

# Mobile Devices Security

Fernando García Sanz(100346043), Francisco Manuel Colmenar Lamas (100346094)

February 26, 2019



# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Development of an Android Application</b>	<b>3</b>
2.1	Flowchart of the Application . . . . .	3
2.2	Detailed description of the functionalities . . . . .	4
2.2.1	Database Implementation . . . . .	4
2.2.2	Title Screen . . . . .	5
2.2.3	Main Screen . . . . .	5
2.2.4	New Record Screen . . . . .	5
2.2.5	Show Password / Export Screen . . . . .	5
2.2.6	Import Record Screen . . . . .	6
2.2.7	Additional Improvements . . . . .	6
2.3	Connection with the Server . . . . .	6
2.4	Listeners . . . . .	8
<b>3</b>	<b>CredHub App Signing</b>	<b>9</b>
<b>4</b>	<b>Signature verification of the application <i>kontaktos.apk</i></b>	<b>10</b>
<b>5</b>	<b>Inspection of digital signatures and digital certificates of the application <i>kontaktos.apk</i></b>	<b>12</b>
<b>6</b>	<b>Interacting with the Activity Manager via ADB</b>	<b>15</b>
<b>7</b>	<b>Extracting an app via ADB</b>	<b>17</b>

# 1 Introduction

Mobile devices have evolved a lot in a short period of time, so their possibilities and, sadly, so their vulnerabilities. Therefore, it does exist the need of developing more secure applications for current days.

In this practice, the focus is on components related to security of applications and the tools which allow the user to access either relevant or confidential information. In this practice, a password manager will be implemented, which will allow the user to manage his private digital credentials.

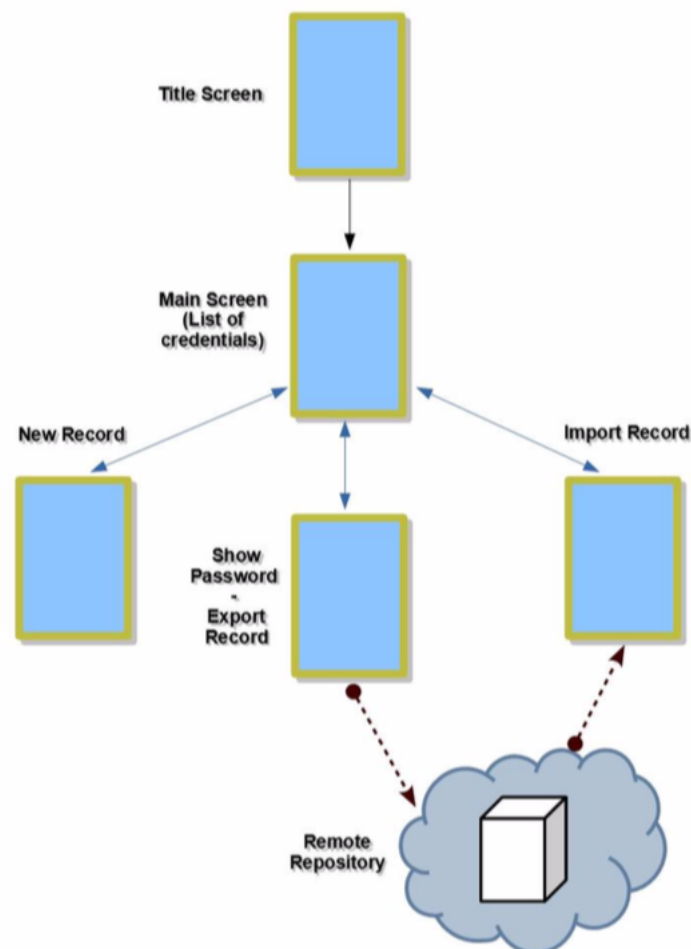
## 2 Development of an Android Application

The first part of this assignment consists of the development of an Android Application. This App is called CredHub. Its main purpose is the management of passwords related to different services, such as Facebook, Twitter,...

The main functionalities of CredHub as well as the different screens of it are going to be described in the following sections.

