¿Qué es Android?

Android: Sistema operativo

Android es un sistema operativo basado en Linux con una interfaz de programación Java.

El Android Software Development Kit (SDK de Android) proporciona todas las herramientas necesarias para desarrollar aplicaciones para Android. Esto incluye un compilador, un depurador y un emulador de dispositivos, así como su propia máquina virtual para ejecutar programas para Android.

Android está desarrollado por Google y proporciona una gran biblioteca de componentes para crear una compleja interfaz de usuario, soporta 2-D y gráficos 3-D usando las librerías OpenGL, el acceso al sistema de archivos y proporciona una base de datos SQLite integrada.

Las aplicaciones para Android se componen de elementos diferentes y pueden volver a utilizarse componentes de otras aplicaciones. Por ejemplo, se puede desarrollar una aplicación que utilice la Galería de Android para seleccionar una foto.

Google Play

Google ofrece el servicio de Google Play en el que los programadores de Google puede ofrecer su aplicación Android a los usuarios de Android. Esta aplicación viene instalada en todos los dispositivos Android por defecto.

Google play también ofrece un servicio de actualización, por ejemplo, si un programador sube una versión nueva de su aplicación, este servicio le notificará a los usuarios existentes que hay una actualización disponible y permitirá que la instalen.

Google play solía ser llamado Android Market.

Herramientas de desarrollo

Qué son las herramientas de desarrollo?

Android ofrece un IDE específico para desarrollar llamado Android Studio el cual usaremos durante el curso ya que es la herramienta más apropiada.

Dalvik Virtual Machine

El sistema Android usa una máquina virtual especial conocida como máquina virtual Dalvik, para ejecutar aplicaciones basadas en Java. Dalvik utiliza un formato de código de bytes propio que es diferente de bytecode de Java.

Por lo tanto, no se puede ejecutar directamente archivos de clase Java en Android.

Cómo desarrollar aplicaciones Android

Las aplicaciones para Android son en su mayoría de las veces desarrolladas en el lenguaje de programación Java.

Android proporciona una herramienta llamada “dx" que convierte los archivos de clase Java en archivos dex (Dalvik Executable). Es decir, todos los archivos de clase de una aplicación se colocan en un archivo comprimido dex. Durante este proceso de conversión de la información redundante en los archivos de clase se optimizan en el archivo .dex. Por ejemplo, si la misma cadena de caracteres se encuentra en dos archivos de clases diferentes, el archivo .dex contiene una sola vez de referencia de esta cadena.

Estos archivos dex son por lo tanto de mucho menor tamaño que los archivos de las clases correspondientes.

Los archivos Dex y los recursos de un proyecto de Android, por ejemplo, las imágenes y archivos XML, se empaquetan en un archivo. apk (Paquete de Android). El programa de AAPT (Activo Android Packaging Tool) realiza este empaquetamiento.

La resultante .apk contiene todos los datos necesarios para ejecutar la aplicación Android, sería el análogo a un archivo con extensión .exe de Windows.

Instalación

Android Studio

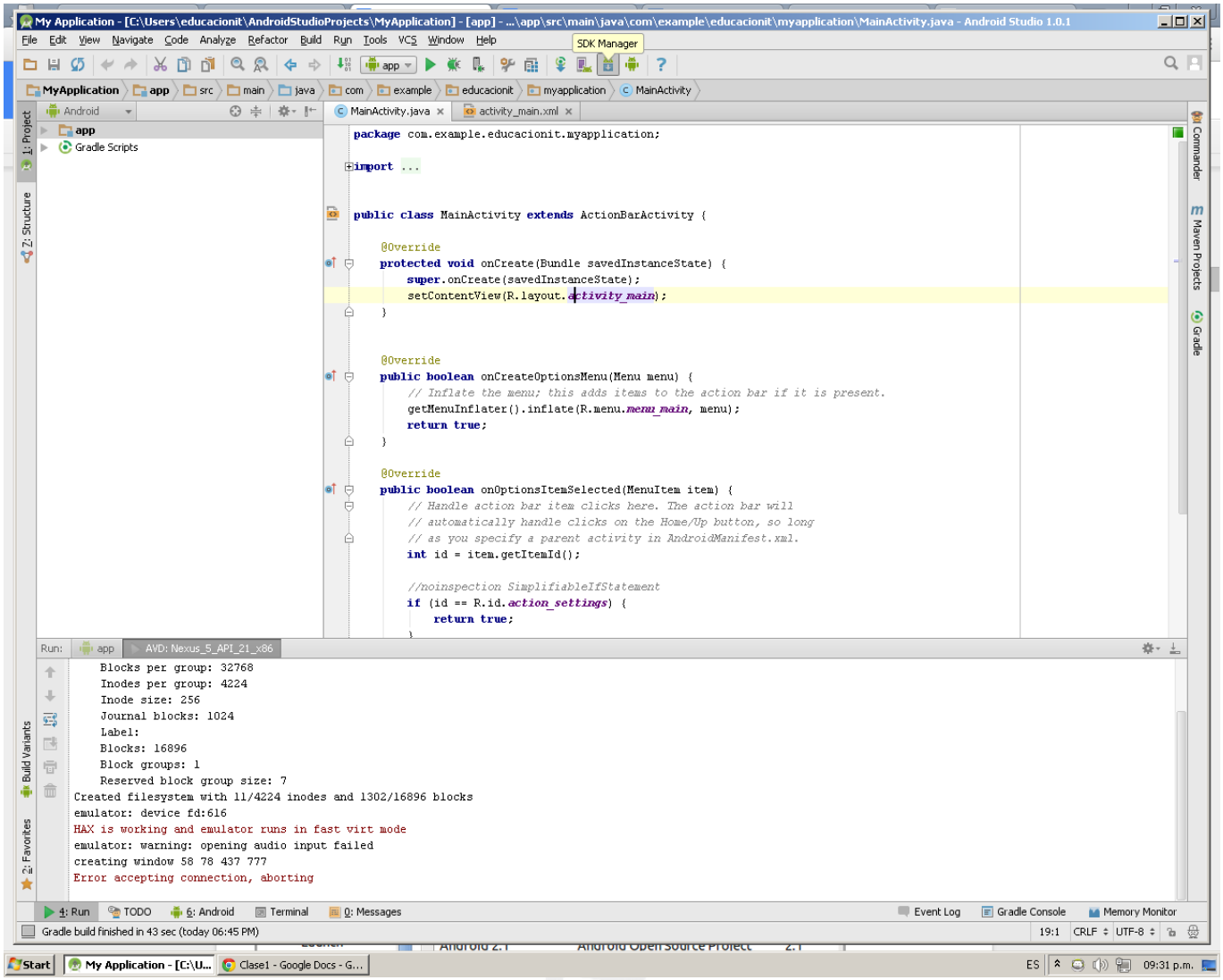
Lo siguiente asume que se conoce como instalar Java.

En la siguiente URL se puede bajar el Android Studio junto a la SDK de Android.

https://developer.android.com/sdk/index.html

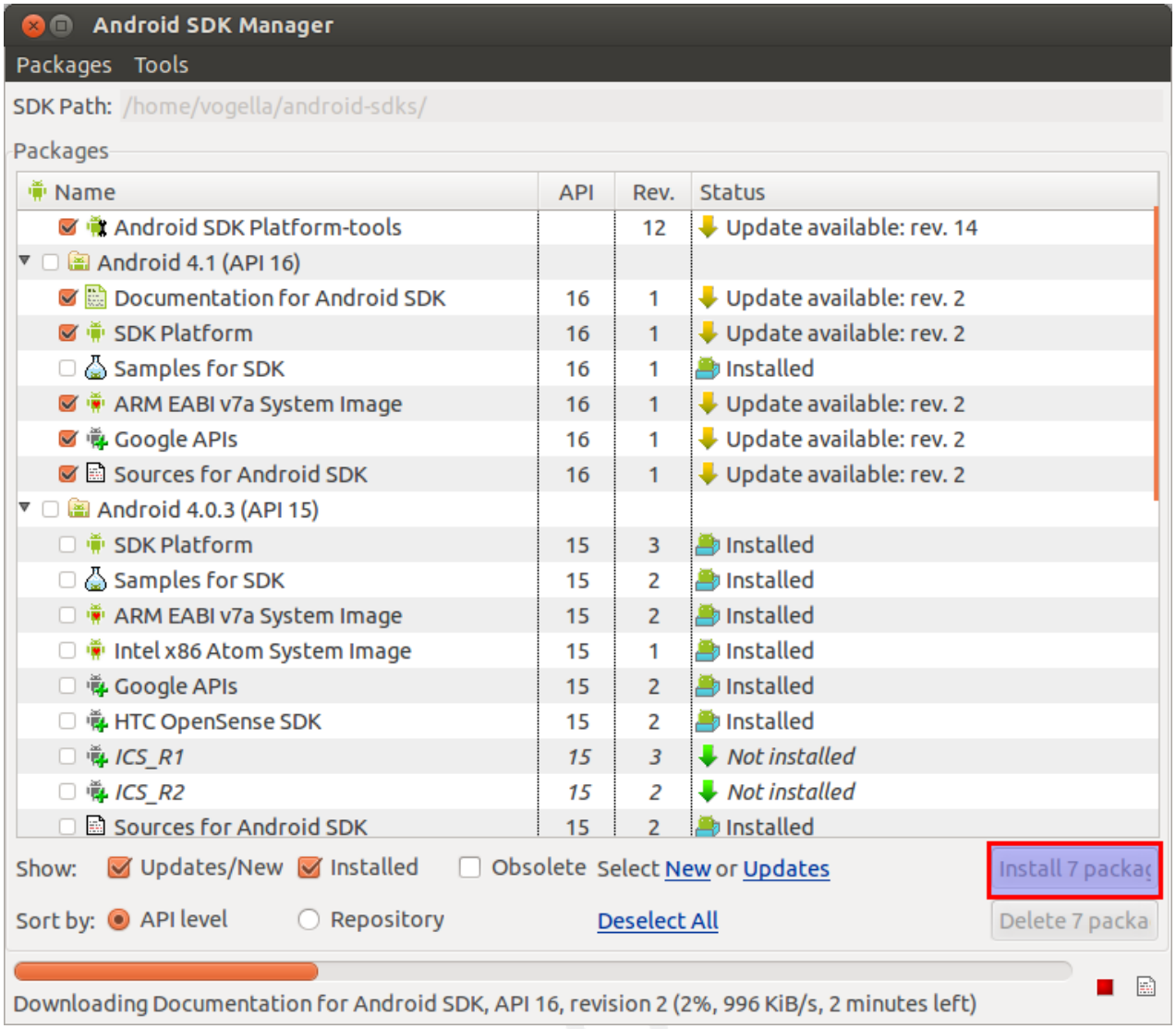
Instalación de una versión específica de Android

El Administrador de Android SDK permite instalar versiones específicas de Android. Clickee sobre el ícono de SDK Manager.



El cuadro de diálogo le permite instalar y eliminar paquetes.

Seleccione los Available packages y abrir Third party components. Seleccione la última API de Google del SDK y pulse el botón Instalar.



Pulse el botón Instalar y confirmar la licencia para todos los paquetes. Después de la instalación, reinicie Eclipse.

Código fuente

Durante el desarrollo de Android, es muy útil tener el código fuente disponible. El mismo se puede descargar a través del Administrador Android SDK mediante la selección de source code de Android SDK.

Los códigos fuente se descargan en el directorio fuente situado en path_to_android_sdk / fuentes / android-xx. xx es el nivel de la API de Android, por ejemplo, 15 para la versión Android 4.0.4.

