

Universidad Carlos III  
Grado en Ingeniería Informática

# Introducción a APKs en Android

Seguridad en dispositivos móviles



Jaime García González - 100346062  
Adán Cano Moreno - 100346105

# Índice general

<b>1. Desarrollo de una app</b>	<b>3</b>
1.1. Descripción de la app . . . . .	3
1.1.1. Inicio . . . . .	4
1.1.2. Listado de credenciales . . . . .	5
1.1.3. Añadir registro . . . . .	6
1.1.4. Más detalle . . . . .	7
1.1.5. Importar registro . . . . .	8
1.2. Mejoras adicionales . . . . .	9
1.3. Firma de la app . . . . .	9
<b>2. Estudio kontaktos.apk</b>	<b>11</b>
2.1. Verificación de firma . . . . .	11
2.2. Firmas y certificados . . . . .	13
<b>3. Uso del ADB</b>	<b>16</b>
3.1. Interactuando con el Activity Manager . . . . .	16
3.2. Extracción de una aplicación . . . . .	19

# Índice de figuras

1.1. Pantalla de bienvenida . . . . .	4
1.2. Pantalla principal . . . . .	5
1.3. Pantalla añadir registro . . . . .	6
1.4. Pantalla más detalle . . . . .	7
1.5. Pantalla importar registro . . . . .	8
1.6. Generando par de claves . . . . .	9
1.7. Firmado del apk . . . . .	10
1.8. Generación del apk . . . . .	10
2.1. Ejecución de apktool para kontaktos.apk . . . . .	11
2.2. Decompilación de kontaktos.apk . . . . .	11
2.3. Aplicación decompilada . . . . .	11
2.4. AndroidManifest.xml de la aplicación . . . . .	12
2.5. Instalación de una apk vía adb . . . . .	13
2.6. Certificado de la apk . . . . .	13
2.7. Certificado de la apk con OpenSSL . . . . .	14
3.1. Listado de paquetes . . . . .	16
3.2. <i>Activities</i> de <i>Kontaktos</i> . . . . .	17
3.3. <i>Activities</i> de <i>CredHub</i> . . . . .	18
3.4. Lanzando una <i>activity</i> de <i>Credhub</i> . . . . .	18
3.5. Arranque y parada de servicio en <i>Kontaktos</i> . . . . .	19
3.6. Permisos de <i>CredHub</i> y <i>Kontaktos</i> . . . . .	19
3.7. Recursos y meta-datos de <i>CredHub</i> y <i>Kontaktos</i> . . . . .	20
3.8. BBDD de <i>CredHub</i> y <i>Kontaktos</i> . . . . .	20
3.9. Estudio de BBDD con <i>sqlite3</i> . . . . .	21
3.10. Estudio de BBDD con <i>DB Browser for SQLite</i> . . . . .	21
3.11. Credenciales almacenadas con <i>DB Browser for SQLite</i> . . . . .	22

# Capítulo 1

## Desarrollo de una app

### 1.1. Descripción de la app

*CredHub* es una aplicación Android que se encarga de gestionar las credenciales de los usuarios. De esta forma, los usuarios pueden almacenar sus datos de registro (Username y password) asociados a un determinado servicio que se identificará por un id único.

Los datos se almacenarán de forma persistente en la memoria local del dispositivo, para ello, se hace uso de una base de datos *SQLite*. Además, es posible utilizar un servicio web que permite la gestión de credenciales de manera remota.

La base de datos de la aplicación está formada por una única tabla, *credenciales*, que se compone de tres atributos; id, username y password, siendo todos los campos de tipo String. Siendo id el identificador único, aunque no se defina como PRIMARY KEY, todas las operaciones lo tratan como si lo fuese.

La aplicación se compone de cinco *activities* con sus correspondientes clases que las dotan de funcionalidad. También, se han creado una serie de clases adicionales que complementan la funcionalidad de estas, son las siguientes:

**Model/Credenciales:** Esta clase contiene la definición de la entidad que se utilizará en la BBDD.

**Constantes:** Esta clase contiene las constantes que se utilizan en el proyecto.

**DatabaseHelper:** Esta clase hereda de la clase SQLiteOpenHelper y se encarga de la gestión de la BBDD.

**EndPoint:** Esta clase se encarga de conectar la aplicación al servicio web y de utilizar la funcionalidad que ofrece el mismo (Listar, importar y exportar).

Como se ha dicho anteriormente, la aplicación consta de cinco *activities*, son las siguientes:

### 1.1.1. Inicio

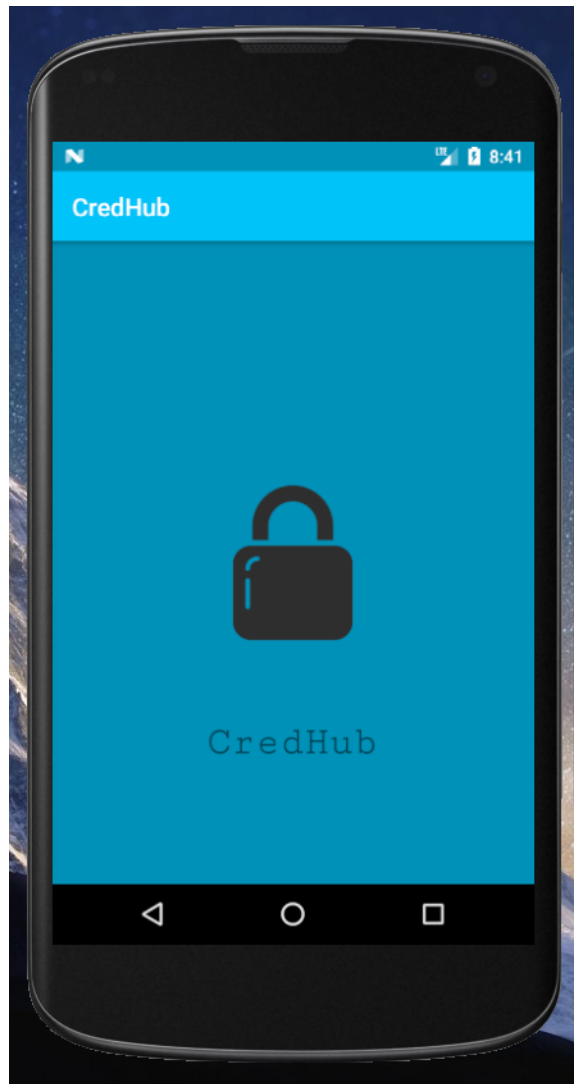


Figura 1.1: Pantalla de bienvenida

Esta pantalla se inicia al ejecutar la aplicación. Muestra un *splash* de bienvenida, con el fondo y el logo de la aplicación. Se utiliza un manejador para que se muestre en el dispositivo durante un tiempo determinado, en este caso 3 segundos. Una vez finalice el tiempo, redireccionará automáticamente a la pantalla principal de la aplicación, *listado de credenciales*.