### 2.1 Flowchart of the Application

The application flow between the different activities can be found in the following picture, provided in the assignment description:



## 2.2 Detailed description of the functionalities

In this section the main functionalities of CredHub are going to be described. Furthermore, the implementation of the connection to the Server from the Application as well as the management of the local database are going to be described.

### 2.2.1 Database Implementation

The database has been implemented by means of the library *android.database.sqlite*, which will provide *SQLite* commands to manage the database. Its implementation has been made using three classes:

- The class *User*, in which the record object is implemented, allowing operating with it as a whole in future.
- The class *UserModel*, which implements the database model, with the table and column names, which will be used later on.
- The class *UserDBHelper*, which implements the methods for creating and updating the database according to the model defined in the previous class.

Once the database has been implemented, its content is stored in the path:

*/data/data/<package>/databases/User.db*

It is only accessible to root users, avoiding content alterations, and it can be accessed by the application to store new records and manage existing ones.

Then, in the class *NewRecord*, the values of the text fields are retrieved when the user press *submit*. Then, a connection to the database is established, getting the database as writable, which allows to manage its content. A *ContentValues* object is created and the previously retrieved values are added to it by means of the *put* command. Once all values have been added, the insert method is performed over the database, creating a new record which will have as identifier the *Description* field value. It is useful to check if the introduced description is already saved into the database, to avoid replications of content, errors and allow future improvements. This can be done by means of a *SQL* query looking for elements with the introduced identifier value in the *Description* field.

Once the pertinent operations have been performed, the database is closed.

Importing records works in a similar way. The system receives from the server the full data

of a record and inserts them as a new record in the database following the previously described operation.

### **2.2.2 Title Screen**

This is the Screen showed when CredHub is opened. It just shows an introductory image of the App for three seconds. After it, the user is redirected to the Main Screen.

### **2.2.3 Main Screen**

In this screen, it will appear the description of all records stored in the local database of our device, allowing the user to press over any of them and accessing to a more detailed view of the record, where the user will be capable of fully seeing the its details and even exporting the record to the server database.

In this screen the user will also be able to find two different buttons in the bottom corners of the device. The left one will lead to the Import Record Screen, while the right one will open the screen in which a new record can be added to the database.

### **2.2.4 New Record Screen**

By pressing the button on the bottom right corner, this screen will be accessed. On it, the user will be capable of adding a new record to the local database of the device by fulfilling the three text fields which appear: description, username and password. Once they three are complete, the user can press the submission button and the new record will be included into the device database.

### **2.2.5 Show Password / Export Screen**

If the user decides to press over any record which appear on the main screen, a new view will be opened, in which will appear a full description of all data included in the record. Here, it is possible to find three different buttons:

- Show Button: By pressing this button, the password field, which by default will show `*****` will reveal its true value during three seconds.
- Export Button: By pressing this button, the record content will be exported to the server database, creating a new record if it did not exist before, or updating the old one if its values

have been modified.

- **Delete Button:** By pressing this button the opened record will be deleted from the device local database.

### 2.2.6 Import Record Screen

By pressing over the bottom left corner button, the user can access to this screen. On it, the different records stored in the server will be displayed, and pressing over any of them will either import the pressed one into the local database of the device, making it appearing on the main screen, or update the values of the local record with the values stored in the server if it was already present in the local database of the device.

### 2.2.7 Additional Improvements

Among the different proposed additional improvements, it has been included both the update and deletion of records.

By means of *SQLite* functions is relatively simple to execute both commands if the database model is adequate, skipping *SQL* queries, just by means of simpler *Java* functions and classes.

To update a record, operation that can be done from an importation of records from the server or overwriting records in the local database by generating new records which own exactly the same descriptor, first a connection to the database is established. Then, a *Cursor* object is generated, which will execute a query to look for the introduced descriptor inside the database. If the descriptor is not found, it will produce an insertion, procedure already explained before. Otherwise, a *ContentValues* object is created, in which descriptor, username and password values are set. Then, a *put* command is executed, overwriting the content related to the descriptor by the new introduced one.

Deletion is a simpler process than the update. The same procedure to find the descriptor is followed, and once it is found, instead of executing a *put* command, a *delete* command is performed over the database, deleting all information contained under the retrieved descriptor.

