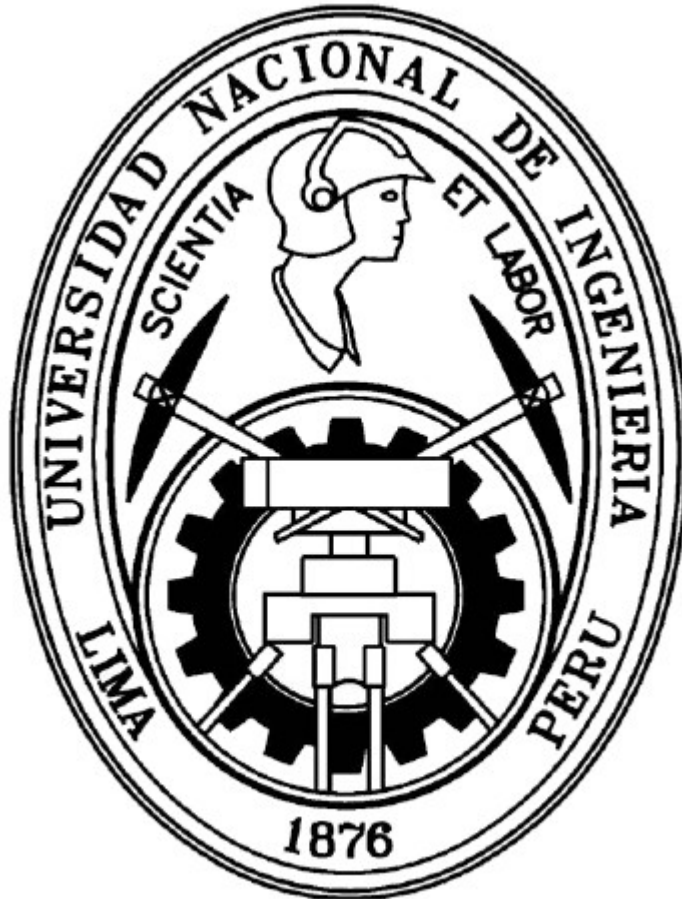


## Laboratorio 3



**Apellidos: Moreno Vera**

**Nombres: Felipe Adrian**

**Código: 20120354I**

**Asignatura: Programación en Dispositivos Móviles  
(CC481)**

**2016 - I**

## **Indice**

<b>Actividad 1 .....</b>	<b>(3)</b>
<b>Actividad 2 .....</b>	<b>(7)</b>
<b>Actividad 3 .....</b>	<b>(10)</b>
<b>Actividad 4 .....</b>	<b>(10)</b>
<b>Practica .....</b>	<b>(14)</b>
<b>Actividad 5 .....</b>	<b>(20)</b>
<b>Actividad 6 .....</b>	<b>(22)</b>

## Actividad 1

### 1. Crea un nuevo proyecto con los conceptos vistos en teoría.

1. Teclea “PrimeraAplicación” en el campo Application Name.

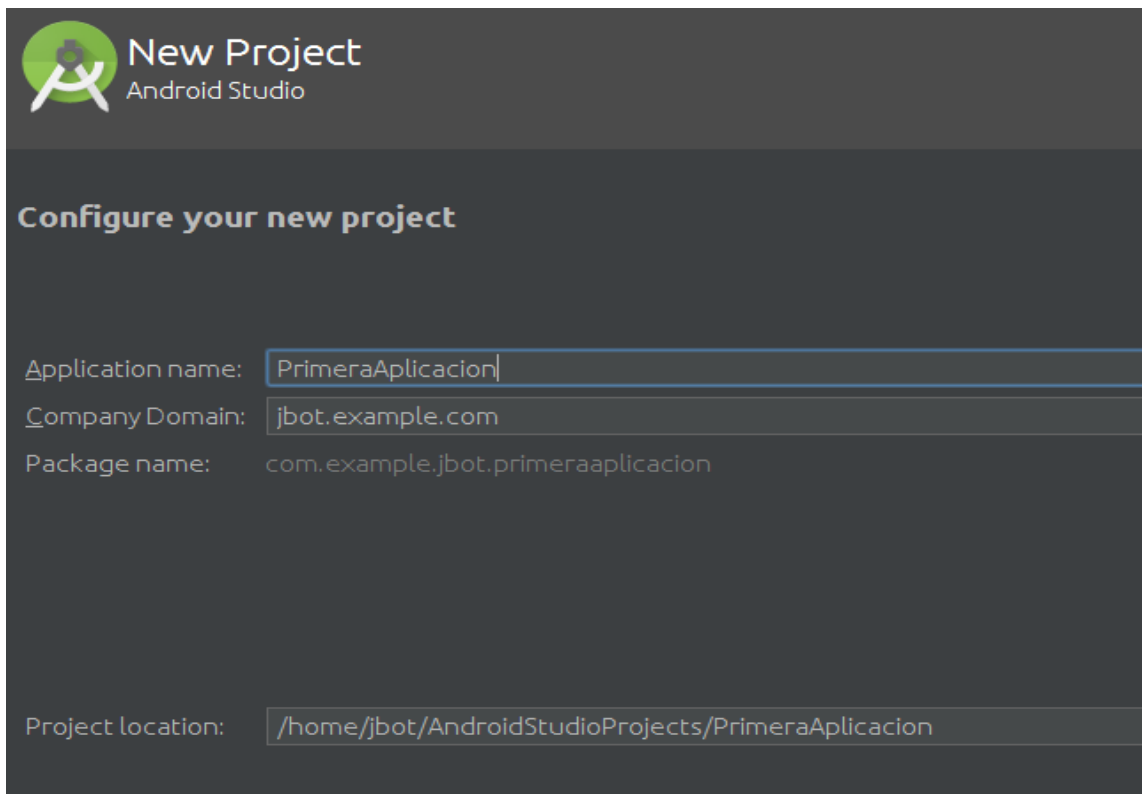
Android Studio y verifica que se crea una carpeta con ese nombre.

Esta adjunto en el ítem 5.

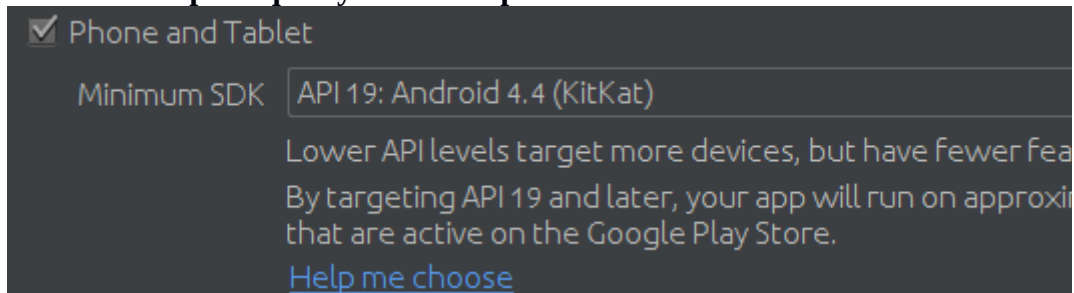
2. Elige el nombre del paquete “Package Name” siguiendo el estilo de Java conocido como convenio de dominio inverso:

