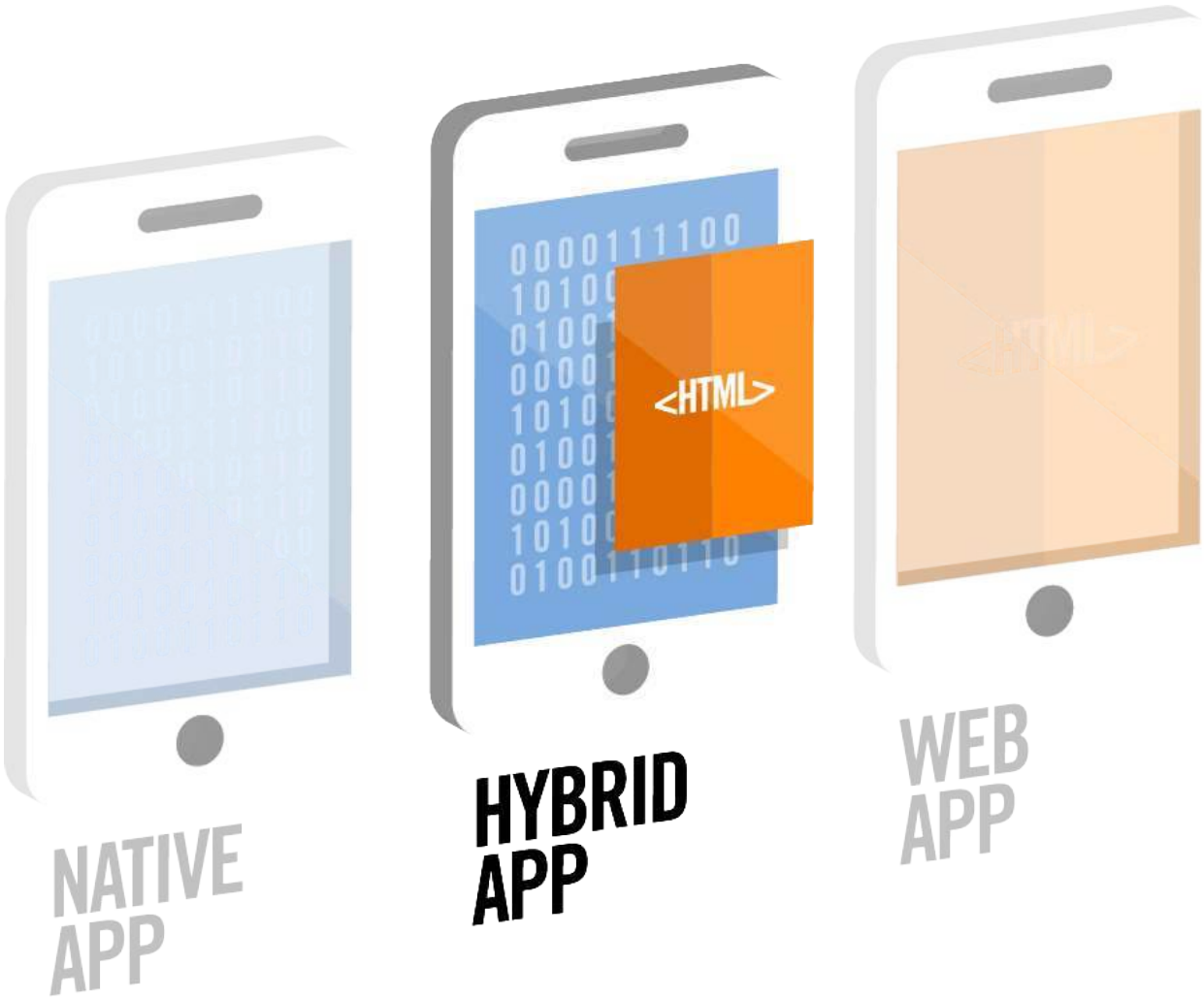
Inicialmente PhoneGap fue desarrollado por Nitobi bajo licencia de software libre, pero en Octubre de 2011 Adobe anunció oficialmente la adquisición de Nitobi, pasando así PhoneGap al control del gigante del software. Esto generó una gran incertidumbre entre los desarrolladores, pues el framework podía pasar a ser una tecnología propietaria, pero en una genial estrategia, Adobe donó PhoneGap a la fundación Apache, conservando de esta forma la integridad libre de PhoneGap.

En la actualidad, el proyecto en el sitio web de la fundación Apache esta nombrado como: "Apache Cordova", pero PhoneGap sigue siendo una especie de marca comercial, por lo que aún se sigue usando ese nombre para identificar al popular framework.

El núcleo de las aplicaciones PhoneGap se crea utilizando lenguajes de programación Web, como JavaScript, HTML5, CSS3, y con la ayuda de otros frameworks de desarrollo y de la propia API de PhoneGap. Esta API nos permite acceder mediante código JavaScript a características nativas del móvil, como por ejemplo: Acelerómetro, cámara, contactos, eventos, geolocalización, redes o almacenamiento.

Posteriormente, y para cada una de las plataformas móviles para las que queramos generar nuestra aplicación, tendremos que incluir este núcleo Web como parte de la aplicación nativa. De esta forma podremos generar una aplicación "nativa" para cada plataforma móvil aprovechando para todas ellas el mismo núcleo de la aplicación.

Las aplicaciones desarrolladas con PhoneGap se consideran aplicaciones *híbridas*. Una aplicación es híbrida cuando es una aplicación nativa con una capa intermedia de herramientas que hacen uso de otros lenguajes de programación. Por el contrario se considera nativa cuando ha sido desarrollada íntegramente utilizando la API y lenguaje de programación que proporciona la compañía que vende el producto. Esta técnica de programación tiene varias ventajas: el usuario sentirá que la aplicación es parte del sistema operativo, nos permitirá distribuir la aplicación como una aplicación nativa, y además nos ahorrará muchísimo trabajo.



Puedes encontrar más información de PhoneGap en la dirección http://phonegap.com/, donde podrás descargar la última versión del *framework*.

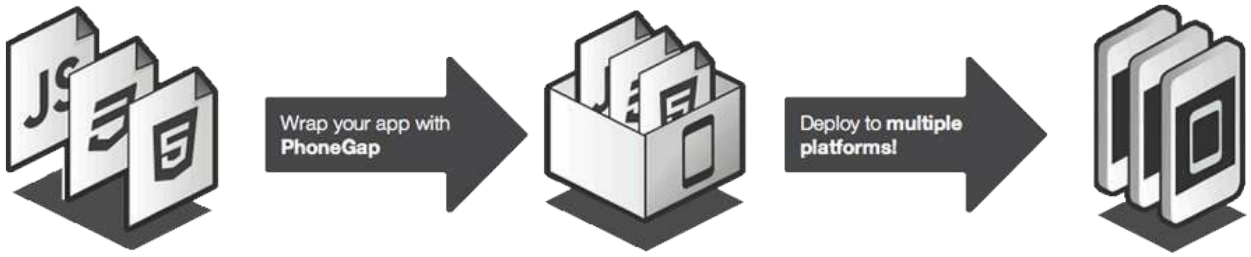
Cómo trabaja PhoneGap

El esquema básico de funcionamiento de PhoneGap es el siguiente:

- Construir la aplicación usando estándares Web: HTML, HTML 5, CSS, CSS3, JavaScript o haciendo uso de otros frameworks para el desarrollo de aplicaciones Web.