## 2.3 Connection with the Server

One of the main and most important parts of CredHub is connecting with the server. The server allows storing records apart from the local device. Furthermore, records can be imported from the

server to the device too.

According to its implementation, it corresponds with the Java Class *TalkToServer*. This class extends *AsyncTask* in order to perform the Network Connections with the server in a different thread from the main one. *TalkToServer* is mainly divided into two blocks of methods: the ones related to *doInBackground* and *onPostExecute*.

- **doInBackground:** the first thing to be done is establishing the URL which is going to be used to connect with the server. In addition to this, the operation to be performed is identified using a special constant for each of the three possible operations to be performed.

In the case that the operation to be performed is an *Export* the method *exportRecord* is called. The main functionality of this method is sending the needed fields of the new record in order to add it to the Web Server.

However, if *Import* is the operation to be executed the method to be called is *importRecord*. It gets the ID of the record as a parameter. This method sends a request to the Server with this ID and gets the different fields of it. An array of Strings with them is returned so as to use these values for further operations.

Finally, if the operation is *Read* the method called is *readRecords*. It gets no parameters and it performs a request to the Server asking for all the records from the Server. Once received, they are returned as an array of Strings by *readRecords*.

- **onPostExecute:** once that the networking operations have been performed it is time for updating the UI or doing further operations. Depending the operation previously performed one method or another is going to be called.

*PostExport* method is called when an *Export* operation has been executed. Its functionality is very easy. It just redirects the user to the MainScreen as the exporting has already finished.

If an *Import* has been executed, *postImport* is called. In this case, the complexity of this method is greater than the previous one. With the received values from the Server, a new Record is created. If the Record was already in the DataBase, its fields are updated with the new ones.



The last possible operation is *Read*. In this case *postRead* is called. It updates the UI with the records received from the Server. Furthermore, the special listener *ImportListener* is assigned to each row of the UI. For further information about this listener read the following section.

## 2.4 Listeners

In order to implement correctly the different functionalities of CredHub, two different Listeners has been created.

- **ExportListener:** it is the listener set in the Show Password Screen. It is in charge of passing the different fields, which the user introduce, to the *TalkToServer* thread in order to perform an *Export*. It has as fields the caller activity and an array of Strings which has the values of the new record.
- **ImportListener:** in this case, this listener is used by *TalkToServer* when the Import Screen is loaded. Once that the records are loaded from the server, they are assigned each of them with an *ImportListener*. It only has the caller activity as a field. Furthermore, it set the *Import* action to the records in the case they are clicked.

### 3 CredHub App Signing

In order to be able to install an application, it is required to have a digital signature. In this section of the Assignment it is required to sign *CredHub* Application. There are two different ways of signing an Application: through command line or using Android Studio. The choice made in this case is the first one.

The first thing which we have to perform is generating the pair of keys with keytool. It is done using the following command

```
C:\Users\FRAN\Desktop\security_Assignment1-master\security_Assignment1-master>keytool -genkeypair -v -keystore C:\jfkStore -alias jfLLae -keyalg RSA -keysize 2048 -validity 10000
```

After executing the command, further information is requested to be introduced in order to create the keys.

```
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
[Unknown]: Francisco Colmenar
¿Cuál es el nombre de su unidad de organización?
[Unknown]: COSEC
¿Cuál es el nombre de su organización?
[Unknown]: UC3M
¿Cuál es el nombre de su ciudad o localidad?
[Unknown]: Madrid
¿Cuál es el nombre de su estado o provincia?
[Unknown]: Madrid
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: es
```

The last action to be performed is signing the *APK*. In order to sign it, we perform the following command.

```
C:\Users\FRAN\Desktop>jarsigner -verbose -keystore C:\jfkStore -sigalg MD5withRSA -digestalg SHA1 C:\Users\FRAN\Desktop\app-release.apk jfLLae
```

Finally, the *APK* is signed getting as output the following messages.

```
signing: res/mipmap-xxxhdpi-v4/ic_launcher.png
signing: res/mipmap-xxxhdpi-v4/ic_launcher_round.png
signing: resources.arsc
jar signed.
```





## 4 Signature verification of the application *kontaktos.apk*

As a result of the previous section in which we sign an Application, in this part the signature is going to be verified. Furthermore, in order to gain more knowledge about the Application used in this section, which is *kontaktos.apk*, its *AndroidManifest.xml* file is going to be analyzed.








