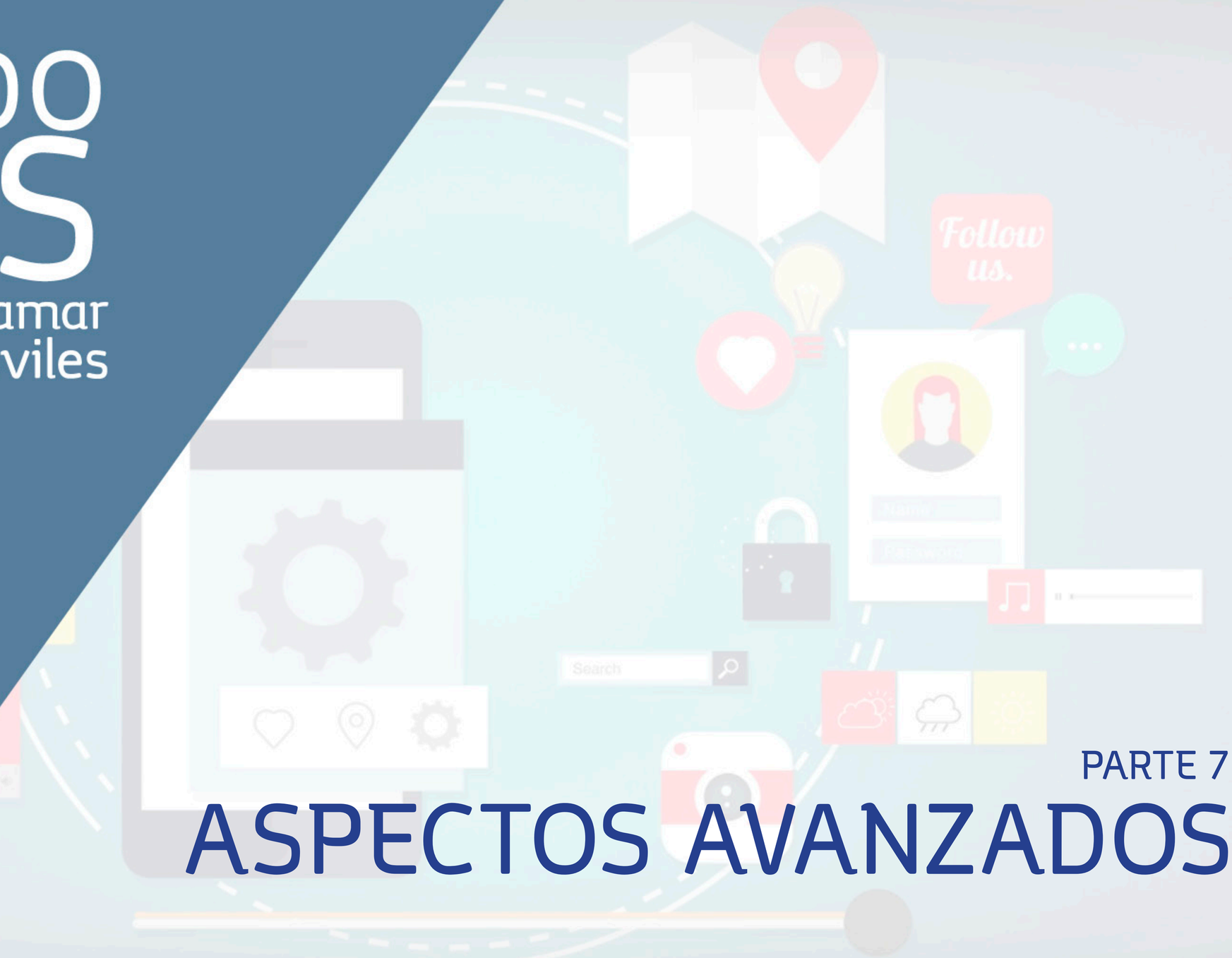
CREANDO APPS

aprende a programar aplicaciones móviles



PARTE 7

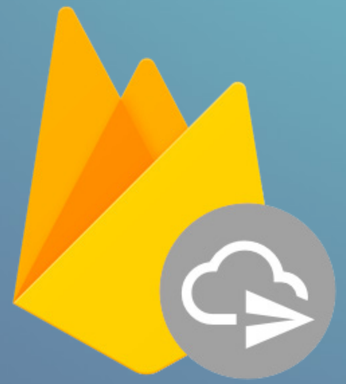
ASPECTOS AVANZADOS

BACKEND DE VERDAD

Para poder hacer aplicaciones reales en muchas ocasiones necesitaremos realizar procesos que no se pueden realizar en el móvil y necesitaremos apoyarnos en un backend remoto para poder hacerlas. Tradicionalmente estos servicios teniamos que programarlos nosotros, pero en la actualidad cada vez tenemos estos servicios disponibles programados por terceras personas sin necesidad de mantenerlas nosotros mismos.

Vamos a ver tres ejemplos de las funcionalidades que nos ofrece uno de los proveedores de servicios que hemos utilizado anteriormente, Firebase.

Firebase Cloud Messaging[1]

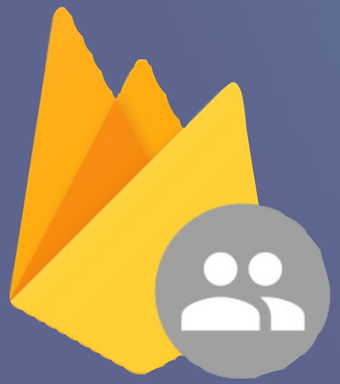


Nos permite mandar mensajes push a nuestros dispositivos, es decir, sin que ellos nos lo soliciten y de forma asíncrona, el servidor puede mandar información a los dispositivos. Estos sistemas se pueden usar en aplicaciones de notificación de chats, mails, etc.

Nos permite mandar mensajes que puedan ver nuestros usuarios o de control del dispositivo. A un grupo de usuarios, a los que estén suscritos a un tema o a un usuario completo. Casi cualquier aplicación puede beneficiarse de este tipo de servicio.

(1) `https://firebase.google.com/docs/cloud-messaging/`

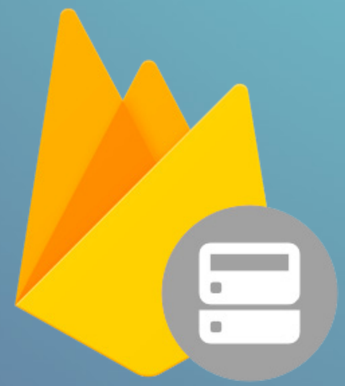
Firebase Authentication[2]



La seguridad en las aplicaciones móviles es cada vez más importante. Para ayudarnos con la identificación de usuarios podemos utilizar esta librería que nos permitirá de manera sencilla proveer a nuestras aplicaciones de un método de identificar a nuestros usuarios. Esta librería ofrece muchas formas de uso aunque hay dos de ellas que son los métodos más frecuentes, autenticación por email y password o usar un proveedor de identidad federado tipo google o facebook.