Después podremos navegar dentro del código fuente.

Dispositivo virtual - Emulador (AVD)

¿Qué es?

Las herramientas de desarrollo de Android incluyen un emulador para ejecutar un sistema Android. El emulador se comporta como un verdadero dispositivo Android (en la mayoría de los casos) y le permite probar la aplicación sin

necesidad de un dispositivo real.

Puede configurar la versión del sistema Android que le gusta correr, el tamaño de la tarjeta SD, la resolución de pantalla y otros ajustes pertinentes. Se pueden definir varios de ellos con diferentes configuraciones.

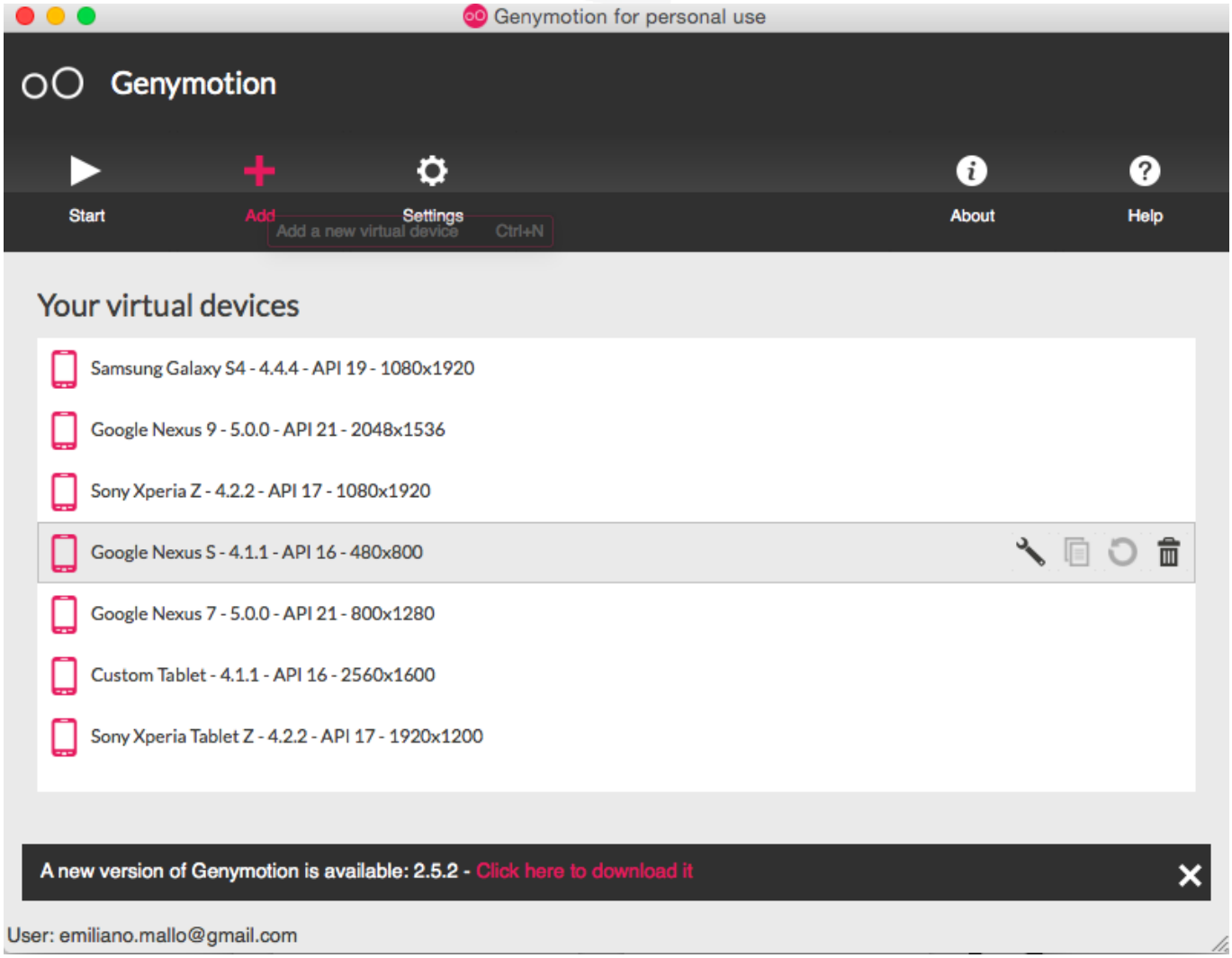
Estos dispositivos se denominan dispositivos Android Virtuales y puede comenzar varios en paralelo.

Recomendamos el uso de Genymotion, el cual se baja del siguiente link(creando previamente una cuenta en la web):

https://www.genymotion.com

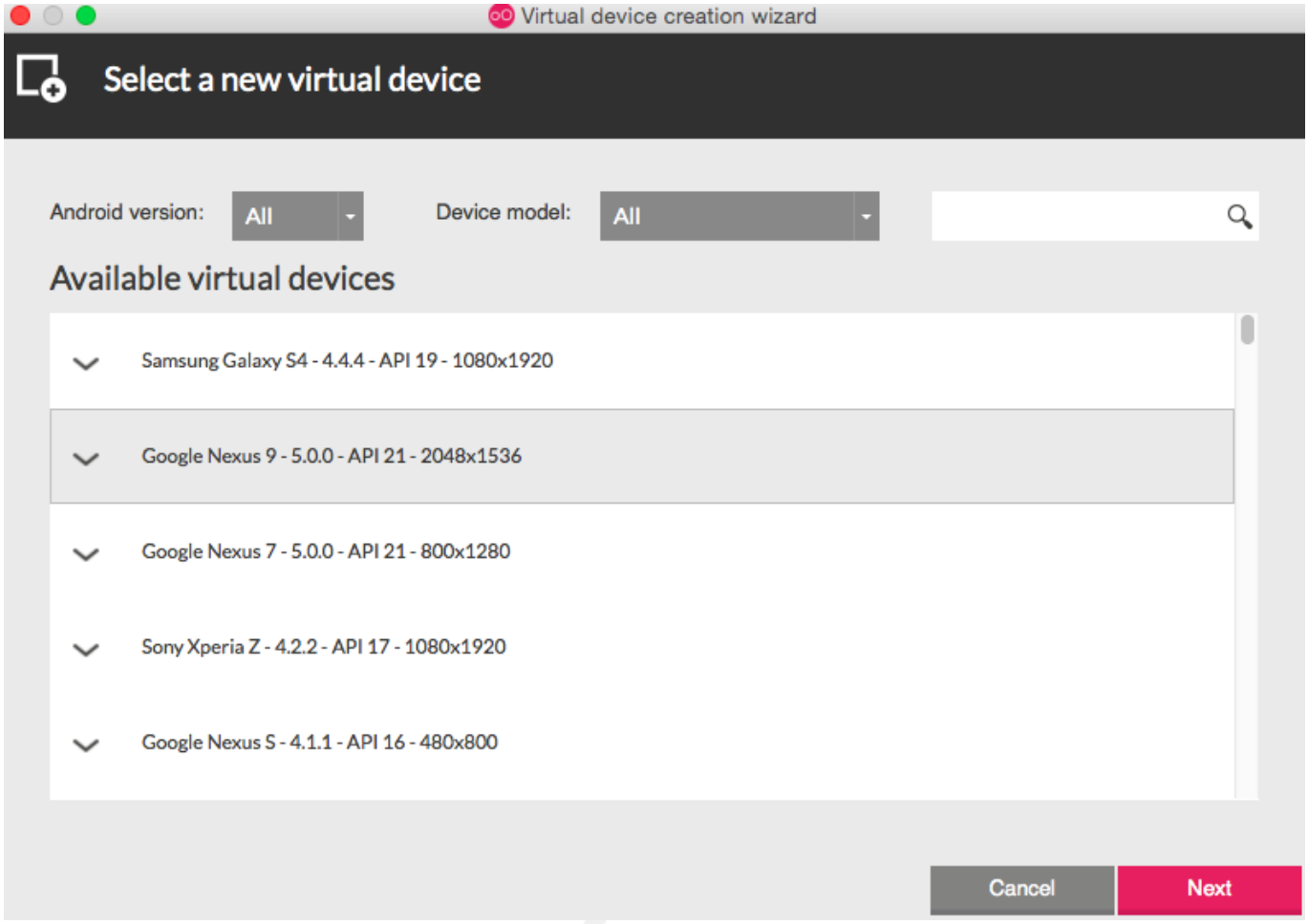
Este programa sirve para emular estos dispositivos de Android, estos dispositivos son máquinas virtuales las cuales se deben descargar y hay muchísima diversidad de las mismas.

Una vez instalado el Genymotion y corriendo, podemos descargar estas máquinas virtuales de la siguiente manera:



(en su caso aparecerá el listado vacío)

y aparecerá la siguiente pantalla para escoger dispositivos (nosotros estamos seleccionando el Google Nexus S para descargar):