To start with, the first action to be perform is de-compiling the Application. In order to accomplish it, the following command is used.

```
C:\Windows\apktool>apktool d kontaktos.apk
```

Once that the process finishes, you can navigate to the folder which contains *AndroidManifest.xml*.

	kontaktos	25/02/2019 19:35	Carpeta de archivos	
	apktool	25/02/2019 19:21	Archivo por lotes ...	1 KB
	apktool	25/02/2019 19:12	Executable Jar File	10.746 KB
	kontaktos	25/01/2019 7:13	BlueStacks Androi...	8.206 KB

Finally we reach *AndroidManifest.xml*.

	assets	25/02/2019 19:35	Carpeta de archivos	
	original	25/02/2019 19:35	Carpeta de archivos	
	res	25/02/2019 19:35	Carpeta de archivos	
	smali	25/02/2019 19:35	Carpeta de archivos	
	unknown	25/02/2019 19:35	Carpeta de archivos	
	AndroidManifest	25/02/2019 19:35	Documento XML	33 KB
	apktool.yml	25/02/2019 19:35	Archivo YML	44 KB

In regards to *AndroidManifest.xml*, it holds most a great amount of information related to its application. In this section of the Assignment five different *Manifest permissions* are going to be described. The chosen ones are:

1. **android.permission.INTERNET:** The main goal of using this permission is being able of performing network operations. In the case this permission is used in an Application, security measures should take into account in order to avoid the expose private data. Some of this measures could be for example sending the information over SSL.

2. **android.permission.WRITE\_CONTACTS:** As its name suggest, it is related to writing into the *Contacts* of the user. This permission is dangerous due to the fact that once it is granted the Application can modify private data, contacts in this case, from the user's device.
3. **android.permission.BATTERY\_STATS:** This permission is very intuitive in its functionality. It gets the statistics from the battery. It is not one of the most important permissions in regards to security and privacy.
4. **android.permission.MODIFY\_PHONE\_STATE:**It allows changing the state of the telephone however placing calls is not included.
5. **android.permission.DISABLE\_KEYGUARD:** This permission permits the keyguard to be disable if it is not secure.

Finally, the difference between permissions required at runtime and static permissions should be explained in order to gain a wider knowledge about this topic. *Static* permissions are the ones which the user accepts when s/he installs the Application. They are considered as non dangerous permissions and the only way of revoking them is to uninstall the App.

On the other hand, *Runtime* permissions are granted while the user is using the Application. They are explicitly accepted and they are more dangerous than the former ones.

## 5 Inspection of digital signatures and digital certificates of the application kontaktos.apk

To proceed with this task, it is necessary to install the application, which can be done by the following command:

```
adb install kontaktos.apk
```

The first of the proposed tasks is obtaining the digital signature from the corresponding certificate of the application. This can be obtained from the following picture:

```
MacBook-Pro-de-Fernando:base fernando$ openssl pkcs7 -inform DER -in 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
      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:eb:
    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
```

Here, it can be seen the digital signature content, besides more useful information such as the encryption algorithms used, the modulus, etc.

It is also required to obtain the cryptographic hashes of the app logo and three different pixel density images. This is done by exploring the file *CERT.SF*, where it is possible to find the following information:

```
Name: res/drawable-hdpi/icon.png
SHA1-Digest: X5/mtN+YdCFiZSlqEC90RQ4eXFo=
```

Here it can be seen the hash of the icon, produced by means of SHA1 algorithm, a pretty secure algorithm practically unbreakable by computational methods, due to its complexity.