(2) `https://firebase.google.com/docs/auth/`

Firebase Realtime Database[3]



Este servicio nos permite sincronizar los datos de un dispositivo de manera automática, ya que cada vez que existe algún cambio en los datos, este servicio se lo notifica a todos los dispositivos conectados. No solo eso, en caso de que algún dispositivo esté offline, se encargará de actualizarlos en cuanto vuelva a estar disponible. Internamente guarda los datos en formato JSON, por lo que estamos capacitados para poder utilizar este servicio cuando queramos.

(3) `https://firebase.google.com/docs/database/`

7.1 Aspectos avanzados / BACKEND DE VERDAD

Back-End	parte de la aplicación que recoge las interacciones con el usuario. Se ejecuta en el dispositivo
Front-End	(Almacenamiento) - se refiere al lugar físico de persistencia, memoria del teléfono o servicio en la nube
Nube	(Cloud Computing) - servicios ofertados por terceros que se pueden usar sin que el usuario tenga que conocer aspectos avanzados de cómo gestionarlos
Notificaciones push	mensajes emitidos por el backend sin necesidad que el frontend los solicite
Autenticación	comprobación de que el usuario es quien dice ser
Proveedor de identificación federado	servicios que permiten identificarnos de manera externa (google, twitter, etc)

7.2 Aspectos avanzados / PATRONES DE USO

Como hemos visto en el curso, desarrollando aplicaciones híbridas podemos crear una app que funciona tanto en plataformas android como iOS. Pero, ¿debemos usar exactamente la misma aplicación en ambos entornos? ¿Hay alguna diferencia en desarrollar una aplicación para android y iOS? ¿Podemos hacer aplicaciones personalizadas para cada plataforma en el mismo proyecto de PhoneGap? Vamos a responder a estas preguntas.