## 1.1.2. Listado de credenciales

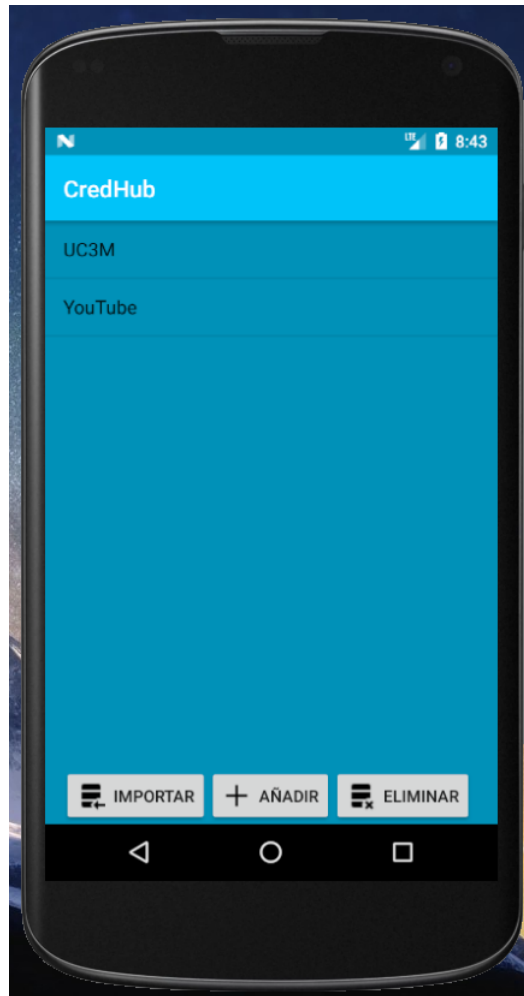


Figura 1.2: Pantalla principal

Esta *activity* es la pantalla principal de la aplicación. Muestra el un listado con el identificador de todas las credenciales almacenadas en la memoria local del dispositivo mediante un *ListView*.

Al pulsar sobre cada elemento de la lista, se nos redireccionará a la pantalla *Más detalles* que mostrará toda la información disponible de la credencial asociada al identificador pulsado.

Como se puede observar en la figura 1.2, esta pantalla dispone de tres botones diferentes, son los siguientes:

**Importar:** Este botón redirecciona a la pantalla *Importar registro*, que conecta con el servicio web.

**Añadir:** Este botón redirecciona a la pantalla *Añadir registro*, que permitirá almacenar localmente una nueva credencial.

**Eliminar:** Este botón permite eliminar todo el contenido de la base de datos almacenada en la memoria local del dispositivo.

## 1.1.3. Añadir registro

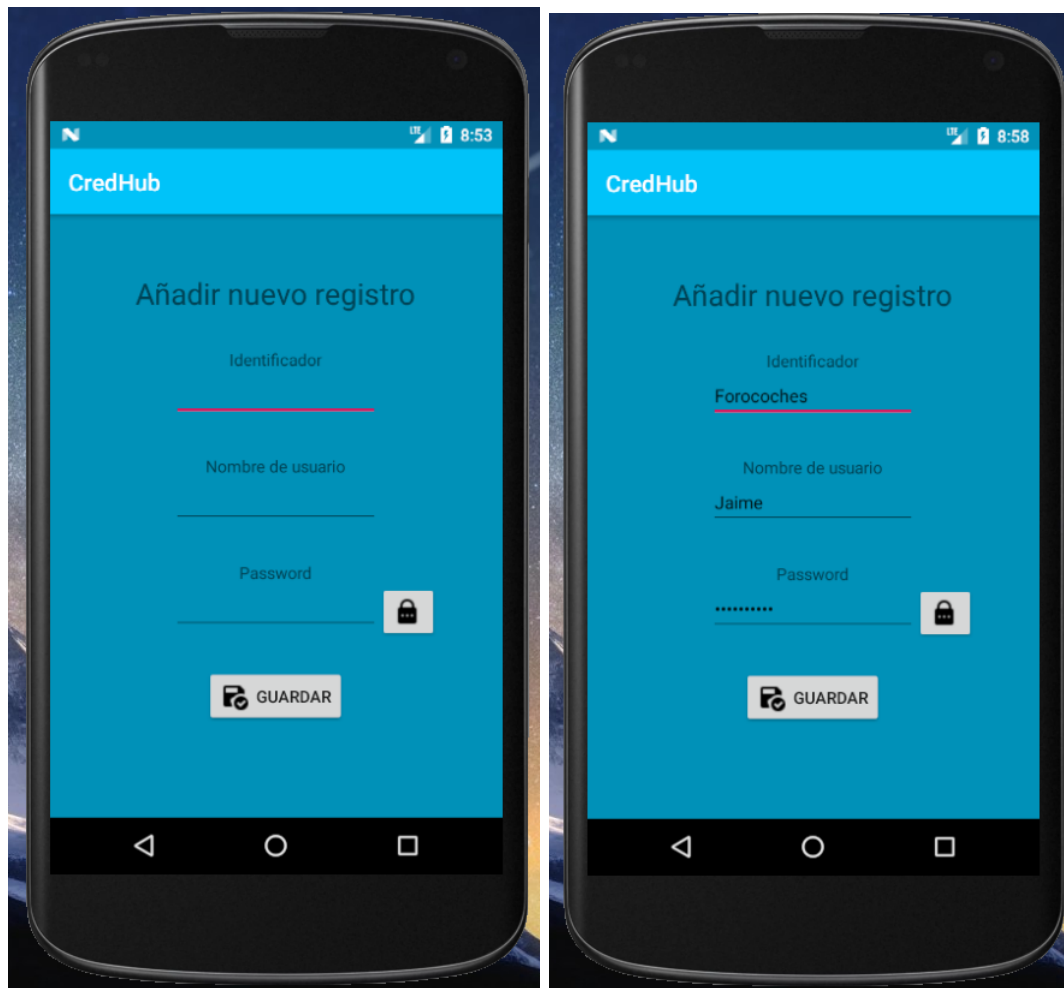


Figura 1.3: Pantalla añadir registro

Esta *activity* permite al usuario almacenar una nueva credencial en la base de datos local del dispositivo. Esta pantalla se compone de tres inputs que el usuario debe completar para poder almacenar la credencial en la base de datos, siendo obligatorio rellenar los inputs, sino no se podrá almacenar.

Además, consta de dos botones que ofrecen la siguiente funcionalidad, son los siguientes:

**Password:** Genera una contraseña aleatoria para la credencial.

**Guardar:** Almacena la credencial en la base de datos y redirecciona al usuario a la pantalla principal de la aplicación.