Other hashes of different elements and images can be seen in the following picture:

```
Name: res/layout/about_us_tablet.xml
SHA1-Digest: 9Xcx5q5L/2H8VEKYn+qTm0qgFiY=

Name: res/drawable/linkedin_clickable_button_more.xml
SHA1-Digest: Ex4RmbfTILcs43btzluWo28oc00=

Name: res/drawable-xhdpi/abs_spinner_ab_disabled_holo_dark.9.png
SHA1-Digest: wK6pwNPN0+3FlkoF9etbrUgak=

Name: res/drawable-hdpi/sms_arrived_icon.png
SHA1-Digest: YZv8zLPONpEZRWZLbq04EKTUPNE=

Name: res/layout/wizard_thank_you.xml
SHA1-Digest: Bbn1NEbYERfYUlyFkERiGIBJ+As=

Name: res/drawable-mdpi/vpi_tab_unselected_holo.9.png
SHA1-Digest: ew9WwLQ2mqsvglxLs5e0tb91R1g=

Name: res/layout/twitter_stage.xml
SHA1-Digest: +k8ovfmYuge0J2xzGzzWEab42l0=

Name: res/drawable-xhdpi/abs_list_longpressed_holo.9.png
SHA1-Digest: +gyZSFuut8L02SG5ZyG5Fjp9E+Q=

Name: res/drawable-hdpi/close.png
SHA1-Digest: oRLJkd0S/FW071Z70qRwE2yYe3U=

Name: res/drawable/log_in_btn_states.xml
SHA1-Digest: sMkSXlvrI+Xn0fCIRxnIOXwvpoI=

Name: res/drawable/tab_chat_log_dark.xml
SHA1-Digest: Bvek+KV+Fna7anHBnFx/Wq0viiY=

Name: res/drawable/abs_list_selector_holo_dark.xml
SHA1-Digest: p/c1S+MNGFgaHuzdnQD03yBqrig=

Name: res/drawable-hdpi/new_indicator.png
SHA1-Digest: ggYAUffdbbIkL/pBnWS3Tyr8too=

Name: res/drawable-xhdpi/emoji_categories_nature_off.png
SHA1-Digest: 3KQ0FGKpnj1vBRZRMk+7/XvFrpc=

Name: res/drawable-hdpi/abs_textfield_search_selected_holo_light.9.png
SHA1-Digest: bfKEzWd0W+kFL0Lwu1RK677Sxa0=

Name: res/layout/facebook_friends_finder.xml
SHA1-Digest: oG0/dke8VS8Eg785vU80HuHFQay=

Name: res/drawable-mdpi/abs_progress_secondary_holo_light.9.png
SHA1-Digest: jFGNeShp/bv741EZVHK70cedTq=

Name: res/drawable-xhdpi/abs_ic_menu_moreoverflow_normal_holo_dark.png
SHA1-Digest: gbECTgqxcuVypdWsZeIQx5ea6M=

Name: res/drawable-hdpi/com_facebook_button_grey_pressed.9.png
SHA1-Digest: qqaNe625R0f0wuhKGZHVfUI2zns=

Name: res/drawable/msg_line_bg_out_dark.xml
SHA1-Digest: ic61Q2B7LYDDyxWvmhzR7xIXKG8=

Name: res/drawable-hdpi/log_in_linkedin_pic.png
SHA1-Digest: mso6sXLvFayE8x3biuGX/3h0XMg=

Name: res/drawable-hdpi/vpi_tab_unselected_holo.9.png
SHA1-Digest: CCI1pmvHS2u4IcLaYY4WkAoHIZM=

Name: res/drawable-hdpi/icon.png
SHA1-Digest: X5/mtn+YdCFiZSlqEC90R04eXFo=

Name: res/layout-land/wizard_new_sms_signup.xml
SHA1-Digest: YX/XbbnuyBiHxC9b0cPbuZHCW8M=

Name: res/drawable/wizard_new_btn_fb_states.xml
SHA1-Digest: KlxTKN9in3yfXkmfi0ydtW60rJ4=

Name: res/drawable-hdpi/wizard_btn_normal.9.png
SHA1-Digest: xm8x9p6ZBZ10A16XAf0/2b5Cw00=

Name: res/drawable-hdpi/attach.png
SHA1-Digest: FK3IWnauw0L0K0BF27bbrz5mnJY=

Name: res/drawable-hdpi/facebook_icon.png
SHA1-Digest: w3XnouXBmFTUdZm5mVNEtFcAU8k=

Name: res/drawable-hdpi/attach_bar_icon_dark.png
SHA1-Digest: Fr7mXZx8Ivbuqcg0x5IrnkqgQTS=

Name: res/drawable-hdpi/notification_icon.png
SHA1-Digest: fs+85JZYIIXm4ZqNwLYYAE/LwAo=

Name: res/drawable-hdpi/dialer_down_icon_dark.png
SHA1-Digest: L/dMvgL5jmwKVXfZRh1Xgs5d5A=
```