`com.example.manwset.primeraplicacion`

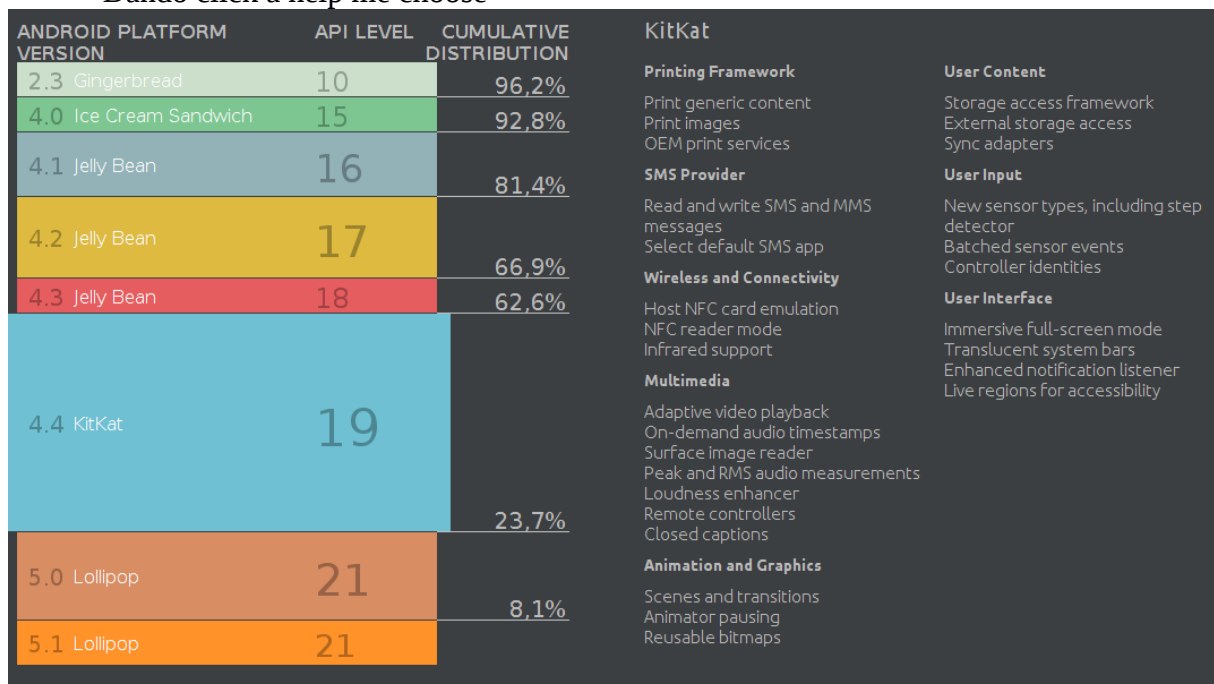
En primer lugar, el nombre de dominio garantiza que los nombres de los paquetes sean únicos evitando colisiones de nombres entre organizaciones. Además, este convenio garantiza que los paquetes dentro de una misma organización quedan bien estructurados. Hay que tener en cuenta que cada '.' del nombre del paquete se traducirá a un directorio. De este modo, los directorios se organizarán de más general ('com' en nuestro caso) a más particular ('manwest').



3. Elige un valor para Minum SDK. Antes de darle “Next” haz click en “Help me choose”. Explica que ayuda tan importante nos da.



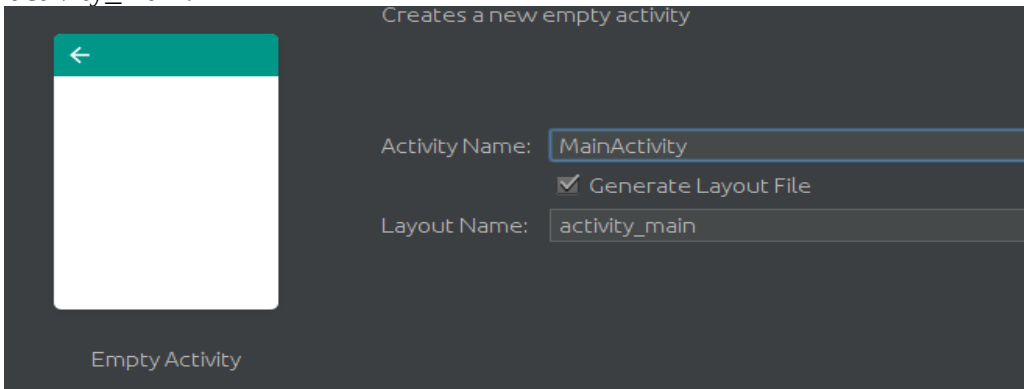
Dando click a help me choose



Help me choose, nos muestra una estadística de la cantidad de usuarios por versión de API. Por lo que si queremos desarrollar, podemos escoger una que este con más porcentaje en mercado. Por ejemplo el 4.4 kitkat con 62.6%

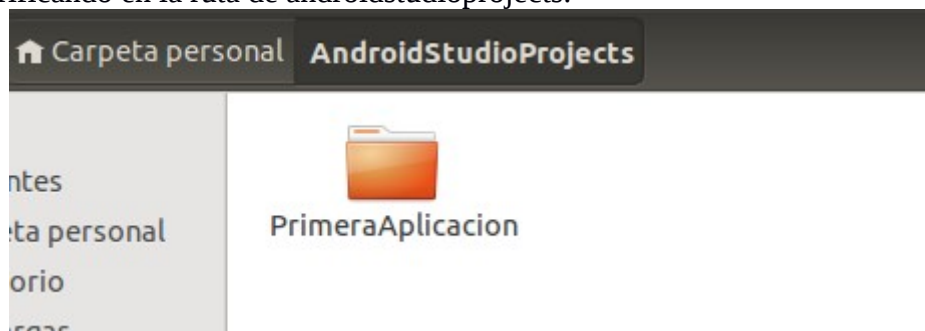
**4. En la siguiente pantalla se pide los nombres para nuestro proyecto. Apuntas cada nombre.**

Al ser solo Empty Activity aparecen Activity Name: MainActivity y Layout Name: activity\_main.

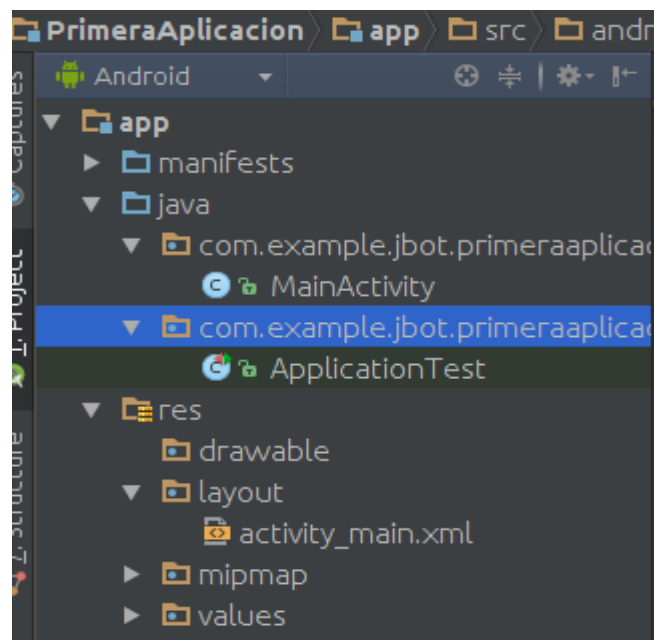


**5. Le damos a finalizar y vemos nuestro proyecto. Según los diferentes nombres puestos anteriormente localizarlos en nuestro proyecto.**

Verificando en la ruta de androidstudioprojects.