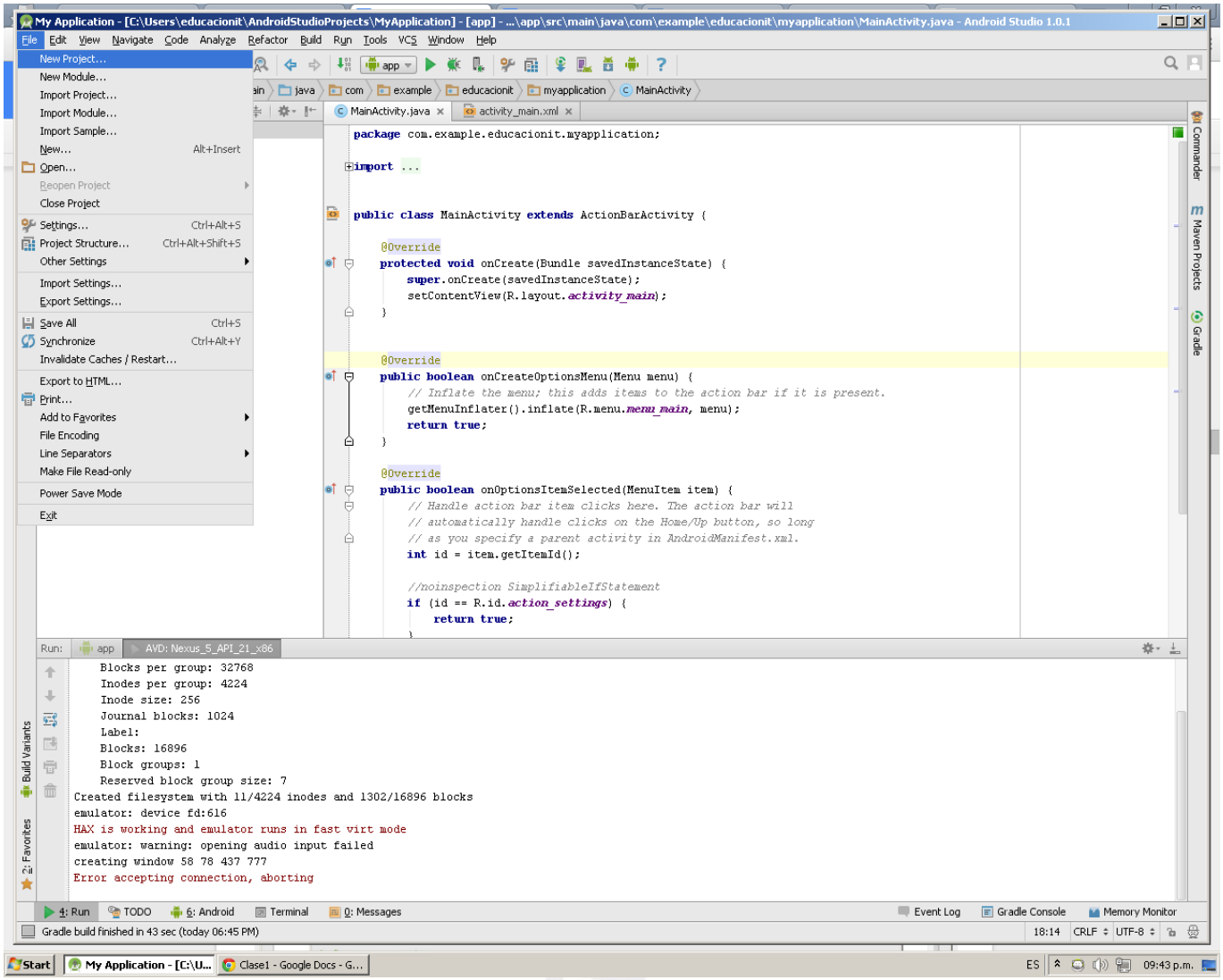


Creación de una aplicación Android

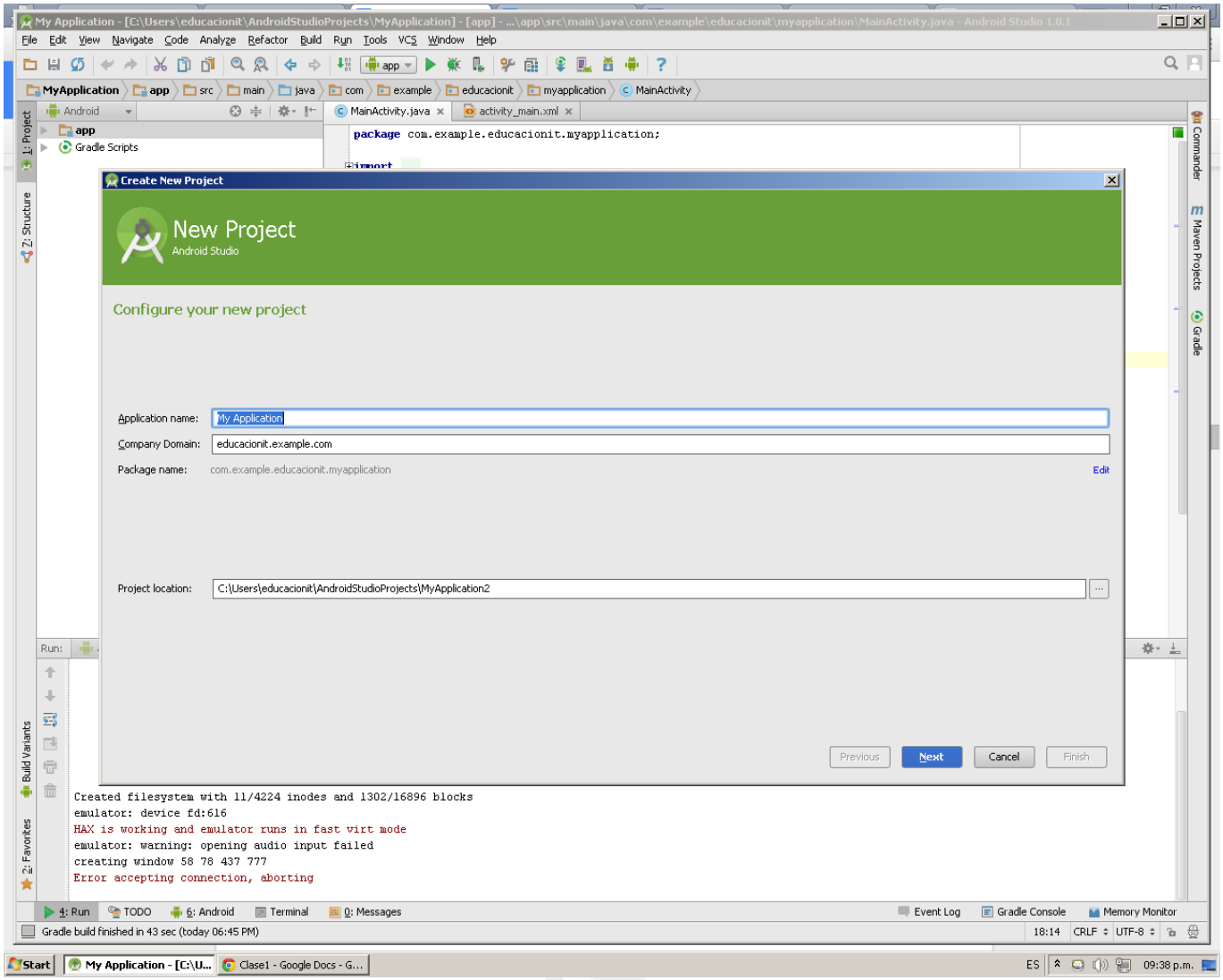
Primera aplicación Android

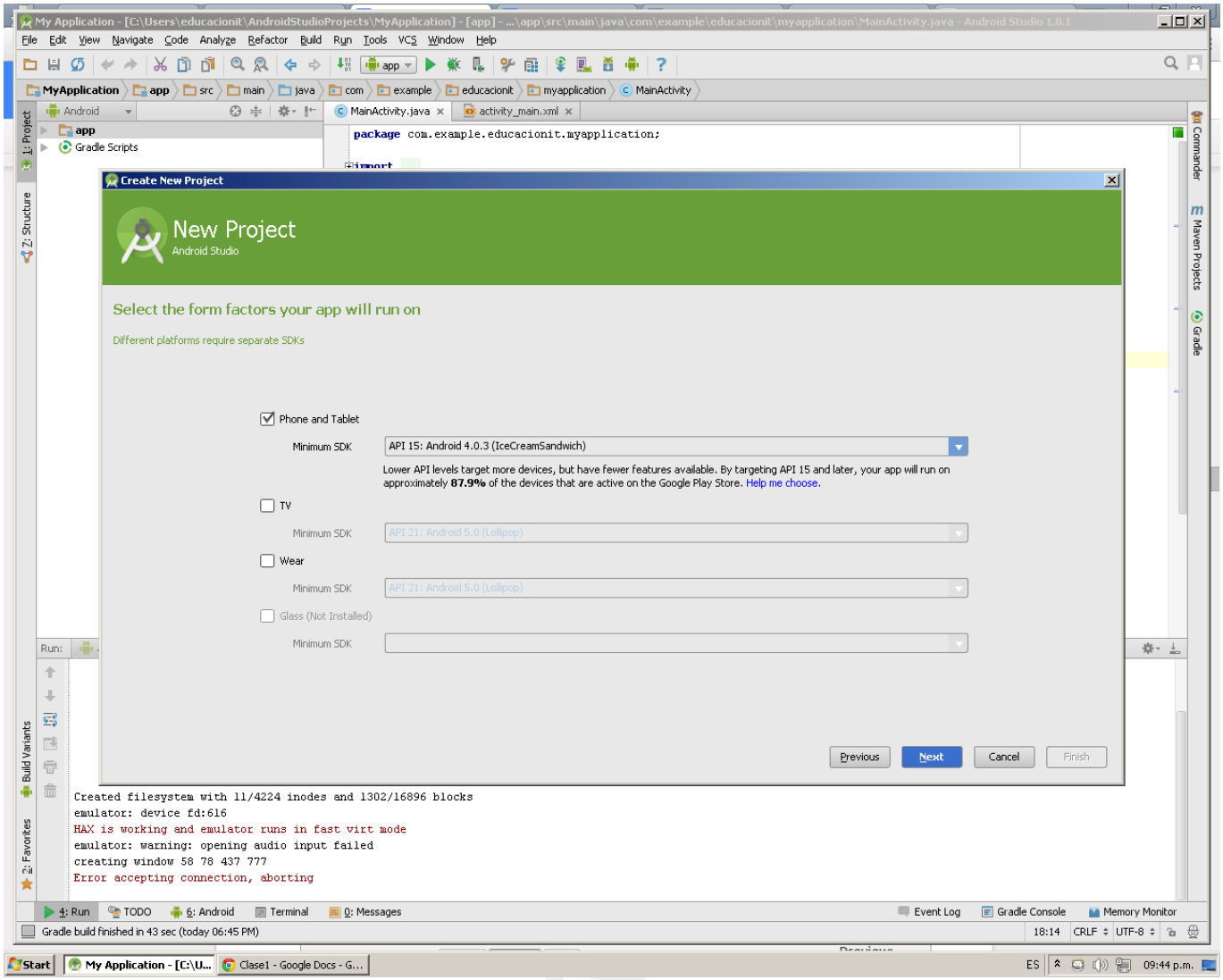
Creación del proyecto

Seleccione File → New Project para crear un nuevo proyecto Android.



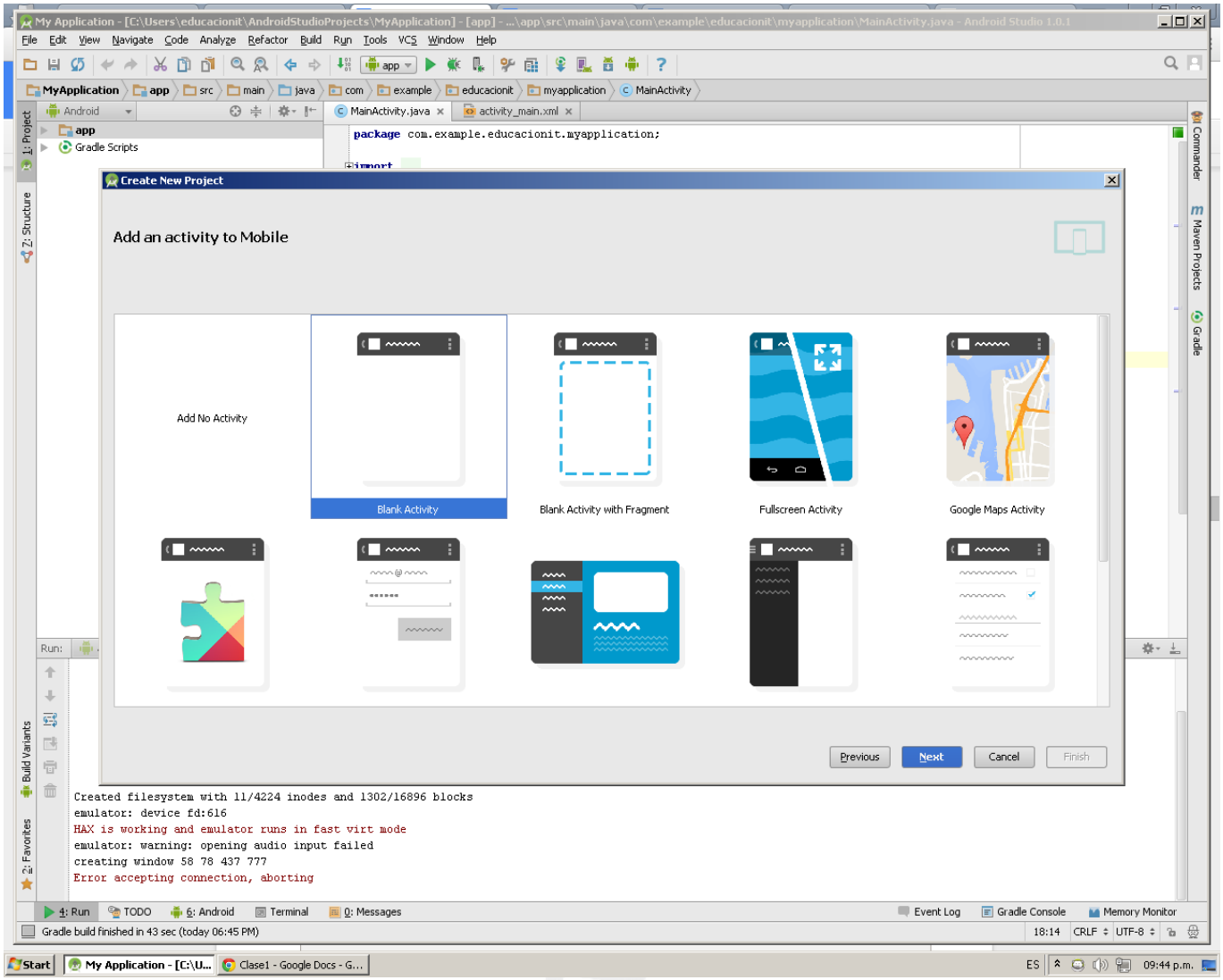
Indique el nombre del proyecto y presione Next.