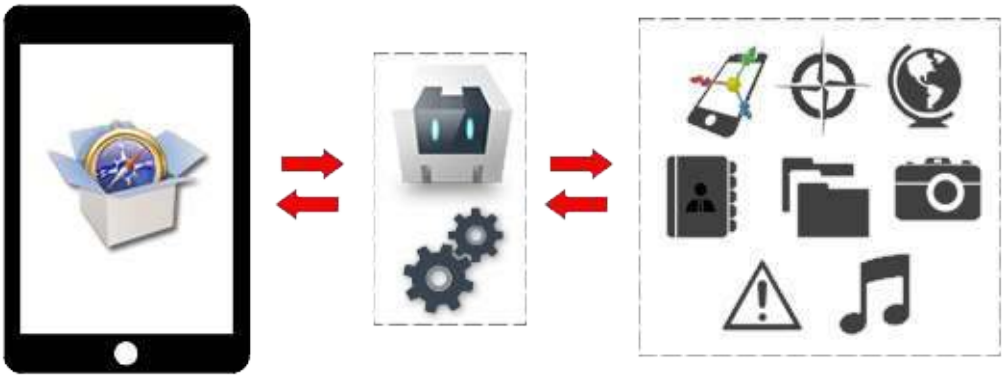
- Combinar la aplicación Web con PhoneGap, esto nos dará acceso a las características nativas de los dispositivos móviles.
- Configurar la aplicación en cada una de las plataformas para las que queramos generar la aplicación nativa.

A continuación se incluye un esquema de este proceso:



Es importante utilizar estándares web para que nuestra aplicación funcione en la mayoría de dispositivos móviles. Para el desarrollo de la aplicación, además de poder utilizar lenguajes de programación web HTML 5, CSS 3 y JavaScript, también soporta perfectamente el uso de frameworks de desarrollo web móvil como: Twitter Bootstrap, jQuery Mobile, Sencha Touch, Dojo, jQTouch, SproutCore, GloveBox, XUI, iScroll, entre otros.

Además, a través de la librería de PhoneGap se nos brindará el acceso desde la aplicación web a las diferentes características nativas o hardware de los dispositivos móviles, como por ejemplo el acelerómetro, la cámara, brújula, etc.



Soporte

Con PhoneGap es posible desarrollar aplicaciones para los siguientes sistemas operativos para móviles:

- Android
- iOS
- Windows Phone
- BlackBerry OS

- Amazon Fire OS
- Firefox OS
- Ubuntu
- Tizen

Permite el acceso a las características nativas o hardware de todas estas plataformas a través de una API en JavaScript que se comunica con el dispositivo. Las características a las que nos permite acceder son:

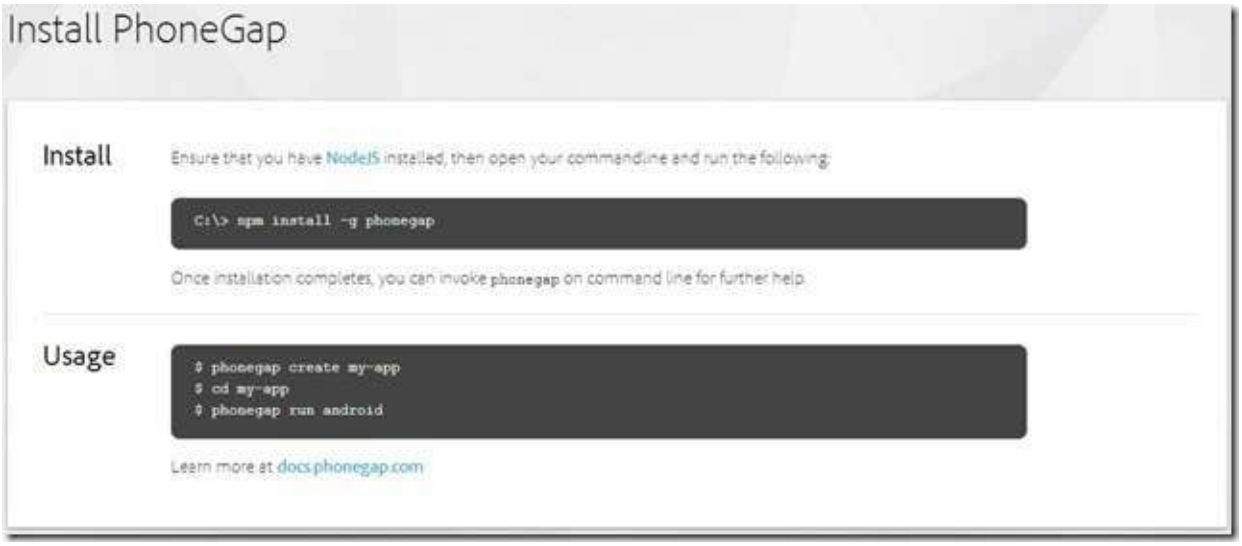
- Acelerómetro
- Estado de la batería
- Cámara
- Brújula
- Estado de la conexión
- Contactos
- Datos del dispositivo
- Eventos
- Gestión de archivos
- Geolocalización
- Navegador
- Contenido multimedia
- Sistema de notificaciones
- Pantalla de bienvenida o *Splashscreen*
- Almacenamiento
- Vibrador

Sin embargo no todas las plataformas soportan el acceso a todas las características nativas desde una aplicación de este tipo. Los principales sistemas operativos para móvil actuales (Android, iOS, WP) no presentan apenas problemas (aunque sí algunas particularidades de configuración), pero otros como Tizen o Firefox OS no permiten el acceso completo. En la siguiente dirección se puede consultar una tabla con el soporte para cada plataforma:

http://docs.phonegap.com/en/edge/guide_support_index.md.html#Platform%20Support

Instalación

Entramos a la página de PhoneGap y hacemos click en "Instalar PhoneGap". Como observaremos PhoneGap desde la versión 3.0 cambió y ya se puede instalar como un módulo de *Node.Js*, para poder crear mediante la línea de comandos la estructura de nuestras aplicaciones y posteriormente compilarlas.



Instalación de Node.Js

Si no tenemos instalado Node.Js entraremos a la página http://nodejs.org/ para descargarlo. El automáticamente reconocerá nuestro Sistema Operativo y procederá a bajar la última versión disponible. Los pasos para la instalación son los siguientes:

- Entramos en la página http://nodejs.org/ y descargamos la librería.
- Descomprimimos el archivo descargado y accedemos al directorio que se genera.
- Compilamos la librería. Este último paso dependerá del sistema operativo que utilicemos:
  - Si usamos Windows tendremos que ejecutar el *script*:

    ```
    vcbuild.bat
    ```

  - Si usamos Linux los siguientes comandos:

    ```
    ./configure
    make
    make install
    ```

- Si usamos Mac simplemente será ejecutar el instalador que se descarga.

Además, si al compilar nos indicara que no encuentra Python tendremos que comprobar que esté correctamente instalado y que lo encuentre en el PATH: `export PYTHON=/path/to/python`. También es posible que nos diese error por cuestión de permisos al realizar la instalación.

Instalación de PhoneGap

Una vez instalado Node.Js procederemos a descargar PhoneGap. Para esto abrimos una consola (o en Windows la consola de Node.Js llamada "*Node.js command prompt*") y ejecutamos:

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

Nota: en Windows ejecutaremos el mismo comando pero sin "sudo".

Este comando descargará e instalará automáticamente PhoneGap. Una vez completado el proceso podemos comprobar que esté correctamente instalado ejecutando el comando `phonegap`, el cual debería mostrar una salida similar a la siguiente:

```
Usage: phonegap [options] [commands]

Description:

  PhoneGap command-line tool.

Commands:
  ...
  ...
```

Interfaz de línea de comandos

Una vez tenemos instalado Node.Js y PhoneGap ya podemos crear nuestra primera aplicación. Pero para ello antes debemos conocer el funcionamiento de la interfaz de línea de comandos (CLI) de PhoneGap, el cual utilizaremos para realizar muchas operaciones, como por ejemplo crear nuevos proyectos, compilar, ejecutar, emular, etc. A continuación veremos las opciones más importantes de las que disponemos.