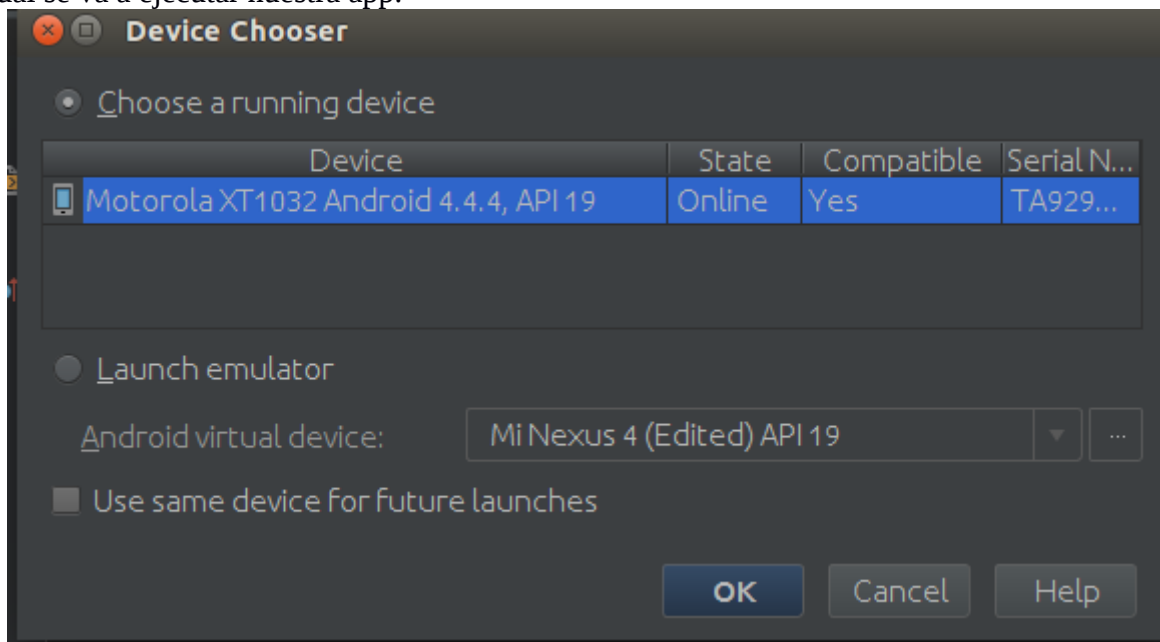


Viendo los nombres puestos.

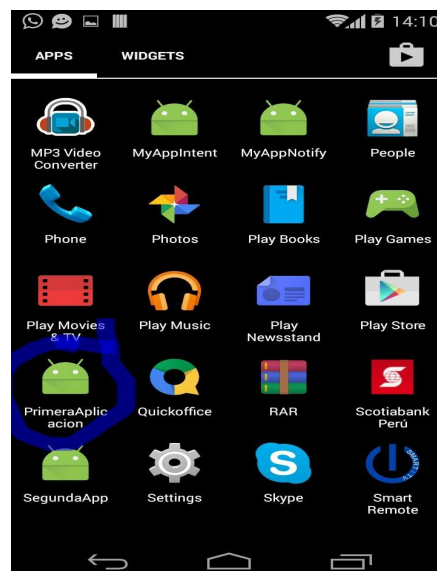
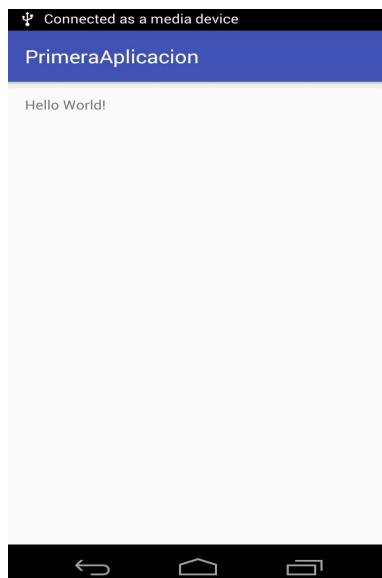


2. Ahora ya puedes ejecutar la aplicación en el emulador que creaste en la tercera unidad. Para ello, selecciona tu proyecto “PrimeraAplicacion” con el cursor en el “Package Explorer” (la ventana de la izquierda) y pulsa el ícono. Aparecerá el siguiente diálogo para que selecciones una máquina virtual.

Al presionar el botón verde de run, nos muestra un campo para escoger la máquina sobre la cual se va a ejecutar nuestra app.



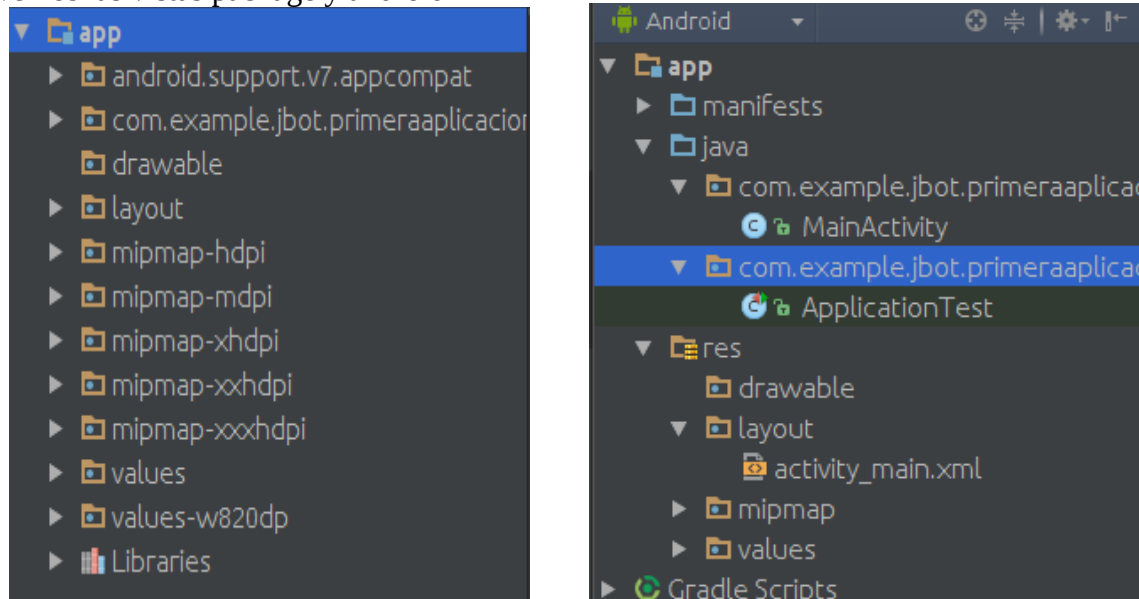
Como se ve, tengo mi Nexus 4 y mi moto G, yo escogeré mi moto G(para evitar que mi computadora consuma RAM). Y esto es lo que nos muestra en la app.



