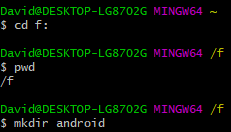
**Inicializando el control de versiones Git para el proyecto**

Utilizaremos Git para controlar de una manera más inteligente nuestro proyecto. Git nos permitirá realizar con mayor eficiencia una “copia” de varios puntos de la aplicación y nos permitirá retroceder a un punto determinado para ver el estado de la aplicación en aquel entonces o avanzar hasta ver el resultado final.

Para inicializar git vamos a la ruta donde vamos a crear el proyecto en este caso en mi unidad f: y creamos un directorio llamado android:



Una vez creado, nos situamos en dicho directorio:

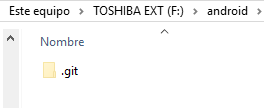


E inicializamos el git para poder trabajar con él:



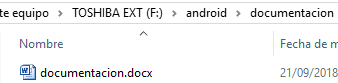
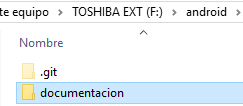
**Revisando el directorio**

Si finalmente ahora abrimos el directorio que hemos creado anteriormente, podemos ver que tras realizar el git init. Git nos ha creado automáticamente un subdirectorio que contiene todos los archivos necesarios del repositorio (el esqueleto de un repositorio Git).

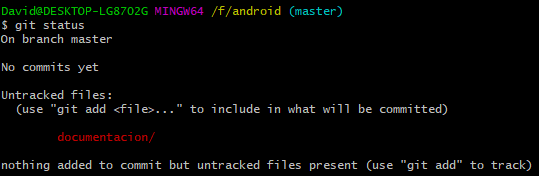


**Añadiendo contenido a nuestro proyecto**

Bien lo siguiente será crear una carpeta documentación y añadir este subdirectorio el documento de Word para que esté dentro del repositorio. Para ello, movemos el archivo al directorio f:/android/documentación e introducimos el documento de word:



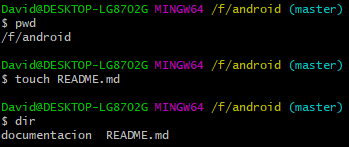
Si ahora visualizamos el directorio desde la consola de comandos de git hacemos git status podemos comprobar que aún no hemos realizado ningún commit. Un commit significaría realizar una copia “una foto” del estado de nuestro proyecto para poder retroceder en caso que lo deseemos más tarde.



**Creado el readme.md**

Vamos a ver cómo realizar un readme.md y para que nos va a servir.

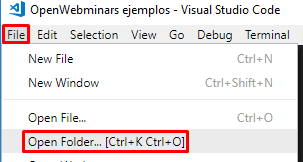
Un readme es el archivo que va a contener la información principal del proyecto. En español y durante la época de los 90, era más característico realizar un leeme.txt está representa su evolución. Si os fijáis tiene la extensión .md que significa markdown. Y normalmente suele escribirse con mayúsculas README.md. En git hub, donde iremos subiendo las diferentes versiones de nuestro proyecto git si un repositorio tiene un archivo README.md este será convertido en html y presentado como página principal dado esto su importancia.



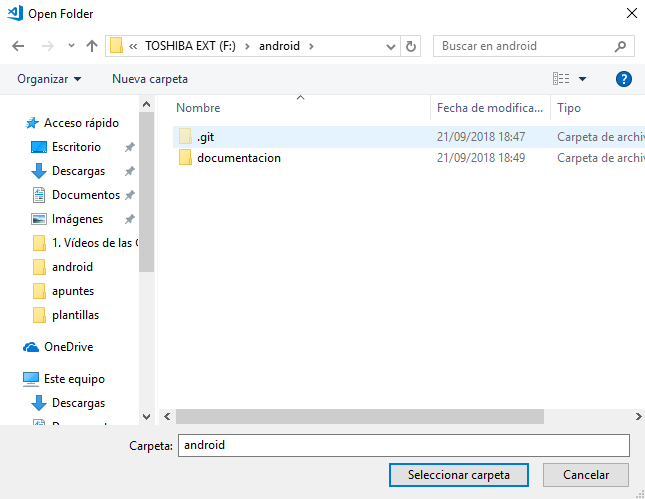
Si hacemos un touch README.md ya hemos creado el archivo y si ahora hacemos un dir podemos ver que tenemos el directorio que hemos creado anteriormente y el archivo README.md.

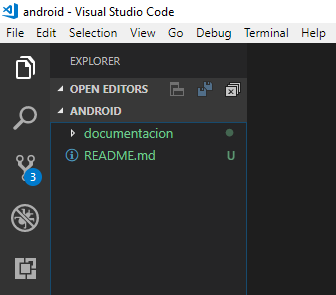
Bien para trabajar con nuestro archivo README.md vamos a utilizar Visual Studio Code. Uno de los beneficios que nos ofrece dicho IDE es que nos permitirá entender mejor que es el README.md y gestionar mejor los commits de git (las fotos con el estado actual).

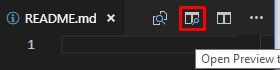
Para ello, vamos a File > Open Folder…



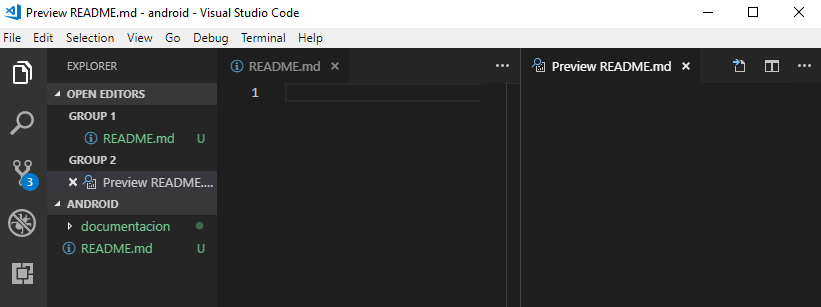
Y seleccionamos el directorio de nuestro proyecto.

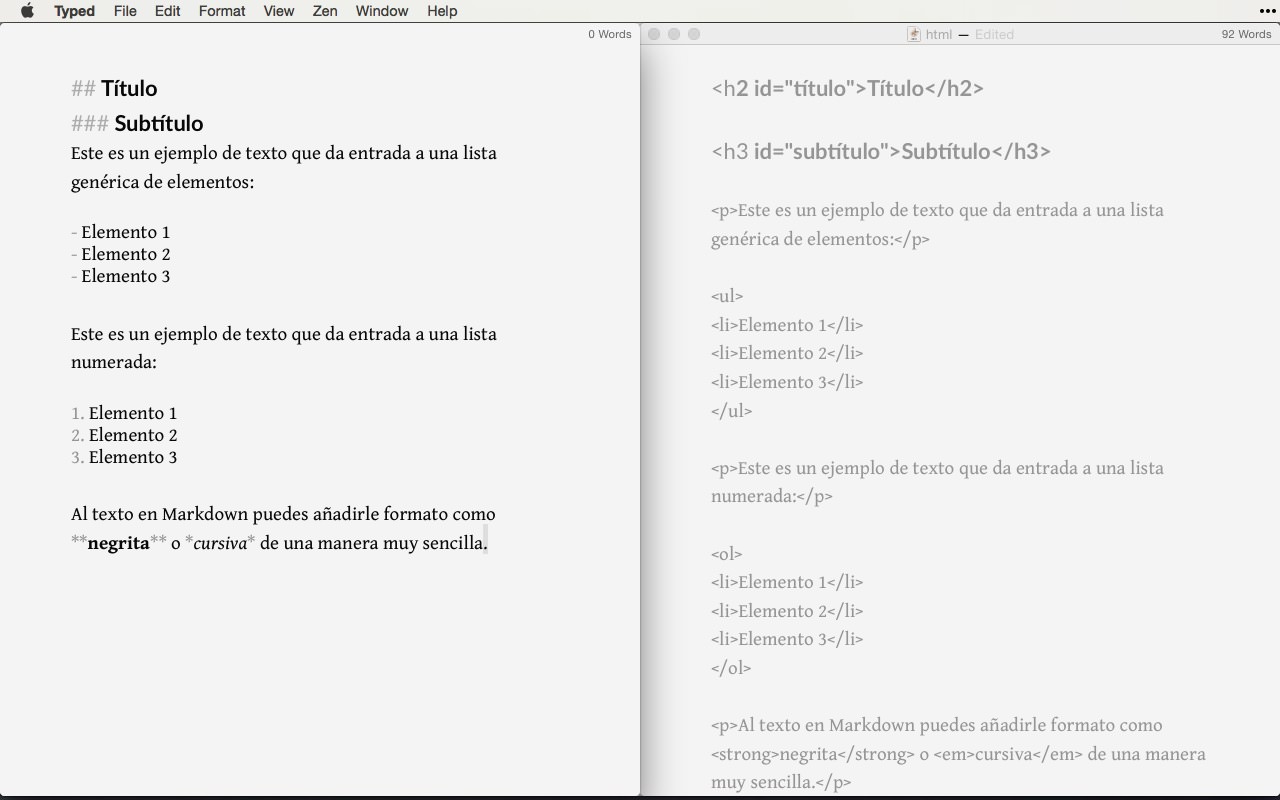


Si abrimos el fichero README.md y le damos a open preview:



Tenemos la división del readme tanto para verlo en formato markdown como en formato que se visualizará en Git:



Un ejemplo para que nos quedé claro del todo lo que es markdown es un html simplificado el ejemplo de la derecha es en markdown y el de la izquierda el equivalente en html.

Bien pues vamos a empezar a redactar el contenido básico de nuestro README.md

**# iMosso**

**## 1.Información del proyecto**

- Dispositivos compatibles: Android

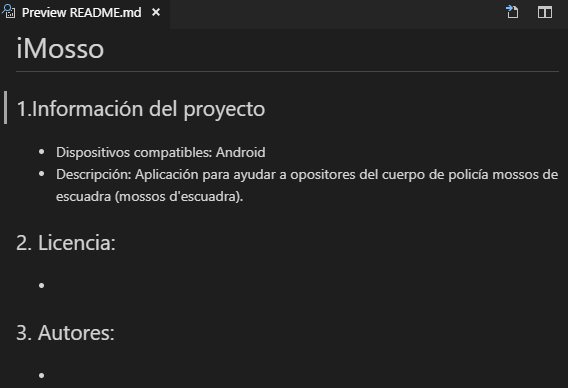
- Descripción: Aplicación para ayudar a opositores del cuerpo de policía mossos de escuadra (mossos d'escuadra).