Ayuda

Para obtener ayuda de todos los comandos disponibles y su sintaxis tenemos que ejecutar:

```
$ phonegap help

# O también:
$ phonegap
```

Si queremos ver la ayuda detallada sobre un comando usaremos:

```
$ phonegap help <comando>

# O también:
$ phonegap <comando> --help
```

El comando "info" muestra un listado de información de ayuda como la versión del propio CLI de PhoneGap y de Node.Js, las plataformas y plugins instalados, las versiones del SDK de cada plataforma, etc. El ejecutar este comando, además de mostrar la información por pantalla, la guarda en un fichero "info.txt" en la misma carpeta.

```
$ phonegap info
```

Crear una aplicación

Para crear una nueva aplicación abriremos una consola (o en Windows la consola de Node.Js llamada "*Node.js command prompt*") y ejecutamos el siguiente comando:

```
$ phonegap create myapp com.example.myapp HelloWorld

# O también podemos ejecutar:
$ phonegap create myapp
```

Donde:

- El primer argumento, " `myapp` ", es el nombre de la carpeta donde se generará el contenido. Si el directorio no existe se creará automáticamente.
- El segundo argumento, " `com.example.myapp` ", es el nombre del paquete del proyecto usado como identificador. Este argumento es opcional y se puede modificar después desde el fichero " `config.xml` " (pero hay que llevar cuidado ya que se utiliza en el código generado, como en el nombre de los paquetes de Java). El valor por defecto es "com.phonegap.helloworld", por lo que se recomienda asignarle un valor apropiado. Normalmente se suele utilizar el nombre inverso de dominio de la organización, pero añadiendo el nombre de la aplicación. Este nombre no puede contener espacios ni números después de un punto.
- El tercer argumento, "HelloWorld", es el nombre del proyecto que se generará para las distintas plataformas (por ejemplo, el nombre del proyecto que se abrirá en Eclipse o XCode). Este argumento es opcional y también se puede modificar después desde el fichero " `config.xml` " (pero hay que llevar cuidado ya que se utiliza en el código generado). El valor por defecto es "HelloWorld", por lo que se recomienda indicar un valor adecuado. Este nombre no puede contener espacios y debe de empezar con mayúsculas.

Al ejecutar este comando se creará una carpeta con el nombre que le hayamos indicado, "*myapp*" en este caso, con todo el contenido necesario para empezar a desarrollar nuestra aplicación. La estructura de carpetas generada es la siguiente:

- *hooks* - Esta carpeta se utiliza para añadir *scripts* que se ejecutarán cuando se produzcan determinados eventos, como por ejemplo antes o después, de la compilación, etc. En la propia carpeta se incluye un fichero con instrucciones para su utilización.
- *merges* - Esta carpeta se utiliza para añadir código que nos permitirá modificar o

personalizar nuestra aplicación para determinadas plataformas. Es posible que esta carpeta no se cree con la instalación inicial, pero la podemos crear nosotros. El contenido que pongamos aquí se unirá o mezclará (de ahí su nombre) con el de la compilación por plataforma. Por ejemplo, si creamos la carpeta *merges/android/css/overrides.css*, al compilar para android se utilizará este CSS en lugar del que esté colocado en la carpeta *www/css*.

- *platforms* - Contiene el código específico de las plataformas móviles para las cuales se va a compilar. El código de esta carpeta es generado y no se ha de modificar manualmente.
- *plugings* - Contiene los *plugins* o módulos instalados para nuestra aplicación, los cuales se utilizan para añadir funcionalidad como el acceso a las características nativas del móvil.
- *www* - Contiene el código fuente de nuestra aplicación web. Esta carpeta es donde tendremos que desarrollar nuestra aplicación web de forma centralizada para después utilizarla en las distintas plataformas móviles que deseemos.

Al crear un nuevo proyecto todas las carpetas se encontrarán vacías a excepción de `www`, la cual incluye una primera aplicación de ejemplo (que mostrará el Hola Mundo de PhoneGap) con una estructura de carpetas básica (css, js, img, etc.). Para empezar a desarrollar un proyecto partiremos de este código de ejemplo, pudiendo modificar todo lo que queramos dentro de la carpeta `www`, incluso las subcarpetas de recursos (css, js, img, etc.).

Además de esta estructura, en la raíz de nuestra aplicación, encontraremos un fichero `config.xml` que se utiliza para la configuración del proyecto.

En las siguientes secciones se tratarán cada uno de estos puntos más en detalle.

> Importante: El resto de comandos que ejecutemos utilizando el CLI de PhoneGap lo tendremos que hacer dentro de la carpeta del proyecto o aplicación, o en una subcarpeta dentro de la misma.

Gestión de plataformas

Antes de poder compilar el proyecto tenemos que especificar las plataformas para las cuales se va a generar. Las plataformas disponibles dependerán del sistema operativo que utilicemos, si soporta el SDK en cuestión o si lo tiene instalado (y si no lo tiene lo tendremos que instalar). En general soporta las siguientes combinaciones:

	Mac	Linux	Windows
iOS	x		
Amazon Fire OS	x	x	x
Android	x	x	x
BlackBerry 10	x	x	x
Windows Phone 8			x
Windows			x
Firefox OS	x	x	x

Añadir plataformas

Para añadir una plataforma para la cual queremos compilar nuestro proyecto usaremos el siguiente comando:

```
$ phonegap platform add <nombre-de-la-plataforma>
```

Por ejemplo, en un Mac podremos añadir las siguientes plataformas a nuestro proyecto:

```
$ phonegap platform add ios
$ phonegap platform add amazon-fireos
$ phonegap platform add android
$ phonegap platform add blackberry10
$ phonegap platform add firefoxos
```

O desde el sistema operativo Windows todas las siguientes:

```
$ phonegap platform add wp8
$ phonegap platform add windows
$ phonegap platform add amazon-fireos
$ phonegap platform add android
$ phonegap platform add blackberry10
$ phonegap platform add firefoxos
```

Ejemplo: Añadir la plataforma Android

Por ejemplo, tras crear nuestro primer proyecto "myapp" en Linux, procedemos a añadir la plataforma Android como destino de la compilación, con el comando:

```
$ phonegap platform add android
```

Si nos indicara que el SDK no está instalado en primer lugar tendríamos que descargarlo e instalarlo nosotros manualmente (el CLI de PhoneGap no incluye opciones para realizar la instalación de los distintos SDKs).

También es posible que no encuentre la ruta del SDK:

```
Error: ANDROID_HOME is not set and "android" command not in your PATH.
```

Para arreglar este problema tenemos que asignar la ruta del SDK de Android a la variable " `ANDROID_HOME` " y al " `PATH` ". Para esto en una consola de Linux ejecutamos:

```
export ANDROID_HOME=/<installation location>/sdk
export PATH=$PATH:$ANDROID_HOME/tools:$ANDROID_HOME/platforms-tools
```

Si estamos en un Mac usaremos:

```
export ANDROID_HOME=/<installation location>/android-sdk-macosx
export PATH=${PATH}:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools
```

Para hacer estos cambios permanentes (de otra forma solo estarían disponibles para la consola actual) tendremos que añadir las líneas indicadas al fichero `.bashrc` (o a `.profile` o `.bash_profile` , dependiendo del sistema).

Después de arreglar estos problemas ya tendríamos que poder instalar la plataforma sin problemas.

Listar plataformas

Para obtener un listado con las plataformas disponibles para nuestro sistema operativo además de un listado con las plataformas que ya hemos añadido a nuestra aplicación tenemos que ejecutar el comando:

```
$ phonegap platforms ls

# O simplemente:
$ phonegap platforms
```

Lo que nos mostraría (en ambos casos):

```
Installed platforms: android 4.1.1, ubuntu 4.0.0
Available platforms: amazon-fireos, blackberry10, browser, firefoxos
```

> Nota: las opciones "platform" y "platforms" son equivalentes.

Eliminar una plataforma

Para eliminar una plataforma añadida podemos ejecutar alguno de los siguientes comandos (los cuales son equivalentes):

```
$ phonegap platform remove android
# O también:
$ phonegap platform rm android
```

Contenido de la carpeta "*platforms*"

Al añadir una plataforma o eliminarla únicamente estamos trabajando sobre la carpeta "*platforms*" de nuestro proyecto. Al añadir una nueva se genera un subdirectorio con el nombre de la plataforma en cuestión, copiando dentro de la misma el contenido de la carpeta "www" (la cual contiene el código de nuestro proyecto). Por ejemplo, para Android se copiaría en la ruta: `platforms/android/assets/www/` o para iOS en la ruta: `platforms/ios/www`.