My Application - [C:\Users\educacionit\AndroidStudioProjects\MyApplication] - [app] - ...\app\src\main\java\com\example\educacionit\myapplication\MainActivity.java - Android Studio 1.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication app src main java com example educacionit myapplication MainActivity
Android
app
Gradle Scripts
MainActivity.java
activity_main.xml
package com.example.educacionit.myapplication;
import
Create New Project
New Project
Android Studio
Configure your new project
Application name: My Application
Company Domain: educacionit.example.com
Package name: com.example.educacionit.myapplication
Edit
Project location: C:\Users\educacionit\AndroidStudioProjects\MyApplication2
Previous
Next
Cancel
Finish
Run:
Created filesystem with 11/4224 inodes and 1302/16896 blocks
emulator: device fd:616
HAX is working and emulator runs in fast virt mode
emulator: warning: opening audio input failed
creating window 58 78 437 777
Error accepting connection, aborting
4: Run
TODO
6: Android
Terminal
0: Messages
Event Log
Gradle Console
Memory Monitor
Gradle build finished in 43 sec (today 06:45 PM)
18:14
CRLF
UTF-8
Start
My Application - [C:\U...
Clase1 - Google Docs - G...
ES
09:38 p.m.

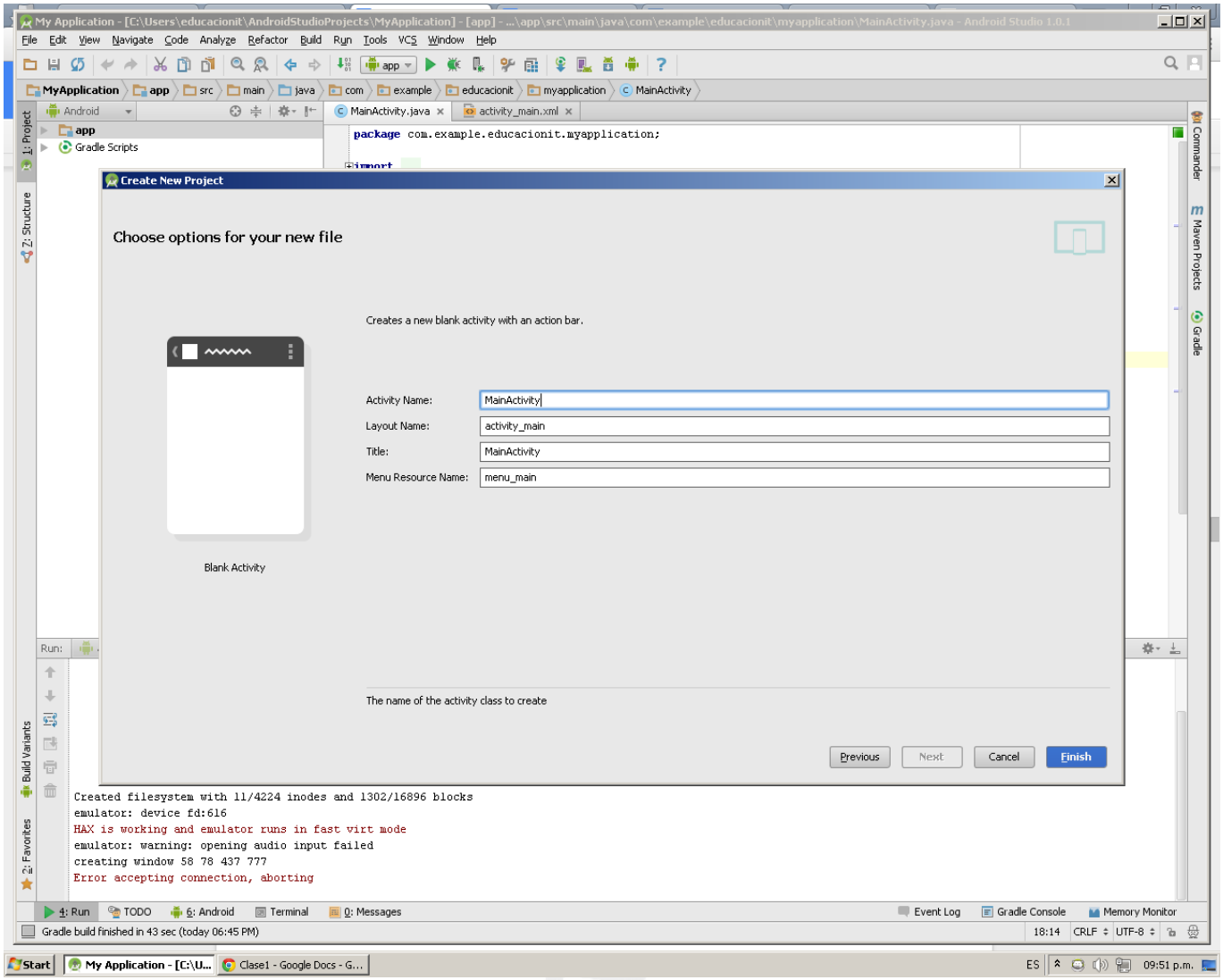


Presione Next.

Seleccionar Blank Activity y clickear Next.

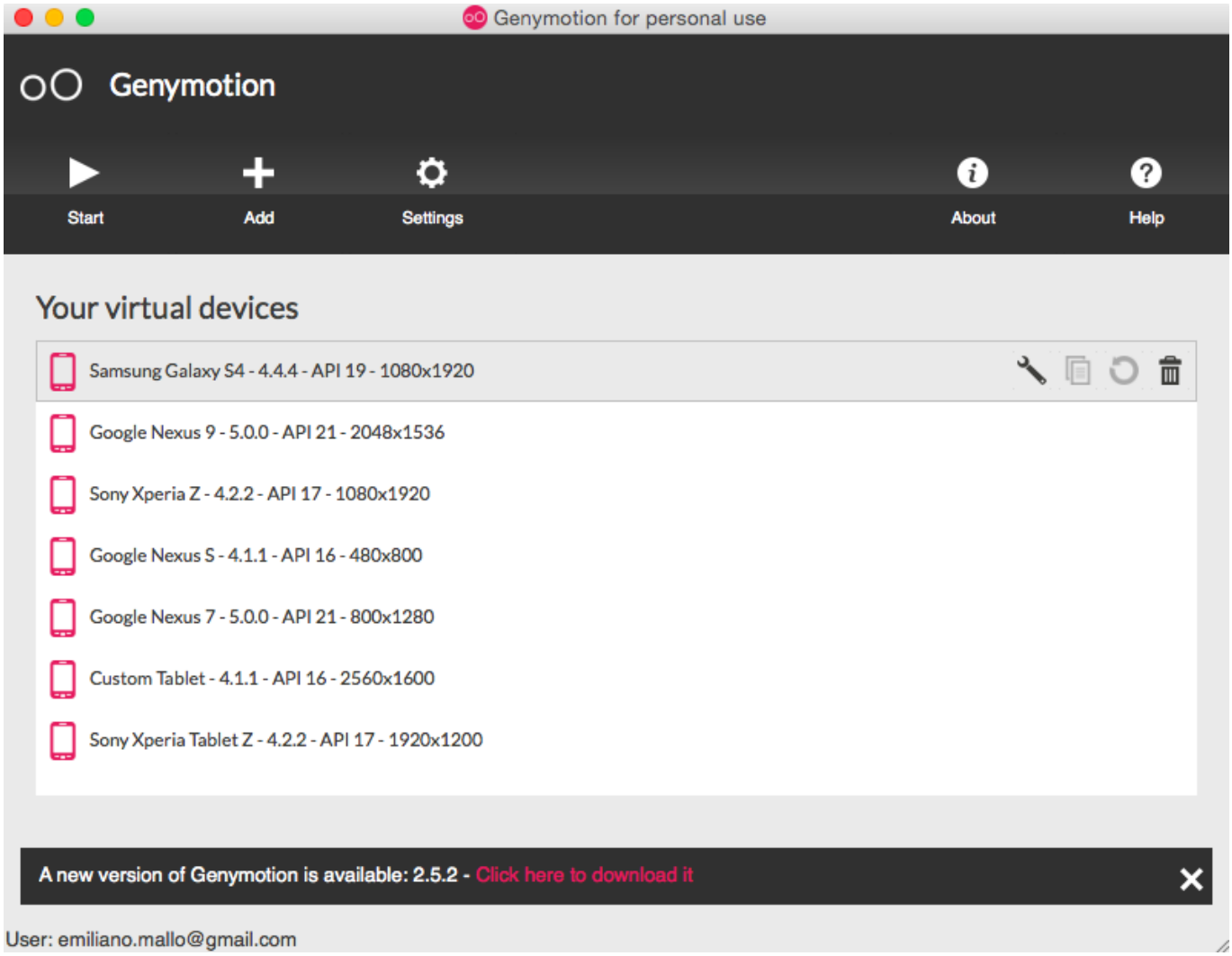


Seleccione la plantilla BlankActivity y pulse el botón Siguiente.