**## 2. Licencia:**

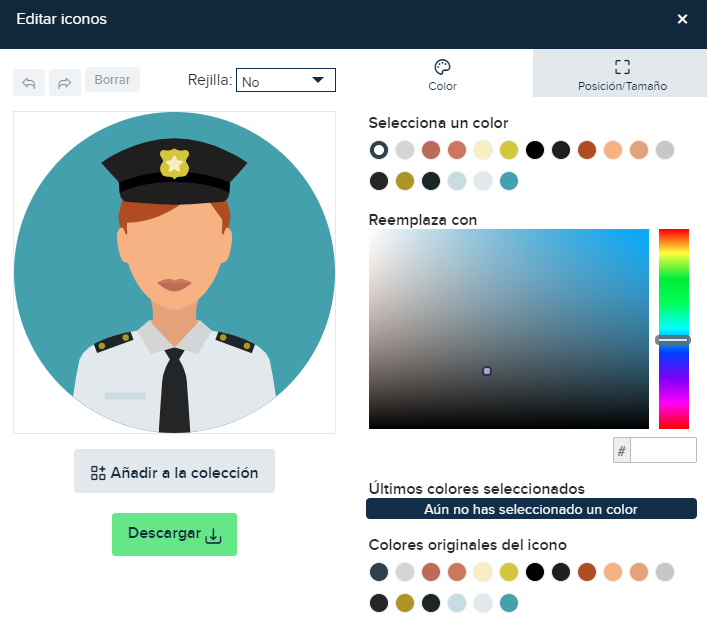
-

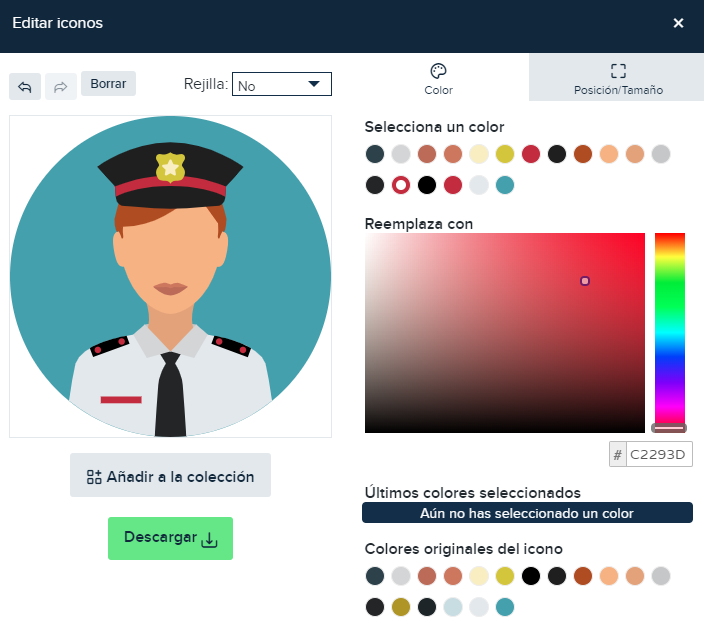
**## 3. Autores:**

-

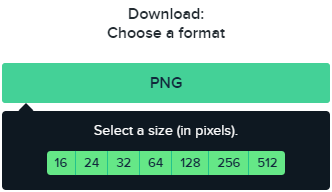
Bien pues una vez escrito si visualizamos el contenido:

Para realizar el icono de la app nos hemos metido en <https://www.flaticon.es/> y basándome meticulosamente en los colores de los uniformes del cuerpo de policía del que vamos a hacer la oposición y con herramientas para sacar el color, hemos buscado el siguiente:

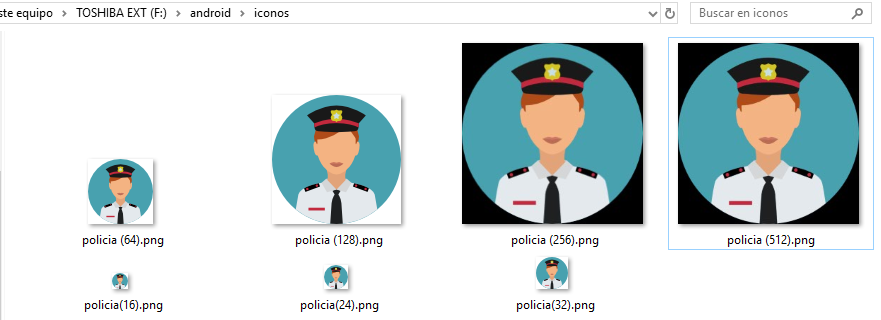




Y una vez tenemos la imagen la vamos a descargar y guardar en todos los formatos



Y vamos a crear una carpeta en el proyecto y vamos a poner los iconos con las diferentes resoluciones.



Y ahora que ya tenemos el icono vamos a añadirle dicho icono a la documentación para mejorar la estructura del README.md mediante a ![](iconos/policia(256).png)

**# iMosso**

![](iconos/policia(256).png)

**## 1.Información del proyecto**

- Dispositivos compatibles: Android

- Descripción: Aplicación para ayudar a opositores del cuerpo de policía mossos de escuadra (mossos d'escuadra).

**## 2. Licencia:**

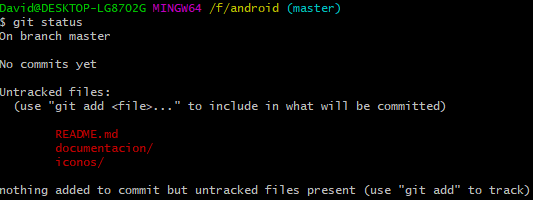
-

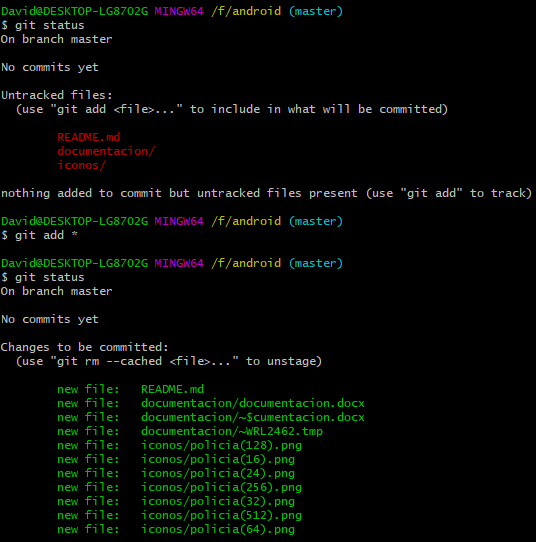
**## 3. Autores:**

-

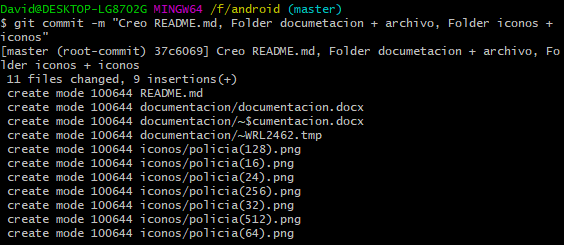
El resultado como podéis comprobar hace que el archivo tenga una presentación mucho más elegante.

Bien llega la hora de realizar el primer commit para guardar el estado de todo el trabajo que hemos realizado hasta ahora.

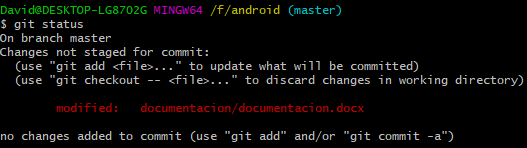




También podríamos hacer un git add . que es equivalente a git add \*

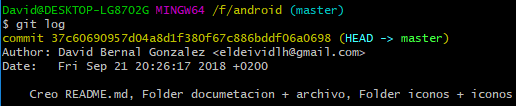


Si ahora compruebo el git status:



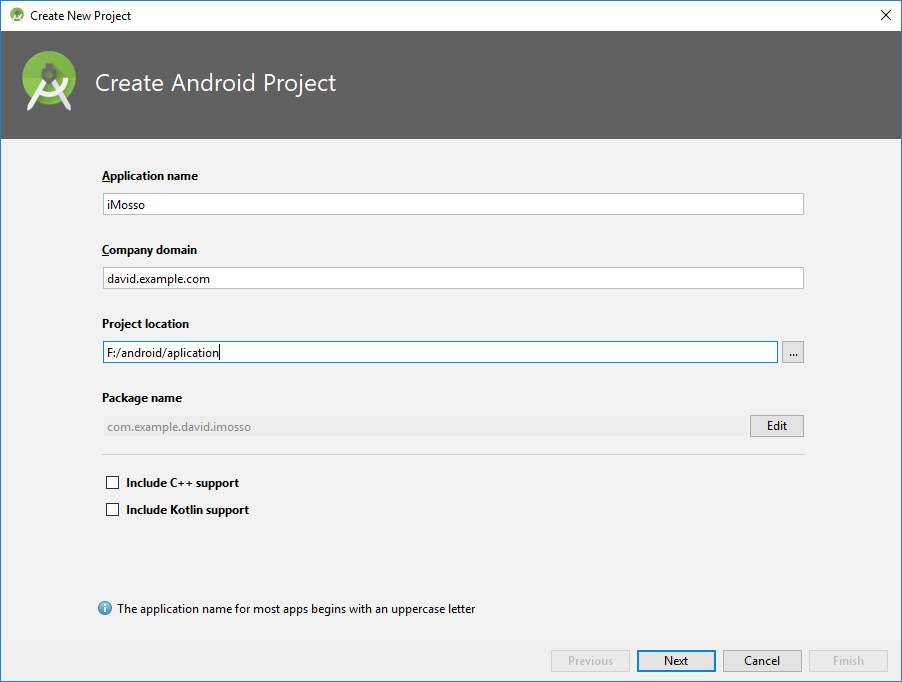
Puedo comprobar que ya han desaparecido todos los archivos a excepción de la documentación que estoy editando conforme realizo el proyecto.

Y aquí puedo ver que he hecho un commit con todo para poder volver a dicho punto cuando quiera.