## 1.1.4. Más detalle

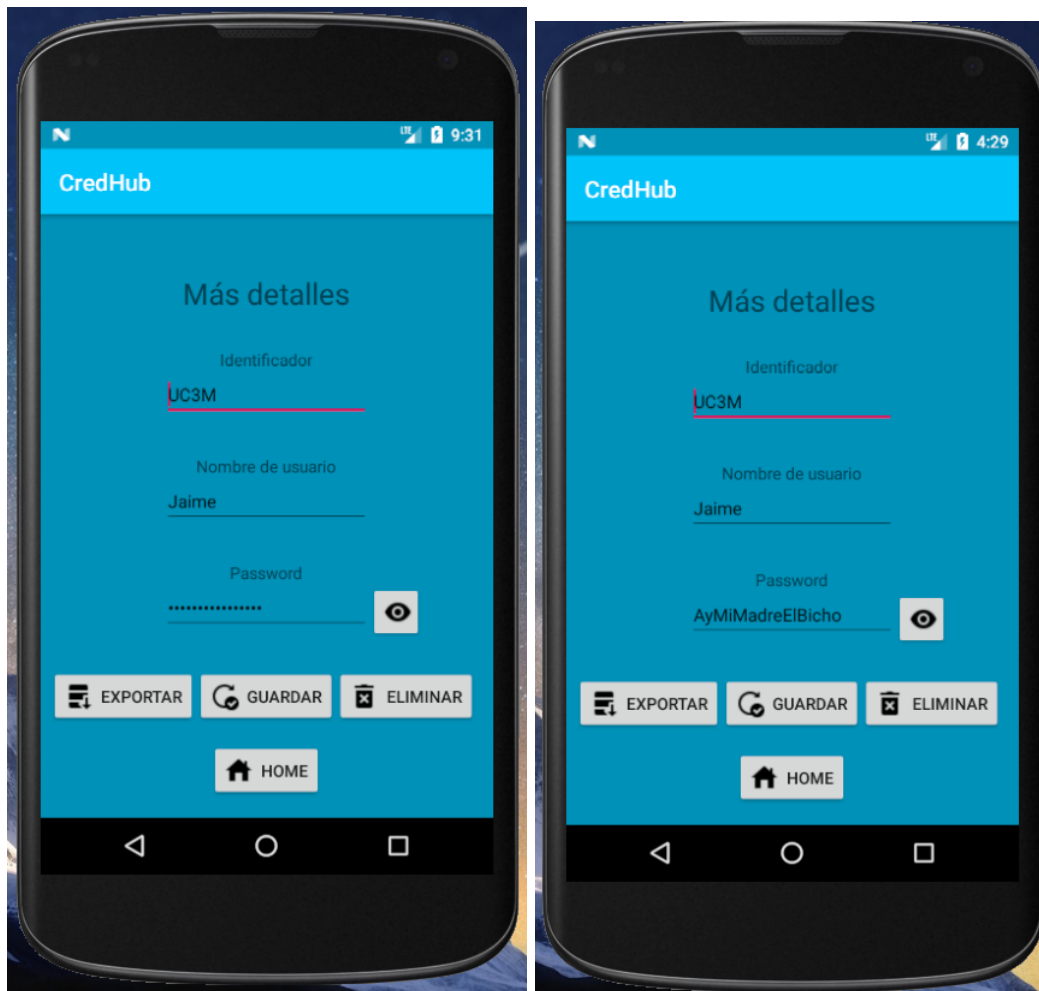


Figura 1.4: Pantalla más detalle

Esta *activity* muestra toda la información disponible de la credencial (identificador, username y password). Inicialmente, la contraseña viene oculta, se debe pulsar el botón situado a su derecha para mostrarla mediante un *toast* en pantalla durante un breve periodo de tiempo.

Además, consta de diferentes botones que ofrecen la siguiente funcionalidad:

**Mostrar:** Muestra la contraseña del usuario en pantalla durante un breve periodo de tiempo.

**Exportar:** Exporta la tripleta (id,username y password) al repositorio remoto.

**Guardar:** Permite realizar un *update* sobre la base de datos local, solo se pueden modificar el nombre de usuario y la contraseña. Una vez se ha realizado la operación, se redireccionará al usuario a la pantalla principal.

**Eliminar:** Elimina la credencial de la base de datos local, una vez se ha realizado la operación, se redireccionará al usuario a la pantalla principal.

**Home:** Redirecciona al usuario a la pantalla principal.



### 1.1.5. Importar registro

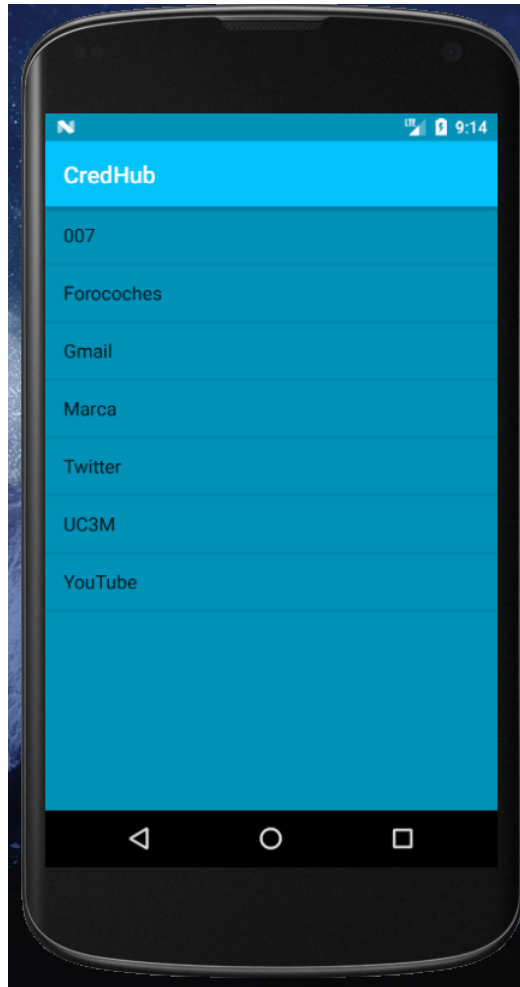


Figura 1.5: Pantalla importar registro

Esta *activity* permite al usuario importar una credencial del repositorio remoto y almacenarla en la memoria local del dispositivo.

Se utiliza un *ListView* para mostrar el listado de identificadores. Cuando el usuario pulsa sobre un identificador, se almacena la tripleta (id, username y password) en la base de datos local del dispositivo y se le redirige a la pantalla principal de la aplicación.

La credencial se almacenará en la base de datos local en caso de que no exista, mientras que, actualizará el nombre de usuario y la contraseña si encuentra ese identificador en la base de datos local.

## 1.2. Mejoras adicionales

Se han implementado todas las mejoras propuestas en el enunciado, son las siguientes:

- Posibilidad de modificar y eliminar credenciales locales. Además, se permite el borrado masivo de datos almacenados en la memoria del dispositivo.
- Se ha introducido un generador de contraseñas aleatorias, que genera caracteres aleatorios desde el 33 al 126 según el código ASCII.
- Se ha mejorado la interfaz de usuario de la aplicación. Toda la aplicación sigue los buenos patrones de diseño (*background* común, jerarquía, colocación de los elementos. . . ), además, todos los botones cuentan con iconos que permiten identificar su funcionalidad fácilmente.
- Se ha utilizado la configuración de *SharedPreferences* para almacenar el par *username/-password* a la hora de crear una nueva credencial, en vez de utilizarlo para parametrizar la URL del repo, ya que, para esta primera entrega solo se ha utilizado el protocolo http.

## 1.3. Firma de la app

En la presente sección se detallarán los pasos que se han seguido para generar el par de claves de la aplicación y el proceso de firmado de la misma.

Se comienza generando las claves correspondientes para firmar la aplicación. Se completan los datos necesarios del certificado y se indica la validez del mismo.