Finally, the obtained certificate *CERT.RSA* is the following:

```
MacBook-Pro-de-Fernando:META-INF fernando$ keytool -printcert -file CERT.RSA
Propietario: CN=Contapps, OU=Contapps, O=Contapps
Emisor: CN=Contapps, OU=Contapps, O=Contapps
Número de serie: 4c5af52d
Válido desde: Thu Aug 05 19:30:21 CEST 2010 hasta: Fri Jul 23 19:30:21 CEST 2060
Huellas digitales del certificado:
    MD5: FB:F5:EC:31:CC:D2:89:27:AC:85:72:EB:F5:02:B6:73
    SHA1: 93:EB:AA:B7:AC:59:98:8A:A2:F0:DA:A7:EB:90:44:D9:56:03:4A:94
    SHA256: 79:27:0A:D6:6A:93:71:4B:86:DA:3C:5E:98:C3:1F:44:00:67:D9:D2:4A:2A:B4:2B:F3:4C:6D:30:E2:AB:6B:9D
Nombre del algoritmo de firma: MD5withRSA (débil)
Algoritmo de clave pública de asunto: Clave RSA de 1024 bits
Versión: 1

Warning:
El certificado utiliza el algoritmo de firma MD5withRSA, lo que se considera un riesgo de seguridad.
```

The following fields can be seen in the picture:

- Serial Number: Unique identifier with a validity period.
- Validity: Period in which the certificate is valid.
- Coding algorithm used for public key: In this case, RSA of 1024 bits.
- Size of public key: As said, 1024 bits.
- Modulus: It can be found in the first picture of this section.
- Signature algorithm: MD5 with RSA.

## 6 Interacting with the Activity Manager via ADB

In this section, it is first required to obtain a list of the applications installed on the emulator. To do so, the following command is used:

```
pm list packages
```

And this is the result:

```
package:com.google.android.backuptransport
package:com.contapps.android
package:jp.co.omronsoft.openwnn
```

```
package:com.android.wallpaper.livepicker
package:com.example.credhub
package:com.android.netspeed
```

It can be seen that both requested applications have been installed on the emulator.

To see all activities installed on each package, it is necessary to use the following command:

```
[generic_x86:/ # dumpsys package | grep -i com.contapps.android | grep Activity
464c832 com.contapps.android/.sms.ComposeNewMessageActivity
464c832 com.contapps.android/.sms.ComposeNewMessageActivity
464c832 com.contapps.android/.sms.ComposeNewMessageActivity
464c832 com.contapps.android/.sms.ComposeNewMessageActivity
f23578c com.contapps.android/.utils.WebViewActivity
1cb0c24 com.contapps.android/.tapps.gplus.ParseDeepLinkActivity
897f3cb com.contapps.android/.sms.NewMessageActivityLauncher
897f3cb com.contapps.android/.sms.NewMessageActivityLauncher
1c9978 com.contapps.android/.WizardActivity
13dd524 com.contapps.android/.tapps.sms.SmsPopupActivity$TempAddContactActivity
1d938b6 com.contapps.android/.sync.LoginActivity
464c832 com.contapps.android/.sms.ComposeNewMessageActivity
b88c9b7 com.contapps.android/.tapps.sms.SmsPopupActivity
c1ce88d com.contapps.android/.messaging.MessagingRegistrationPageActivity
dc01851 com.contapps.android/.help.WhatsNewActivity
52187af com.contapps.android/.shortcuts.ShortcutActivity
464c832 com.contapps.android/.sms.ComposeNewMessageActivity
com.contapps.android/.sms.NewMessageActivityLauncher
com.contapps.android/.sms.NewMessageActivityLauncher
```

With it, all activities installed on a package are displayed.