## Actividad 2

**1. Antes de ver los componentes haga un pequeño resumen de la estructura de proyecto que vimos en teoría con “PrimeraAplicacion”. Seguramente tengas la visión de packages cámbiela a Android para verlos de forma muy resumida los ficheros.**

Vemos las vistas package y android ->



**2. AndroidManifest.xml: En la teoría hemos visto que este fichero identifica el paquete principal y las actividades. Ábralo en el editor y entienda su contenido.**

El AndroidManifest.xml nos da información sobre nuestra app, nos da a modificar con cual actividad debe iniciar nuestra app al presionar un launcher (cuyo icono puede ser modificado en este fichero) y permisos de usuario que tiene, acceder a servicios por ejemplo.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jbot.primeraplicacion">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PrimeraAplicacion"
        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>
```

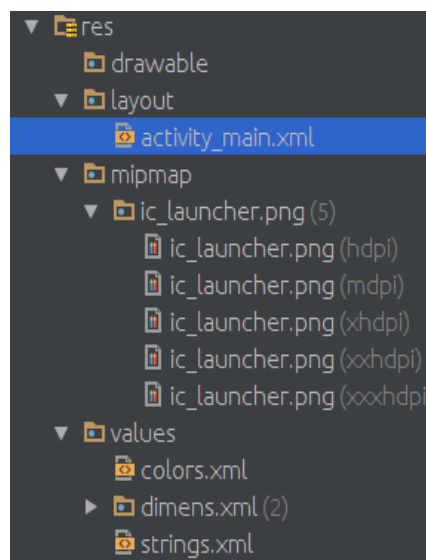
### 3. Actividades: Desde la actividad principal se carga el Layout o el diseño.

**1. Lo primero que vemos en nuestra clase principal Java es que hay una herencia. Explique esa clase padre e identifique su paquete de importación explicándolo también.**

Clase AppCompatActivity: Es la clase base que tiene soporte con actividades que usan el action bar, hereda de Activity. Una actividad es una sola cosa, enfocado que el usuario puede hacer. Casi todas las actividades interactúan con el usuario, por lo que el tipo de actividad se encarga de crear una ventana para usted en la que usted puede poner su interfaz de usuario con setContentView (Ver). Mientras que las actividades se presentan a menudo al usuario como ventanas de pantalla completa, también pueden ser utilizados en otras formas: como ventanas flotantes (a través de un tema con el conjunto windowIsFloating) o incrustados en el interior de otra actividad (usando ActivityGroup).

Su paquete de importación es android.support.v7.app.AppCompatActivity: Hay un tipo de librerías que están diseñadas a ser usadas con Android 2.1 (API 7) y superior. Estas librerías proveen configuraciones específicas y pueden ser incluidas en tu aplicación independientemente una con otra (otra versión).

**2. Un aspecto importante es el método setContentView. Este método dibuja el fichero xml en pantalla. Identificar dónde se encuentra este fichero xml en nuestro proyecto. Fijarse que en esa dirección se encontrarán todas las interfaces gráficas.**



Se ve que está en la carpeta res/layout, y además en la carpeta res, se encuentra todas las imágenes, y ficheros de cambio de aspecto como colors, dimens, etc.. La línea setContentView(R.layout.activity\_main), llama a este fichero y su configuración.



### 3. Explique los otros métodos que encontramos en dicho fichero.

Encontramos el método onCreate, es el que se ejecutará siempre que se cargue la actividad, dentro de el está el setContentView que ya explicamos.

### 4. Layout: en esta carpeta nos encontramos los xml que definen nuestra interfaz gráfica de la aplicación.

#### 1. En nuestra aplicación tenemos un Relative Layout. Explicar brevemente 2 layout más.

Linear Layout: A diferencia del Relative Layout, un linear layout abarca tal cual dice el nombre, una línea, puede meter objetos dentro pero todo estará como un bloque de línea, puede ser orientado en vertical u horizontal.

List View: Es un grupo de views que muestra una lista de elementos desplazables. Los elementos de la lista se insertan automáticamente en la lista con un adaptador que lanza el contenido de una fuente, como una consulta de matriz y convertir cada elemento resultante en una vista que se coloca en la lista.

#### 2. También hemos aprendido que toda variable string se encuentra en su fichero. Verificar la ruta de estas variables.

Su ruta es res / values / strings.xml

#### 3. Aunque ya lo veremos más adelante explicar los otros elementos de este fichero que no se explicaron en teoría.

Se ve el elemento TextView: que es un cuadro en el cual puedes escribir texto.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res-
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.jbot.primeraplicacion.MainActivity"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

**5. Strings: otro aspecto a entender es el fichero strings.xml. Cambia la variable en nuestro xml Activity y ejecute la aplicación. Comenta los cambios.**

Cambiamos la línea de `<string name="app_name">PrimeraAplicacion</string>` por `<string name="app_name1">PrimeraAplicacion</string>` y nos aparece un error.

```
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/AppTheme" >
```



Nos dice que no existe la variable `app_name` en nuestro manifest

```
/home/jbot/AndroidStudioProjects/PrimeraAplicacion/app/build/intermediates/manifests/full/debug
! No resource found that matches the given name (at 'label' with value '@string/app_name').
```

## Actividad 3

**1. Abra la carpeta física Android/sdk/Extras/Intel y dentro podrá observar el archivo. Instálelo y verá que mejora notablemente el arranque.**

Bueno yo estoy sobre linux, y linux no necesita, es por eso que el Extras del SDK me dice que no es compatible con linux.

<input type="checkbox"/>  Google Web Driver	2	<input checked="" type="checkbox"/> Installed
<input type="checkbox"/>  Intel x86 Emulator Accelerator (HAXM) inst	6.0.1	<input checked="" type="checkbox"/> Not compatible with Linux

## Actividad 4

**1. A continuación vamos a crear una segunda actividad. Para crearla creamos una clase pública que herede de AppCompatActivity como hemos estado estudiando hasta ahora. Hay que crearla donde está la otra actividad, es decir, src/main/java/comXXX.**

**2. Llamamos a nuestra clase “SegundaActividad”.**

**3. Para que sea considerada Actividad deberemos hacer que herede de AppCompatActivity.**

**4. Acordarse de hacer la importación de la librería correspondiente.**

**5. implementar el método onCreate de la clase anterior.**

**6. Verificar que nos solicita la importación de la librería “Bundle”.**

```
import android.support.v7.app.AppCompatActivity;
//import android.os.Bundle;

public class SegundaActividad extends AppCompatActivity {

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

7. cambiamos el nombre de la interfaz gráfica, llamándose ahora “activity\_segunda\_actividad”. Comprobamos que nos aparece un error y es porque no reconoce el Layout, es decir, no existe esta interfaz.

```
public class SegundaActividad extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_segunda_actividad);  
    }  
}
```

error: cannot find symbol variable activity\_segunda\_actividad

8. Copiamos y pegamos el archivo xml de Layout para crear nuestra nueva interfaz.

9. En el archivo “activity\_segunda\_actividad.xml” cambiamos “tools:context” por “SegundaActividad”.

10. Cambiamos el mensaje que muestra el TextView. Para ello tendremos que crear una nueva variable string y asignarle el nuevo valor en el fichero strings.xml.

```
activity_segunda_actividad.xml  
  
:paddingRight="16dp"  
:paddingTop="@dimen/activity_vertical_margin"  
tools:context="com.example.jbot.primeraproyecto.SegundaActividad"  
  
TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/app_name1" />
```

Muestra el texto que esta en el textview de la 2da actividad que tiene como nombre app\_name1.