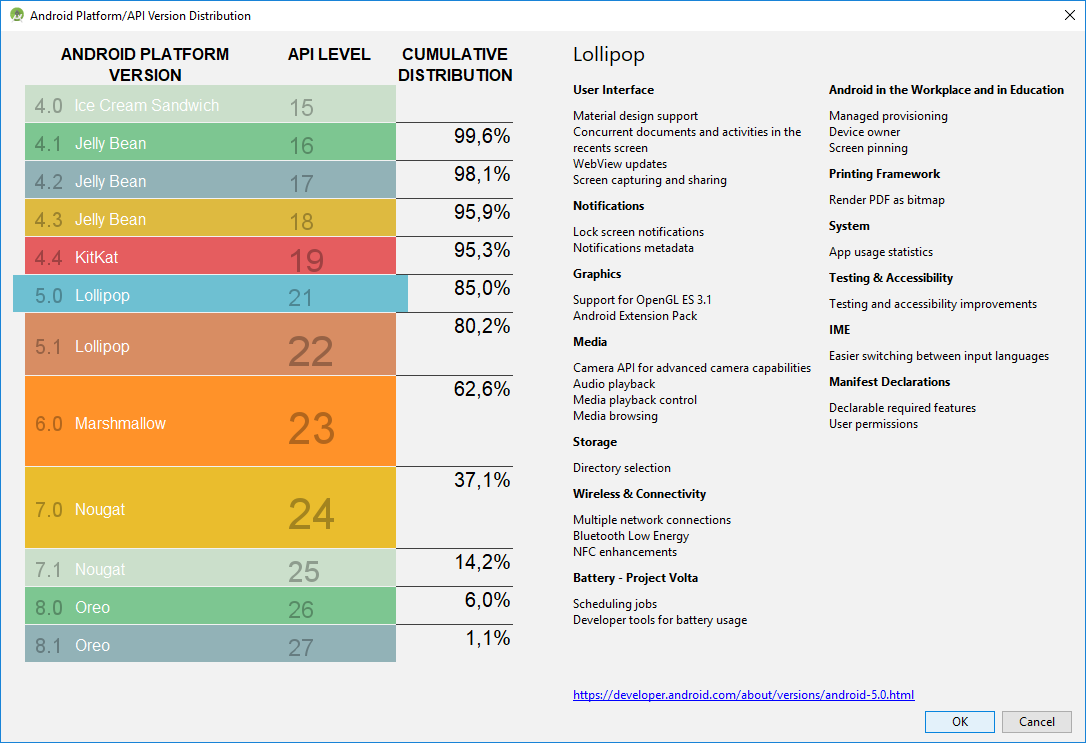


**Creando el proyecto de Android**

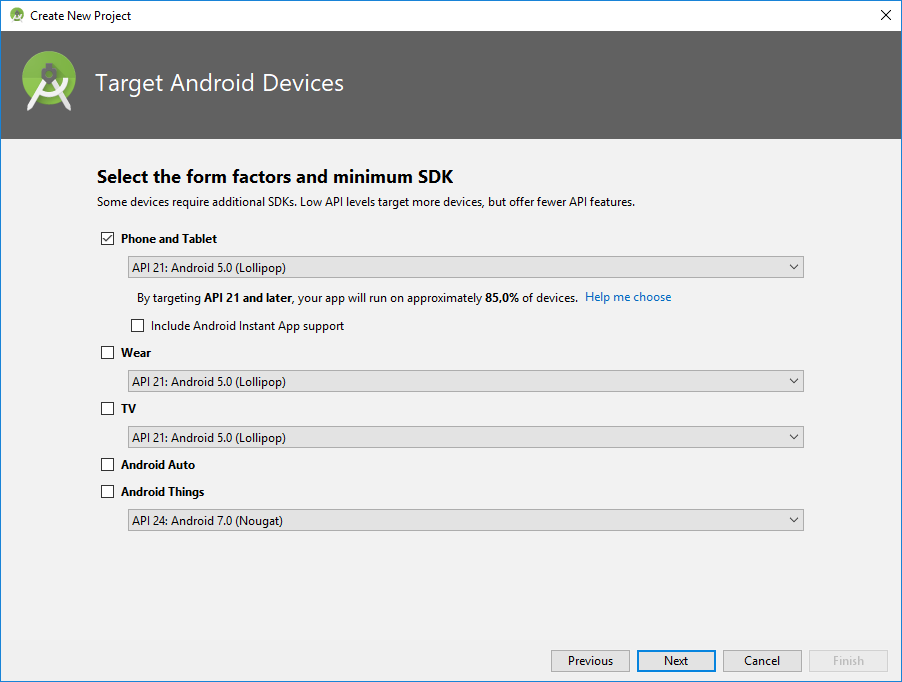
Llega la hora de ponernos ya con Android studio, lo primero que vamos a hacer es crear el proyecto con File > New > Proyect e indicar la ruta de nuestro proyecto git y concretamente se guardará en el subdirectorio aplication.

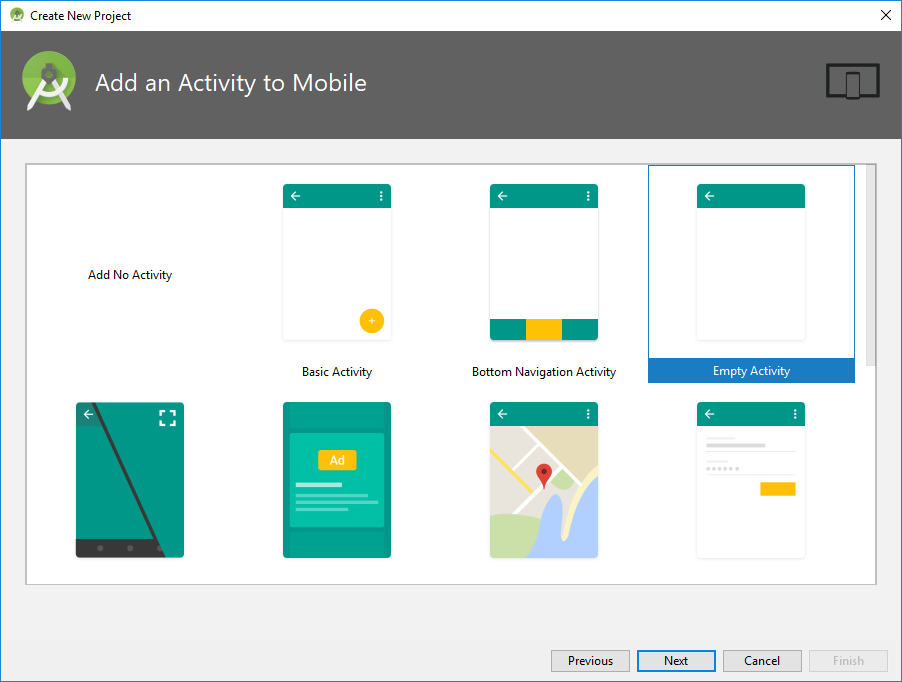


**Seleccionando una versión Android para el proyecto**

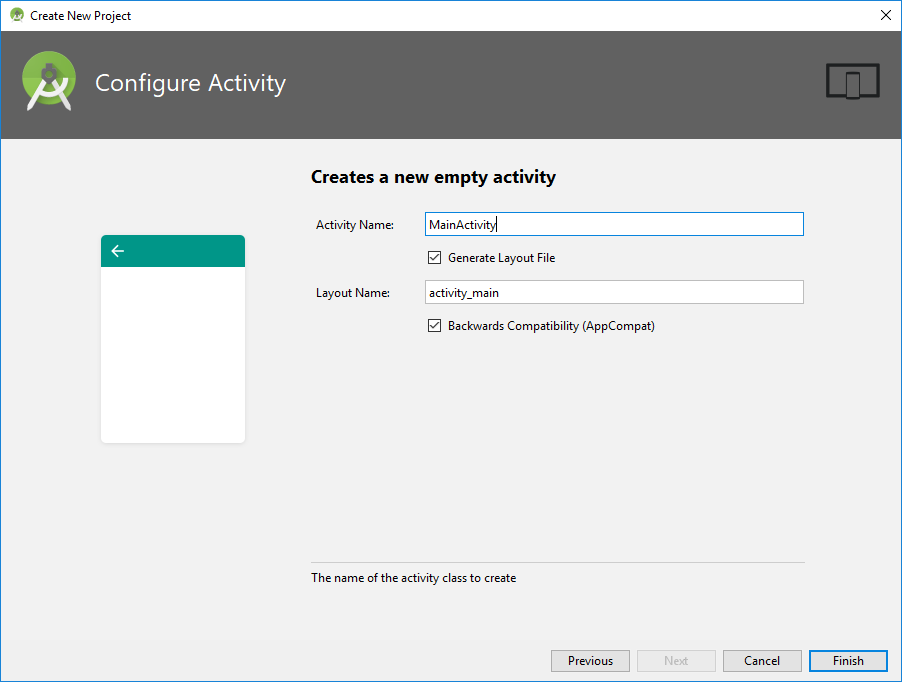
Bien, llega la hora uno de los momentos más difíciles, el momento de elegir una versión de android. La versión afortunada en este caso es la 5.0 Lolipop. Con un 85 % de compatibilidad sobre todos los dispositivos Android, deja muy pocos dispositivos fuera sobre todo pensando que el público objetivo de la aplicación es de entre 18 años y 35 años aproximadamente. Ya que han quitado desde hace poco el máximo de edad pero ese es nuestro público objetivo.

Por lo tanto, esos usuarios, suelen tener dispositivos que normalmente no superan los 3 – 4 años de media y por tanto prefiero trabajar con una api algo más avanzada para que la aplicación sea lo más estable posible. Por tanto, y a modo de resumen api nivel 21 con una compatibilidad de un 85 % pero debido a que gran parte de nuestros usuarios son millennials, creemos que la aproximación será de un 90 – 92 % aproximadamente.