Diferencias de diseño

En primer lugar, existen algunas diferencias que deberían reflejarse en el propio diseño de la aplicación. Vamos a ver 3 de ellas.

1. La plataforma iOS sólo funciona en terminales Apple. En cambio android funciona en terminales de muchos fabricantes diferentes, que realizan personalizaciones sobre la plataforma. Así, debemos tener en cuenta cuáles son nuestros terminales objetivo en android para hacer un diseño adaptado al terminal y su personalización.
2. En android los terminales tienen un botón back (atrás) físico, que los terminales iOS no tienen. Por tanto, las apps que desarrollemos deben tener esto en cuenta, por ejemplo, mostrando un botón atrás sólo para terminales iOS.
3. Las listas, menús y cabeceras tienen un diseño específico en cada plataforma, y los usuarios de cada una esperan que sean de esa forma.

Por suerte ambas plataformas cuentan con guías de estilo que debemos usar, el guía de interfaces de iOS[4] y la de Material Design de Android[5].

(4) Guía de estilo iOS
`https://developer.apple.com/ios/human-interface-guidelines/`

(5) Guía de estilo de Material Design
`https://material.google.com/material-design/environment.html`

El plugin device[6]

En Apache Cordova contamos con un plugin que nos permite detectar el dispositivo sobre el que se está ejecutando nuestra aplicación. De esta forma, podemos personalizar desde nuestro código JavaScript si queremos por ejemplo mostrar un botón sólo para iOS o cambiar el estilo de un menú para android.

El plugin share[7]

Otra característica que puede requerir nuestra aplicación es poder compartir un recurso (documento, imagen, video) con otras apps. Esta acción se realiza de forma distinta en función del sistema, y la implementa una llamada nativa. Para ello existe un plugin, share, que permite compartir un recursos con otras aplicaciones y que funciona en varias plataformas. Este plugin no viene incluido de serie en PhoneGap y no puede usarse con aplicaciones de la PhoneGap Developer App sino que tendríamos que construir directamente la aplicación para probarla en nuestro dispositivo.

Back	atrás en inglés, usado para describir el botón atrás físico de los dispositivos android
Tabs	del inglés pestañas, referido a un menú que usa pestañas
Material Design	guía de estilo para el diseño de aplicaciones android
Device	del inglés dispositivo, es un plugin de Cordova
Share	del inglés compartir, es otro plugin de Cordova

(6) Apache Cordova device plugin
`https://github.com/apache/cordova-plugin-device/`

(7) Apache Cordova share plugin
`https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin`

7.3 Aspectos avanzados / RENDIMIENTO

Las aplicaciones móviles se ejecutan en un entorno, el terminal móvil, que tiene capacidad limitada de recursos: memoria, CPU, almacenamiento en disco, acceso a la red, batería, etc. Por ejemplo, si nuestra aplicación está accediendo a la red tenemos que tener en cuenta que puede tener tiempos de respuesta largos, desconexiones, etc. O si queremos pintar una imagen con millones de píxeles, el sistema tienen que cargarla en memoria y tomará un tiempo hacerlo, incluso podemos consumir toda la memoria disponible y la aplicación pasa a un estado de error (crash). Por eso, es importante hacer un uso eficiente de los recursos para que nuestra aplicación se ejecute de forma fluida y el usuario tenga una buena experiencia de uso.

Vamos a dar una serie de tips para que el rendimiento de nuestras apps no decaiga.