Por este motivo es importante que no modifiquemos el contenido de nuestro proyecto o aplicación directamente en la carpeta *platforms*, ya que sería **borrado** o sobreescrito en la siguiente compilación. Cualquier cambio que queramos hacer en el mismo lo tendremos que realizar siempre sobre la carpeta base `www`.

Actualización de PhoneGap

En el futuro, cuando queramos actualizar PhoneGap, también podemos usar el *Package Manager* de Node.Js (`npm`). Para ello simplemente tenemos que ejecutar en una consola:

```
$ sudo npm update -g phonegap
```

Después de ejecutar este comando es posible que tengamos que actualizar también las plataformas de nuestros proyectos. En este caso tendremos que situarnos dentro de la carpeta de cada proyecto que queramos actualizar y ejecutar los siguientes comandos para cada plataforma:

```
$ phonegap platform update android
$ phonegap platform update ios
...etc.
```

Compilar una aplicación

Para compilar un proyecto ejecutamos el comando:

```
$ phonegap build
```

Este comando tiene que se ejecutado dentro de la carpeta del proyecto a compilar y además tienen que haber plataformas añadidas para el mismo, en otro caso nos mostraría el error:

```
[phonegap] executing 'cordova build'...
No platforms added to this project. Please use `cordova platform add <platform>`.
```

Al compilar se genera el código del proyecto para todas las plataformas añadidas, en caso de querer compilar solamente para una plataforma lo podríamos indicar de la forma:

```
$ phonegap build ios
```

El comando `build` en realidad realiza dos operaciones que son las de preparar el código (copiar el código actual del proyecto desde la carpeta `www` a todas las plataformas destino) y compilarlo. Estos dos comandos también los podríamos ejecutar por separado:

```
$ phonegap prepare
$ phonegap compile

# O también para solo una plataforma:
$ phonegap prepare ios
$ phonegap compile ios
```

Después de llamar al comando `prepare` podríamos abrir de forma manual el proyecto desde la carpeta de la plataforma que queramos con el SDK correspondiente, por ejemplo con Xcode, con Eclipse o con Android Studio.

Esta alternativa es interesante si lo que se desea es crear una versión inicial del proyecto mediante el CLI de PhoneGap y después cambiar al IDE del SDK correspondiente. En este caso tenemos que llevar cuidado ya que si volvemos a compilar para actualizar el código de la carpeta `www` es posible que perdiésemos algunos cambios.

Para más información sobre como desarrollar aplicaciones con cada uno del os IDEs soportados podéis consultar la web:

http://docs.phonegap.com/en/edge/guide_platforms_index.md.html#Platform%20Guides

> Nota: en caso de querer abrir nuestro proyecto para Android desde Eclipse tendremos que utilizar la opción "*File > New > Other... > Android > Android project from existing code*" (y no la opción de importar "*Existing projects into workspace*").

Probar una aplicación en un emulador o dispositivo real

Cada SDK para una plataforma móvil suele incluir un emulador que permite probar las aplicaciones desarrolladas. Estos emuladores también se pueden llamar desde el CLI de PhoneGap mediante el siguiente comando:

```
$ phonegap emulate

# O para una plataforma:
$ phonegap emulate android
```

Para que se abra el emulador correspondiente debe de estar bien configurado. Algunas plataformas ya traen uno por defecto (como iPhone), pero en otras será necesario configurar el emulador por defecto a utilizar (como en Android).

Es importante que antes de llamar al emulador ejecutemos el comando `build` el cual copiará la última versión de nuestro código, generará los proyectos correspondientes y los compilará. En otro caso podríamos tener errores.

Si actualizamos el código tendríamos que llamar de nuevo a `build` y a `emulate`, el cual actualizará los emuladores abiertos con la última versión.

También podemos probar nuestro código en un dispositivo real mediante el comando:

```
$ phonegap run android
```

El comando `run` en realidad es una combinación de los comandos `build` (por lo que no haría falta llamarlo previamente) e `install` (el cual instala un proyecto en una plataforma). Por defecto `run` instala y ejecuta el proyecto en una plataforma real (lo cual podemos forzar mediante el parámetro `--device`), o también lo podemos utilizar para ejecutar en un emulador añadiendo el parámetro `--emulator`.

Antes de ejecutar en un dispositivo real tenemos que configurarlo para permitir la instalación de aplicaciones de prueba. Este proceso varia para cada plataforma, por ejemplo, para Android tendremos que habilitar las opciones de desarrollador y la depuración a través de

USB, además, dependiendo del sistema operativo que utilicemos, es posible que tengamos que instalar drivers o realizar alguna configuración adicional.

Crear nuestra primera aplicación

Una vez instalado *Node.Js* y *PhoneGap* ya podemos crear nuestra primera aplicación. Para ello, y como se ha visto en secciones anteriores, tendríamos que ejecutar:

```
$ phonegap create myapp com.example.myapp HelloWorld
```

Este comando creará una carpeta llamada `myapp` con las subcarpetas: `hooks`, `platforms`, `plugings` y `www`. La carpeta `www` es la que contendrá el código fuente de nuestra aplicación web, la cual por defecto incluye una aplicación "Hola Mundo" de ejemplo con las subcarpetas *css*, *js* e *img* con los recursos correspondientes. El nombre de estas carpetas y código son los que se utilizan de forma común en el desarrollo web, pero se pueden modificar como queramos.

Es importante destacar que todo el contenido que se incluya dentro de la carpeta `www` se copiarán dentro de las aplicaciones que se generen para cada plataforma.

La carpeta y html llamados "*spec*" se utilizan para realizar pruebas unitarias a nuestro código Javascript mediante el *Framework Jasmine*, por lo que si no lo vamos a utilizar se pueden eliminar sin problemas.

El archivo " `index.html` " es el que se abrirá por defecto al iniciar nuestra aplicación. Este archivo, al igual que el resto de contenidos, se pueden modificar como queramos, incluso el nombre del html inicial siempre que lo indiquemos en el fichero de configuración " `config.xml` ".

Para el desarrollo de nuestra aplicación lo podemos realizar en HTML nativo (con funcionalidades de HTML5, CSS3, JavaScript, etc.), pero también podemos utilizar algunos de los frameworks existentes para el desarrollo de aplicaciones web para móviles, como por ejemplo:

- Bootstrap: http://getbootstrap.com/
- JQueryMobile: http://jquerymobile.com/
- SenchaTouch: http://www.sencha.com/products/touch/
- Ionic: http://ionicframework.com/
- Kendo UI: http://www.telerik.com/kendo-ui1
- SideTap: http://harvesthq.github.io/Sidetap/
- etc.

Además, para completar el desarrollo de nuestra aplicación podemos utilizar la gestión de eventos que incorpora PhoneGap mediante su librería JavaScript o un gran número de *plugins* para acceder a las características nativas de los dispositivos móviles, como por ejemplo la cámara, la geolocalización, etc. Estas características se abordarán en el siguiente apartado.

Configuración

PhoneGap permite configurar nuestras aplicaciones desde un fichero centralizado para indicar, por ejemplo, el icono de la aplicación, una imagen o *splash screen* de inicio, los plugins a utilizar, etc.

Además también veremos en esta sección como personalizar nuestra aplicación para cada una de las plataformas destino mediante la utilidad de *merges*.

Fichero de configuración

En cada proyecto se incluye un fichero de configuración `config.xml` que nos permite establecer las principales opciones de configuración de nuestra aplicación de forma global (para todas las plataformas destino).