11. Una vez creada nuestra segunda actividad recordad que se debe de indicar en el archivo “AndroidManifest.xml”.

1. Escribir el siguiente código dentro de la etiqueta “application”.

```
<activity  
    android:name=".SegundaActividad"  
    android:label="@string/seg_actividad">  
</activity>
```

2. Vemos que nos aparece error, eso es porque no hemos creado dicho string. Procedemos a crearlo con, por ejemplo, el valor “Mi segunda actividad”.

```
<activity  
    android:name=".SegundaActividad"  
    android:label="@string/seg_actividad">  
></activity>
```

Vemos que nos manda error, entonces corregimos eso en strings.xml

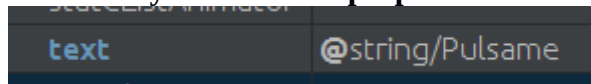
```
<resources>
    <string name="app_name">PrimeraAplicacion</string>
    <string name="app_name1">SegundaActividad</string>
    <string name="seg_actividad">Mi Segunda Actividad</string>
</resources>
```

lista de variables en strings.xml

12. En la pestaña design creamos un button en el fichero “activity\_primera\_aplicacion” que es el xml que teníamos en primer momento. Para ello borramos el TextView de “Helloworld!”.

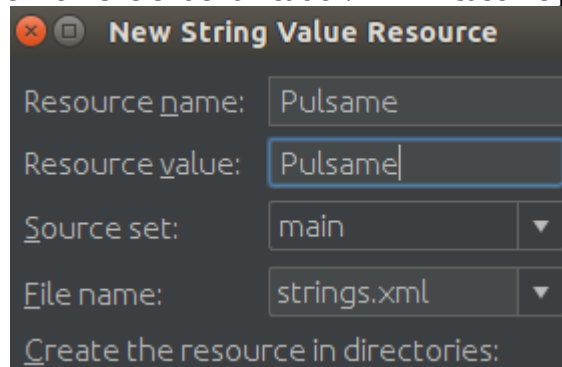
13. vemos como la pestaña Text se ha creado el código de <Button>, Además nos aparece una advertencia que es que tenemos que definir dicho string. En esta ocasión vamos a cambiarlo a través de “Properties” en la vista “Design”.

1. Pulsamos sobre Button y buscamos la propiedad “text”.

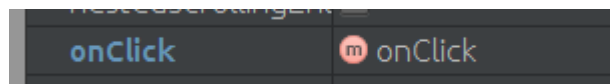


2. Hacemos click en los puntos suspensivos y luego en New Resource -> New string value ...

3. Le indicamos el nombre e identificador. En mi caso he puesto “Pulsame”.



4. En la propiedad onClick vamos a poner nuestro nombre al método nuevo al hacer click sobre “Pulsame”. En mi caso le he llamado onClick.



El diseño queda así:

```
android:layout_width="100dp"
android:layout_height="40dp"
android:layout_alignParentTop="true"
android:layout_alignParentStart="false"
android:text="@string/Pulsame"
android:onClick="onClick" />
```

**5. En el fichero “PrimeraAplicacion.java” añadimos el siguiente método:**

```
public void onClick(View view){
    Intent intent = new Intent(this,SegundaActividad.class);
    startActivity(intent);
    finish();
}
```

**6. Del código anterior explique cada línea su funcionalidad.**

Public void onClick(View view), función de retorno tipo void, que toma como parámetro una vista.

Intent intent = new intent(this,SegundaActividad.class), instancia un objeto de la clase Intent, que toma como parámetros en su constructor. Una conexto(que seria nuestra actividad) y un .class que es el código de una clase que vamos a lanzar(en este caso, la segunda actividad).

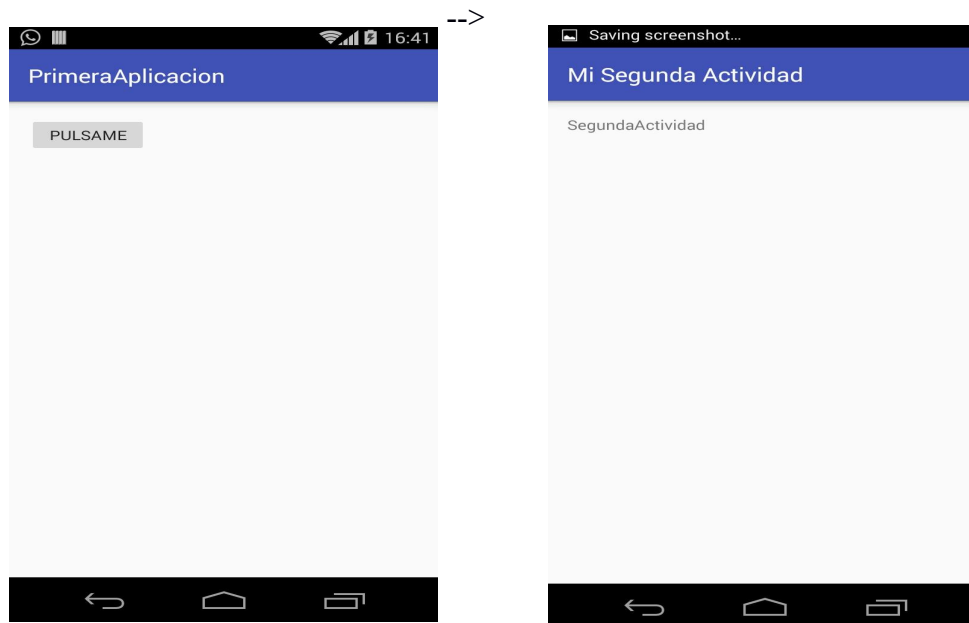
startActivity(intent) inicia la actividad definida en intent, que tomo como parámetro el código de la segund actividad.

Finish(); cierra esta actividad(la primera).