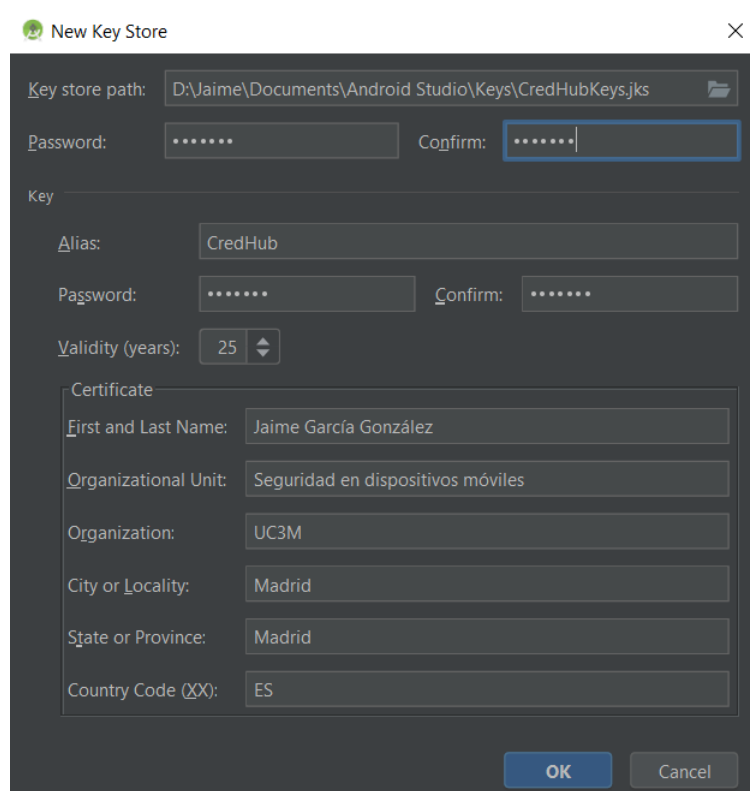


Figura 1.6: Generando par de claves

Una vez se han generado las claves correspondientes para firmar, se introduce la contraseña definida en las claves para firmar la aplicación.

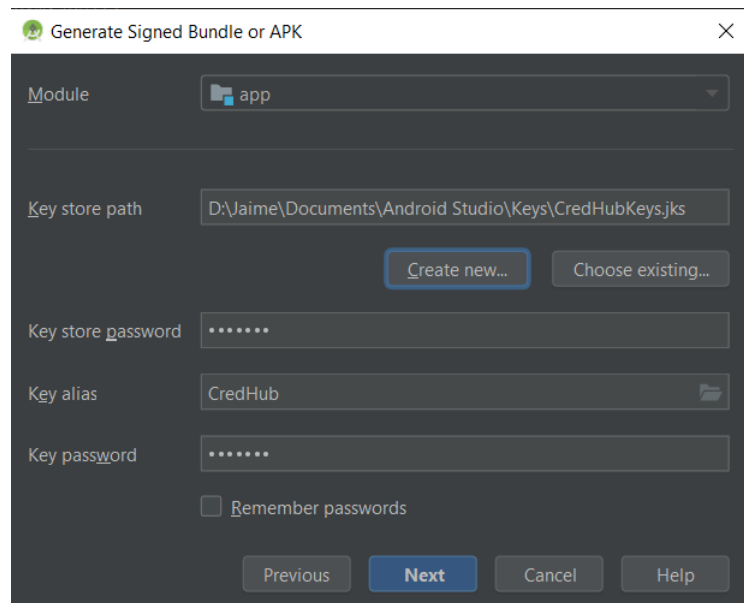


Figura 1.7: Firmado del apk

Finalmente, se indica la ruta de destino del apk que se va a generar. También se debe especificar que es una *release*, de esta forma sabremos que es la versión final. Además, se elige la V2 de firmado.

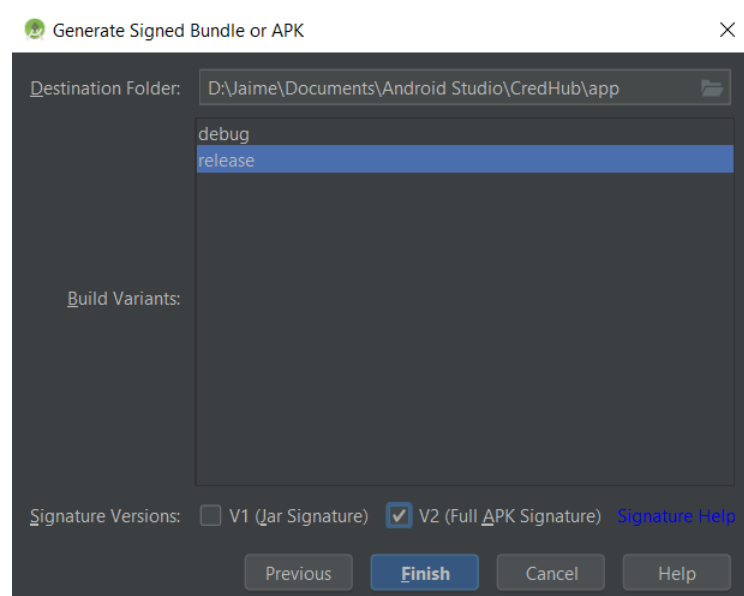


Figura 1.8: Generación del apk

## Capítulo 2

# Estudio kontaktos.apk

### 2.1. Verificación de firma

En este apartado se procederá a verificar la firma de la aplicación *kontaktos.apk*, para ello, se utilizará la herramienta *apktool*. Se descargarán las herramientas necesarias, la aplicación a analizar y se moverán al mismo directorio. Los pasos seguidos han sido:

1. Ejecución de la aplicación, se generarán los archivos necesarios en un directorio temporal. Véase la figura 2.1.

```
D:\Jaime\Programas\Apktool>java -jar apktool.jar if kontaktos.apk
S: WARNING: Could not write to (C:\Users\Jaime\AppData\Local\apktool\framework), using C:\Users\Jaime\AppData\Local\Temp\ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the default storage directory is unavailable
I: Framework installed to: C:\Users\Jaime\AppData\Local\Temp\127.apk
```

Figura 2.1: Ejecución de apktool para kontaktos.apk

2. Se procede a decompilar la aplicación para su estudio. Véase la figura 2.2.

```
D:\Jaime\Programas\Apktool>java -jar apktool.jar d kontaktos.apk
I: Using Apktool 2.3.4 on kontaktos.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (C:\Users\Jaime\AppData\Local\apktool\framework), using C:\Users\Jaime\AppData\Local\Temp\ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the default storage directory is unavailable
I: Loading resource table from file: C:\Users\Jaime\AppData\Local\Temp\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Figura 2.2: Decompilación de kontaktos.apk

3. Se generará automáticamente un directorio que contiene la aplicación decompilada. Véase la figura 2.3.





 kontaktos	06/02/2019 9:44	Carpeta de archivos	
 apktool	06/02/2019 9:30	Archivo por lotes ...	1 KB
 apktool	06/02/2019 9:27	Executable Jar File	10.746 KB
 kontaktos.apk	06/02/2019 9:33	Archivo APK	8.206 KB

Figura 2.3: Aplicación decompilada

4. Abrimos el fichero *AndroidManifest.xml* para comprobar los permisos. Véase la figura 2.4.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS"/>
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.BATTERY_STATS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE"/>
<uses-permission android:name="android.permission.WRITE_APN_SETTINGS"/>
<uses-permission android:name="android.permission.READ_APN_SETTINGS"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<permission android:name="com.android.launcher.action.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<permission android:name="com.contapps.android.permission.C2D_MESSAGE" android:protectionLevel="signature
<uses-permission android:name="com.contapps.android.permission.C2D_MESSAGE"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
<permission android:name="com.contapps.android.permission.MAPS_RECEIVE" android:protectionLevel="signature
<uses-permission android:name="com.contapps.android.permission.MAPS_RECEIVE"/>
<uses-feature android:name="android.hardware.telephony" android:required="false"/>
<supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" and
<application android:allowBackup="true" android:icon="@drawable/icon" android:label="@string/app_name" an
<uses-library android:name="com.google.android.maps"/>
```