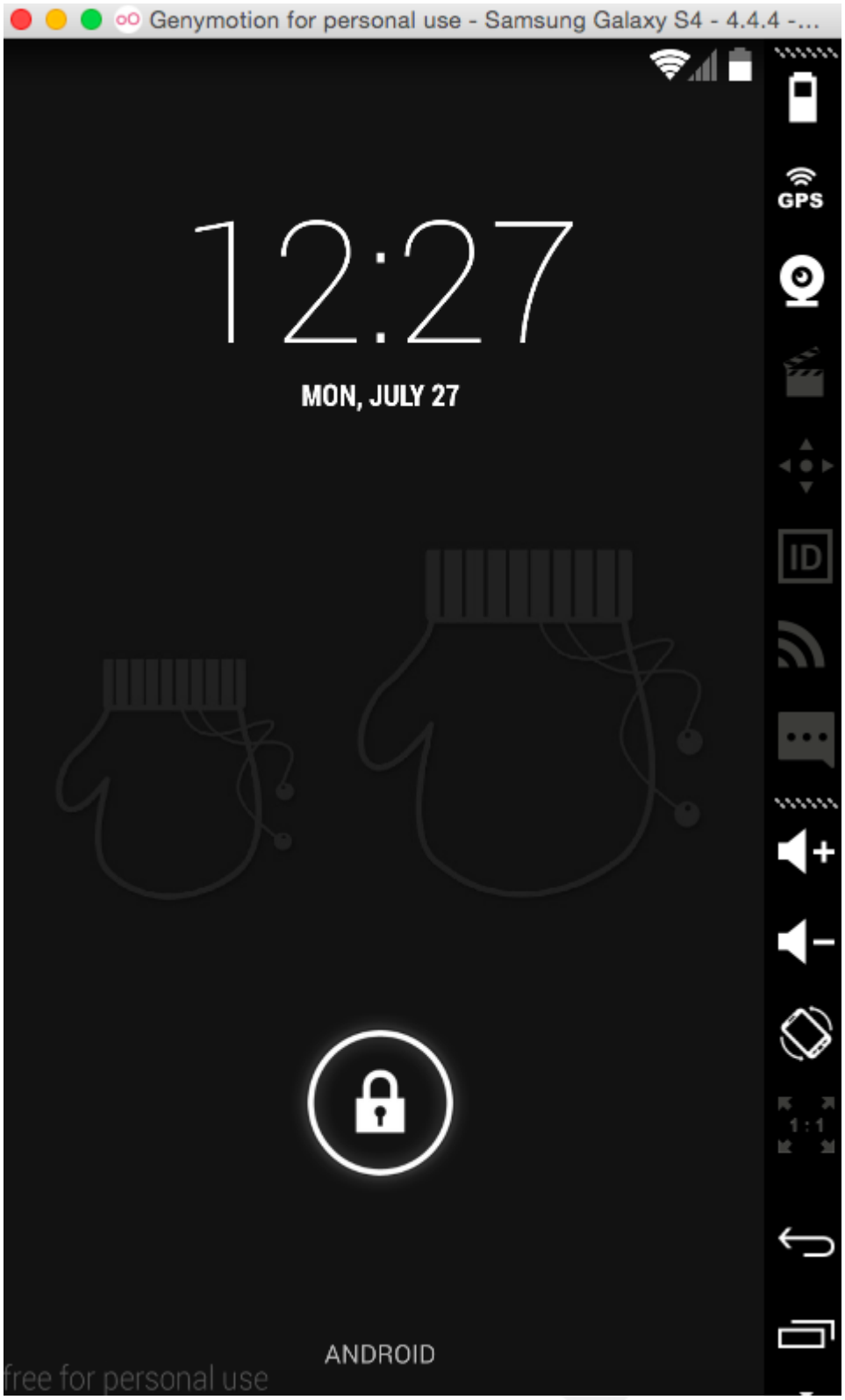
Correr nuestra primera aplicación con Genymotion y Android Studio

Para correr nuestra primera aplicación una vez creada abriremos el genymotion y visualizaremos lo siguiente:



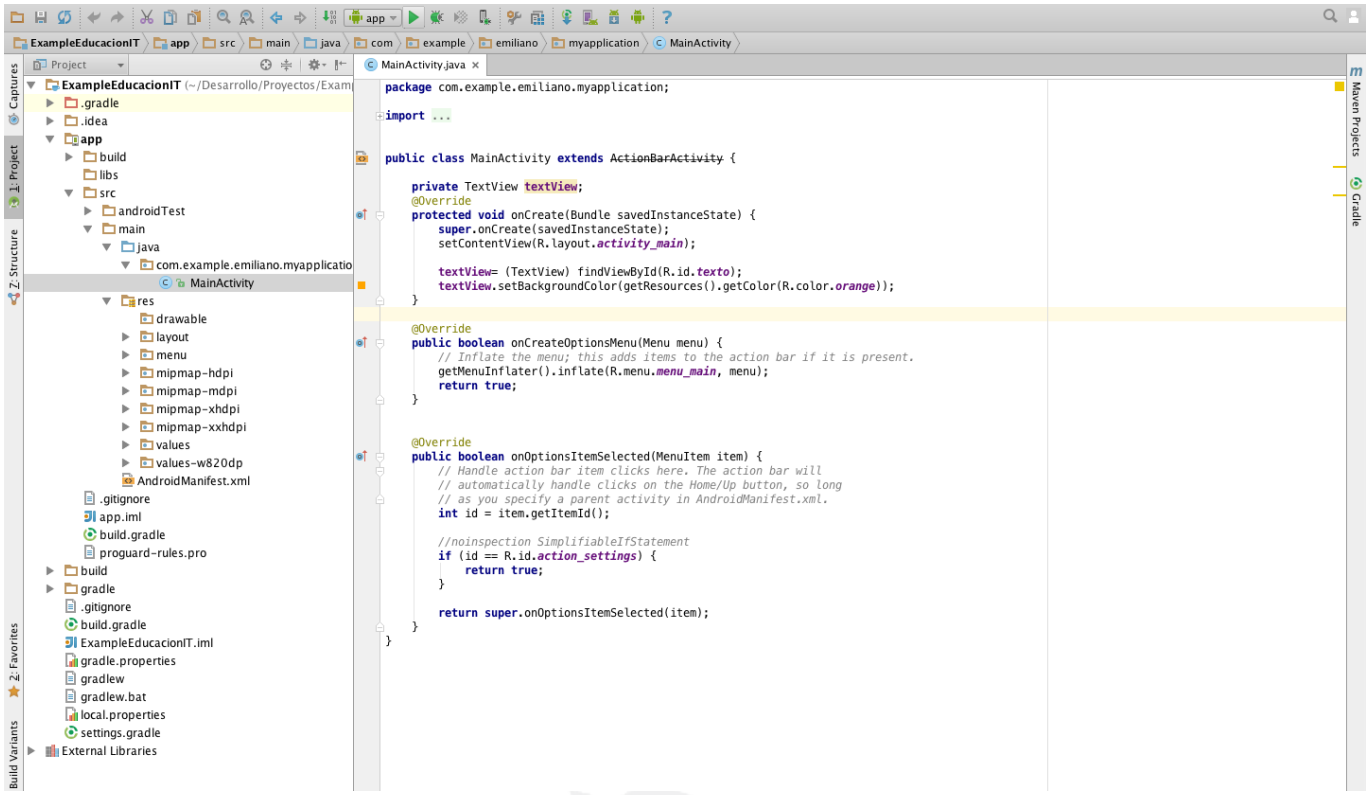
En esta pantalla seleccionaremos el dispositivo que querramos usar para correr la aplicación (en mi caso está Samsung Galaxy S4 y tocaremos el botón de Start que se ve en la zona superior del programa.

Y veremos que se abrirá lo siguiente:

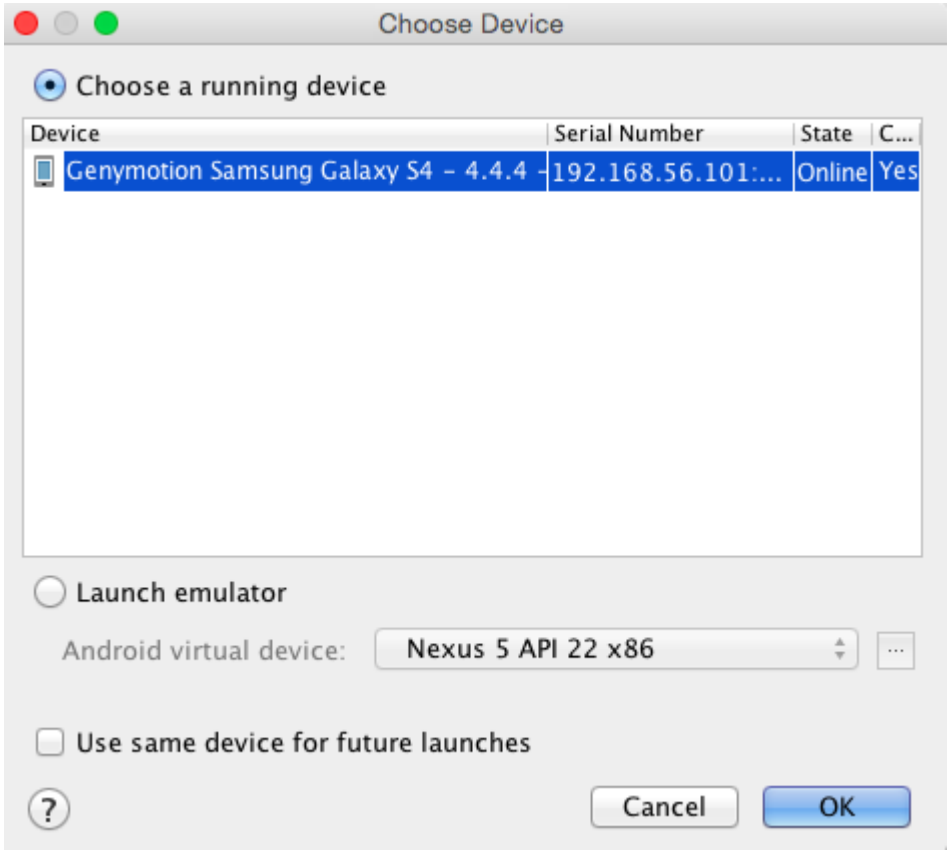


Esto significa que el emulador se inició correctamente! y ya estamos listos para correr la aplicación desde Android Studio.

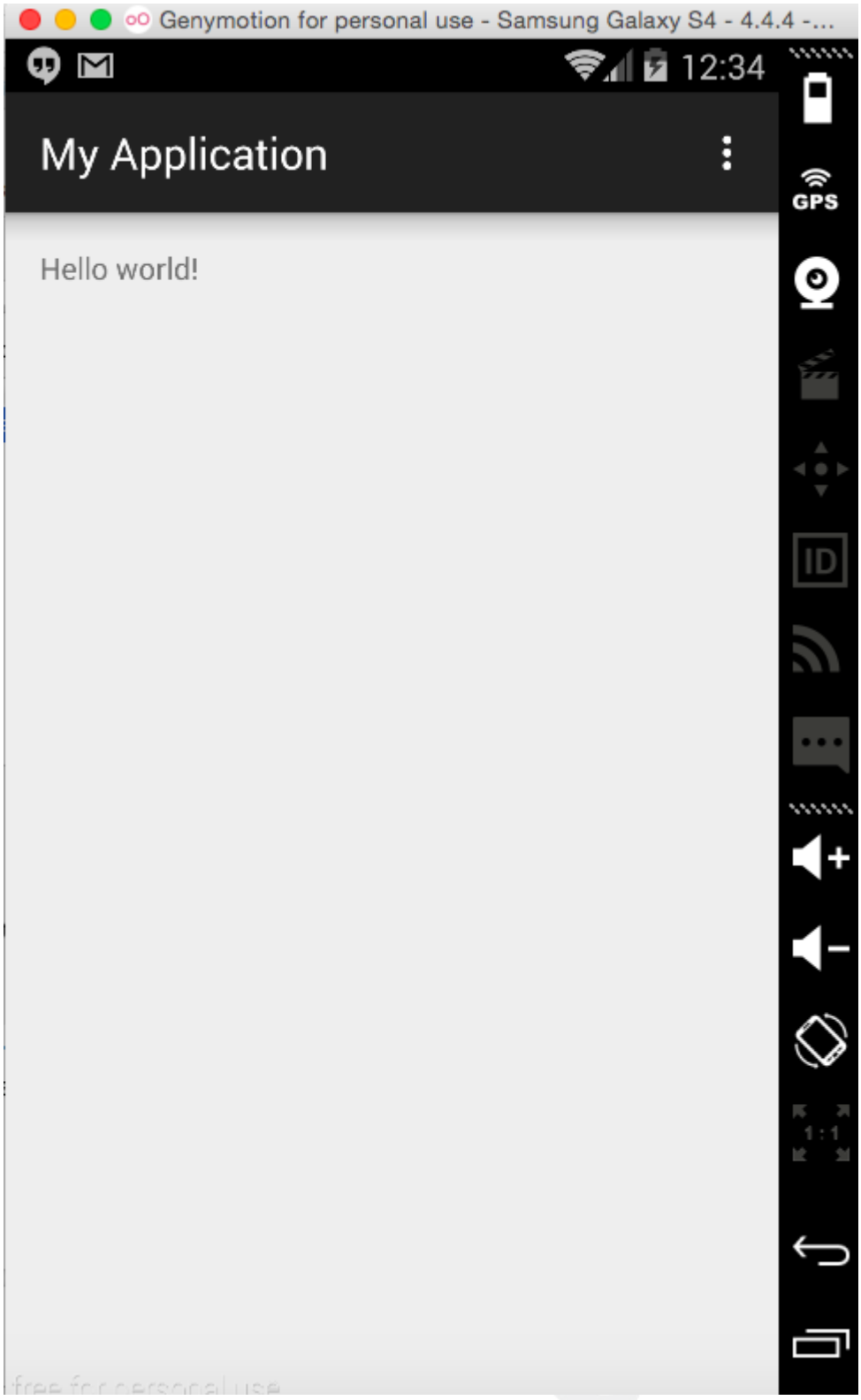
Vamos al Android Studio y presionamos el botón de play verde que se ve en la barra de tareas del programa:



Esto hará que nos aparezca este cartel luego de unos instantes:



Este cartel es para escoger un emulador o dispositivo para correr el programa, en este caso nos detectó el dispositivo que iniciamos desde genymotion asi que presionaremos el botón "OK".



Y ya hemos creado nuestra primer aplicación de Android!!!

Activity

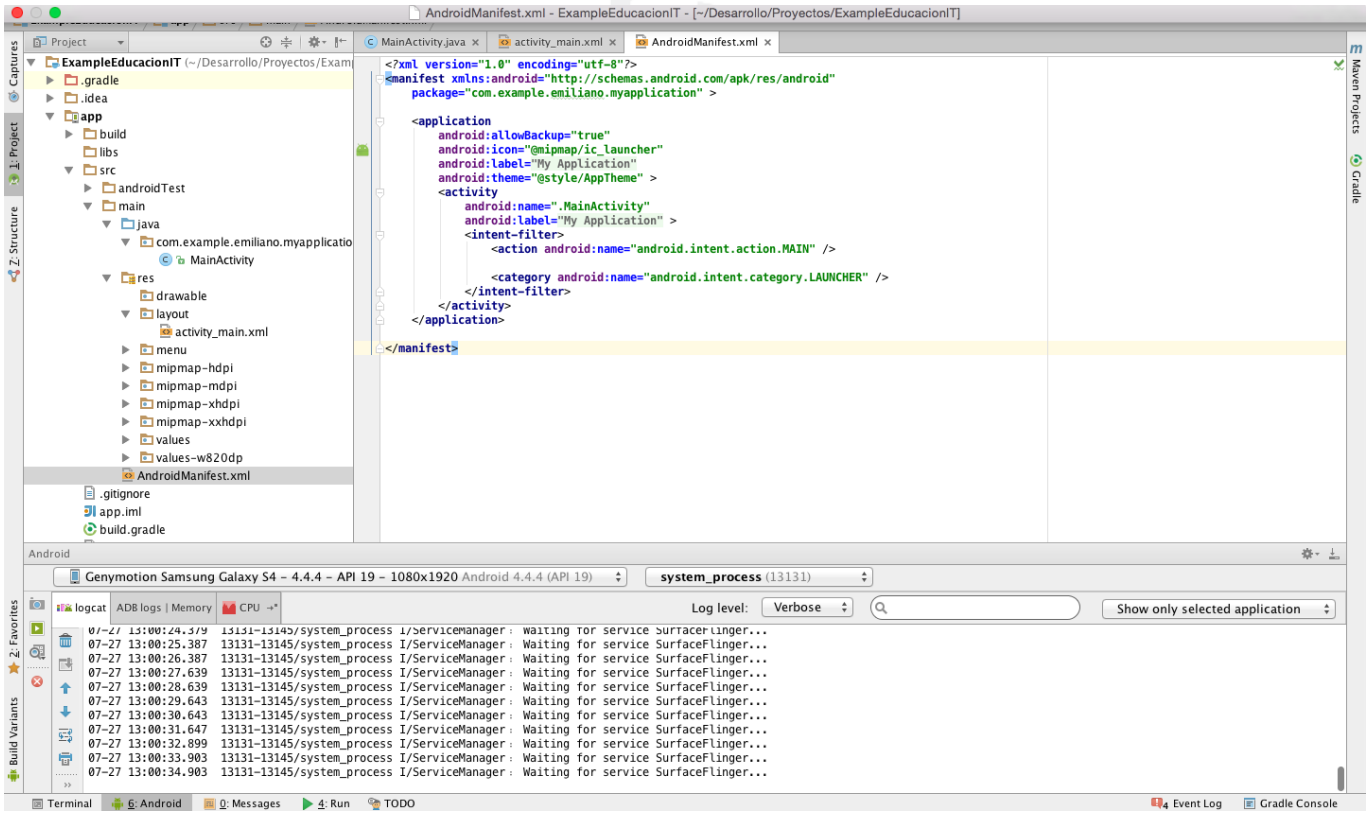
Un Activity está enfocado a lo que el usuario puede hacer en una pantalla. Casi todos los activities deben interactuar con el usuario, por lo que el activity se encarga de crear una ventana para que puedas poner una interfaz de usuario a través del método: setContentView.

Si bien los activities se presentan a menudo al usuario como ventanas a pantalla completa, también pueden ser utilizados en otras formas: como ventanas flotantes o incrustados en el interior de otro activity (usando ActivityGroup). Hay dos métodos que casi todas las subclases de Activity implementarán:

onCreate (Bundle): Este método se ejecuta al crearse el activity. Lo más normal es que llame setContentView (int) con un recurso de diseño que define la interfaz de usuario, y el uso de findViewById (int) para recuperar los widgets en que la interfaz de usuario que necesites para interactuar dinamicamente (en tiempo de ejecución).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Todo Activity debe ser declarado en el archivo Android Manifest de la siguiente manera:



Cada activity tiene que tener una vista asociada (en este caso es activity_main), que podemos encontrarlo en activity_main.xml y tiene el siguiente contenido:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:id="@+id/texto" android:text="Hello world!" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Estas vistas se encuentra dentro de la carpeta res/layout/

AndroidManifest.xml

Los componentes y la configuración de una aplicación Android se describen en el archivo AndroidManifest.xml. Por ejemplo, todos los Activities deben ser declarados en el archivo.

También debe contener los permisos necesarios para la aplicación. Por ejemplo, si la aplicación requiere acceso a la red se debe especificar aquí.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ar.com.educacionit.android"
    android:versionCode="1"
    android:versionName="1.0">
   <application android:icon="@drawable/icon" android:label="@string/app_name">
      <activity android:name=".Convertir"


      android:label="@string/app_name">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>

   </application>
   <uses-sdk android:minSdkVersion="9" />

</manifest>
```

El atributo package define el paquete de base para los objetos Java a que se refiere en este archivo. Si un archivo de Java se encuentra dentro de un paquete diferente, se debe declarar con el nombre del paquete completo.

Google Play requiere que todas las aplicaciones de Android tengan su paquete único. Por lo tanto, es una buena práctica usar el nombre de dominio inverso al nombre del paquete. Esto evitará las colisiones con otras aplicaciones de Android cuando se quiera subir la aplicación al Google Play.

Android: versionName y android: versionCode especifican la versión de su aplicación. versionName es lo que el usuario ve y puede ser cualquier string.

VersionCode debe ser un número entero. El Google Play determinará sobre la base de la versionCode, si se debe realizar una actualización de las aplicaciones para las instalaciones existentes. Por lo general, comienzan con "1" y luego se debe aumentar el valor en uno a medida que se van subiendo nuevas versiones de la aplicación.

El tag <activity> define un Activity, en este ejemplo apunta a la clase Convertir en el paquete ar.com.educacionit.android. Lo que se encuentra dentro del "intent-filter" declara que el Activity se iniciará cuando la aplicación se lance.

El valor de la cadena de @ / app_name se refiere a una cadena de caracteres, esto se explicará posteriormente cuando se vea el archivo de constantes "strings".

El "uses-sdk" define la mínima versión del SDK para los cuales la aplicación será válida. Esto evitará que su aplicación se instale en los dispositivos con versiones anteriores de SDK.

R.java y Resources

El directorio gen de en un proyecto de Android se encuentran los valores generados. R.java es una clase generada que contiene referencias a ciertos recursos del proyecto.

Estos recursos deben ser definidos en el directorio "res" y pueden ser archivos XML, iconos o imágenes. Por ejemplo, puede definir valores, los menúes, presentaciones o animaciones a través de archivos XML.

Si creamos un nuevo recurso, la referencia correspondiente se crea automáticamente en R.java. Estas referencias son estáticas valores int y definir los ID de los recursos.

El sistema Android proporciona métodos para acceder a los recursos correspondientes a través de estos identificadores que se verán más adelante.

Vista XML (de manera estática)

Se dice que estas vistas denotadas en formato xml son estáticas ya que no pueden cambiar en tiempo de ejecución desde este formato.

Analizaremos el xml anteriormente mencionado:

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:id="@+id/texto" android:text="Hello world!" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

En esta vista podemos notar que existe un componente de tipo RelativeLayout como padre del resto de las vistas, y tiene un hijo, un componente de tipo TextView (el cual es un label común y corriente)

Cada tag de la vista se abre con el signo "<" continuado por el nombre del tipo del componente y se cierra con "/>".

Pero a diferencia del resto de las vistas los componentes de tipo ViewGroup (en este caso el RelativeLayout) permiten declarar vistas adentro de la misma por lo que se cierra recién luego de la última vista declarada con lo siguiente "<"continuado por el nombre del componente y luego ">" como se visualiza en el ejemplo.

Dentro de cada vista podemos ver atributos los cuales nos facilitarán la personalización de la misma o inclusive podremos declarar su tamaño como asi también su posición entre muchas otras utilidades. (más adelante se verá en profunidad la utilidad de los atributos principales).

Cómo conectar una vista hija desde un Activity?

Para esto los activities tienen un método llamado findViewById

Un ejemplo del uso de findViewById:

```java
private TextView textView;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    textView= (TextView) findViewById(R.id.texto);
}
```

Para que este ejemplo funcione lo que figura luego del R.id. (en este caso se llama texto) debe concordar con el id de un TextView que se encuentre dentro de activity_main.xml como se ve anteriormente.

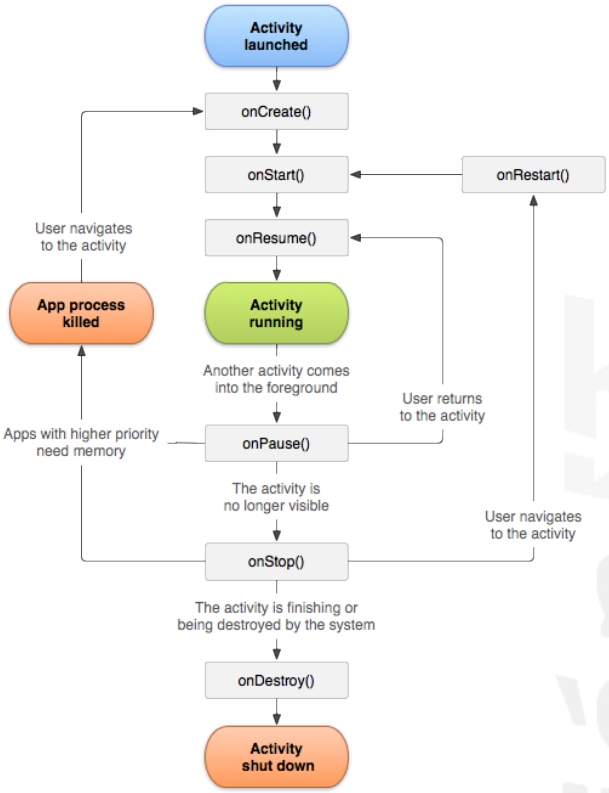
Una vez teniendo la vista linkeada al activity podemos modificarle sus atributos en tiempo de ejecución lo cual nos brinda una increíble flexibilidad

onResume: Este método se ejecuta cada vez que el activity se vuelve a mostrar en pantalla.

Ciclo de vida del Activity

Ciclo de vida, es una secuencia de métodos que se van ejecutando del activity a medida que suceden eventos.

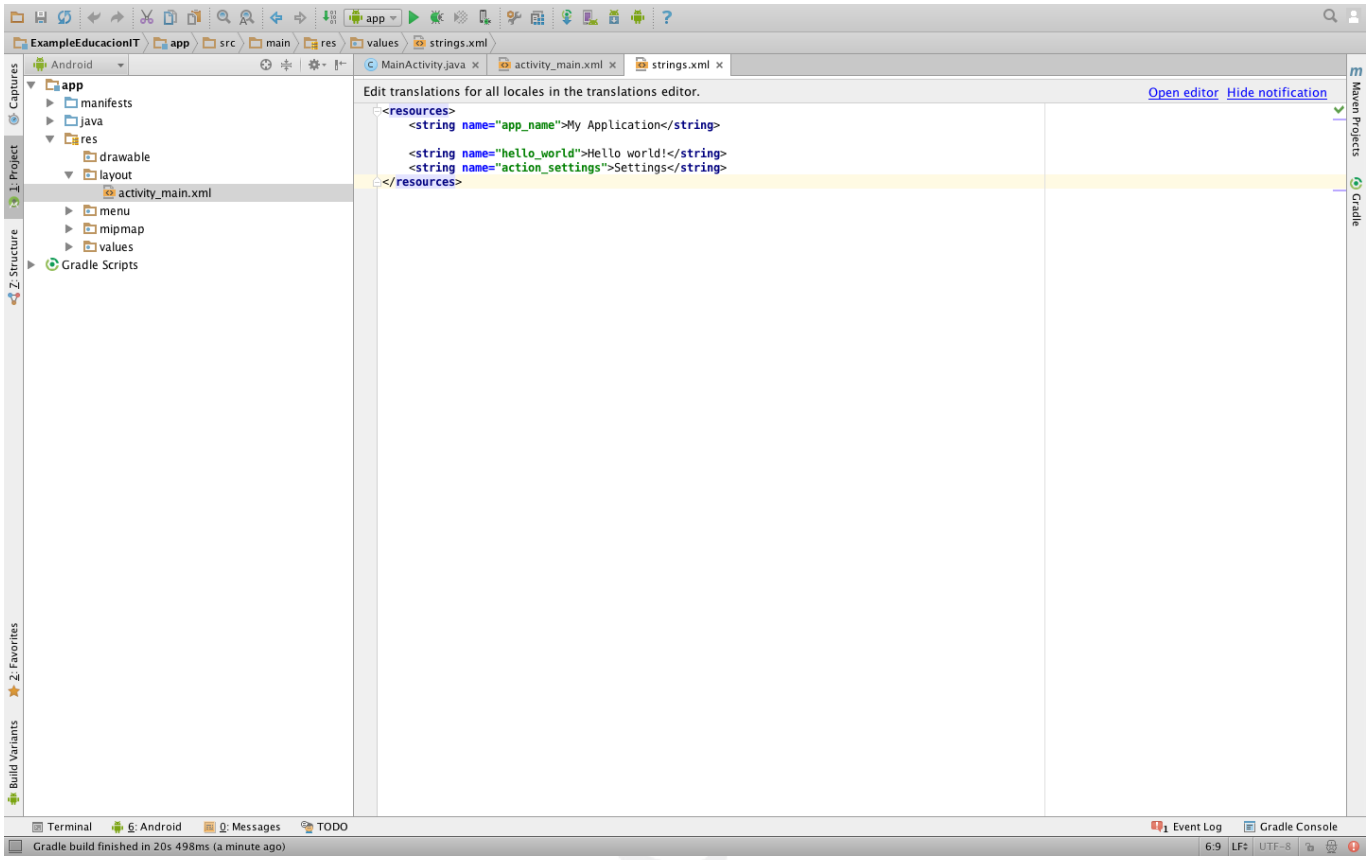
Se denota claramente en la siguiente imagen:



Crear constantes String

Android permite crear atributos estáticos, por ejemplo, String. Estos atributos pueden utilizarse por ejemplo en archivos XML o de diseño a que se refiere el código fuente a través de Java.

Seleccione el archivo strings.xml y verá lo siguiente:



Acá se puede notar que hay varias constantes de string, y esto nos va a servir a futuro para internacionalizar la aplicación en caso de ser necesario.

En xml se utiliza de la siguiente forma:

```
<TextView android:id="@+id/texto" android:text="@string/hello_world" android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Desde java se utiliza de la siguiente forma:

```
    textView= (TextView) findViewById(R.id.texto);
    textView.setText(R.string.hello_world);
```

Crear constantes de colores

Buscar dentro de la carpeta values un archivo color.xml y si no existe crearlo de la siguiente forma:

```
<resources>
    <color name="orange">#ffa500</color>
</resources>
```

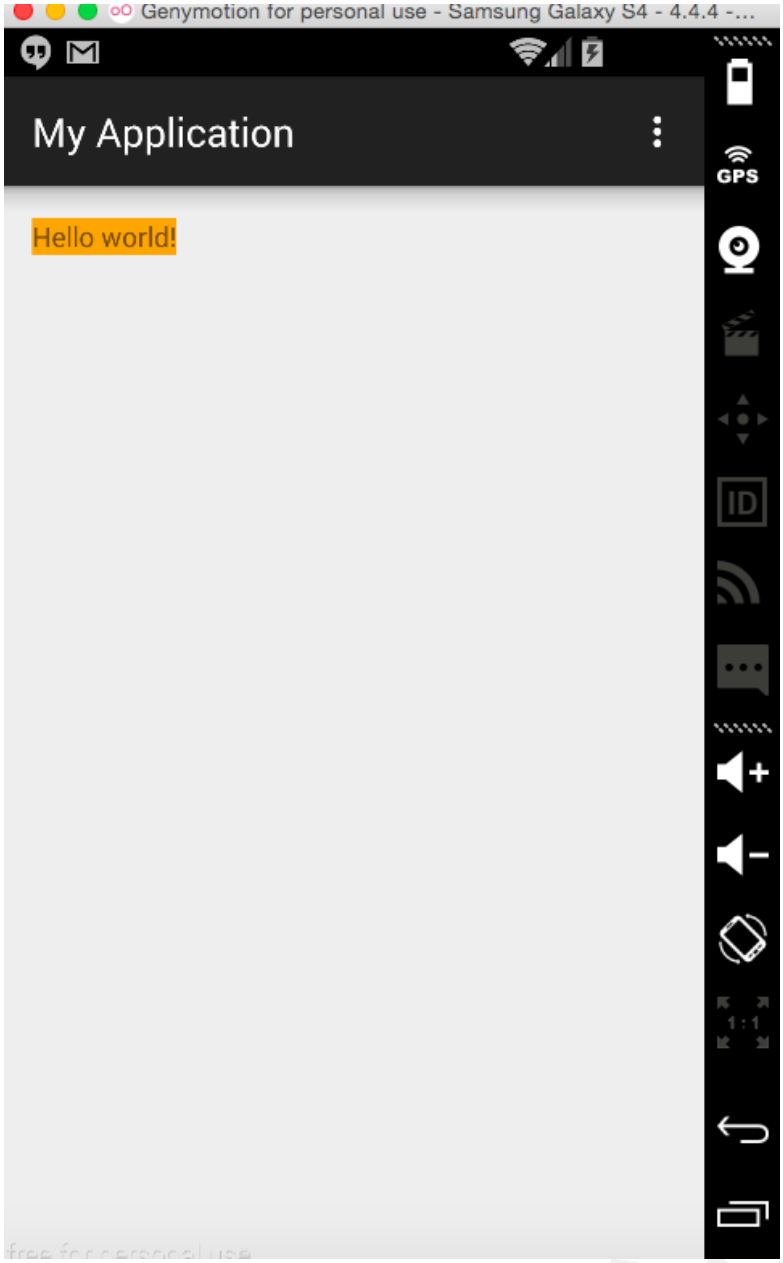
Para usarlo desde un xml:

```
<TextView android:id="@+id/texto" android:text="@string/hello_world" android:background="@color/orange" android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Para usarlo desde java:

```
textView= (TextView) findViewById(R.id.texto);
textView.setBackgroundColor(getResources().getColor(R.color.orange));
```

En ambos casos al correr se verá de la siguiente manera



Tomar el Click de un botón

Vamos a escribir en el activity lo siguiente:El par

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onClick(View view){
        Toast.makeText(this,"Click en el botón.",Toast.LENGTH_LONG).show();
    }
}
```

