Información sobre deploy de aplicaciones Ionic

iOS:

- XCode (ide) para poder compilar nuestra versión en modo productivo.
- Una cuenta de developer de Apple.
- Un certificado de provisioning profile en modo productivo vigente.
- Un certificado de app development y distribution válido.

Primero tendremos que compilar nuestra app en modo productivo de la siguiente manera:

```
ionic capacitor build ios --prod
```

Esto compilará nuestro código que nos permitirá abrirlo como un proyecto de iOS nativo, para ello utilizaremos el Xcode y abriremos el proyecto de extensión .xcworkspace ubicado en ./platforms/ios/ de nuestro proyecto.
Una vez que tenemos los certificados correctos, XCode los manejará de manera automática según el tipo de compilación que elijamos.
Para subir nuestra app al store, iremos a la opción Archive, del menú Product-> Archive, esto nos abrirá el Xcode Organizer que contiene todos los builds que realizamos, seleccionamos este último y le damos a “Upload to App Store” y con esto damos por finalizada la subida, pero todavía nuestra app no está disponible en la tienda.
Con ello lo que hicimos hasta ahora fue subir nuestra app compilada al store, pero antes de que esté disponible para todos, debe pasar por una

revisión por parte de Apple, y recién finalizado y aprobado por ellos, nuestra app estará disponible en la tienda.

Certificados iOS

Como Desarrollador de Apple (Apple Developer), cuando se crea un proyecto necesita ser firmado con un Certificado de Distribución o Distribution Certificate. Este certificado te autentifica como el creador de la app. Es por ese motivo, que el nombre con el que te hayas registrado o el de la empresa (si te has registrado como una organización) aparecerá como "Vendedor" de la app dentro de la App Store.
Para generarlo, se necesita subir el archivo Certificate Signing Request (CSR). Para crearlo se necesita recurrir al Keychain Access en una Mac y completado el proceso de subida del CSR, nos descargaremos el archivo ios_distribution.cer que se instalará en nuestro KeyChain.
Con la cuenta de desarrollador se pueden publicar varias apps y utilizar el mismo Certificado de Distribución para ellas. Sin embargo, hay que tener en cuenta que el Certificado de Distribución caduca después de un año.
Lo que identifica a cada app como única es lo que se denomina App ID, tiene que existir un App Id por cada App. Si queremos habilitar las notificaciones push dentro de una app, cuando configuremos el App ID tendremos que habilitar la casilla "Push notifications" antes de terminar y validar el proceso de registro.
La creación del Provisioning Profile es otro paso obligatorio. Es el link entre el desarrollador de la app y el proyecto (un App ID). Se necesita el Provisioning Profile tanto para el Ad Hoc (Distribution - AdHoc) como para el App Store distribution. En este caso, el Provisioning Profiles caduca junto con el Certificado de Distribución.
Por último, si queremos incluir las Notificaciones Push, necesitaremos un último certificado, llamado Push SSL Certificate, este nos permite la conexión entre el servidor de notificaciones y el servicio de notificaciones push de Apple. Cada app que utilice Notificaciones Push, necesitará de este certificado en concreto. El proceso para generar el certificado de push es el mismo que para generar el certificado de distribución, es decir necesitamos generar un archivo CSR del KeyChain

de una Mac, subirlo en la interfaz de Apple developer y luego descargar en este caso el aps_production.cer

Vencimiento certificados

En el caso de que la cuenta de Apple developer expire, las apps se eliminarán del App Store, pero continuarán funcionando en los dispositivos donde han sido instaladas. Si se renueva la licencia, las apps aparecerán de nuevo en la tienda.
Para los Distribution Certificate se tiene que crear un nuevo certificado para poder compilar la aplicación, realizar una actualización o publicarla de nuevo. Las apps que ya han sido publicadas en la tienda no se verán afectadas.
Si el Provisioning Profiles caducó, hay que generarlo de nuevo para poder actualizar la app conectada a este.
Por último, en el caso de que el Push Certificate caduque, no se podrá enviar notificaciones push desde la app que está conectada a este certificado.

Android

Para android los requerimientos cambian, solo necesitaremos una cuenta de developer de google play que cuesta unos $25USD por única vez.
Generamos nuestro archivo compilado de la siguiente manera:

```
ionic capacitor build android --prod --release
```

Esto nos generará el compilado según las configuraciones establecidas en el archivo config.xml y el archivo se encontrará en la ruta:

```
platforms/android/build/outputs/apk
```

Dicha apk está sin firmar, por eso todavía no se puede subir al store, para firmarla debemos utilizar la llave (signing key), en el caso de que no la tengamos, tendremos que generarla de la siguiente manera.

Vamos a utilizar una herramienta que viene con el android SDK, dentro de esta carpeta por línea de comando haremos lo siguiente:

```
keytool -genkey -v -keystore nombreLlave.keystore -alias nombreAlias
-keyalg RSA -keysize 2048 -validity 10000
```

Con este ya tendremos nuestro archivo .keystore que es la llave para firmar nuestra apk. (Con esta llave tendremos que firmar todos los updates de la apk una vez subida al store, si esta llave la perdemos, no podremos actualizar nuestra apk).
Para firmarla, utilizaremos otra herramienta que viene con el android SDK que se llama jarsigner.

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
nombreLlave.keystore nuestroAPK-unsigned.apk nombreAlias
```

Con la apk ya firmada, debemos utilizar una última herramienta llamada zipaling para optimizar nuestro APK, para encontrar la herramienta debemos ir a la carpeta:

/path/to/Android/sdk/build-tools/VERSION/zipalign

y para macOS por ejemplo estaría en:

~/Library/Android/sdk/build-tools/VERSION/zipalign

y corremos este comando:

zipalign -v 4 nuestroAPK-unsigned.apk nuestroAPK.apk

Con esto terminamos todos los pasos y estamos listos para subir nuestro apk al Google Play Store, para ello debemos ingresar aquí:

https://play.google.com/apps/publish/signup/

y una vez que ingresamos vamos al botón de Crear Aplicación y completamos todos los datos requeridos.

Como nota, cabe aclarar que cada vez que subamos una nueva versión de la apk, deberemos modificar el config.xml cambiando el número de versión en el atributo “versión” del .XML.

Créditos

Brian Ducca
https://www.joshmorony.com/deploying-capacitor-applications-to-ios-development-distribution/
https://www.joshmorony.com/deploying-capacitor-applications-to-android-development-distribution/