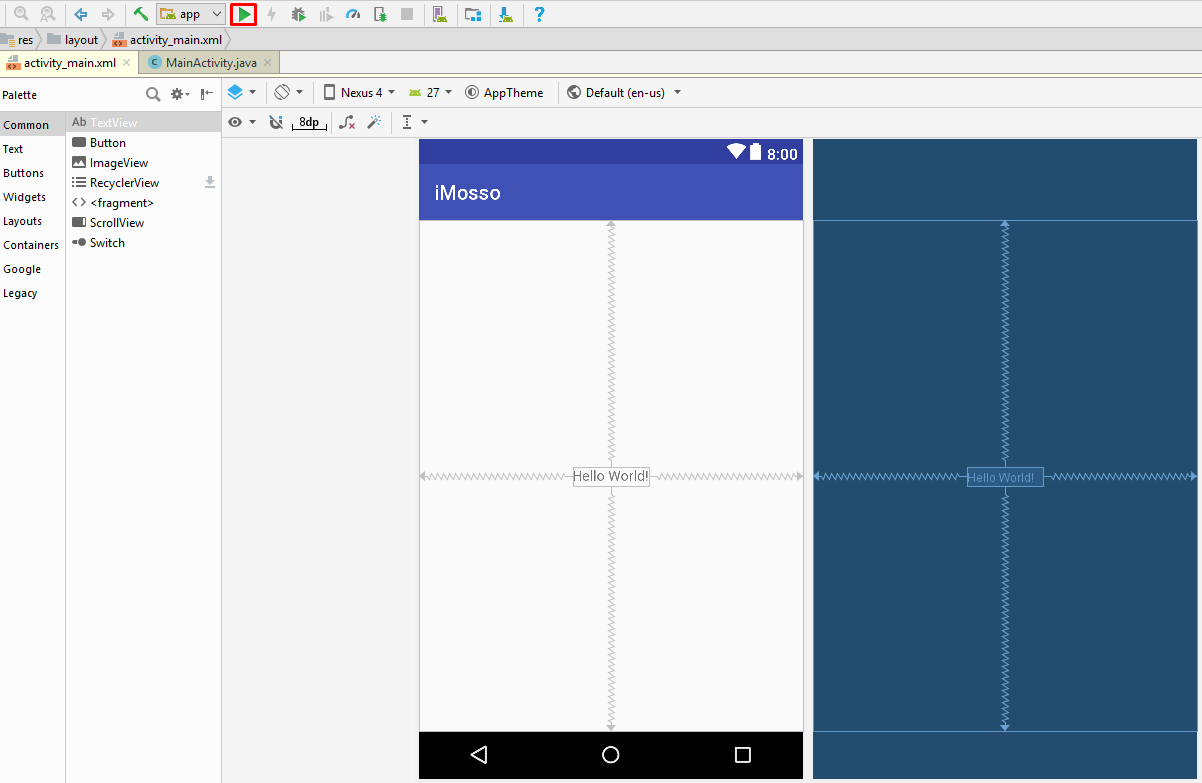
Y finalmente, creamos la actividad principal con el nombre de MainActivity.

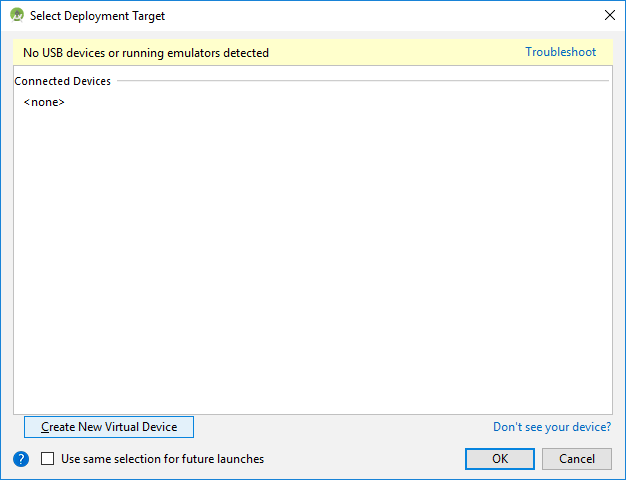


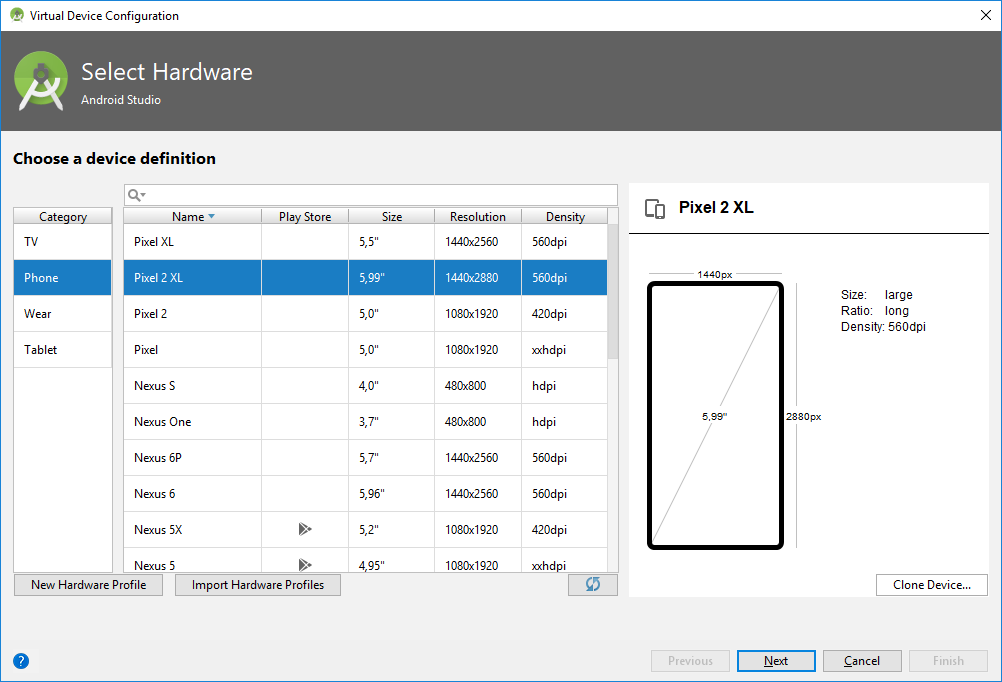
Al finalizar la creación del proyecto, nos aparece un mensaje de si queremos que VCS Version Control System no Visual Studio Code pero en este caso preferimos pulsar sobre ignorar y gestionar todo desde git por línea de comandos.

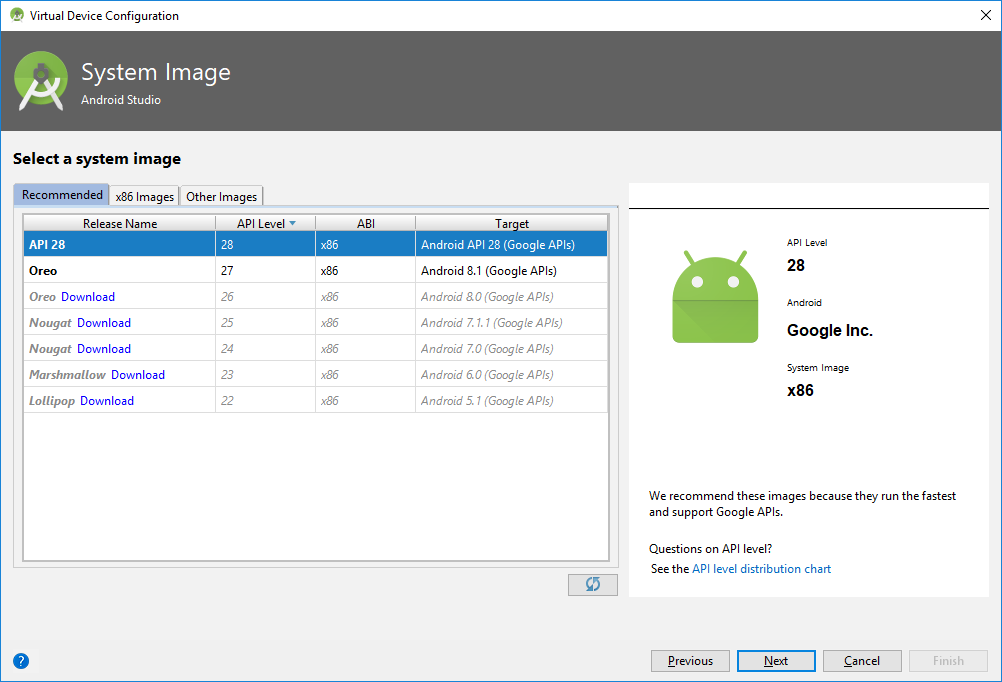


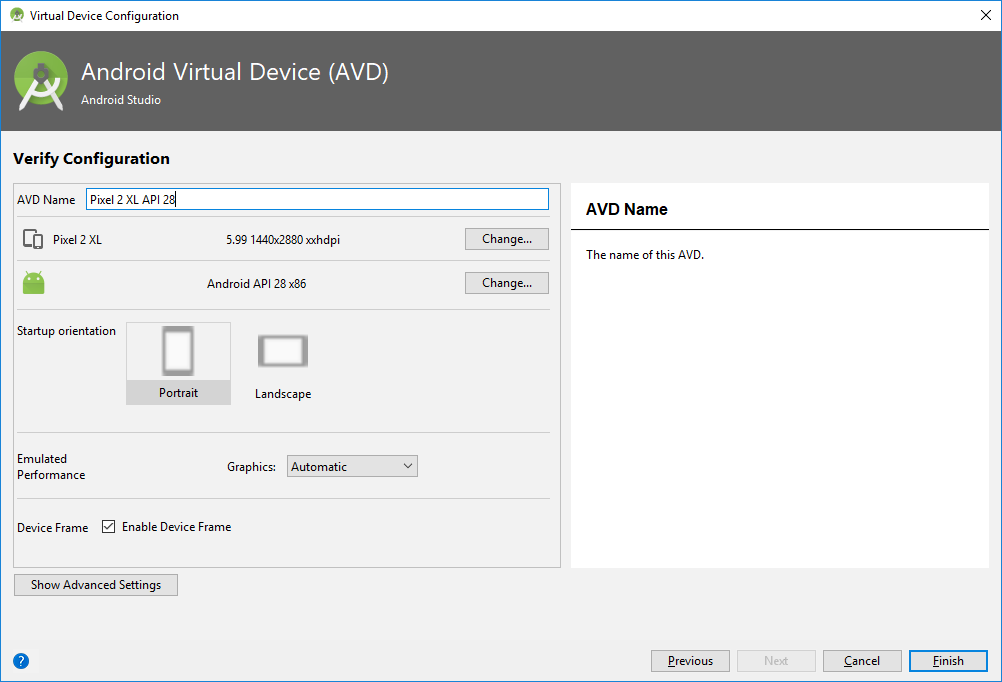
**Creando la máquina virtual y la primera vista previa del nuestra aplicación**