To run an already installed activity, it is required to do what is shown in this image:

```
[generic_x86:/ # am start com.contapps.android
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] pkg=com.contapps.android }
```

With it, the corresponding activity is started and displayed on the emulator. just the activity, not the whole application.



To run a service, the command is not strictly identical, but similar:

```
generic_x86:/ # dumpsys package | grep -i com.contapps.android | grep Service
    ad9123e com.contapps.android/.sync.ContactsSyncAdapterService
    35ae333 com.contapps.android/.sync.AccountAuthenticatorService
generic_x86:/ # am startservice com.contapps.android/.sync.ContactsSyncAdapterService
Starting service: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.contapps.android/.sync.ContactsSyncAdapterService }
```

First, the service is selected among the ones the package provides and it is started. No graphical content will be shown in the emulator display, but the process is in execution.

To kill an activity or service it is necessary to look for its related process. The Process Identifier (Pid) is needed to do so:

```
u0_a1      6963  1398  1383948 43512 Sys_epoll_ ad05b424 S android.process.acore
u0_a80     6985  1398  1391308 45596 Sys_epoll_ ad05b424 S com.contapps.android:contacts
root       7038   2     0         0 worker_thr 00000000 S kworker/0:0
```

```
generic_x86:/ # kill 6985
```

```
root       6939   2     0         0 worker_thr 00000000 S kworker/u5:0
u0_a1      6963  1398  1383948 43512 Sys_epoll_ ad05b424 S android.process.acore
root       7038   2     0         0 worker_thr 00000000 S kworker/0:0
```

It can be seen that after executing the kill command with the identifier of the process as argument, the process disappears from the list of running processes.

## 7 Extracting an app via ADB

In the last section of this Assignment, the Application *CredHub* and *kontaktos* will be extracted getting several valuable information.

Firstly, all the list of permissions as well as the the creation dates for both Applications are obtained.

```
drwxr-x--x  6 u0_a80 u0_a80 4096 2019-02-26 10:32 com.contapps.android
```

```
drwx-----  4 u0_a81 u0_a81 4096 2019-02-26 10:26 com.example.credhub
```

