Firma digital de la aplicación

Toda aplicación Android debe estar firmada digitalmente para poder instalarse y ejecutarse sobre un sistema Android, bien sea sobre un emulador o sobre un dispositivo Android. Esto permite, en particular, identificar al desarrollador de la aplicación.

La aplicación se distribuye bajo la forma de un archivo con formato APK (*Android Package*).

El archivo `.apk` es un archivo de formato zip que contiene toda la aplicación: los archivos Java compilados como .class, los archivos de recursos y los archivos XML compilados con formatoWBXML (wap binary xml).

Durante la fase de desarrollo de una aplicación, el desarrollador puede compilarla, instalarla y ejecutarla numerosas veces. Esto supone firmar cada una de estas compilaciones, sin lo cual no es posible instalarlas sobre el sistema Android. No obstante, hasta aquí, no se ha requerido ninguna solicitud de firma digital y todas las compilaciones de las aplicaciones han podido instalarse sin problema sobre el emulador o sobre un dispositivo Android. Lo que contradice los principios expuestos.

La razón es sencilla. Para no entorpecer el trabajo del desarrollador solicitándole firmar cada una de las compilaciones, el proceso de firma difiere según el modo de compilación usado.

1. Compilación en modo debug

Es el modo de compilación usado por defecto en Eclipse con el plug-in ADT durante la fase de desarrollo de la aplicación. Durante la compilación de una aplicación en este modo, el plug-in ADT crea automáticamente un almacén de claves y crea una clave privada específicamente dedicada a la depuración (debug).

Esta clave se utiliza a continuación para firmar automáticamente el archivo apk, optimizarlo e instalarlo sobre el sistema Android. Todas estas fases se realizan de forma totalmente transparente para el desarrollador. Esto le permite, por tanto, no tener que preocuparse de este aspecto y centrarse en el desarrollo de la aplicación.

La aplicación firmada de esta forma no puede distribuirse en Play Store.

2. Compilación en modo release

Una vez terminada la aplicación, ésta debe compilarse en modo release (final). Este modo, a diferencia del modo debug (depuración), no firma automáticamente la aplicación. Esto permite firmar la aplicación con la clave privada de la cuenta de desarrollador. La aplicación firmada con esta clave podrá, a continuación, ser publicada y distribuida a terceros.

La firma de una aplicación final se realiza mediante un certificado digital auto-firmado cuya clave privada pertenece únicamente al desarrollador de la aplicación. Las etapas de creación de esta clave privada y del almacén de claves que las contiene hay que hacerlas sólo la primera vez. Las compilaciones siguientes podrán, en lo sucesivo, utilizar la clave creada anteriormente.

Es posible utilizar una clave nueva para cada aplicación, incluso para cada versión. No obstante, se recomienda utilizar la misma, especialmente en el caso de actualizar una aplicación, con el fin de:

- Permitir una actualización rápida y sencilla de la aplicación a los usuarios existentes. Una actualización de una misma aplicación requiere obligatoriamente el uso de una misma clave, sin lo cual la actualización se considerará como una nueva aplicación completa, sin vínculo alguno con la versión anterior. Y, dado que esta nueva aplicación utiliza el mismo nombre de paquete, el sistema rechazará su instalación.
- Poder ejecutarlas en el mismo proceso de sistema si lo desean y compartir los mismos datos propios de la aplicación como archivos de preferencias. Esta funcionalidad permite, por

ejemplo, instalar por separado la aplicación principal, gratuita, y los módulos suplementarios independientes, de pago.

La clave utilizada deberá, por tanto, tener una fecha de validez lo suficientemente extensa como para permitir cubrir todas las aplicaciones y todas las duraciones de sus actualizaciones. La validez de esta fecha se verificará únicamente en la fase de instalación de la aplicación sobre el sistema Android.

El desarrollador debe proteger y conservar el almacén y la clave secreta, así como las contraseñas asociadas, bajo el riesgo de que le roben su identidad. Sin ello, la seguridad de las aplicaciones y de los datos de usuario asociados no estará garantizada.

a. Protección del código

Proguard es una herramienta que permite, entre otros, reducir, optimizar y ofuscar el código Java compilado, sin modificar su funcionamiento. Esto permite, a la vez, optimizar el tamaño del archivo binario y hacer que la aplicación sea más complicada de estudiar.

Para ello, Proguard toma en cuenta únicamente el código Java utilizado y deja de lado el código no utilizado. Renombra las clases, los métodos y los campos utilizando nombres cortos sin significado alguno. Por último, optimiza el archivo binario generado.