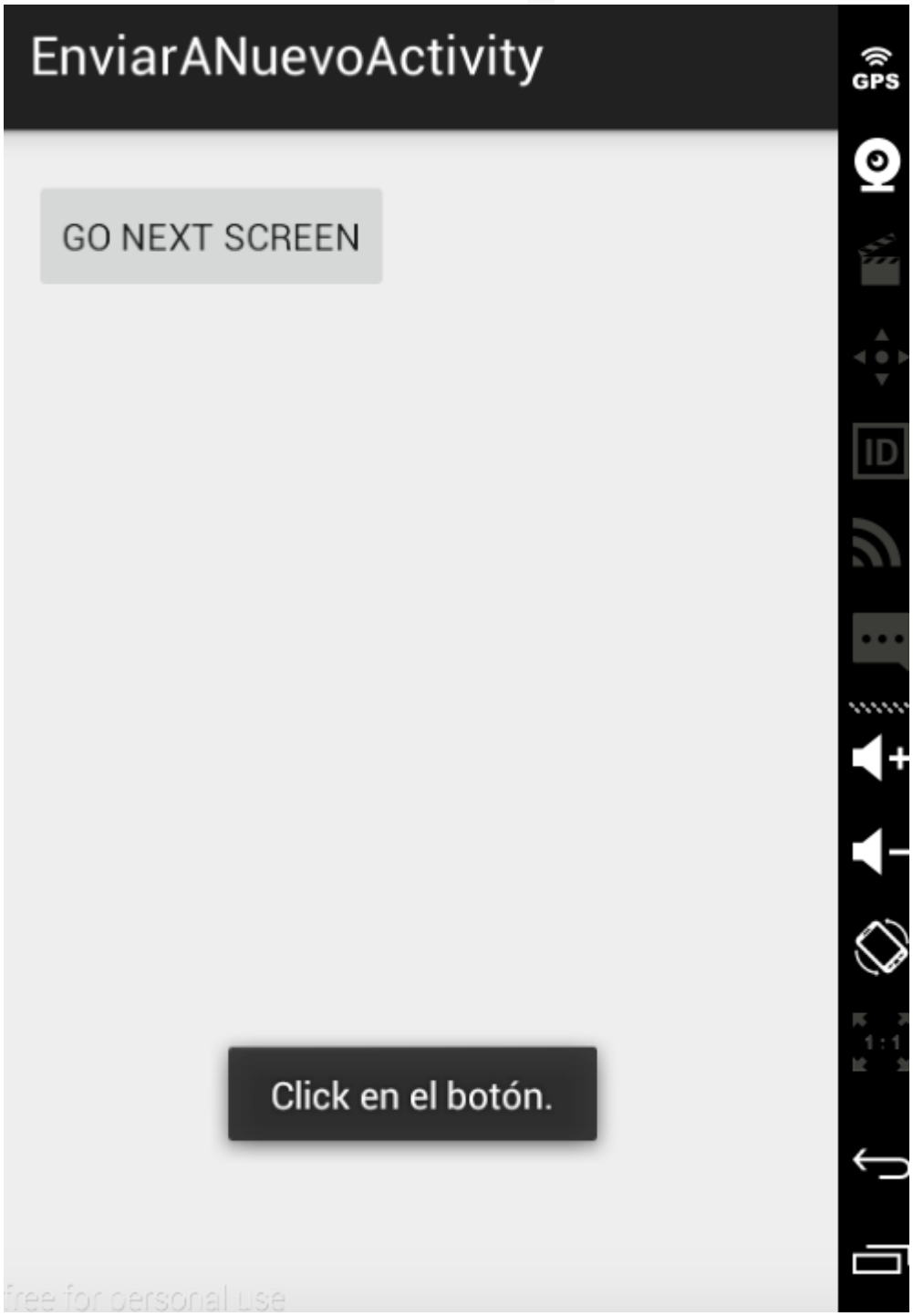
El parámetro que viene en el método onClick es el boton mismo.

Y en el xml lo siguiente:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Go Next Screen" android:onClick="onClick" />
```

En esto último estamos diciéndole a la vista que cuando el botón se clickea ejecute el método del activity llamado "onClick".

Al clickear el botón veremos lo siguiente:

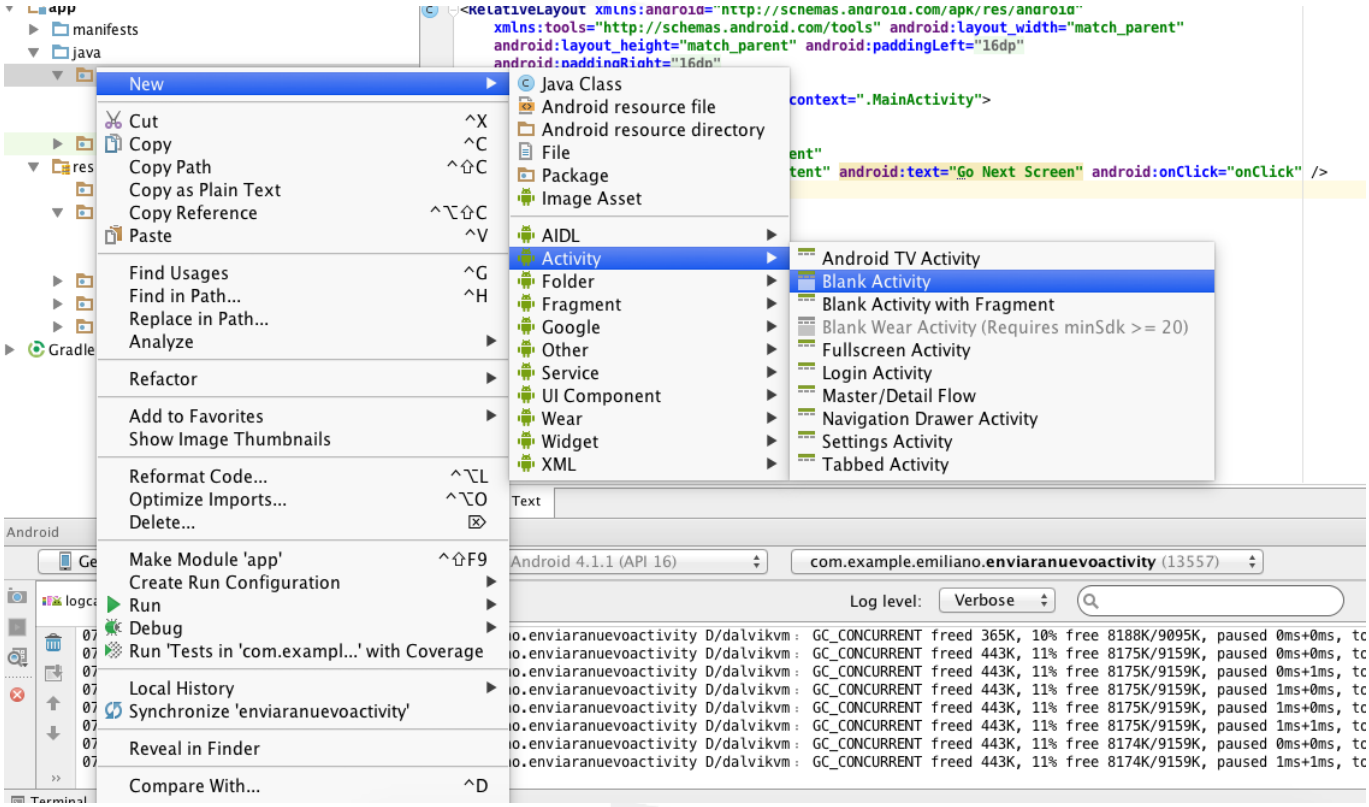


Pasar de un Activity a otro:

Sobre el ejemplo anterior vamos a cambiar el contenido del método onClick por:

startActivity(new Intent(this,SecondActivity.class));

y crearemos un nuevo activity de la siguiente forma:



Y le pondremos de nombre: SecondActivity

Luego correremos la aplicación y veremos que al clickear el botón nos llevará al segundo activity.

Aclaración: Este ejemplo está subido en la sección de Descargas/c1 su nombre es: "EnviarANuevoActivity".

Pasar de un Activity a otro (enviando un parámetro):

Es muy parecido al ejemplo anterior pero dentro del méotodo onClick se debe poner lo siguiente:

Intent intent=new Intent(this,SecondActivity.class);

Bundle bundle=new Bundle();

bundle.putString("clave","nombre");

intent.putExtras(bundle);

startActivity(intent);

Con el anterior código lo que hacemos es crear un Bundle (que es como un diccionario, o sea cada objeto que enviamos tiene una clave asociada);

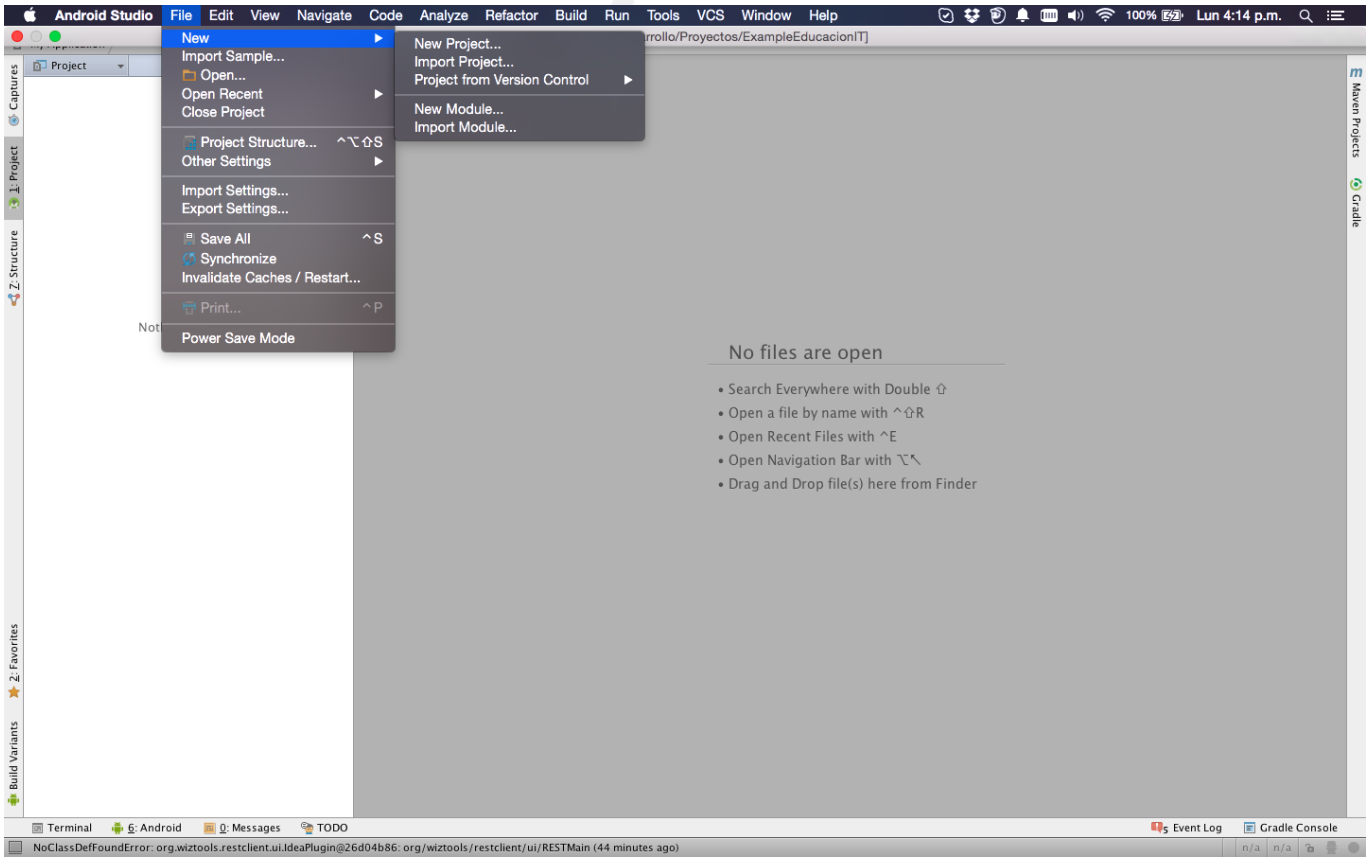
y en el método onCreate del SecondActivity pondremos el siguiente código:

```
String valorRecibido=getIntent.getExtras().getString("clave");
```

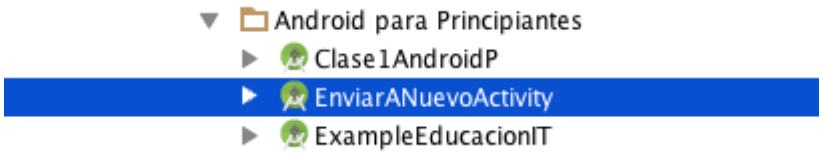
Ahora tenemos el valor enviado por parámetro en la variable valorRecibido.

Abrir proyectos desarrollados desde el Android Studio

Para abrir proyectos en el Android Studio seguir los siguientes pasos:



Y clickear en Import Project, a continuación se verá la siguiente pantalla dónde habrá que buscar el proyecto que deseamos abrir:



Y seleccionar Ok.

Por último haremos la práctica del Laboratorio 1 para integrar todos estos conocimientos adquiridos.