Este fichero se situa en la carpeta raíz del proyecto (`app/config.xml`), pero por cuestiones de compatibilidad con versiones anteriores de PhoneGap también está soportado desde la ruta: `app/www/config.xml`

Al compilar un proyecto (mediante el comando `build` o `run` del CLI) se crean versiones específicas de este fichero de configuración para cada plataforma, lo que nos permite establecer configuraciones más específicas. La ruta donde el fichero es copiado varía dependiendo de la plataforma, por ejemplo:

```
app/platforms/ios/AppName/config.xml
app/platforms/blackberry10/www/config.xml
app/platforms/android/res/xml/config.xml
```

Para más información sobre configuraciones específicas podéis consultar la sección "*Configuration*" de cada plataforma en:

http://docs.phonegap.com/en/edge/guide_platforms_index.md.html#Platform%20Guides

A continuación se describen las opciones de configuración generales para todas las plataformas, pero además de estas hay muchas específicas dependientes de la plataforma:

- iOS Configuration: http://docs.phonegap.com/en/edge/guide_platforms_ios_config.md.html#iOS%20Configuration
- Android Configuration: http://docs.phonegap.com/en/edge/guide_platforms_android_config.md.html#Android%20Configuration
- BlackBerry 10 Configuration: http://docs.phonegap.com/en/edge/guide_platforms_blackberry10_config.md.html#BlackBerry%2010%20Configuration

Elementos de configuración principales

A continuación se muestran las primeras líneas del fichero `config.xml` generado por defecto al crear un nuevo proyecto de PhoneGap. Esta configuración es compatible para todas las plataformas:

```
<widget id="com.example.hello" version="1.0.0">
    <name>HolaMundo</name>
    <description>
        Ejemplo de aplicación con PhoneGap
    </description>
    <author email="email@email.com" href="http://miweb.com">
        Nombre autor
    </author>
    <content src="index.html" />
    <access origin="*" />
    <preference name="permissions" value="none" />
    <preference name="..." value="..." />
    ...
</widget>
```

Donde:

- `<widget>` : su atributo `id` especifica el paquete de la aplicación (usando el *reverse-domain-name*), y su atributo `version` es el número completo de versión de la app (siguiendo la notación *major/minor/patch*, ver más información en la sección "Versión de la aplicación").
- `<name>` : especifica el nombre del proyecto y de la app, es el que aparecería al instalar la app junto a su icono y en los *markets*.
- `<description>` y `<author>` : son metadatos e información de contacto que se utilizará al publicar la app en los *markets*.
- `<content>` : es un atributo opcional para indicar la página inicial de nuestro código. Por defecto se utilizará `index.html` .
- `<access>` : define un conjunto de dominios externos a los que se le permite acceder. Por defecto, al indicar `*` se le permite acceder a cualquier servidor. Para más información consultar la ayuda sobre *Domain Whitelist*.
- `<preference>` : establece opciones de configuración siguiendo la notación de pares nombre-valor, sin distinguir mayúsculas y minúsculas. En las siguientes secciones se tratarán estas opciones de configuración, las cuales pueden ser genéricas o específicas para algunas plataformas.

Las preferencias globales se aplicarán para todas las plataformas para las que se compile el proyecto.

- *fullscreen* permite mostrar u ocultar la barra de estado de la pantalla. El valor por defecto es *false*. Ejemplo:

```
<preference name="fullscreen" value="true" />
```

- *orientation* permite controlar si se ha de permitir que la pantalla gire o no. Los valores permitidos son: *default*, *landscape*, o *portrait*, siendo *default* el valor por defecto (el cual permite ambas orientaciones). Ejemplo:

```
<preference name="orientation" value="landscape" />
```

Además de todas estas opciones de configuración también es posible establecer los iconos y *splashscreens* que se utilizarán para cada plataforma. En las siguientes secciones veremos estos temas más en profundidad.

Iconos

Los iconos se pueden especificar utilizando la etiqueta `<icon>` en el fichero `config.xml`, por ejemplo:

```
<icon gap:platform="ios" height="57" src="www/res/icon/ios/icon-57.png" width="57" />
```

Donde:

- `src`: (requerido) localización de la imagen relativa a la carpeta del proyecto.
- `gap:platform`: (opcional) plataforma para la que se utilizará el icono.
- `width`: (opcional) ancho en píxeles.
- `height`: (opcional) alto en píxeles.

Para establecer un icono que se utilizará en todas las plataformas podemos establecer la siguiente configuración:

```
<icon src="icon.png" />
```

En caso de no especificar ningún logo por defecto se utilizará el icono de Apache Cordova.

Podemos utilizar tantas etiquetas como sea necesario para establecer los iconos para cada plataforma, incluso podemos indicar varios para una misma plataforma con distintos tamaños o densidades. Por ejemplo:

```
<icon gap:platform="android" gap:qualifier="ldpi" src="www/res/icon/android/icon-36-ld
pi.png" />
<icon gap:platform="android" gap:qualifier="mdpi" src="www/res/icon/android/icon-48-md
pi.png" />
<icon gap:platform="android" gap:qualifier="hdpi" src="www/res/icon/android/icon-72-hd
pi.png" />
<icon gap:platform="android" gap:qualifier="xhdpi" src="www/res/icon/android/icon-96-x
hdpi.png" />
```

Otro ejemplo para establecer todos los iconos de iOS:

```
<icon gap:platform="ios" height="57" src="www/res/icon/ios/icon-57.png" width="57" />
<icon gap:platform="ios" height="72" src="www/res/icon/ios/icon-72.png" width="72" />
<icon gap:platform="ios" height="114" src="www/res/icon/ios/icon-57-2x.png" width="114"
 />
<icon gap:platform="ios" height="144" src="www/res/icon/ios/icon-72-2x.png" width="144"
 />
```

Splash Screens

Igual que la configuración de los iconos, PhoneGap también permite definir un *splashscreen* para las distintas plataformas, por ejemplo para Android y para iOS:

```
<gap:splash gap:platform="android" gap:qualifier="port-ldpi" src="www/res/screen/andro
id/screen-ldpi-portrait.png" />
<gap:splash gap:platform="android" gap:qualifier="port-mdpi" src="www/res/screen/andro
id/screen-mdpi-portrait.png" />
<gap:splash gap:platform="android" gap:qualifier="port-hdpi" src="www/res/screen/andro
id/screen-hdpi-portrait.png" />
<gap:splash gap:platform="android" gap:qualifier="port-xhdpi" src="www/res/screen/andr
oid/screen-xhdpi-portrait.png" />

<gap:splash gap:platform="ios" height="480" src="www/res/screen/ios/screen-iphone-port
rait.png" width="320" />
<gap:splash gap:platform="ios" height="960" src="www/res/screen/ios/screen-iphone-port
rait-2x.png" width="640" />
<gap:splash gap:platform="ios" height="1136" src="www/res/screen/ios/screen-iphone-por
trait-568h-2x.png" width="640" />
<gap:splash gap:platform="ios" height="1024" src="www/res/screen/ios/screen-ipad-portr
ait.png" width="768" />
<gap:splash gap:platform="ios" height="768" src="www/res/screen/ios/screen-ipad-landsc
ape.png" width="1024" />
```

Versión de la aplicación

Tanto Android como iOS soportan una cadena con una nombre (o número) de versión alternativo además del que se utiliza en el *market*. En Android es el *versionCode* y en iOS el *CFBundleVersion*. A continuación se muestra como podemos establecer estas versiones en el `config.xml`:

```
<widget id="io.cordova.hellocordova"
    version="1.0.0"
    android-versionCode="7"
    ios-CFBundleVersion="3.3.3">
```

Si esta versión no se especifica se construirá a partir del atributo `version`. Como se ha indicado este atributo se construye siguiendo la notación *major/minor/patch*, por lo que se usarán estos valores de la forma:

```
// version = MAJOR.MINOR.PATCH-whatever
versionCode = PATCH + MINOR * 100 + MAJOR * 10000
CFBundleVersion = "MAJOR.MINOR.PATCH"
```

Personalizar plataformas mediante *merges*

En ocasiones se hace necesario especificar algunos recursos de forma separada para una plataforma. En este caso no deberíamos modificar la carpeta correspondiente de `platforms` ya se sobreescribiría en la próxima compilación.