De este modo, resulta muy difícil comprender el código Java de un programa binario producido mediante la herramienta Proguard que se haya podido descompilar.

El uso de la herramienta Proguard es opcional. Está indicada para aquellos desarrolladores que quieran proteger el código de sus aplicaciones y se recomienda encarecidamente cuando la aplicación utiliza la librería de verificación de licencias para reforzar la seguridad de la clave del dispositivo.

El sistema de compilación de Android gestiona el uso de la herramienta de forma automática y transparente al desarrollador. La herramienta se utiliza solamente para las compilaciones en modo release.

Por defecto, un proyecto Android no utiliza la herramienta Proguard. Para activar el uso de esta herramienta, hay que agregar la propiedad `proguard.config` en el archivo`default.properties` que se encuentra en la raíz del proyecto. Esta propiedad debe especificar la ruta y el nombre del archivo de configuración destinado a la herramienta Proguard. Existe un archivo de configuración por defecto, `proguard.cfg`, que se provee en la raíz del proyecto y es apropiado para la mayoría de los proyectos.

Sintaxis

```
proguard.config=/carpetas/proguard.cfg
```

Ejemplo

```
proguard.config=proguard.cfg
```

El ejemplo utiliza el archivo `proguard.cfg` que se provee por defecto.

Una vez ejecutada, la herramienta crea varios archivos en la carpeta `proguard` del proyecto. Esos archivos son `dump.txt`, `mapping.txt`, `seeds.txt` y `usage.txt`. Permiten, en particular, convertir las trazas de excepción ofuscadas con los nombres de las clases, de los métodos y de los miembros originales.

Las compilaciones siguientes borrarán estos archivos. Se recomienda, por tanto, salvaguardarlos, así como el proyecto completo, para cada aplicación publicada.

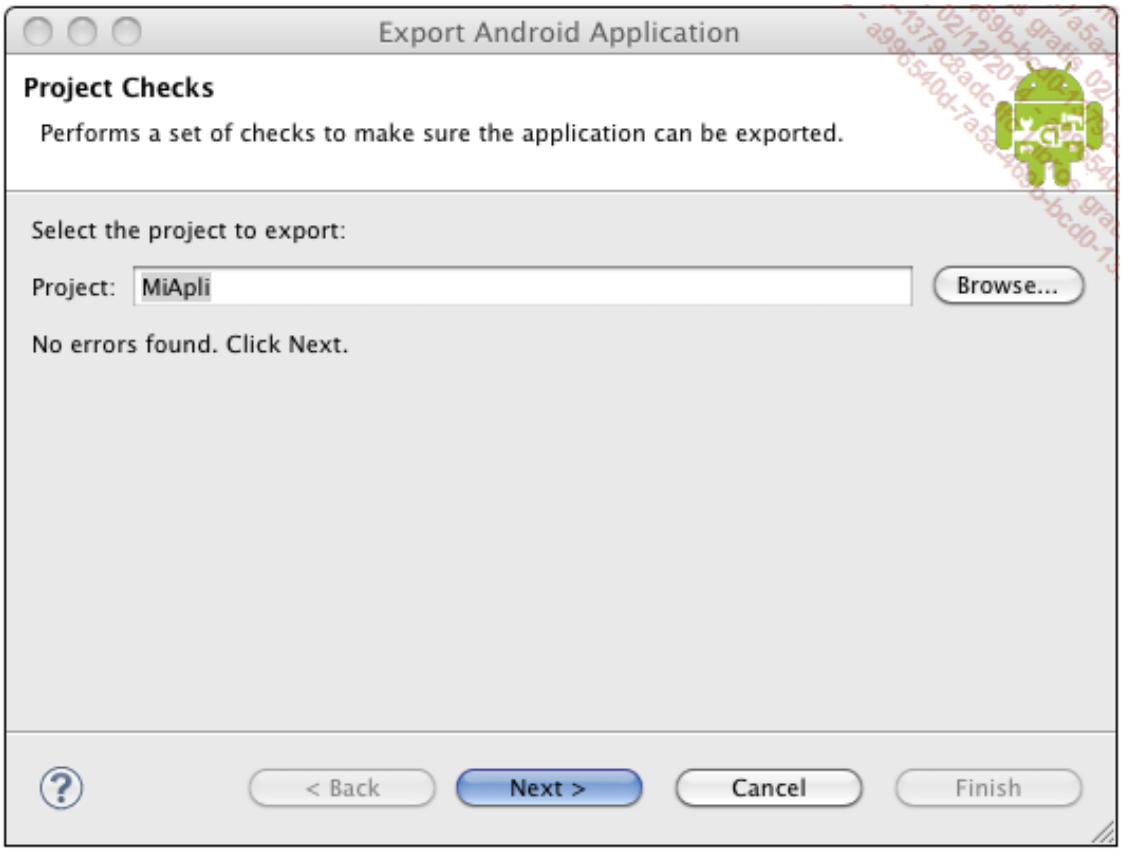
Es posible que sea necesario parametrizar Proguard para que no modifique los nombres de los métodos especificados en los atributos `android:onClick` de los widgets con el objetivo de mantener funcionales estas relaciones.