**7. Importamos las clases que nos sugiere.**

```
import android.view.View; import android.content.Intent;
```

Mostrando la ejecución:

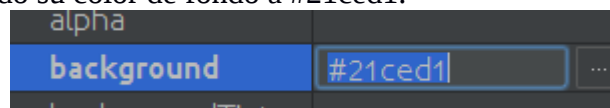


## Practica

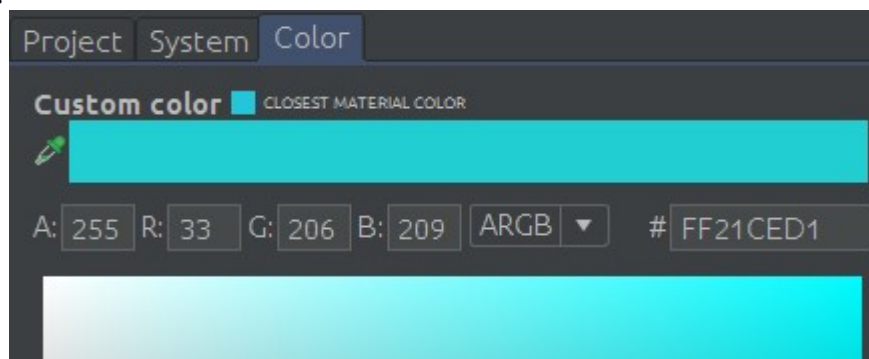
### 1. En la pestaña propiedades de nuestra vista diseño:

**1. cambiar el color de fondo de “Pulsame”. Exponga la propiedad y como funcionan sus valores.**

Hemos cambiado su color de fondo a #21ced1.



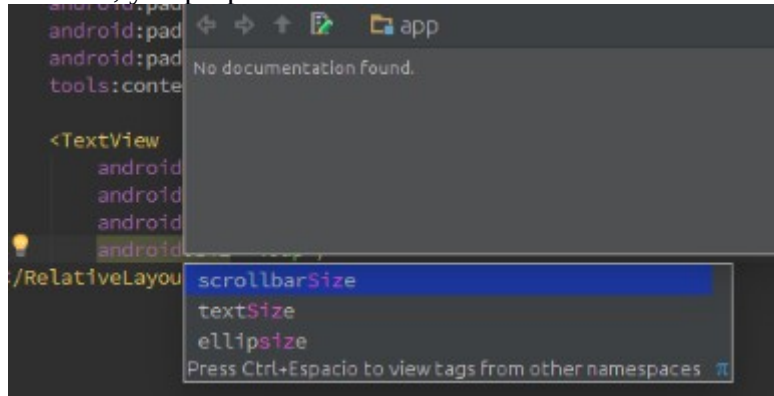
Nos muestra propiedades de system que contiene estilos predeterminados, estilos del proyecto y en color, nos permite escoger a nuestro gusto, de modo ARGB, RGB y HSB.



Donde HSB es por porcentajes de RGB(red green, blue) y ARGB es alpha RGB.

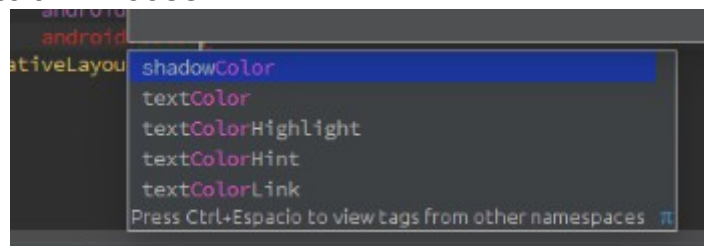
**2. En la actividad del texto hacemos click en Properties y buscamos size. De esta forma podemos buscar propiedades que tiene esa palabra. Cambiamos el tamaño del texto con la propiedad adecuada.**

Se busco con size, y la propiedad adecuada es textSize.



**3. Cambiamos color de texto también.**

Se puso el color #ffcc33.



**4. Para cambiar el color de fondo tendremos que hacer click en nuestro layout de Component Tree y en Properties buscamos la adecuada. Cambiamos de color.**

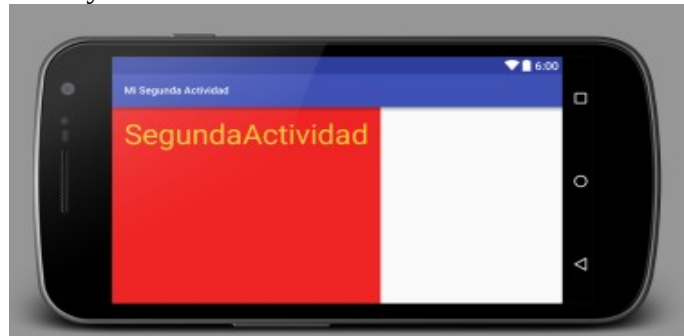
En background se cambia el color de fondo a rojo.



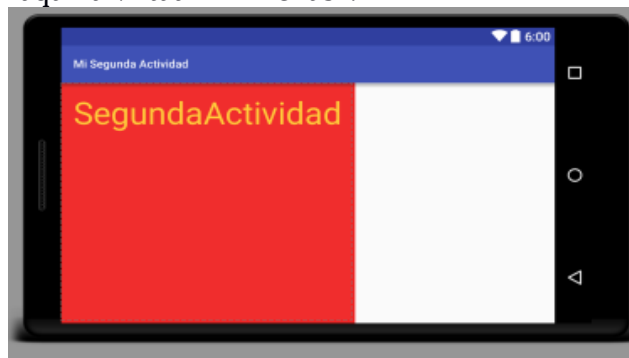
**2. En la vista diseño también podemos ver la apariencia de nuestro dispositivo en vertical y horizontal, que se encuentra entre el elemento que selecciona el dispositivo y “AppTheme”.**

**1. También podemos ver como quedan en múltiples dispositivos, navegar por esta opción.**

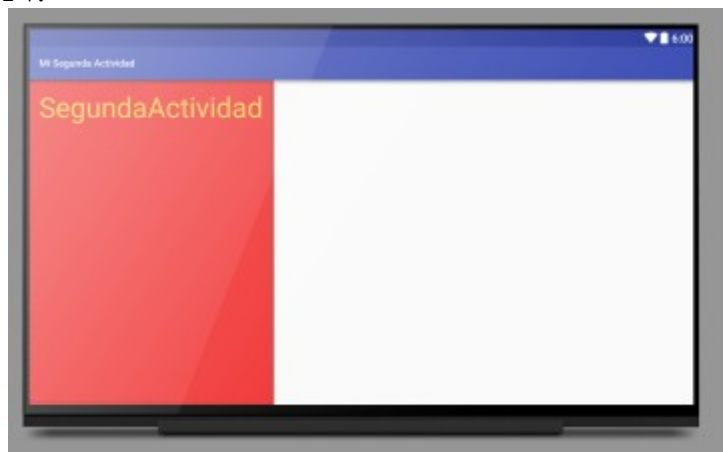
Viendo en el Galaxy Nexus4.



Viendo en mi máquina virtual “mi nexus”.



Viendo en TV.





**2. Donde está el icono Android en la misma pestaña podemos ver como quedaría nuestra aplicación para diferentes versiones.**

Como use de la API 19 hacia arriba, solo me muestra API 23 y API 19.



API 19



API 23

**3. En la última actividad hemos visto como crear una actividad con una instancia.**

**Teniendo esta actividad como referencia, realice los siguientes pasos:**

**1. Cree un nuevo proyecto.**

Se llama MiPrueba.