Figura 2.4: AndroidManifest.xml de la aplicación

Como se puede observar, esta aplicación requiere numerosos permisos, algunos de ellos son:

**READ\_CONTACTS.** Permite a la aplicación la lectura de contactos del dispositivo.

**VIBRATE.** Permite a la aplicación vibrar.

**INTERNET.** Permite a la aplicación abrir sockets de red.

**CALL\_PHONE.** Permite a la aplicación efectuar una llamada telefónica sin hacer uso del teclado.

**BATTERY\_STATS.** Permite a la aplicación conocer las estadísticas de la batería del dispositivo.

Los *permisos runtime* solo están disponibles a partir de Android 6.0 (API 23), son notificados al usuario cuando se ejecuta la aplicación, no en el momento de instalación. Estos permisos se pueden aceptar o denegar, para que no se pregunte al usuario cada vez que inicia la aplicación. Sin embargo, los *permisos estáticos* son aquellos que están disponibles en versiones anteriores a 5.1 (API 22) y son notificados al usuario en el momento de instalación de la aplicación.

## 2.2. Firmas y certificados

En esta sección, se procederá a estudiar las firmas y los certificados digitales de la aplicación anterior.

Lo primero que se debe hacer es instalar la aplicación en nuestro dispositivo virtual (AVD), para ello, se inicializa nuestro dispositivo virtual y una terminal para instalar el .apk, véase la figura .

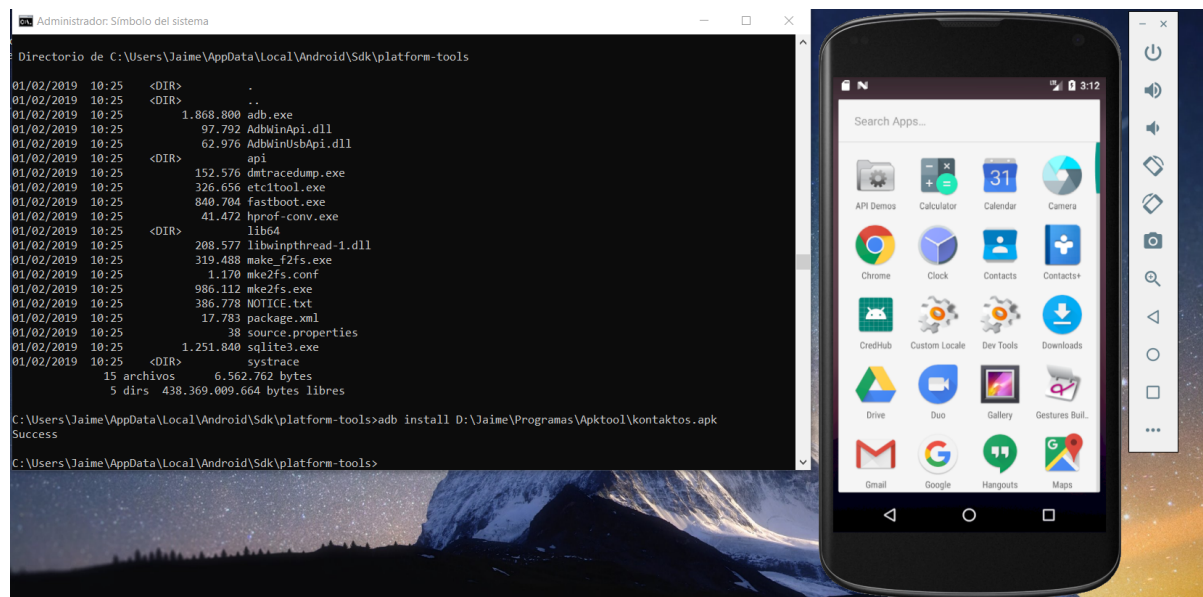


Figura 2.5: Instalación de una apk vía adb

Seguidamente, se procede a descomprimir el .apk para visualizar los certificados, que se encuentran en el directorio "META-INF". Se descomprime la aplicación haciendo uso de la herramienta *Winrar*. A continuación, hacemos uso de la herramienta *keytool* para visualizar el contenido del certificado digital declarado por el dueño de la llave pública, véase la figura 2.6.

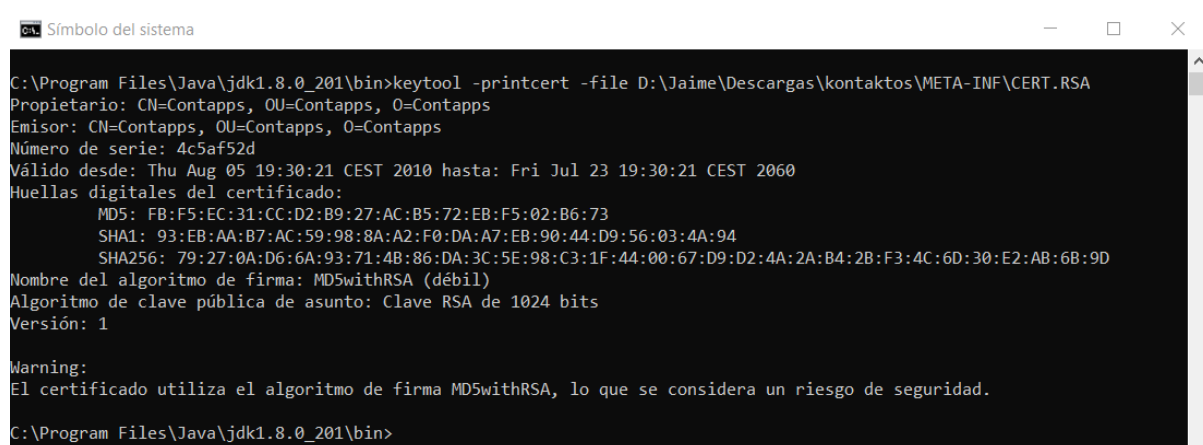


Figura 2.6: Certificado de la apk

Como se puede comprobar, el certificado consta de varias partes, son las siguientes:

**Propietario.** Propietario de la aplicación.

**Emisor.** Emisor de la aplicación.

**# Serie.** Número de serie de la aplicación.

**Validez.** Indica el periodo de validez del certificado.

**Huellas digitales.** Funciones resumen únicas que identifican la aplicación.

**Algoritmo de firma.** Algoritmo de firmado utilizado en la aplicación.

**Algoritmo de clave pública.** Algoritmo utilizado para generar la clave pública.

**Versión.** Versión de la aplicación.

Los valores obtenidos de la aplicación *kontaktos* después de utilizar *OpenSSL*, son los siguientes:

Serial number	0x4c5af52d
Validez	Aug 2010 - Jul 2060
Codificación clave pública	RSA
Tamaño clave pública	1024 bits
Modulus	Véase la figura 2.7
Algoritmo firma	MD5withRSA

```
D:\Jaime\Programas\OpenSSL-Win64\bin>openssl.exe
OpenSSL> pkcs7 -inform DER -in D:\Jaime\Descargas\kontaktos\META-INF\CERT.RSA -noout -print_certs -text
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 1281029421 (0x4c5af52d)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: O=Contapps, OU=Contapps, CN=Contapps
    Validity
      Not Before: Aug  5 17:30:21 2010 GMT
      Not After : Jul 23 17:30:21 2060 GMT
    Subject: O=Contapps, OU=Contapps, CN=Contapps
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (1024 bit)
      Modulus:
        00:a3:36:bd:7e:93:c5:a8:e9:2a:44:4f:5d:89:d6:
        19:54:af:88:82:64:26:39:42:71:4c:10:c0:cc:2e:
        81:93:00:5d:32:95:28:80:9d:9d:e8:7d:5a:32:38:
        02:42:59:a8:b5:d1:dd:6e:e2:d2:8d:3a:02:42:af:
        ae:4d:21:3a:7a:f6:d8:4d:e7:07:d2:5c:02:1f:8f:
        1a:35:8f:82:72:3b:d1:de:52:ed:8e:10:9b:46:88:
        98:2e:4d:19:d2:0f:68:b3:9d:10:44:c8:c6:25:c7:
        6e:cd:bd:43:1d:84:ad:02:c2:f7:e6:bd:16:cf:5e:
        88:45:79:26:f4:09:d5:f5:37
      Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
      58:d2:9e:74:d2:da:ef:9c:09:d1:aa:c3:c2:62:58:af:00:6c:
      cf:f3:9d:78:dc:8e:dc:f1:a8:71:e7:e8:d6:ba:c3:44:5c:2f:
      7d:a0:08:14:f2:dd:3f:cd:91:1f:6b:28:28:23:1a:06:66:b8:
      9d:67:76:50:93:0e:5f:8c:1a:b5:9a:ef:13:a3:71:aa:e6:97:
      26:ea:e5:3e:21:2d:fb:01:c5:d5:61:ab:41:9e:15:5c:c1:cb:
      a8:7c:29:f4:4e:55:2a:b4:98:41:a6:91:30:3f:60:c4:23:a9:
      60:78:61:45:0b:96:86:c3:7a:45:2d:2a:07:ad:2b:de:e7:ac:
      33:b9
```

Figura 2.7: Certificado de la apk con OpenSSL

La siguiente tarea consiste en obtener los hash criptográficos del logotipo de la aplicación y de al menos tres imágenes distintas que tengan densidad de píxeles diferentes, por ello, se debe analizar el fichero CERT.SF que se puede visualizar con cualquier editor de texto, en este caso, *Notepad ++*.

El fichero contiene más de 5000 líneas de código, así que, para localizar el nombre del icono de la aplicación se accede al directorio `res` se busca el icono en los diferentes subdirectorios. El logo de la aplicación recibe el nombre de *icon*, así que, se busca en el fichero este archivo y se encuentra su información:

```
Name: res/drawable-hdpi/icon.png
SHA1-Digest: X5/mtN+YdCFiZSlqEC9QRQ4eXFo=
```

Para localizar tres imágenes diferentes se realiza de manera análoga, siendo cada directorio la densidad de píxeles de cada una, por ejemplo, `mdpi` (medium-dpi), `hdpi` (large-dpi) ... Por tanto, tres archivos válidos serían:

```
Name: res/drawable-hdpi/welcome_pic.png
SHA1-Digest: kEtv1qXDVEaipT5tpE6yJFbTVag=
```

```
Name: res/drawable-ldpi/wizard_thank_you.png
SHA1-Digest: oaQL3lC7JfkqM+rjezbyhW9iO3A=
```

```
Name: res/drawable-mdpi/wizard_free_sms_pic.png
SHA1-Digest: aiczu6u6zKX4kkjBzKZmODovr44=
```

Para la codificación del hash, se ha utilizado el algoritmo *SHA-1*, que separa la información en bloques de 512 bits y luego añade 80 vueltas utilizando una serie de vectores y mezclando la información con los siguientes hasta obtener un resumen de 160 bits.



## Capítulo 3

# Uso del ADB

### 3.1. Interactuando con el Activity Manager

Para comenzar, se deben iniciar el emulador y se hará uso del ADB para comunicarse con el dispositivo. Lo primero que se debe de hacer es listar los dispositivos, para ello se utiliza el siguiente comando:

```
adb.exe devices
```

Se obtendrá que el dispositivo conectado es `.emulator-5554`, lo siguiente que se debe hacer es conectarse al dispositivo, por tanto, se hará uso del siguiente comando:

```
adb.exe -s emulator-5554 shell
```

Se nos abrirá la shell del dispositivo, de esta forma ya se podrá interactuar con él. Para listar los paquetes instalados se utiliza el siguiente comando:

```
pm list packages
```

Los paquetes de la aplicación *Kontaktos* y de *CredHub* han sido resaltados, véase la figura 3.1.

```
generic_x86:/ # pm list packages -3
package:com.android.smoketest
package:com.example.android.livecubes
package:com.example.android.apis
package:com.example.credhub
package:com.contapps.android
package:com.android.gesture.builder
package:com.android.smoketest.tests
package:com.example.android.softkeyboard
package:com.android.widgetpreview
generic_x86:/ #
```