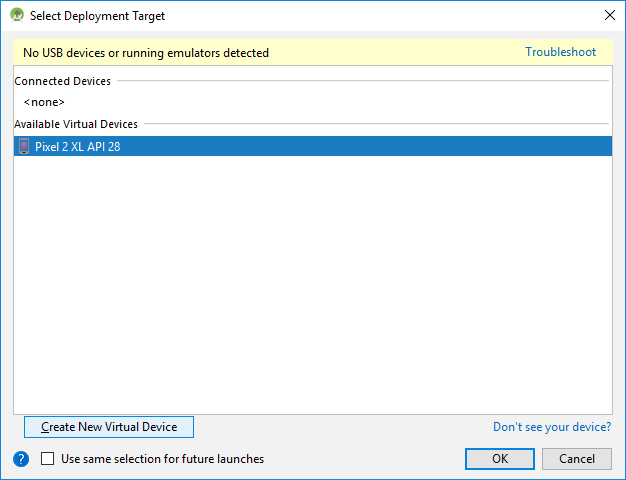


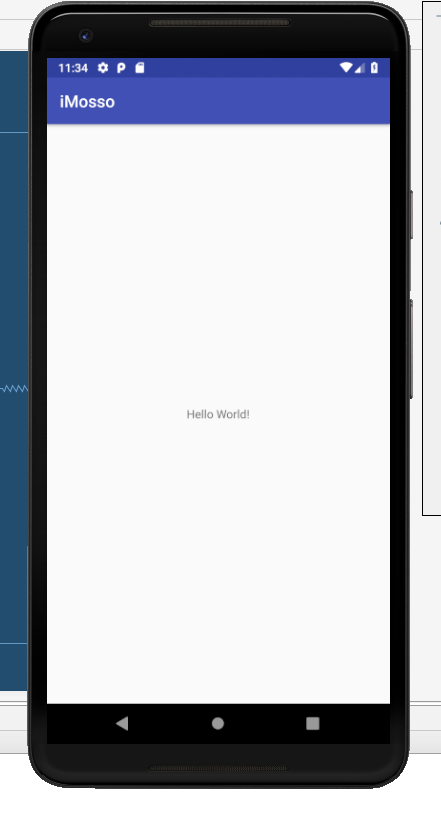












**Añadiendo información al README.md**

Aprovechamos para añadir toda la información que vamos conociendo al README.md

**# iMosso**

![](iconos/policia(256).png)

**## 1.Información del proyecto**

- Dispositivos compatibles: Android

- Compatibilidad:

- Versión mínima de Android : 5.0 Lolipop

- Nivel de api mínimo: api 21

- Compatibilidad mínima: 85 % (90-92 % estamación real).

- Descripción: Aplicación para ayudar a opositores del cuerpo de policía mossos de escuadra (mossos d'escuadra).

**## 2. Licencia:**

- Software Comercial

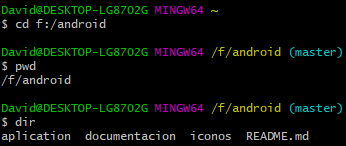
**## 3. Autor:**

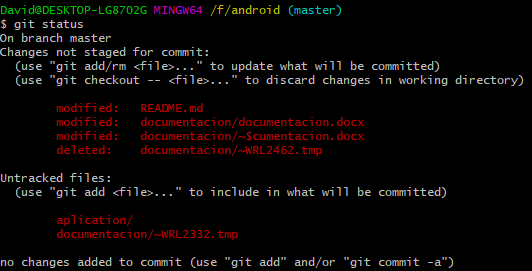
- David Bernal Gonzalez

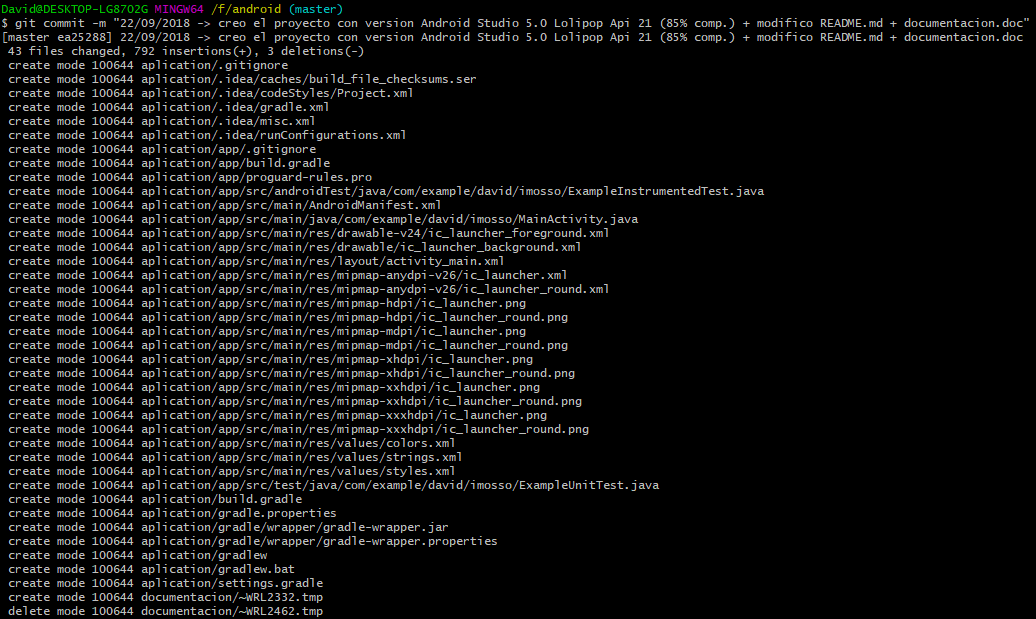


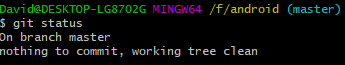
**Realizando un commit de nuevo**

Bien una vez tenemos el proyecto Android creado, considero que no nos encontramos en un buen momento para crear una copia de Git.

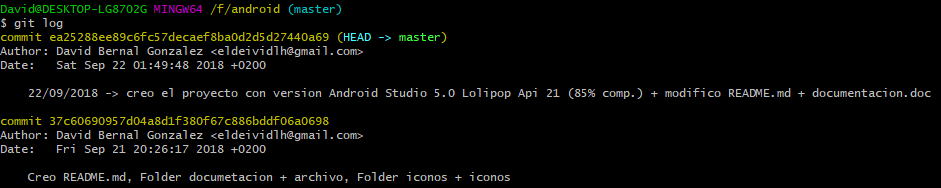




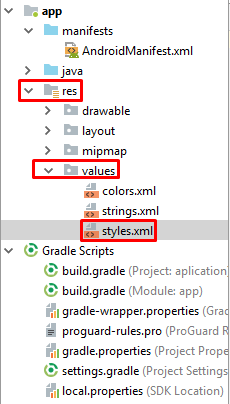


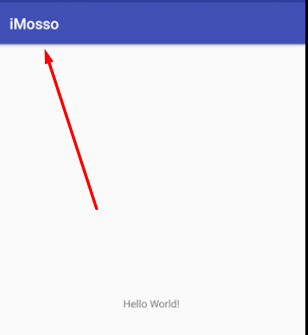


Y ya podemos ver que tenemos dos puntos guardados en el git:



**Eliminando la barra superior iMosso:**

Lo primero que vamos a hacer en el proyecto es eliminar la barra superior iMosso de color azul para disponer de más espacio en nuestra aplicación y a su vez porque considero que es algo antiestético.



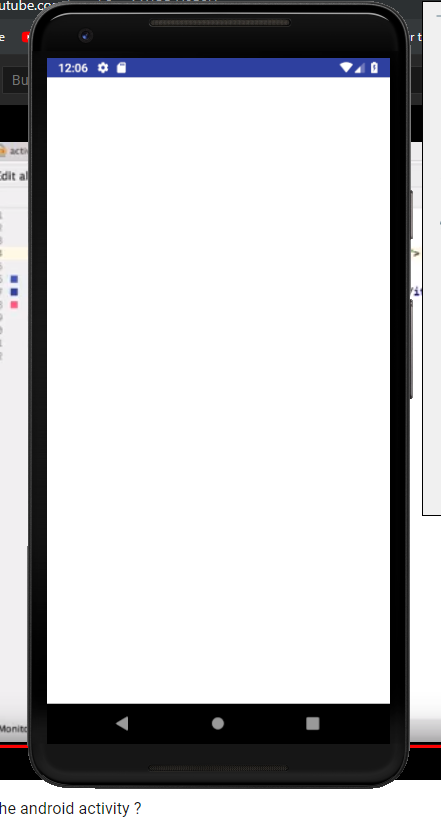
Para ello, vamos a:

Y tenemos el siguiente archivo XML:

<**resources**>  
  
 *<!-- Base application theme. -->* <**style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
 </**style**>  
  
</**resources**>

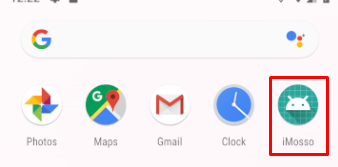
Eliminamos del parent del style **Light.DarkActionBar** y lo substituimos por **NoActionBar** de tal manera que nos quede de la siguiente manera:

<**resources**>  
  
 *<!-- Base application theme. -->* <**style name="AppTheme" parent="Theme.AppCompat.NoActionBar"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
 </**style**>  
  
</**resources**>

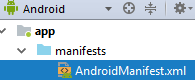


**Cambiando el icono de la aplicación:**

Si nos metemos en la aplicación podemos ver que el icono de iMosso es horroroso. Si queremos crearnos nuestra propia marca, debemos definir un icono que nos diferencie del resto y no podemos utilizar el que viene por defecto en cualquier aplicación de android.



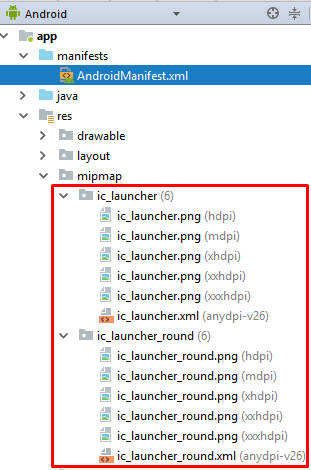
Por ello, vamos a utilizar el icono que habíamos creado anteriormente. Bien para cambiar el icono, lo primero que tenemos que hacer es abrir el fichero de android manifest.



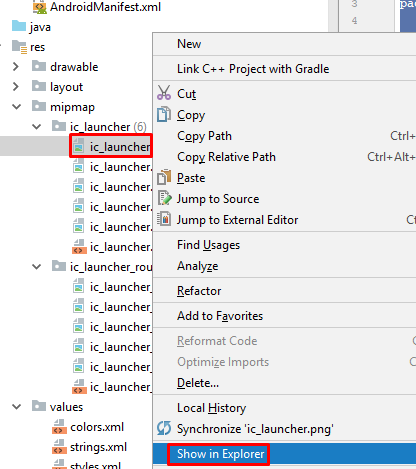
*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.david.imosso"**>  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 </**application**>  
  
</**manifest**>

En android manifest, podemos comprobar que tenemos dos iconos y cambiaremos ambos.