Furthermore, the following action is displaying their resources as well as their metadata.

```
./com.contapps.android:
total 56
drwxr-x--x  6 u0_a80 u0_a80 4096 2019-02-26 10:32 .
drwxrwx--x 104 system system 4096 2019-02-26 10:26 ..
drwxrwx--x  2 u0_a80 u0_a80 4096 2019-02-26 10:32 cache
drwxrwx--x  2 u0_a80 u0_a80 4096 2019-02-26 10:32 databases
drwxrwx--x  2 u0_a80 u0_a80 4096 2019-02-26 10:34 files
drwxrwx--x  2 u0_a80 u0_a80 4096 2019-02-26 11:25 shared_prefs

./com.contapps.android/cache:
total 16
drwxrwx--x 2 u0_a80 u0_a80 4096 2019-02-26 10:32 .
drwxr-x--x 6 u0_a80 u0_a80 4096 2019-02-26 10:32 ..

./com.contapps.android/databases:
total 80
drwxrwx--x 2 u0_a80 u0_a80 4096 2019-02-26 10:32 .
drwxr-x--x 6 u0_a80 u0_a80 4096 2019-02-26 10:32 ..
-rw-rw---- 1 u0_a80 u0_a80 20480 2019-02-26 10:32 ContappsDB
-rw----- 1 u0_a80 u0_a80 8720 2019-02-26 10:32 ContappsDB-journal

./com.contapps.android/files:
total 24
drwxrwx--x 2 u0_a80 u0_a80 4096 2019-02-26 10:34 .
drwxr-x--x 6 u0_a80 u0_a80 4096 2019-02-26 10:32 ..
-rw-rw---- 1 u0_a80 u0_a80 36 2019-02-26 10:32 gaClientId

./com.contapps.android/shared_prefs:
total 64
drwxrwx--x 2 u0_a80 u0_a80 4096 2019-02-26 11:25 .
drwxr-x--x 6 u0_a80 u0_a80 4096 2019-02-26 10:32 ..
-rw-rw---- 1 u0_a80 u0_a80 379 2019-02-26 10:32 com.contapps.android_preferences.xml
-rw-rw---- 1 u0_a80 u0_a80 65 2019-02-26 10:34 com.crittercism.crashes.xml
-rw-rw---- 1 u0_a80 u0_a80 65 2019-02-26 10:32 com.crittercism.exceptions.xml
-rw-rw---- 1 u0_a80 u0_a80 65 2019-02-26 10:32 com.crittercism.loads.xml
-rw-rw---- 1 u0_a80 u0_a80 409 2019-02-26 11:25 com.crittercism.prefs.xml
-rw-rw---- 1 u0_a80 u0_a80 206 2019-02-26 10:32 timestamps.prefs.file.xml
```

```

./com.example.credhub:
total 40
drwx----- 4 u0_a81 u0_a81 4096 2019-02-26 10:26 .
drwxrwx--x 104 system system 4096 2019-02-26 10:26 ..
drwxrwx--x 2 u0_a81 u0_a81 4096 2019-02-26 10:26 cache
drwxrwx--x 2 u0_a81 u0_a81 4096 2019-02-26 10:26 databases

./com.example.credhub/cache:
total 16
drwxrwx--x 2 u0_a81 u0_a81 4096 2019-02-26 10:26 .
drwx----- 4 u0_a81 u0_a81 4096 2019-02-26 10:26 ..

./com.example.credhub/databases:
total 80
drwxrwx--x 2 u0_a81 u0_a81 4096 2019-02-26 10:26 .
drwx----- 4 u0_a81 u0_a81 4096 2019-02-26 10:26 ..
-rw-rw---- 1 u0_a81 u0_a81 20480 2019-02-26 10:26 User_DB
-rw----- 1 u0_a81 u0_a81 8720 2019-02-26 10:26 User_DB-journal

```

Once that the resources has been obtained, the databases are going to be the following data to be displayed. The databases related to both Applications are going to be listed.

```

com.contapps.android/databases/:
total 80
drwxrwx--x 2 u0_a80 u0_a80 4096 2019-02-26 10:32 .
drwxr-x--x 6 u0_a80 u0_a80 4096 2019-02-26 10:32 ..
-rw-rw---- 1 u0_a80 u0_a80 20480 2019-02-26 10:32 ContappsDB
-rw----- 1 u0_a80 u0_a80 8720 2019-02-26 10:32 ContappsDB-journal

com.example.credhub/databases/:
total 80
drwxrwx--x 2 u0_a81 u0_a81 4096 2019-02-26 10:26 .
drwx----- 4 u0_a81 u0_a81 4096 2019-02-26 10:26 ..
-rw-rw---- 1 u0_a81 u0_a81 20480 2019-02-26 10:26 User_DB
-rw----- 1 u0_a81 u0_a81 8720 2019-02-26 10:26 User_DB-journal

```

After this step, the tables and columns of CredHub's database are showed. In order to accomplish it we need to use Sqlite.

```





[MacBook-Pro-de-Fernando:databases fernando$ sqlite3 User_DB
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
[sqlite> .tables
User          android_metadata

[MacBook-Pro-de-Fernando:databases fernando$ sqlite3 User_DB
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
[sqlite> .schema User
CREATE TABLE User (Description TEXT PRIMARY KEY,Username TEXT NOT NULL>Password TEXT NOT NULL);

```

Finally, using *DB Browser for SQLite* we extract all the credentials from the database.

▼	Tablas (2)		
▼	User		CREATE TABLE User (Description TEXT PRIMARY KEY,Username TEXT NOT NULL>Password TEXT NOT NULL)
	Description	TEXT	"Description" TEXT
	Username	TEXT	"Username" TEXT NOT NULL
	Password	TEXT	"Password" TEXT NOT NULL
▶	android_metadata		CREATE TABLE android_metadata (locale TEXT)

Tabla:     

	Description	Username	Password
	Filtro	Filtro	Filtro
1	Facebook	Prueba	Password
2	Twitter	Twit	Pass