Con este propósito se creo la carpeta *merges*, la cual contiene los assets que se han de copiar o reemplazar para las plataformas especificadas. La estructura de esta carpeta es la siguiente:

- En primer lugar tendremos que crear subcarpetas con el mismo nombre que la carpeta de `platforms` que deseemos sobreescribir.
- Dentro de esa subcarpeta crearemos una estructura espejo a la carpeta `www`, pero únicamente situando los ficheros y carpetas que deseemos reemplazar.

Por ejemplo, el fichero `merges/android/css/overrides.css` sobreescribiría el css situado en `www/css/overrides.css` pero solamente para la plataforma Android.

Si por ejemplo quisieramos especificar un estilo distinto para la plataforma iOS, podríamos almacenar dicho recurso en:

```
merges/ios/css/overrides.css
```

El cual cargaríamos desde el fichero `index.html` situado en `www/index.html` de forma normal:

```
<link rel="stylesheet" type="text/css" href="css/overrides.css" />
```

Además tendríamos que crear un css vacío (o con algún estilo por defecto) en `www/css/overrides.css` que utilicen el resto de plataformas que no sean "iOS" para evitar que apareciera un error de fichero no encontrado.

Eventos

La librería de PhoneGap incluye una serie de eventos JavaScript a los cuales podemos escuchar para realizar determinadas acciones cuando se produzcan. Estos eventos son:

- *deviceready*
- *pause*
- *resume*
- *backbutton*
- *menubutton*
- *searchbutton*
- *startcallbutton*
- *endcallbutton*
- *volumedownbutton*
- *volumeupbutton*

Si instalamos el *plugin* para el control de la batería (*org.apache.cordova.battery-status*) se añadirán los siguientes eventos (ver sección de *plugins*):

- *batterycritical*
- *batterylow*
- *batterystatus*

Si instalamos el *plugin* para obtener información de la red (*org.apache.cordova.network-information*) se añadirán los siguientes eventos (ver sección de *plugins*):

- *online*
- *offline*

En general para suscribirnos a un evento utilizaremos el siguiente código:

```
document.addEventListener("nombre-evento", funcionCallback, false);
```

Donde:

- *nombre-evento*: es el evento al que queremos escuchar.
- *funcionCallback*: es la función que se llamará cuando el evento se produzca.
- *false*: afecta al orden en el que se lanzará el evento, al inicio (cuando se producte) o al final (cuando ya ha terminado).

A continuación se describen los principales eventos de PhoneGap y se incluyen ejemplos de uso.

deviceready

Este evento se ejecuta cuando la librería de PhoneGap se ha cargado completamente. Su uso es esencial para cualquier aplicación y lo tendremos que utilizar de forma inicial para asegurarnos de que la API se ha cargado completamente.

A continuación se incluye un ejemplo de uso:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo</title>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">

    // Esperamos a que se cargue la API de PhoneGap
    function onLoad() {
        document.addEventListener("deviceready", onDeviceReady, false);
    }

    // La API de PhoneGap ya está disponible
    function onDeviceReady() {
        // Ya podemos usar la librería de PhoneGap
    }
    </script>
  </head>
  <body onload="onLoad()">
  </body>
</html>
```

pause

Este evento se produce cuando la aplicación pasa a segundo plano. A continuación se incluye un ejemplo completo de uso:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pause Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">

    function onLoad() {
        document.addEventListener("deviceready", onDeviceReady, false);
    }

    // Añadimos el evento cuando la API esté lista
    function onDeviceReady() {
        document.addEventListener("pause", onPause, false);
    }

    // Se ha producido el evento pause!
    function onPause() {
    }
    </script>
  </head>
  <body onload="onLoad()">
  </body>
</html>
```

> En todos los eventos tendremos que esperar a que la API de PhoneGap termine de cargar.

resume

Este evento se produce cuando la aplicación estaba en segundo plano y se vuelve a mostrar en primer plano. A continuación se incluye un ejemplo completo de uso:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Resume Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">

    function onLoad() {
        document.addEventListener("deviceready", onDeviceReady, false);
    }

    // Añadimos el evento cuando la API esté lista
    function onDeviceReady() {
        document.addEventListener("resume", onResume, false);
    }

    // Se ha producido el evento "resume"!
    function onResume() {
    }
    </script>
  </head>
  <body onload="onLoad()">
  </body>
</html>
```

backbutton

Este evento se produce cuando el usuario pulsa el botón atrás. A continuación se incluye un ejemplo completo de uso:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Back Button Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">

    function onLoad() {
        document.addEventListener("deviceready", onDeviceReady, false);
    }

    // Añadimos el evento cuando la API esté lista
    function onDeviceReady() {
        document.addEventListener("backbutton", onBackKeyDown, false);
    }

    // Se ha producido el evento "backbutton"!
    function onBackKeyDown() {
    }
    </script>
  </head>
  <body onload="onLoad()">
  </body>
</html>
```

Salir de la aplicación

Por defecto, al pulsar el botón *back* no se cerrará la aplicación. Si queremos que se cierre tendremos que hacer:

```
document.addEventListener("backbutton", onBackKeyDown, false);

function onBackKeyDown() {
   navigator.app.exitApp()
}
```

Hemos de tener cuidado con esta funcionalidad porque cerrará la aplicación siempre que se pulse dicha tecla. Si nuestra aplicación tiene varias páginas y queremos que se use el botón *back* para volver a la página anterior y además que en la primera página cierre la aplicación tendremos que distinguir entre las páginas y solo cerrar en la primera.

menubutton

Este evento se produce cuando el usuario pulsa el botón menú. Para su utilización, igual que en el resto de casos, simplemente tendremos que hacer (ejemplo abreviado):

```
document.addEventListener("menubutton", onMenuKeyDown, false);

function onMenuKeyDown() {
   // Se ha producido el evento "menubutton"!
}
```

volumedownbutton

Este evento se produce cuando el usuario pulsa el botón para bajar el volumen. A continuación se incluye un ejemplo abreviado de uso:

```
document.addEventListener("volumedownbutton", onVolumeDownKeyDown, false);

function onVolumeDownKeyDown() {
    // Se ha producido el evento "volumedownbutton"!
}
```

volumeupbutton

Este evento se lanza cuando se pulsa el botón para subir el volumen. Igual que en el resto de casos, para su utilización tenemos que hacer:

```
document.addEventListener("volumeupbutton", onVolumeUpKeyDown, false);

function onVolumeUpKeyDown() {
    // Se ha producido el evento "volumeupbutton"!
}
```

Plugings

PhoneGap, además de permitir encapsular aplicaciones Web y compilarlas como aplicaciones nativas para distintas plataformas móviles, nos brinda el acceso a características hardware de los dispositivos a través de su sistema de *plugins*. Por ejemplo tenemos disponibles *plugins* para trabajar con la cámara, brújula, geolocalización, etc.

Estos *plugins* se pueden añadir a nuestros proyectos según los necesitemos mediante el CLI de PhoneGap. A partir de la versión 3.0 de PhoneGap, al crear un nuevo proyecto no incluye ningún *plugin*, por lo que si queremos alguno tendremos que añadirlo manualmente.

PhoneGap también permite crear nuestros propios plugins (para más información consultar: "http://docs.phonegap.com/en/edge/guide_hybrid_plugins_index.md.html"), pero la opción más común es utilizar alguno de los ya disponibles.

En la siguientes secciones vamos a ver como gestionar estos *plugins* y algunos ejemplos de uso.

Gestión de *plugins*

En esta sección vamos a ver como buscar, añadir, listar y eliminar *plugins*, además de algunas características avanzadas adicionales.

Buscar *plugins*

En la dirección "http://plugins.cordova.io" podemos consultar la lista completa de todos los *plugins*, incluyendo los que han sido desarrollados por la comunidad.