Para ello, vamos a:

Donde si os fijáis tenemos varias resoluciones del icono. Por lo que pondremos



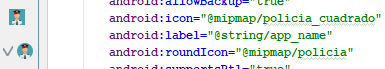
Lo teníamos de está manera:

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
package="com.example.david.imosso"**>  
  
<**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
</**application**>  
  
</**manifest**>

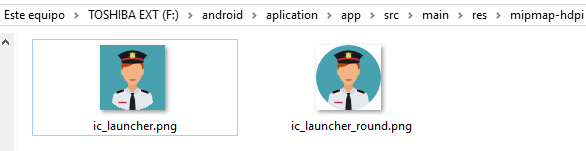
Y lo que vamos a hacer es cambiar los nombre y substituir los iconos:

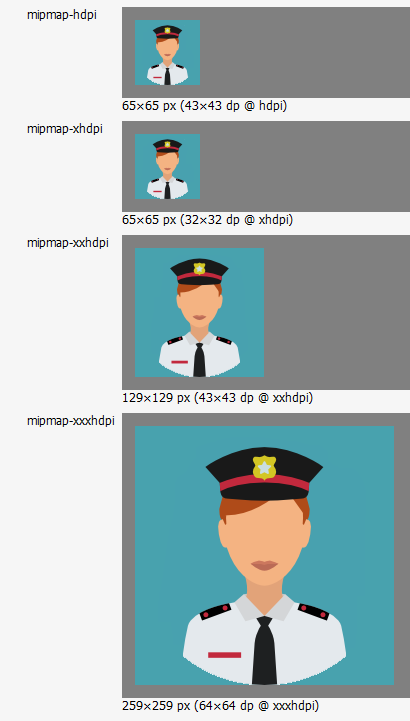
*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
package="com.example.david.imosso"**>  
  
<**application  
 android:allowBackup="true"  
 android:icon="@mipmap/policia\_cuadrado "  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/policia "  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
</**application**>  
  
</**manifest**>

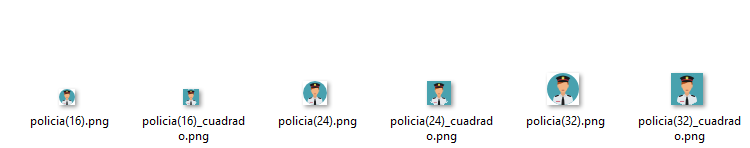
Y una vez cambiados nos aparece en las imágenes de tal manera:

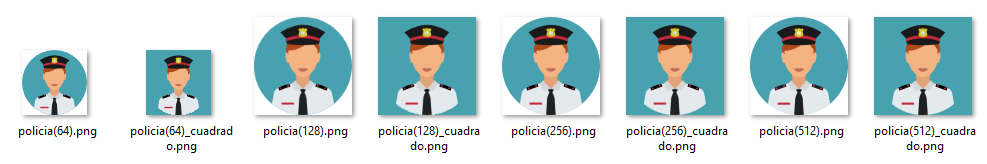


Y lo que vamos a ir haciendo es eliminar las dos imágenes y substituirlas por las de policia.png y policia\_cuadrado.png.

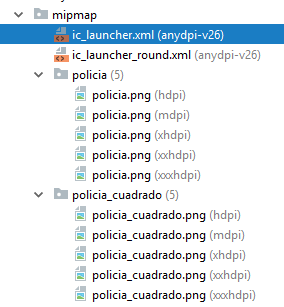
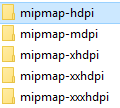
 Esto lo haremos en todas las resoluciones para que quede de la siguiente manera.

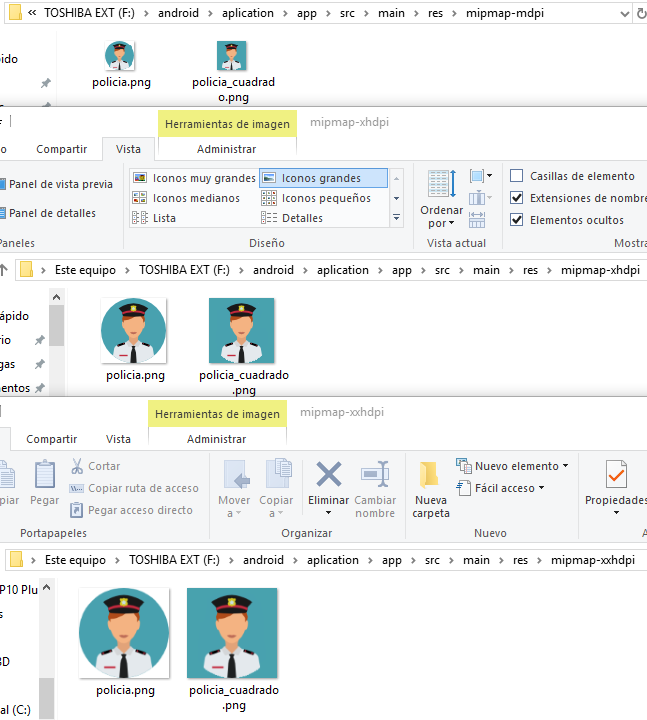


Por cierto también hemos tenido que editar el icono para que en el caso de que sea el icono cuadrado se vea con dicho formato de todas las imágenes con photoshop.

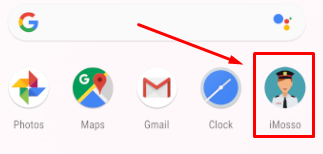


Y en cada resolución tenemos los dos iconos adaptados a esa medida el redondo y el cuadrado.

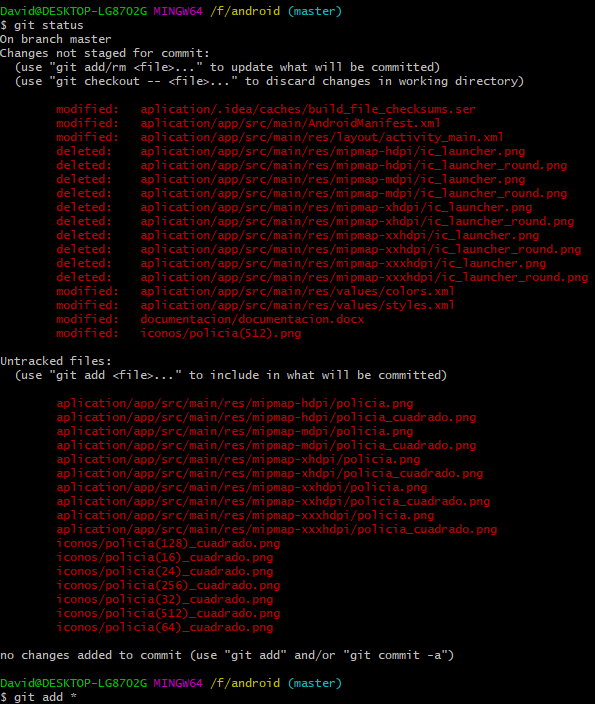
 

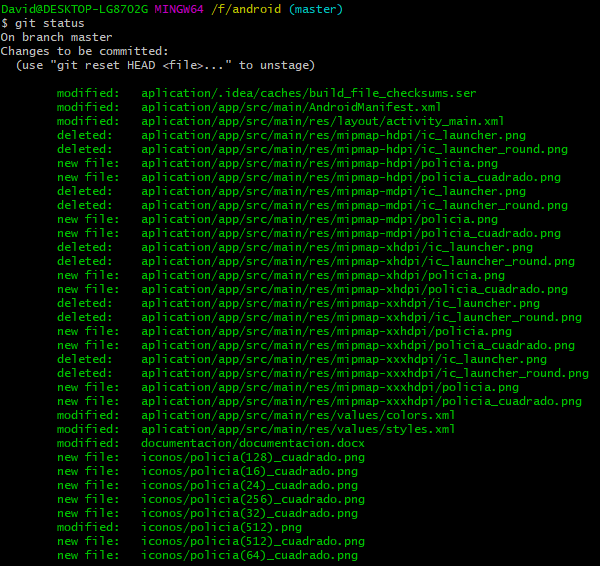


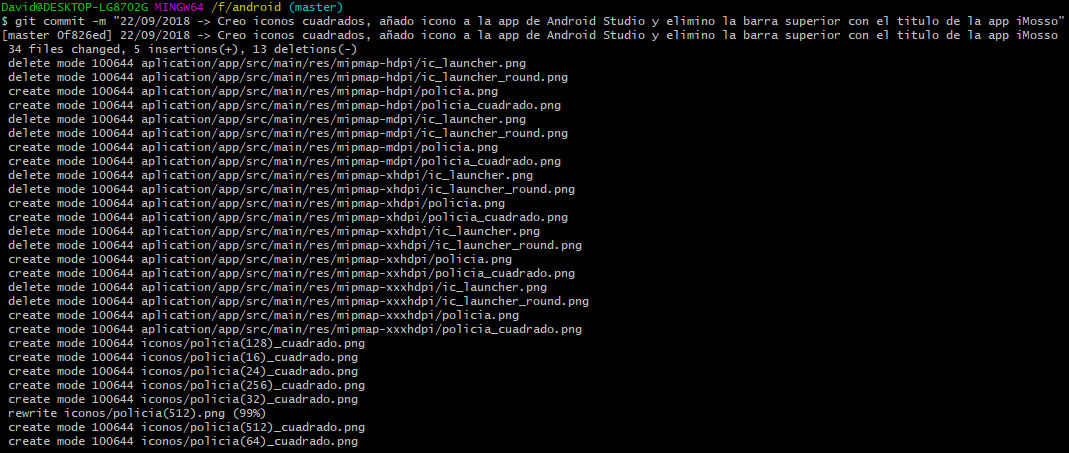
Y finalmente, ya tenemos nuestro icono:

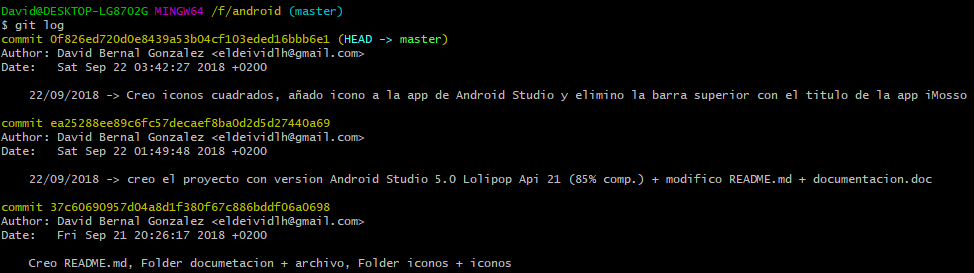


**Realizando un commit de nuevo**





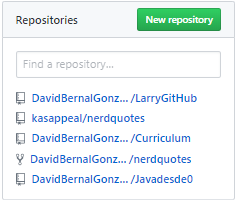


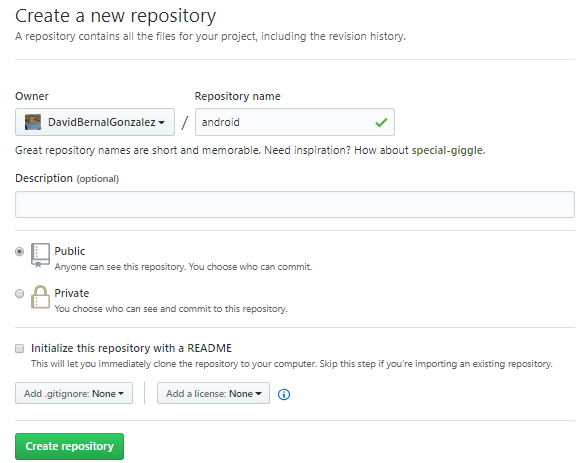


**Creando un proyecto en Git Hub y subiéndolo con git pull**

En este caso queremos compartir en Git Hub nuestra primer repositorio y de esta manera tener seguro el proyecto para ello tenemos que realizar un push.