Figura 3.1: Listado de paquetes

Las *activities* que forman la aplicación *Kontaktos* son:

```

$ adb -s emulator-5554 shell
generic_x86:/ # dumpsys package | grep -Eo "[^[:space:]]+[0-9a-f]+[[:space:]]+com.contapps.android/[[:space:]]+" | grep -oE "[^[:space:]]+ $"
com.contapps.android/.CallLog
com.contapps.android/.Messages
com.contapps.android/.BoardPicker
com.contapps.android/.Messages
com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.BoardPicker
com.contapps.android/.ContappsBoard
com.contapps.android/.PhonePicker
com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.Messages
com.contapps.android/.PhonePicker
com.contapps.android/.CallLog
com.contapps.android/.BoardPicker
com.contapps.android/.ContappsBoard
com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.Messages
com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.CursorContact
com.contapps.android/.utils.WebViewActivity
com.contapps.android/.tapps.facebook.FacebookAuthenticator
com.contapps.android/.viral.EmailInviter
com.contapps.android/.tapps.gplus.GPlusAuthenticator
com.contapps.android/.preferences.TappSettings
com.contapps.android/.Preferences
com.contapps.android/.tapps.gplus.ParseDeepLinkActivity
com.contapps.android/.sms.NewMessageActivityLauncher
com.contapps.android/.SmartDialer
com.contapps.android/.sms.NewMessageActivityLauncher
com.contapps.android/.SmartDialer
com.contapps.android/.BoardPicker
com.contapps.android/.ContappsBoard
com.contapps.android/.CallLog
com.contapps.android/.SmartDialer
com.contapps.android/.ContappsBoard
com.contapps.android/.viral.EmailInviter
com.contapps.android/.messaging.MessagingRegistrationPageActivity
com.contapps.android/.tapps.sms.SmsPopupActivity$TempAddContactActivity
com.contapps.android/.WizardActivity
com.contapps.android/.help.WhatsNewActivity
com.contapps.android/.tapps.sms.SmsPopupActivity
com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.sync.LoginActivity
com.contapps.android/.Preferences
com.contapps.android/.SmartDialer
com.contapps.android/.CallLog
com.contapps.android/.shortcuts.ComposeNewShortcut
com.contapps.android/.shortcuts.DialerShortcut
com.contapps.android/.shortcuts.MessagingShortcut
com.contapps.android/.shortcuts.CallLogShortcut
com.contapps.android/.shortcuts.ShortcutActivity
com.contapps.android/.ContappsBoard
com.contapps.android/.Messages
com.contapps.android/.PhonePicker
com.contapps.android/.BoardPicker
com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.CallLog
com.contapps.android/.ContappsBoard
com.contapps.android/.sms.MmsReceiver
com.contapps.android/.sms.MmsReceiver
com.contapps.android/.utils.SMSUtils$NotifyBroadcastReceiver
com.contapps.android/com.google.android.c2dm.C2DMBroadcastReceiver
com.contapps.android/.utils.ContactActionReceiver
com.contapps.android/com.google.analytics.tracking.android.CampaignTrackingReceiver
com.contapps.android/.VersionUpgrader$updateReceiver
com.contapps.android/.wavelauncher.PluginUpdateRequestReceiver
com.contapps.android/com.google.android.c2dm.C2DMBroadcastReceiver
com.contapps.android/.wavelauncher.PluginUpdateRequestReceiver
com.contapps.android/.dailyTask.OnBootReceiver
com.contapps.android/.tapps.sms.SmsReceiver
com.contapps.android/.tapps.sms.MessageStatusReceiver
com.contapps.android/.tapps.sms.PrivilegedSmsReceiver
com.contapps.android/.tapps.sms.SmsReceiver
com.contapps.android/.desktopWidget.ContappsDesktopWidget1Liner
com.contapps.android/.desktopWidget.ContappsDesktopWidget4Liner
com.contapps.android/.desktopWidget.ContappsDesktopWidget2Liner
com.contapps.android/com.google.android.c2dm.C2DMBroadcastReceiver
com.contapps.android/.sms.MmsReceiver
com.contapps.android/.sync.ContactsSyncAdapterService
com.contapps.android/.sync.AccountAuthenticatorService
generic_x86:/ #

```

Figura 3.2: *Activities* de *Kontaktos*

Las *activities* que forman la aplicación *CredHub* son:

```
C:\Users\Jaime\AppData\Local\Android\Sdk\platform-tools>adb -s emulator-5554 shell
generic_x86:/ # dumpsys package | grep -Eo "[^[:space:]]+[0-9a-f]+[[:space:]]+com.example.credhub/[^\[:space:]]+" | grep -oE "[^[:space:]]+ $"
com.example.credhub/.Inicio
com.example.credhub/.ListadoDeCredenciales
com.example.credhub/.MostrarRegistro
com.example.credhub/.AnadirRegistro
com.example.credhub/.ImportarRegistro
generic_x86:/ #
```

Figura 3.3: *Activities* de *CredHub*

Para lanzar la *activity* de añadir un nuevo registro se ha ejecutado el siguiente comando:

```
am start -n com.example.credhub/.AnadirRegistro
```

Como resultado, se abre la *activity* correspondiente de la aplicación, véase la figura 3.4.

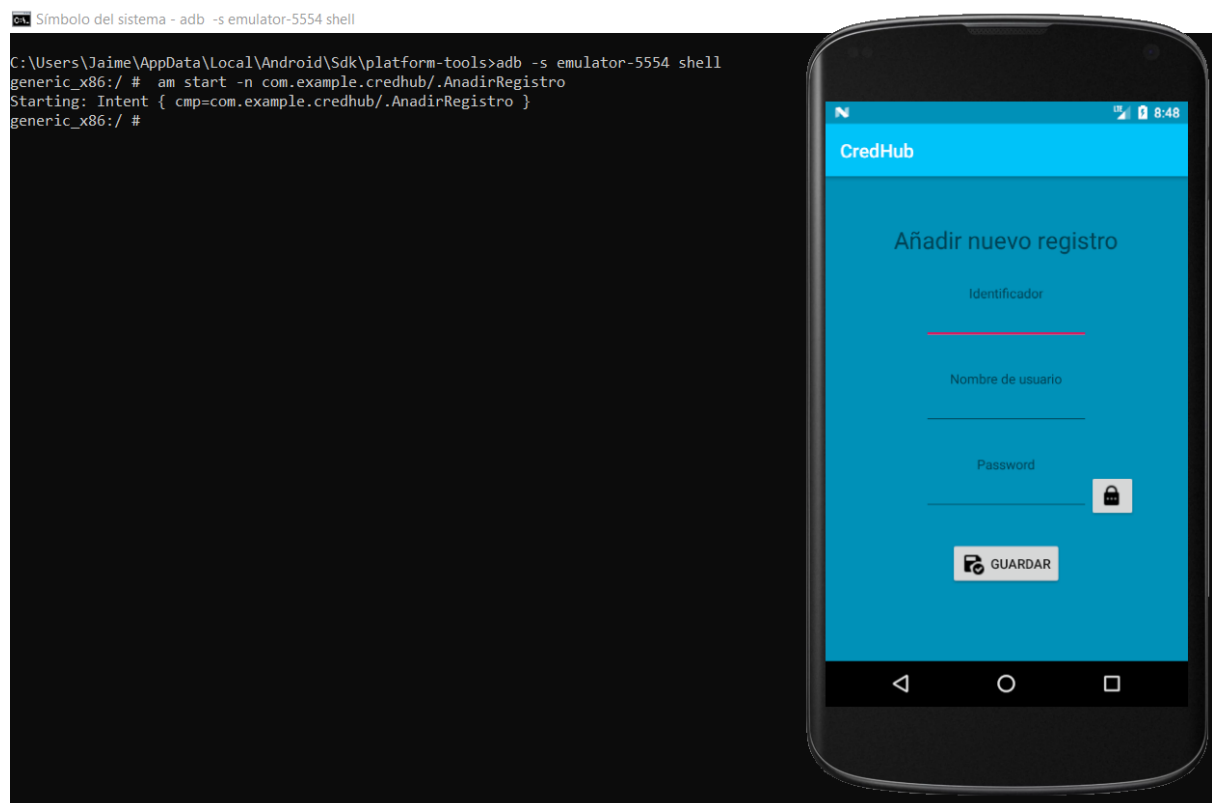


Figura 3.4: Lanzando una *activity* de *Credhub*

Para finalizar la *activity* se ejecuta el siguiente comando:

```
am force-stop com.example.credhub
```

Para lanzar un servicio de la aplicación *Kontaktos*, se ha estudiado su archivo *AndroidManifest* y se ha lanzado uno de ellos, el comando ejecutado es el siguiente:

```
am startservice -n com.contapps.android/.sync.ContactsSyncAdapterService
```

Para finalizar el servicio, se ha ejecutado el siguiente comando:

```
am force-stop com.contapps.android
```

Todo el proceso relacionado con el servicio se puede observar en la figura 3.5.

```
generic_x86:/ # am startservice -n com.contapps.android/.sync.ContactsSyncAdapterService
Starting service: Intent { cmp=com.contapps.android/.sync.ContactsSyncAdapterService }
generic_x86:/ # am force-stop com.contapps.android
generic_x86:/ #
```