b. Firmar la aplicación

He aquí cómo compilar y firmar una aplicación en modo release en Eclipse:

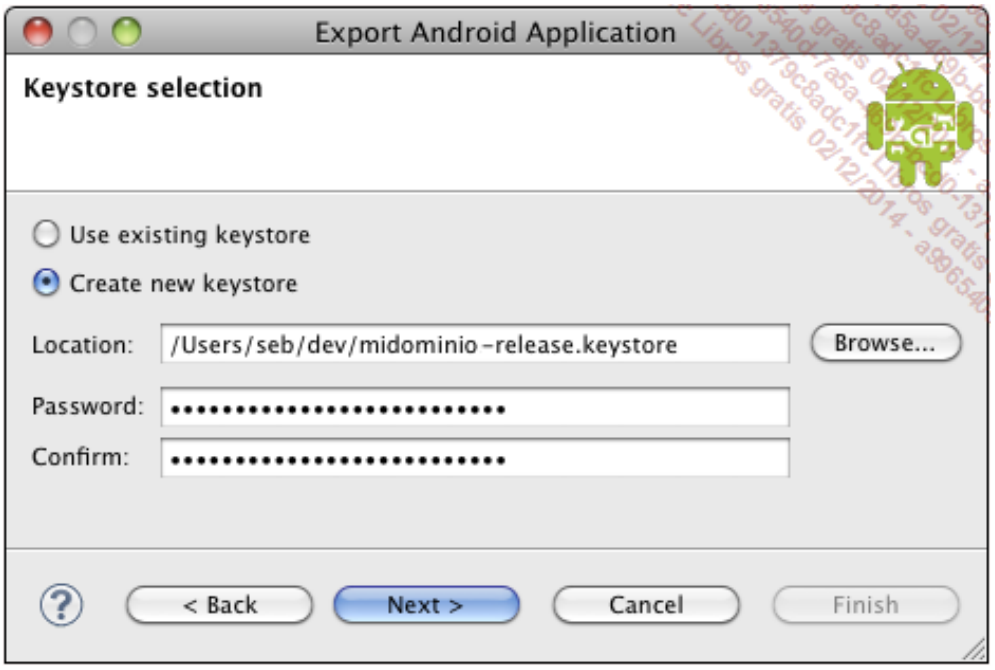
- En la vista **Package Explorer** de Eclipse, haga clic con el botón derecho sobre el proyecto y, a continuación, haga clic en **Android Tools - Export Signed Application Package...**.

Aparece la ventana **Export Android Application**.



- Verifique que el proyecto está presente en el campo **Project**. Si no fuera el caso, haga clic en el botón **Browse...** para seleccionar el proyecto.
- Haga clic en el botón **Next >** y seleccione **Create new keystore** para solicitar la creación de un nuevo almacén de claves. Haga clic sobre el botón **Browse...** para especificar la ubicación y el nombre del archivo de almacén que se creará. Introduzca una contraseña con seis caracteres como mínimo en el campo **Password** y confírmela introduciéndola de nuevo en el campo**Confirm**.

Esta contraseña debe memorizarse bien pues se le solicitará con cada compilación en modo release. Se recomienda encarecidamente escoger una contraseña segura para proteger el almacén.



- Haga clic en el botón **Next**.

La etapa siguiente consiste en crear la clave privada.

- Informe la clave privada en el campo **Alias**. Indique una contraseña de seis caracteres como mínimo en el campo **Password** y confírmela introduciéndola de nuevo en el campo **Confirm**.

Esta contraseña debe memorizarse bien pues se le solicitará con cada compilación en modo release. Por motivos de seguridad evidentes, se recomienda utilizar una contraseña distinta a la del almacén de claves, y que a su vez sea segura.