1. Evita los accesos innecesarios a la red. Muchas veces nuestra aplicación necesita varios recursos que se descargan de la red. En lugar de descargarlos de uno en uno, podemos descargar todos en una única petición. También haciendo uso de cache y almacenamiento local evitamos estos accesos. Otro caso es que si vamos a mostrar un listado de recursos muy grandes, en lugar de descargarnos todos lo vayamos haciendo poco a poco, según los vamos mostrando al usuario. Como regla general sólo debemos descargar la información que vamos a mostrarle al usuario, ni más ni menos.

2. No esperar a los datos para empezar a pintar la interfaz. Hay interfaces que para pintarse depende de datos de la red. Pero podemos ir pintando la interfaz en el navegador vacía y cuando tengamos datos la actualizamos. También podemos usar splash screens para que el usuario no pierda la paciencia.

3. Usar funciones que usen aceleración por hardware, por ejemplo, las animaciones CSS están optimizadas por encima de las que hacemos con JavaScript.

4. Usar fastclick para evitar el retardo de 300ms para escuchar eventos móviles.

5. Optimiza las imágenes al tamaño de la pantalla, para evitar descargar imágenes de megas cuando van a pintarse en un espacio pequeño de la pantalla.

6. No usar frameworks pesados (en tamaño) de JavaScript, como jQuery, a no ser que de verdad estemos haciendo uso de sus características.

`http://coenraets.org/keypoint/phonegap-performance/#0`

El debugger de Phonegap

Phonegap cuenta con una aplicación para hacer debugging, weinre[8] (inspector web remoto), que permite usar las devtools de Chrome para revisar nuestra aplicación. Usando esta herramienta podemos detectar problemas de rendimiento al ver cómo se cargan los recursos de nuestra app y ver si hay, por ejemplo, alguno muy pesado o algún proceso consume mucha memoria.

Para utilizar weinre debemos primero instalarlo usando npm, y luego lanzarlo desde nuestro terminal. A continuación incluimos un script en nuestra app de PhoneGap para que se conecte al servidor a través de la red local. Cuando ya tenemos nuestra app funcionando en la PhoneGap Developer App, usamos una URL que nos da la aplicación para abrir una web en nuestro navegador. En esta web aparecen las devtools que inspeccionan directamente nuestra app en el móvil. Podemos usar el inspector, la consola JavaScript y en general todas las funcionalidades de las devtools.

(8) *Debugging PhoneGap with weinre*
`http://docs.phonegap.com/references/developer-app/debugging/`

Performance	del inglés rendimiento
Reflow	cálculo que hace el navegador para pintar un elemento en pantalla que ha cambiado
jQuery	librería JavaScript de propósito general
Debugger	aplicación que usamos para encontrar bugs (errores) en un programa
Devtools	herramientas del navegador web que facilitan el desarrollo de aplicaciones web
Inspector Web Remoto (weinre)	herramientas que permite usar las devtools sobre una aplicación móvil hecha con PhoneGap ejecutándose en un terminal móvil

DISTRIBUCIÓN DE APLICACIONES MÓVILES

Tradicionalmente la distribución de aplicaciones móviles ha sido bastante tediosa, lenta y requería de máquinas potente. Ahora con PhoneGap Build[9] tenemos la opción de realizarla de manera sencilla. Lo único que necesitamos es saber la url del repositorio git de la aplicación que queremos empaquetar y una aplicación de escaneo de códigos qr instalada en el dispositivo en el que queremos instalarla.

Una vez hemos indicado el repositorio y tras unos breves instantes tenemos el código qr. Lo escaneamos y comienza la descarga de la aplicación. Tras esto... ¡ya tenemos nuestra aplicación corriendo en nuestro dispositivo!

(9) PhoneGap build `https://build.phonegap.com/`

Código QR sistema de almacenamiento de información. Es similar al código de barras pero en dos dimensiones

7.5 Aspectos avanzados / APIs HTML5

En el curso hemos usado los plugins de Cordova para acceder a las capacidades del dispositivo: la cámara, el acelerómetro, etc. Pero en el futuro podremos acceder a todas estas capacidades directamente desde el navegador mediante las APIs HTML5[10]. Con estas APIs vamos a poder acceder directamente (a través del navegador) a las capacidades del dispositivo, como la cámara, el acelerómetro, la geolocalización, etc.

(10) WebAPI en Mozilla Developer Network
`https://developer.mozilla.org/es/docs/WebAPI`