**2. En la actividad principal cambie el fondo de pantalla y alguna propiedad más.**

Se cambio el color de fondo a verde y el tamaño de "HelloWorld!" a 50dp.

**3. Haz click derecho sobre la carpeta Layout y cree una nueva actividad. Fijarse en la carpeta Java y en el archivo "AndroidManifest.xml", Qué a ocurrido ?**

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".Main2Activity"></activity>
```

Se ha creado automáticamente la sección activity, con atributo name ".Main2Activity".

**4. Volver a realizar una instancia pero cambiando propiedades a elementos de cada actividad.**

Volviendo a instanciar creamos la actividad 3, con otros colores.

Mostrando el original(activityMain) y las 2 instancias creadas, con modificaciones.

## ActivityMain

```
activity_main3.xml x activity_main.xml x Main3Activity.java x
<?xml version="1.0" encoding="utf-8"?>
<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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.jbot.miprueba.MainActivity"
    android:background="#55e916">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="50dp"
        android:textColor="#ed6d12" />
</RelativeLayout>
```

## Instancia 1.(ejercicio parte 3)

```
activity_main3.xml x land/activity_main2.xml x Main3Activity.java x
<?xml version="1.0" encoding="utf-8"?>
<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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.jbot.miprueba.Main2Activity"
    android:background="@android:color/holo_purple">

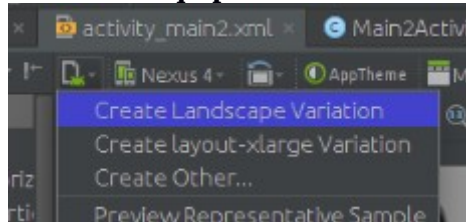
</RelativeLayout>
```

## Instancia 2. (ejercicio parte 4)

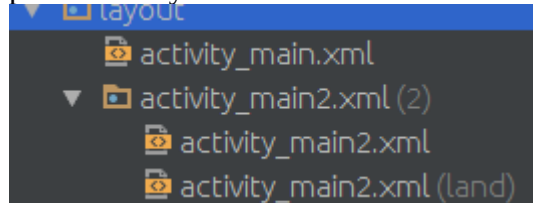
```
activity_main3.xml x activity_main.xml x Main3Activity.java x
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.jbot.miprueba.Main3Activity">
    <Button android:layout_width="match_parent"
        android:layout_height="50dp"
        android:text="New Button"
        android:id="@+id/botonQueFalla"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:background="#f9e908"
        android:onClick="informacion"/>
</RelativeLayout>
```

4. Otro aspecto importante es establecer diseños cuando la aplicación esta en vertical y horizontal. Buscamos el icono que se especifica en la imagen y elegimos “Create Landscape Variation”.

1. Vemos como ha creado un nuevo “Layout”, Donde se ha creado dicho “Layout” ? Buscar la carpeta “layout-land”, de landscape.  
También podemos observar que los xml se llaman igual, Android se encarga de escogerlo según la rotación del equipo.



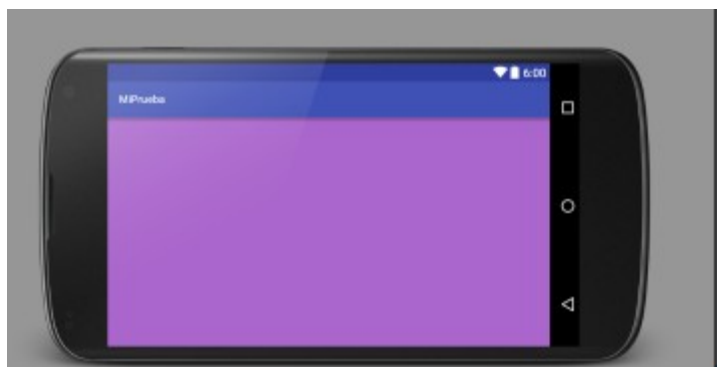
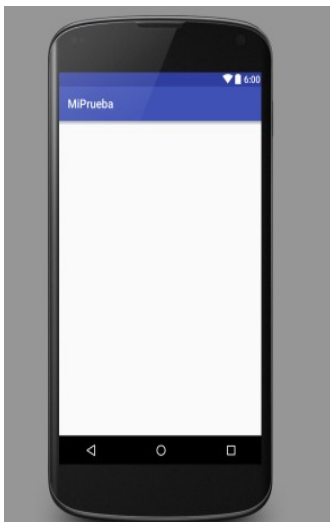
Creando el landscape variation y verificamos la creación de otro más y son iguales.



2. Damos a la propiedad Background y cambiamos el fondo en, por ejemplo, System -> holo\_purple.

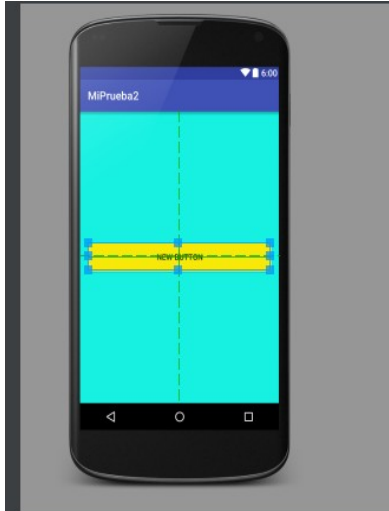
Entonces viendo la parte texto del activitiy main xml, se ve que background ha tomado la forma de android:background="@android:color/holo\_purple".

3. Con esto vemos que tiene una nueva variante para mi orientación de pantalla.
4. En la misma vista diseño, rotamos la vista previa, Qué nos muestra ?

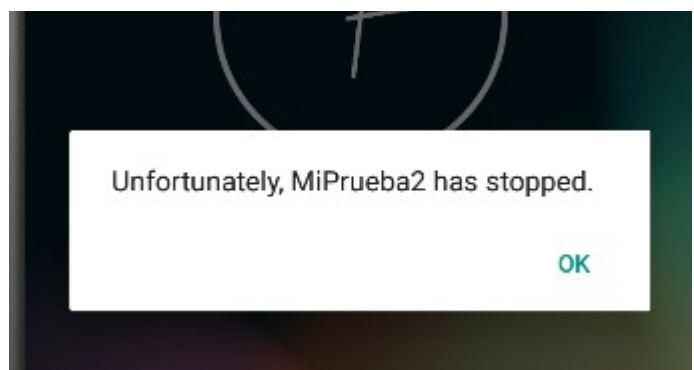
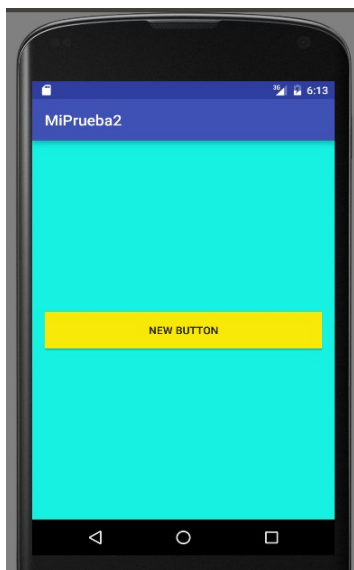


## Actividad 5

Creamos la interfaz dicha ...



1. Una vez creada nuestra interfaz, le damos el valor “conductaIncorrecta” a la propiedad `onClick` de `Button`.
2. Este método no existe en el código Java.
3. Hacemos doble click en la vista de diseño de nuestro `Button` y le damos el valor `id` “`botonQueFalla`”.
4. Le damos a `Run` y vemos que se compiló correctamente aunque da error al pulsar. Ejecutando en mi máquina virtual Nexus 4 Marshmallow, se ejecuta correctamente. Pero al pulsar, nos da error, pues no hay método `new button`.



**5. En la parte posterior abrimos el apartado “6:Android” y se activa la herramienta LogCat.**

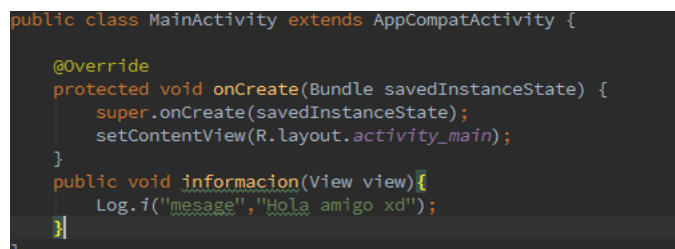
**6. Volvemos a hacer click y vemos que nos dice LogCat. Podemos observar que nos muestra que hubo un “FATAL ERROR”. Transcribe hasta aquí lo que nos muestra.**

*\*El siguiente texto está en letra 10, para evitar el desorden.*