También es posible utilizar el CLI para buscar *plugins* en este registro. Por ejemplo, para buscar un *pluging* para códigos de barras usaríamos:

```
$ phonegap plugin search bar code

com.phonegap.plugins.barcodescanner - You can use the BarcodeScanner plugin...
...
```

En este caso devuelve varios resultados, un *plugin* oficial de PhoneGap y varios desarrollados por la comunidad. Es importante destacar que las búsquedas no distinguen mayúsculas y minúsculas, por lo que también podríamos haber buscado "Bar Code".

Si buscasemos únicamente por "Bar" obtendríamos muchos más resultados, incluyendo un *plugin* para trabajar con la barra de estado y otro para mostrar notificaciones:

```
$ phonegap plugin search bar

org.apache.cordova.statusbar - Cordova StatusBar Plugin
org.chromium.notifications - This plugin allows apps to show notifications in the stat
us bar.
...
```

Añadir un *plugin*

Para añadir un plugin a un proyecto simplemente tenemos que indicar el repositorio que deseemos y el sistema se encargará de descargarlo e instalarlo. A continuación se incluye una lista de los *plugins* oficiales más utilizados y como tendríamos que instalarlos.

Obtener información del dispositivo:

```
$ phonegap plugin add cordova-plugin-device
```

Información sobre la conexión y la batería: (ver sección sobre eventos)

```
$ phonegap plugin add cordova-plugin-network-information
$ phonegap plugin add cordova-plugin-battery-statusbattery-status
```

Acelerómetro, brújula y geolocalización:

```
$ phonegap plugin add cordova-plugin-device-motion
$ phonegap plugin add cordova-plugin-magnetometer
$ phonegap plugin add cordova-plugin-device-orientation
$ phonegap plugin add cordova-plugin-geolocation
```

Cámara, reproducción y captura de contenidos multimedia:

```
$ phonegap plugin add cordova-plugin-camera
$ phonegap plugin add cordova-plugin-media-capture
$ phonegap plugin add cordova-plugin-media
```

Acceso a ficheros del dispositivo y de la red:

```
$ phonegap plugin add cordova-plugin-file
$ phonegap plugin add cordova-plugin-file-transfer
```

Notificaciones mediante diálogos y vibración:

```
$ phonegap plugin add cordova-plugin-dialogs
$ phonegap plugin add cordova-plugin-vibration
```

Contactos:

```
$ phonegap plugin add cordova-plugin-contactscontacts
```

Globalización:

```
$ phonegap plugin add cordova-plugin-globalization
```

***Splashscreen* o pantalla inicial:**

```
$ phonegap plugin add cordova-plugin-splashscreen
```

Abrir nuevas ventanas de navegador (InAppBrowser):

```
$ phonegap plugin add cordova-plugin-inappbrowser
```

Consola de depuración:

```
$ phonegap plugin add cordova-plugin-console
```

> Nota: los *plugins* se descargarán para las plataformas que tenga el proyecto actualmente instaladas. Si posteriormente se añaden más plataformas estos *plugins* se descargarán también de forma automática para la nueva plataforma.
>
> Nota: al instalar un *plugin* de forma automática también se actualizarán los permisos necesarios para cada plataforma. Por ejemplo, en Android se actualizará el Manifest para solicitar los permisos necesarios.

Ver los *plugins* instalados

Para ver los *plugins* añadidos al proyecto actual podemos usar alguno de los siguientes comandos (todos son equivalentes), los cuales mostrarán un listado de paquetes instalados, junto con su versión y nombre:

```
$ phonegap plugin

# O también:
$ phonegap plugin ls

# O también:
$ phonegap plugin list

cordova-plugin-battery-status 1.1.1 "Battery"
cordova-plugin-console 1.0.2 "Console"
cordova-plugin-device 1.1.0 "Device"
cordova-plugin-network-information 1.1.0 "Network Information"
cordova-plugin-vibration 2.0.0 "Vibration"
```

Eliminar un *plugin*

Para eliminar un *plugin* simplemente tenemos que referirnos a él por el mismo nombre del paquete que utilizamos para su instalación (o que podemos ver si ejecutamos `phonegap plugin ls`). Por ejemplo, para eliminar el soporte a la consola de depuración ejecutaríamos:

```
$ phonegap plugin rm cordova-plugin-console

# O también:
$ phonegap plugin remove cordova-plugin-console
```

Añadir o elminar varios *plugins* a la vez

Si queremos añadir o eliminar varios *plugins* a la vez simplemente tenemos que indicarlos usando el mismo comando pero separados por espacios, de la forma:

```
$ phonegap plugin add cordova-plugin-console cordova-plugin-device

$ phonegap plugin rm cordova-plugin-console cordova-plugin-device
```

Advanced Plugin Options

Para opciones más avanzadas sobre la gestión de *plugins*, como por ejemplo instalar una versión en concreto de un *plugin* o instalar desde otro repositorio, podéis consultar la dirección:

http://docs.phonegap.com/en/edge/guide_cli_index.md.html#The%20Command-Line%20Interface_advanced_plugin_options

Uso de la API de PhoneGap

A continuación vamos a ver ejemplos de uso de algunos de estos *plugins*.

> **Importante**: En todos los casos tendremos que esperar a que la API de PhoneGap se haya terminado de cargar.

Información del dispositivo (*cordova-plugin-device*)

Este *plugin* permite obtener información del dispositivo como modelo, sistema operativo, etc. Para su utilización tenemos que esperar que se cargue la API de PhoneGap y ya podremos acceder a estos valores a través de las propiedades del objeto "`device`":

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
    console.log(device.cordova);
}
```

Las propiedades que podemos utilizar son:

- *device.cordova*: Obtiene la versión de PhoneGap.
- *device.model*: Obtiene el nombre o modelo del dispositivo.
- *device.platform*: Obtiene el nombre del sistema operativo.
- *device.uuid*: Devuelve el *Universally Unique Identifier* (UUID) del dispositivo.
- *device.version*: Obtiene la versión del sistema operativo utilizado.

Para más información podéis consultar la siguiente página:

http://plugins.cordova.io/#/package/org.apache.cordova.device

Información sobre la conexión (*cordova-plugin-network-information*)

Este *plugin* proporciona información sobre el tipo de conexión a la red y además añade eventos para saber cuando el dispositivo se conecta o se desconecta de la red.

Para obtener información sobre el tipo de conexión utilizamos el objeto `connection.type` el cual podrá contener los siguientes valores:

- Connection.UNKNOWN
- Connection.ETHERNET
- Connection.WIFI
- Connection.CELL_2G
- Connection.CELL_3G
- Connection.CELL_4G
- Connection.CELL
- Connection.NONE

A continuación se incluye un ejemplo de uso:

```
function checkConnection() {
    var networkState = navigator.connection.type;

    var states = {};
    states[Connection.UNKNOWN]  = 'Tipo de conexión desconocida';
    states[Connection.ETHERNET] = 'Ethernet';
    states[Connection.WIFI]     = 'WiFi';
    states[Connection.CELL_2G]  = 'Cell 2G';
    states[Connection.CELL_3G]  = 'Cell 3G';
    states[Connection.CELL_4G]  = 'Cell 4G';
    states[Connection.CELL]     = 'Cell generic';
    states[Connection.NONE]     = 'Sin conexión';

    alert('Tipo de conexión: ' + states[networkState]);
}

checkConnection();
```

Para escuchar a los eventos de *online* y *offline*, igual que para el resto de eventos, simplemente tendremos que hacer:

```
document.addEventListener("online", onOnline, false);
document.addEventListener("offline", onOffline, false);

function onOnline() {
    // Se ha conectado a la red
}

function onOffline() {
    // Se ha perdido la conexión a la red
}
```

Brújula (*cordova-plugin-device-orientation*)

Este *plugin* proporciona el acceso a la brújula, el cual devuelve la orientación del dispositivo con respecto al norte como si de una brújula se tratase. El valor retornado estará en grados entre 0 y 359.99.

Una vez haya cargado la API de PhoneGap podremos utilizar los siguientes métodos:

- `navigator.compass.getCurrentHeading`
- `navigator.compass.watchHeading`
- `navigator.compass.clearWatch`

Para obtener el valor actual utilizamos el método `navigator.compass.getCurrentHeading`, el cual puede devolver la orientación o un error en caso de no pode acceder. A continuación se incluye un ejemplo de uso:

```
function onSuccess(heading) {
    alert('Orientación: ' + heading.magneticHeading + 'º');
};

function onError(error) {
    alert('Compass Error: ' + error.code);
};

navigator.compass.getCurrentHeading(onSuccess, onError);
```

Pero este método solo devuelve un valor. Si queremos que el valor se actualice según se mueva el dispositivo tenemos que utilizar el método `navigator.compass.watchHeading`. Este método llamará a la función "success" cada vez que obtenga la orientación con la frecuencia que se indique en las opciones, por ejemplo:

```
function onSuccess(heading) {
    var element = document.getElementById('heading');
    element.innerHTML = 'Orientacion: ' + heading.magneticHeading;
};

function onError(compassError) {
    alert('Compass error: ' + compassError.code);
};

var options = {
    frequency: 3000
}; // actualizar cada 3 segundos

var watchID = navigator.compass.watchHeading(onSuccess, onError, options);
```

El valor de retorno de esta función (`watchID`) nos permite detener el listener cuando queramos mediante la función:

```
navigator.compass.clearWatch(watchID);
```

> Este *plugin* tiene un funcionamiento muy similar al del acelerómetro o al de la geolocalización, todos disponen un método para obtener el valor actual y métodos para obtener los valores en tiempo real (con una determinada frecuencia) y para detener el listener.

Ejercicios 1

Ejercicio 1 - Instalación y prueba (1 punto)

En este ejercicio vamos a realizar la instalación de las librerías necesarias para utilizar PhoneGap y a continuación crearemos una primera aplicación de ejemplo y la probaremos. Para ello seguiremos los siguientes pasos:

- Instalar las librerías de *Node.Js* y de *PhoneGap* siguiendo los pasos explicados en la teoría.
- Añadir plataformas de compilación para Android e iOS.
- Compilar y probar el código de ejemplo inicial en los emuladores de Android e iOS.
- Modificar el código de ejemplo para que en la página "index.html" únicamente se muestre "¡Hola Mundo!", en grande y centrado en la pantalla. Al pulsar sobre este texto se irá a una segunda página que pondrá "¡Adiós!", también en grande y centrado.
- Importar los proyectos generados para cada plataforma en el IDE correspondiente (Android e iOS), analiza el código generado y prueba a ejecutarlos.

Ejercicio 2 - Importar nuestros proyectos (2 puntos)

En este ejercicio vamos a empaquetar como aplicaciones PhoneGap el resultado obtenido en los últimos ejercicios de esta asignatura. En todos los proyectos que vamos a crear tendremos que:

- Instalar las plataformas de Android e iOS, compilar las aplicaciones y probarlas (en emulador o dispositivo real).
- Incluir todos los assets (css, js, imágenes, etc.) que se utilicen dentro de la propia aplicación. Es decir, si por ejemplo algún proyecto accede a alguna librería desde su CDN tendríamos que descargarla, incluirla entre los recursos de la aplicación y modificar el código para que acceda de forma local. De esta forma las aplicaciones podrán funcionar sin necesidad de tener conexión a Internet.
- Es importante que nos aseguremos de haber completado cada uno de los ejercicios correspondientes antes de incluirlos en la aplicación de PhoneGap.

Vamos a crear dos nuevos proyectos PhoneGap, con los nombres de carpetas y contenidos siguientes:

- *html*: en este primer proyecto con PhoneGap empaquetaremos el resultado del ejercicio 3 de la primera sección "Introducción al desarrollo Web", llamado "Ejercicio 3 - Estructura de HTML" en el que había que utilizar las nuevas etiquetas semánticas de HTML5.
- *bootstrap*: en esta segunda aplicación usaremos la web realizada en el último ejercicio de la sección sobre Bootstrap en el que había que crear una web de contenido libre que utilizara dicha librería.

Ejercicios 2

Ejercicio 1 - Configuración de PhoneGap (0.5 puntos)

Para completar el último ejercicio se pide que realicéis los siguientes pasos para todas las aplicaciones:

- Desactivar el *fullscreen* de las aplicaciones (ya que está activado por defecto).
- Crear un icono para cada aplicación con una resolución de 128x128 píxeles. Podéis hacer solamente uno por aplicación que se utilice para ambas plataformas y para todas las resoluciones. El dibujo del icono podéis diseñarlo vosotros mismos o simplemente poner el logo de la tecnología utilizada: HTML5 o Bootstrap, y el nombre de la misma.
- Crear un *splashscreen* para cada aplicación con una resolución de 480x800px. Podéis hacer solamente uno por aplicación que se utilice para ambas plataformas y para todas las resoluciones. Al igual que el icono, el dibujo del *splashscreen* podéis crearlo vosotros mismos o simplemente poner el logo de la tecnología utilizada y el nombre de la misma. Revisad que la configuración del fichero `config.xml` sea correcta y apunte a vuestras imágenes, y que tras compilar las imágenes se hayan copiado a la ruta correspondiente dentro de la carpeta *platforms*.

Ejercicio 2 - Usando la API de PhoneGap (2.5 puntos)

En este ejercicio vamos a probar algunas de las funcionalidades que nos ofrece la API de PhoneGap.

En primer lugar creamos un nuevo proyecto con el nombre "PhoneGapAPI" y modificamos el HTML del "Hola Mundo" para que cargue la librería de PhoneGap y espere hasta que esté lista (ver el código básico de ejemplo que se incluye en la sección "Eventos - *deviceready*").

A continuación vamos a crear un log de eventos en esta aplicación. En primer lugar creamos la sección HTML donde mostraremos el log:

```
<div id="log" style="margin:5px;padding:5px;border:1px solid silver;color:gray">
</div>
```

Desde JavaScript podremos añadir texto a esta sección haciendo:

```
var element = document.getElementById("log");
element.innerHTML += "- API de PhoneGap cargada.<br/>";
```

Se pide que mostréis mensajes de log cuando se produzcan los siguientes eventos:

- Se cargue la API de PhoneGap (*deviceready*).
- La aplicación se ponga en *background* (*pause*).
- La aplicación salga de *background* (*resume*).
- Se pulse el botón atrás (*backbutton*).
- Se pulse el botón menú (*menubutton*).
- Se pulse el botón para bajar el volumen (*volumedownbutton*).
- Se pulse el botón para subir el volumen (*volumeupbutton*).

Instalar el *plugin* para obtener información sobre el dispositivo utilizado y mostrarla en una tabla antes del log.

Instalar el *plugin* para obtener información sobre la conexión y añadir esta información:

- Añadir al log los eventos de *online* y *offline*.
- Añadir el tipo de conexión a la tabla inicial de información del dispositivo.

Instalar el *plugin* de la brújula y configurar el listener para que actualice la orientación del dispositivo cada segundo (mostrar junto a la información inicial del dispositivo).

Instala otro *plugin* (el que tu quieras de entre los oficiales de PhoneGap) y añade la información del mismo a la aplicación.

A continuación se puede ver una captura de ejemplo de la aplicación a desarrollar:

Dispositivo:

- PhoneGap version: 3.6.4
- Modelo: GT-I9505
- Plataforma: Android
- UUID: 5d039a5229fa686e
- Versión: 4.4.2
- Tipo de conexión: WiFi
- Orientación: 269.4º
- Aceleración X: -2.0
- Aceleración Y: 5.8
- Aceleración Z: 8.6

Log de eventos:

- API de PhoneGap cargada.
- Conexión online.
- Evento volumeupbutton.
- Evento volumedownbutton.
- Evento backbutton.
- Evento menubutton.
- Evento pause.
- Evento resume.