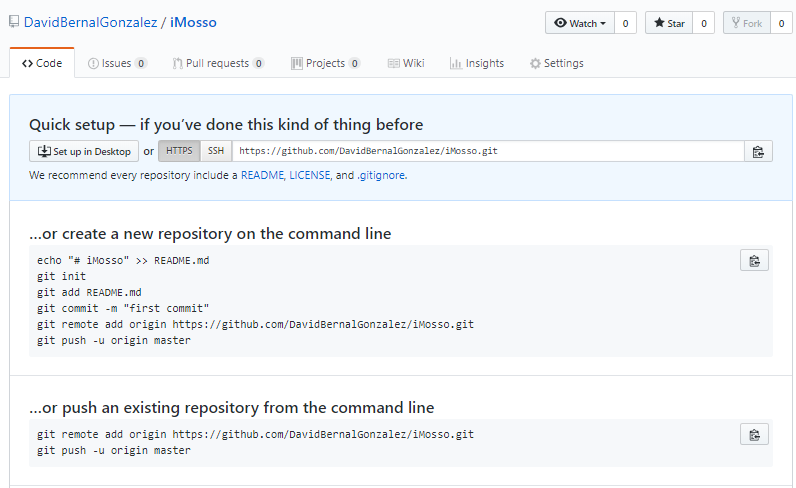
Para poder realizar el push, necesitamos previamente crear un proyecto para eso vamos a new repository:



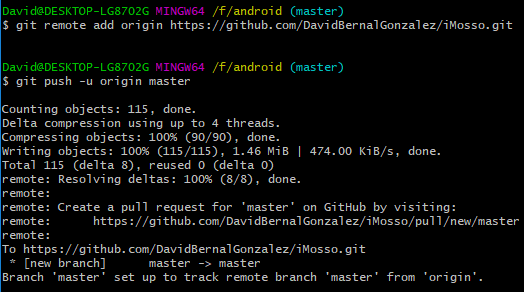


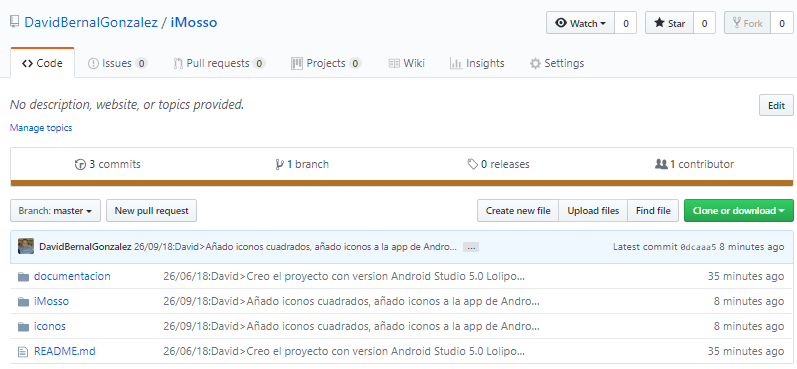
En este caso, solo se puede poner la modalidad Publica si haces la privada requiere un pago de 7$ mensuales. Git Hub, podría decirse que es la red social más popular del mundo de código.

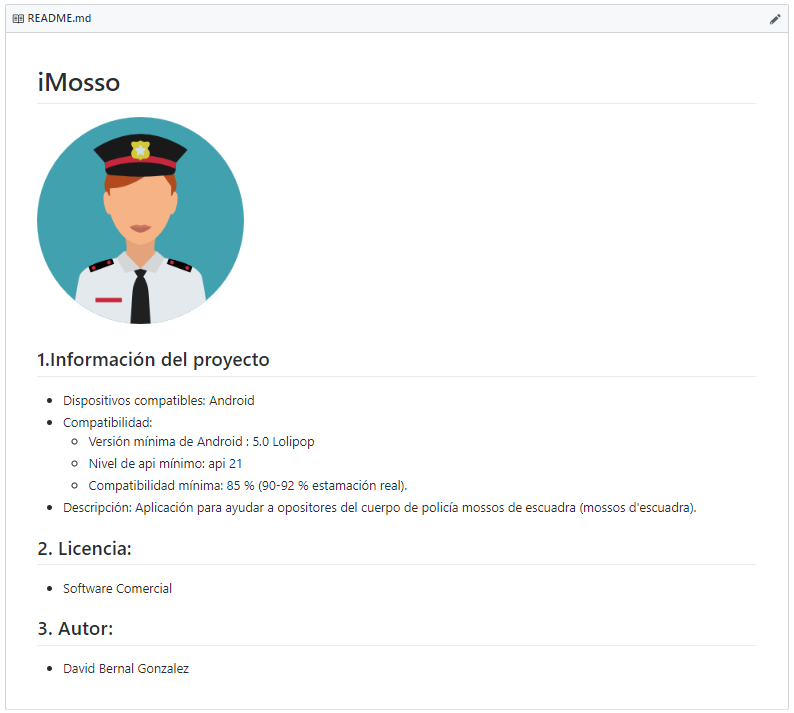
Bien una vez tenemos el proyecto, tenemos que seguir los pasos subrayados en azul.



Si copiamos y pegamos en la consola de Git Bash



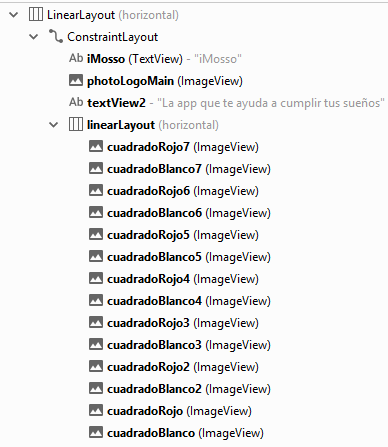




**Dándole un lavado de cara a nuestra Main Activity**

Este es el resultado por el momento:

****

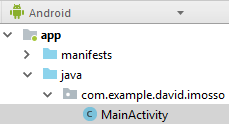


**Barra de estado/navegación transparente:**

Bien, vamos a hacer que la barra de estado sea transparente por una cuestión estética y que coincida con el color de la app.



Para modificar el color del status bar debemos ir a:



**package** com.example.david.imosso;  
  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.Window;  
**import** android.view.WindowManager;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

Y añadimos:

**package** com.example.david.imosso;  
  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.Window;  
**import** android.view.WindowManager;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 Window barratop = getWindow();  
 barratop.setFlags(WindowManager.LayoutParams.***FLAG\_TRANSLUCENT\_NAVIGATION***, WindowManager.LayoutParams.***TYPE\_STATUS\_BAR***);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

Si ahora finalmente compilamos de nuevo:



Vemos que la parte superior es transparente y que tiene el mismo color que la app.

**Loggin con Google**

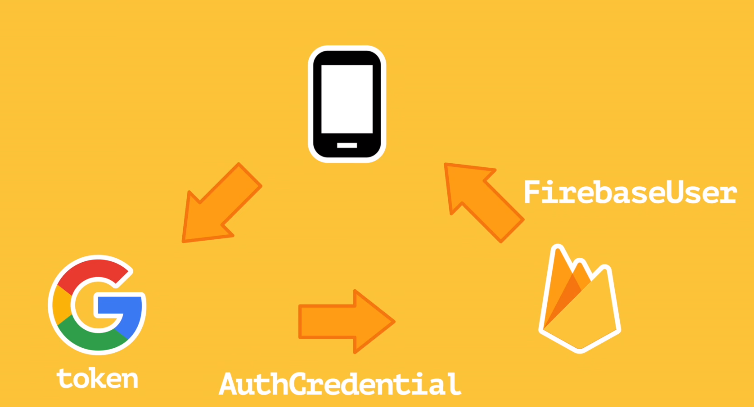
<https://www.youtube.com/watch?v=rPzHPNOt7Js> Google

<https://www.youtube.com/watch?v=O3aemJ9eAAA> Facebook

Para conectarse existen dos alternativas utilizar directamente el SDK de Android o utilizar el Firebase UI un proyecto de código abierto que nos ayuda a implementar la autentificación entre otras cosas. Para tener control total utilizaremos SDK.



Lo primero que tenemos que hacer es que nuestra aplicación se autentifique con google al hacer esto obtendremos un token con la ayuda de este toquen nos creamos una credencial que nos permitirá autentificarnos con firebase. Una vez autentificados, podremos acceder al objeto firebaseuser este es el objeto que tiene toda la información del usuario y la podremos mostrar en nuestra aplicación.