Figura 3.5: Arranque y parada de servicio en *Kontaktos*

## 3.2. Extracción de una aplicación

Para visualizar los permisos y la fecha de modificación de la aplicación *CredHub* y *Kontaktos* se ha navegado hasta el directorio `/data/data` y se ha ejecutado el siguiente comando:

```
ls -al
```

Los permisos y la fecha de modificación de ambas aplicaciones se pueden visualizar en la figura 3.6.

```
drwxr-x--x  6 u0_a81 u0_a81 4096 2019-02-19 09:05 com.contapps.android
drwxr-x--x  2 u0_a73 u0_a73 4096 2019-02-02 18:17 com.example.android.apis
drwx----- 2 u0_a74 u0_a74 4096 2019-02-02 18:17 com.example.android.livecubes
drwx----- 2 u0_a78 u0_a78 4096 2019-02-02 18:17 com.example.android.softkeyboard
drwx----- 4 u0_a80 u0_a80 4096 2019-02-13 16:44 com.example.credhub
```

Figura 3.6: Permisos de *CredHub* y *Kontaktos*

Para listar los recursos y meta-datos de las aplicaciones se ha ejecutado el siguiente comando:

```
ls 45aIR */
```

Los recursos y meta-datos de ambas aplicaciones se pueden visualizar en la figura 3.7.

```
com.contapps.android/:
cache databases files shared_prefs

com.example.android.apis/:

com.example.android.livecubes/:

com.example.android.softkeyboard/:

com.example.credhub/:
cache databases shared_prefs
```

Figura 3.7: Recursos y meta-datos de *CredHub* y *Kontaktos*

Como se puede observar, ambas aplicaciones tienen su caché de datos, sus bases de datos y sus ficheros *SharedPreferences*. Además, la aplicación *Kontaktos*, tiene una serie de ficheros propios.

Para listar las bases de datos de las aplicaciones se ha ejecutado el siguiente comando:

```
ls 45aIR */databases/
```

Las bases de datos de ambas aplicaciones se pueden visualizar en la figura 3.8.

```
com.contapps.android/databases/:
ContappsDB ContappsDB-journal

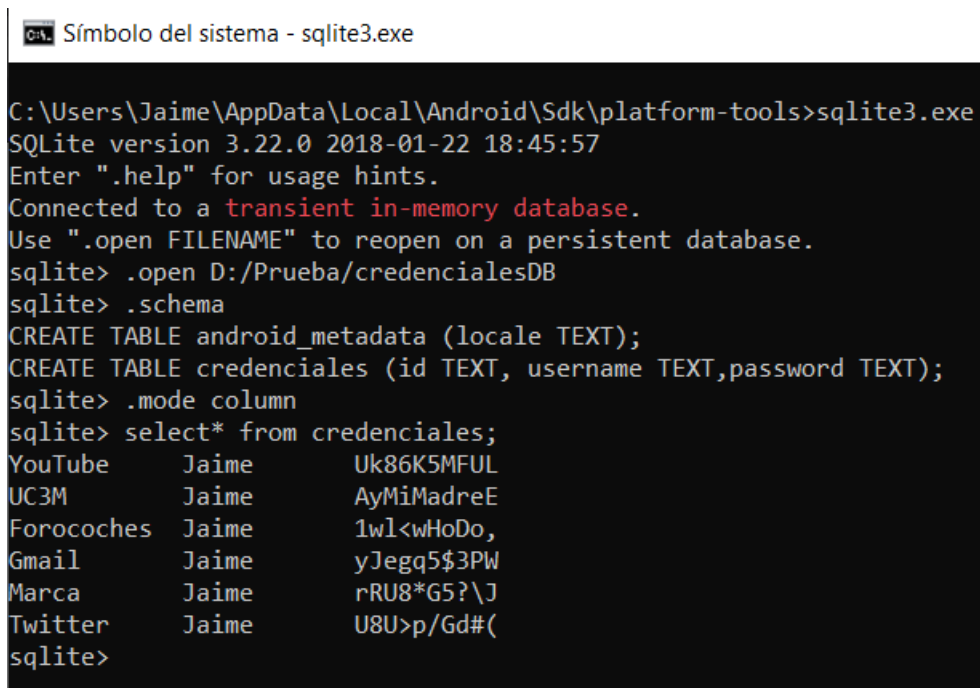
com.example.credhub/databases/:
credencialesDB credencialesDB-journal
```

Figura 3.8: BBDD de *CredHub* y *Kontaktos*

Para extraer el archivo de bases de datos se ejecuta el siguiente comando desde la terminal:

```
adb -s emulator-5554 pull
/data/data/com.example.credhub/databases/credhubDB D:/Prueba
```

Una vez se ha terminado de copiar el fichero de BBDD en el directorio indicado, se procede al estudio del mismo. Para ello, se abre una terminal y se ejecuta el programa *sqlite3*. Se procede a ejecutar los siguientes comandos:



```

C:\Users\Jaime\AppData\Local\Android\Sdk\platform-tools>sqlite3.exe
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open D:/Prueba/credencialesDB
sqlite> .schema
CREATE TABLE android_metadata (locale TEXT);
CREATE TABLE credenciales (id TEXT, username TEXT,password TEXT);
sqlite> .mode column
sqlite> select* from credenciales;
YouTube      Jaime      Uk86K5MFUL
UC3M         Jaime      AyMiMadreE
Forocoches   Jaime      1wl<wHoDo,
Gmail        Jaime      yJegq5$3PW
Marca        Jaime      rRU8*G5?\J
Twitter      Jaime      U8U>p/Gd#(
sqlite>

```

Figura 3.9: Estudio de BBDD con *sqlite3*

Como se puede comprobar, el fichero extraído del dispositivo consta de dos tabla; *android\_metadata* y *credenciales*. La tabla *credenciales* es la que ha sido generada para el almacenamiento de claves y dispone de tres campos (*id*, *username* y *password*), todos ellos de tipo TEXT (String). Finalmente, se ha ejecutado un *Select \** de dicha tabla para visualizar todos los datos almacenados.

Finalmente, se ha realizado un estudio del mismo fichero de base de datos con la herramienta *DB Browser for SQLite*, véase la figura 3.10.

Nombre	Tipo	Esquema
▼ Tablas (2)		
▼ android_metadata		
locale	TEXT	CREATE TABLE android_metadata (locale TEXT)
▼ credenciales		
id	TEXT	CREATE TABLE credenciales (id TEXT, username TEXT,password TEXT)
username	TEXT	"id" TEXT
password	TEXT	"username" TEXT
Índices (0)		
Vistas (0)		
Disparadores (0)		

Figura 3.10: Estudio de BBDD con *DB Browser for SQLite*

Las credenciales almacenadas también se han obtenido utilizando esta herramienta, véase la figura 3.11

	id	username	password
	Filtro	Filtro	Filtro
1	Forocoches	Jaime	1wl<wHoDo,
2	Gmail	Jaime	yJegg5\$3PW
3	Marca	Jaime	rRU8*G5?\J
4	Twitter	Jaime	U8U>p/Gd#(
5	UC3M	Jaime	AyMiMadreElBicho
6	YouTube	Jaime	Uk86K5MFUL

Figura 3.11: Credenciales almacenadas con *DB Browser for SQLite*