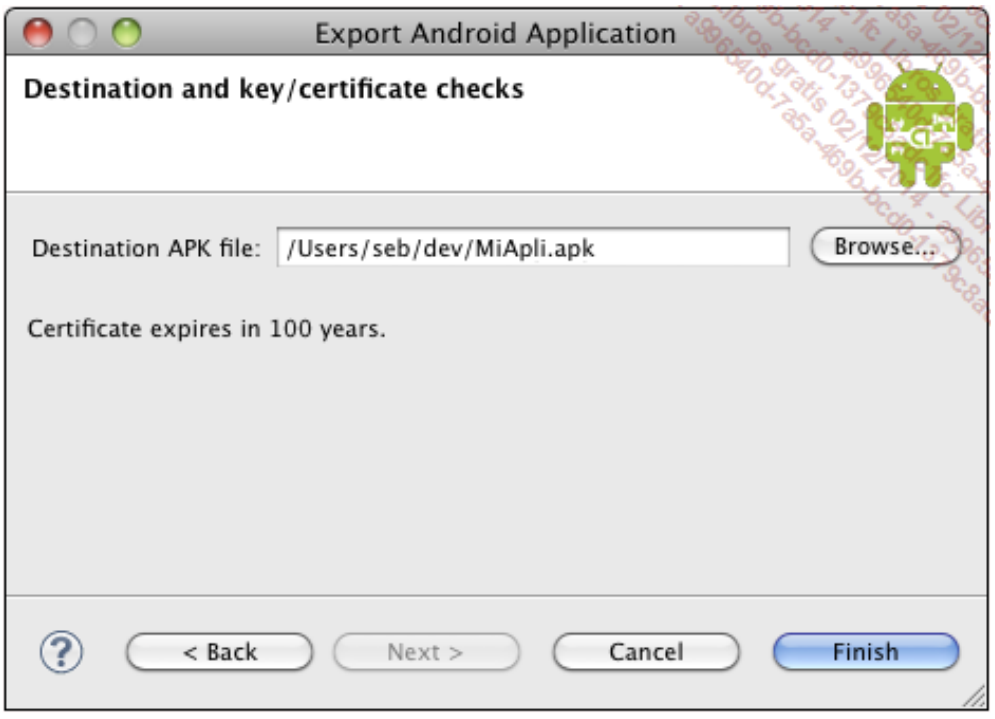
- Indique a continuación el número de años durante los cuales será válida esta clave, entre uno y mil años.

La publicación en Play Store requiere obligatoriamente una fecha de fin de validez superior al 22 de octubre de 2033. Puesto que no cuesta nada, no dude en especificar una fecha lo suficientemente lejana.

- Finalice la etapa especificando la identidad del creador de la clave.

Esta información tiene un carácter meramente informativo limitado a la clave. No se utiliza posteriormente, por ejemplo en Play Store.

- Haga clic sobre el botón **Next**.
- Introduzca la ubicación y el nombre del archivo `.apk` que quiere crear y firmar con esta clave privada.



- Haga clic en el botón **Finish**.

La clave se genera y almacena en el nuevo almacén. El proyecto se compila en modo release. El archivo `.apk` se crea, a continuación, en la ubicación indicada anteriormente, optimizado y firmado con la clave privada.

c. Instalar la aplicación

Antes de distribuir el archivo `.apk` del modo release, se recomienda encarecidamente instalarlo y ejecutarlo sobre el emulador o sobre un dispositivo Android para verificar que el archivo no se ha corrompido y que la aplicación funciona correctamente.

Para ello, es necesario suprimir previamente la aplicación compilada en modo debug del sistema Android puesto que las claves utilizadas por los archivos `.apk`, debug y release no son idénticas. El sistema Android lo detectará y denegará la instalación del nuevo archivo apk.

- Desinstale la aplicación compilada en modo debug del sistema Android. Bien utilizando el menú del sistema Android, o bien utilizando la herramienta `adb` por línea de comandos especificándole el paquete que desea borrar.

Sintaxis

```
adb uninstall [opciones] paquete
```

La opción `-k` indica que no se borren los datos de la aplicación y la caché de la aplicación.

Ejemplo

```
$ adb uninstall es.midominio.android.miaplicacion
Success
```

El sistema Android no contiene ninguna traza de la aplicación, y podemos instalar la aplicación final.

Para ello, utilizaremos de nuevo la herramienta `adb` por línea de comandos especificándole el archivo `apk` a instalar.

Sintaxis

```
adb install [opciones] archivo.apk
```

La opción `-r` es útil para realizar una actualización. Permite conservar los datos ya instalados por la versión anterior.

La opción `-s` permite instalar la aplicación sobre el almacenamiento externo del dispositivo en lugar de sobre su almacenamiento interno.

Ejemplo

```
$ adb install /Users/seb/dev/MiAplicacion.apk
475 KB/s (13404 bytes in 0.027s)
    pkg: /data/local/tmp/MiAplicacion.apk
Success
```