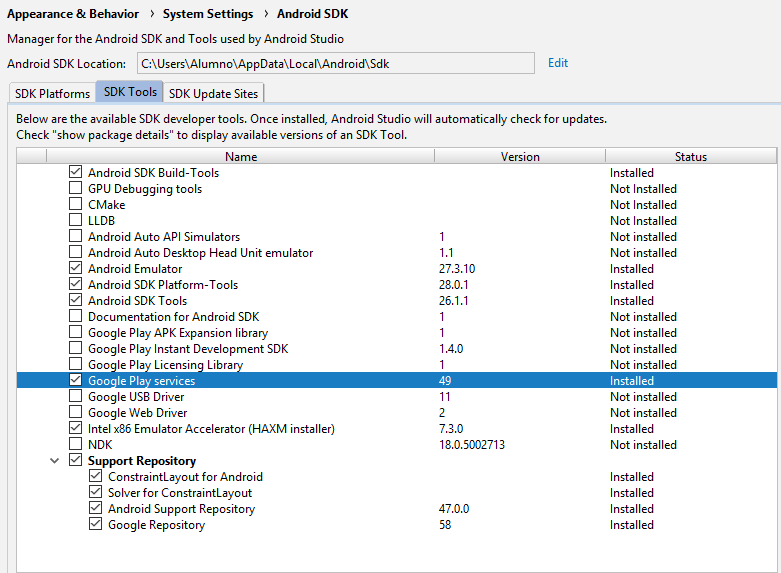


De la misma forma, si queremos cerrar sesión primero cerramos sesión en Firebase y luego cerramos sesión en google con sus respectivos métodos y así, quedamos como en el inicio.



Lo primero que tenemos que hacer entonces es implementar un inicio de sesión con Google tenemos un vídeo completo de cómo hacer esto desde 0 y que esto será nuestra base.

Antes de empezar debemos tener instalado como requisito el SDK de Google Play Services. Para verificar o instalar vamos a ir a SDK Manager en Android Studio. Para ello, vamos a la sección SDK Tools y lo actualizamos o lo instalamos.

* 

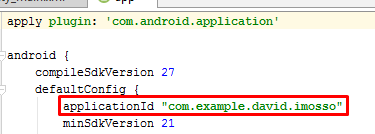
Creamos un nuevo proyecto en firebase.

Nos registramos en: <https://console.firebase.google.com> 

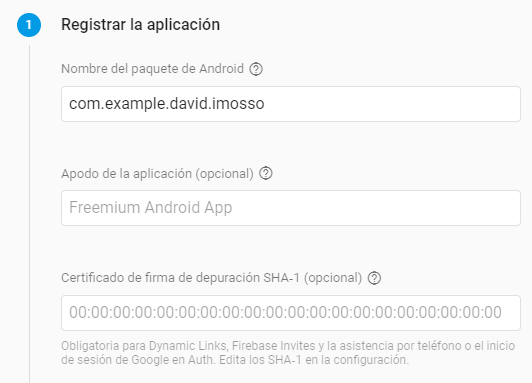
Seleccionamos alguna región que no influye en lo que haremos. Después de la creación pulsamos a Android para el asistente de configuración.



Para conocer el nombre del paquete nos dirigimos a Android Studio y abrimos el build.grandle del módulo y copiamos el valor de applicationId



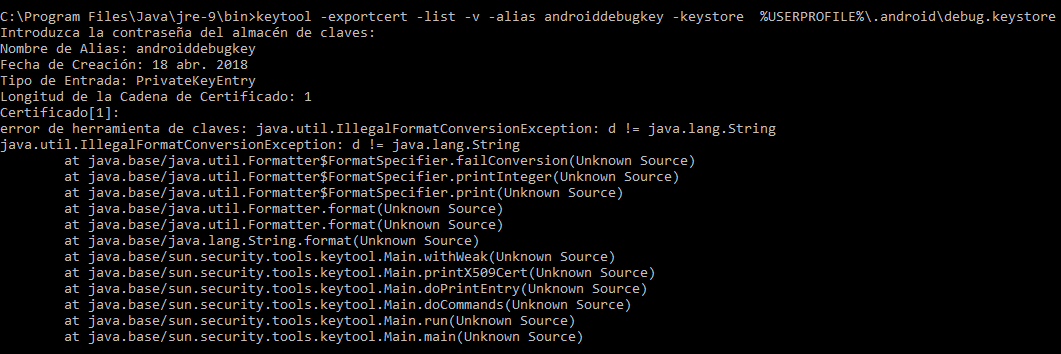
El segundo campo, es opcional y no lo necesitamos. Pero si el tercero. Para obtener el SHA-1 podemos presionar el botón de ayuda y la página nos mostrará cómo obtenerlo para MAC, Windows o Linux.



En Windows para ejecutar el SHA-1 debemos ir primeramente a la carpeta donde tengamos instalado java pulsamos SHIT + click derecho y pulsamos sobre la opción abrir en terminal. Y ahora pegamos el chorizo:

Keytool -exportcert -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore

La contraseña por defecto es la palabra android en minúsculas.

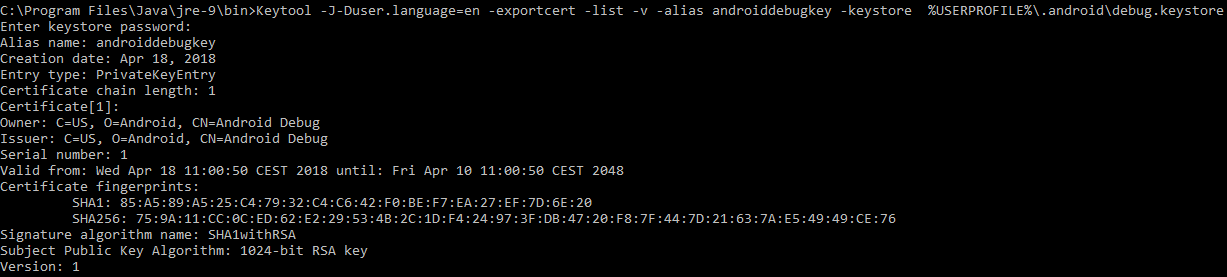


Descubrí que el problema proviene de la última versión de Keytool en JDK8.151y JDK9. De forma predeterminada, keytool usa el idioma predeterminado del sistema y parece que ya no funciona correctamente en algunos idiomas (en mi caso, español).

Fue suficiente para forzar la salida en inglés utilizando el parámetro -J-Duser.language=en

Intente usar esta línea de comando en su lugar:

Keytool -J-Duser.language=en -exportcert -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore



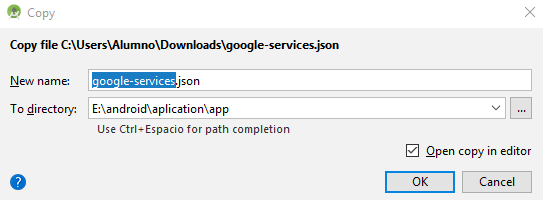
SHA1: 85:A5:89:A5:25:C4:79:32:C4:C6:42:F0:BE:F7:EA:27:EF:7D:6E:20

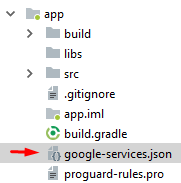
Pegamos el SHA1 en la consola de firebase y continuamos.



Ahora descargamos el archivo de configuración de google services. En android studio cambiamos el modo de ver el proyecto a Project y arrastramos y soltamos el archivo de configuración dentro de la carpeta app.

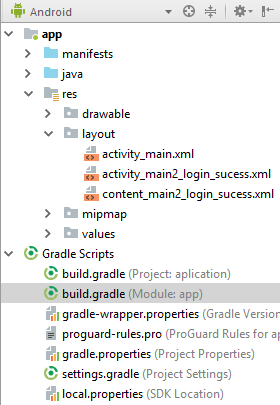


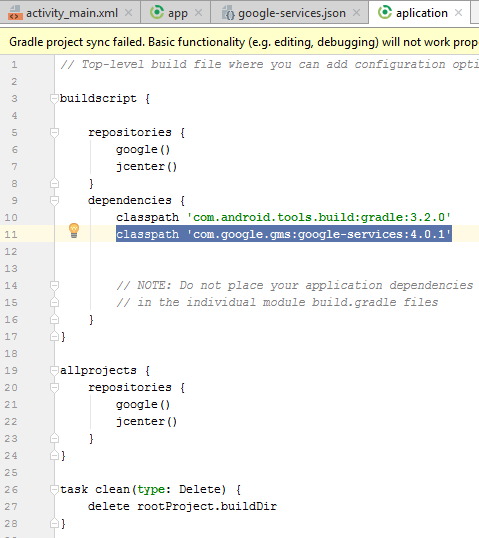


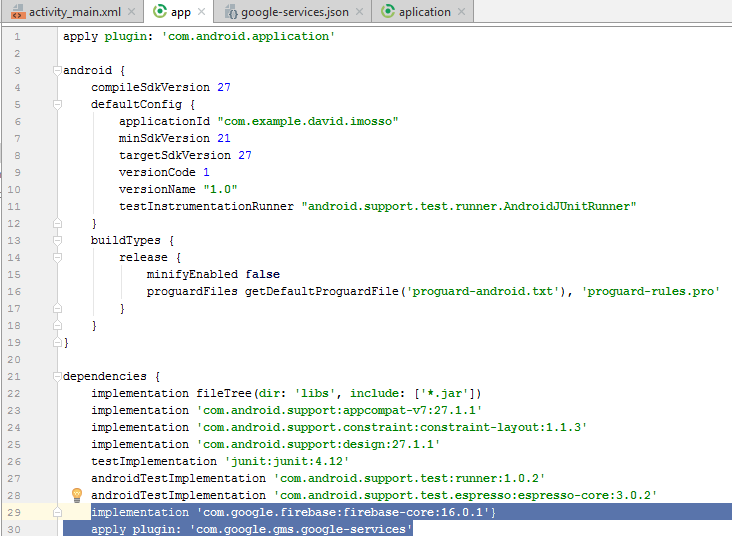


Lo que tenemos que hacer a continuación es adicionar estas dos líneas al build.grande del módulo y al build.grande del proyecto.



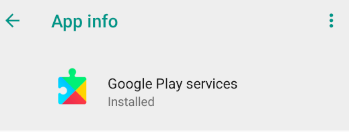


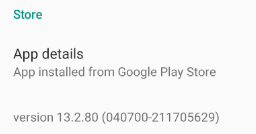




Y ya terminamos con la consola de firebase. Lo último que haremos para la configuración es adicionar una dependencia en el build.grande del módulo:

Para saber la versión de google play services que tenemos vamos a configuración aplicaciones y seleccionamos google play services:





Por lo tanto, si queremos probar la aplicación en este emulador, deberíamos coger una versión igual o menor. En este caso, vamos a escoger la 9.8 colocamos en la dependencia 9.8.0 y pulsamos sincronizar en la parte superior. Y ya podemos empezar para poder utilizar el api de google siempre vamos utilizar como intermediario a GoogleApiClient como necesitamos autentificarnos con google en nuestra activity, nos crearemos una variable global con private GoogleApiclient googleApiClient;

<https://www.youtube.com/watch?v=O3aemJ9eAAA>