```
01-17 18:25:06.304 2132-2132/com.example.jbot.miprueba2 I/art: Not late-enabling -Xcheck:jni (already on)
01-17 18:25:06.334 2132-2132/com.example.jbot.miprueba2 W/System: ClassLoader referenced unknown
path: /data/app/com.example.jbot.miprueba2-2/lib/x86_64
01-17 18:25:06.364 2132-2145/com.example.jbot.miprueba2 D/OpenGLRenderer: Use
EGL_SWAP_BEHAVIOR_PRESERVED: true
01-17 18:25:06.404 2132-2145/com.example.jbot.miprueba2 I/OpenGLRenderer: Initialized EGL, version 1.4
01-17 18:25:06.424 2132-2145/com.example.jbot.miprueba2 W/EGL_emulation: eglSurfaceAttrib not
implemented
01-17 18:25:06.424 2132-2145/com.example.jbot.miprueba2 W/OpenGLRenderer: Failed to set
EGL_SWAP_BEHAVIOR on surface 0x7f61e7552b40, error=EGL_SUCCESS
01-17 18:25:43.164 2132-2132/com.example.jbot.miprueba2 D/AndroidRuntime: Shutting down VM
01-17 18:25:43.164 2132-2132/com.example.jbot.miprueba2 E/AndroidRuntime: FATAL EXCEPTION: main
Process: com.example.jbot.miprueba2, PID: 2132
java.lang.IllegalStateException: Could not find method
conductaIncorrecta(View) in a parent or ancestor Context for android:onClick attribute defined on view class
android.support.v7.widget.AppCompatButton with id 'botonQueFalla'
        at
        android.support.v7.app.AppCompatActivity$DeclaredOnClickListener.resolveMethod(AppCompatActivityInfl
ater.java:307)
        at
        android.support.v7.app.AppCompatActivity$DeclaredOnClickListener.onClick(AppCompatActivityInflater.jav
a:266)
        at android.view.View.performClick(View.java:5198)
        at android.view.View$PerformClick.run(View.java:21147)
        at android.os.Handler.handleCallback(Handler.java:739)
        at android.os.Handler.dispatchMessage(Handler.java:95)
        at android.os.Looper.loop(Looper.java:148)
        at android.app.ActivityThread.main(ActivityThread.java:5417)
        at java.lang.reflect.Method.invoke(Native Method)
        at
        com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:726)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
```

**7. Cambiamos el valor de onClick por “informacion” donde en el código java tendrás el siguiente código.**

**public void informacion(View view){ Log.i(“message”,”Hola amigo xd”); }**



```
public class MainActivity extends AppCompatActivity {

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

    public void informacion(View view){
        Log.i("message","Hola amigo xd");
    }
}
```

```
01-17 18:34:34.953 2124-2124/com.example.jbot.miprueba2 I/mesage: Hola amigo xd
01-17 18:34:38.973 2124-2124/com.example.jbot.miprueba2 I/mesage: Hola amigo xd
01-17 18:34:39.623 2124-2124/com.example.jbot.miprueba2 I/mesage: Hola amigo xd
```

## 8. Explicar la ejecución ahora y sobre todo los distintos niveles para filtrar la información que tenemos.

Ahora el mensaje de “Hola amigo xd” lo muestra en la ventana de LogCat ( que es la consola debugger de salida).

Los niveles son:

La clase Log se utiliza en el código para imprimir mensajes en el LogCat.

Los métodos más comunes incluyen:

v(String, String) (verbose) : Dependiendo del parámetro, lanza tiempo, tag, id de proceso, la fecha, lanza un texto al LogCat

d(String, String) (debug) : Es un mensaje como los que aparecen con fecha, lanza paso a paso cada instrucción hecha con fecha y hora y paquete de origen de la app.

i(String, String) (información) : Lanza un mensaje de información.

w(String, String) (advertencia) : Es usado para dar advertencias, como por ejemplo que no se encontró un path de alguna clase.

e(String, String) (error) : Lanza mensajes de error de ejecución o compilación.

f(String, String) (fatal error) : Lanza un Fatal error (generalmente es rojo).

## Actividad 6

### 1. Ejecute la aplicación anterior de manera que esté su máquina virtual activada.

Listo, esta ejecutando.

### 2. Explique la importancia del icono para revisar toda la estructura de archivos.

Muestra todos los ficheros .java y sus respectivos métodos, si en algún momento falla algo en algún método, podremos buscar en esa lista.



**3. Explique la importancia del icono para revisar todos los hilos que se están ejecutando.**

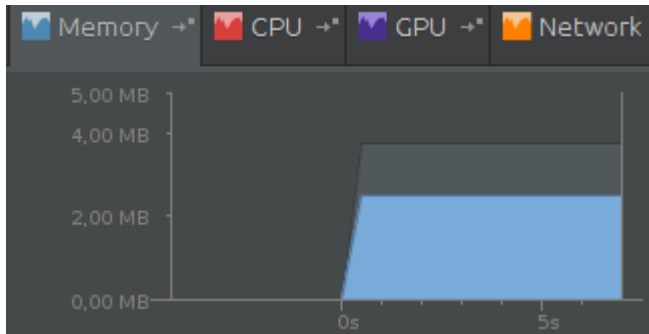
com.android.externalstorage	1456		8600
system_process	1323		8601
com.android.systemui	1448		8602
com.android.inputmethod	1653		8603
android.process.media	1675		8604
com.android.phone	1687		8605
com.android.launcher3	1698		8606
com.android.printspooler	1719		8607
android.process.acore	1749		8608
com.android.music	1826		8609
com.android.deskclock	1848		8610
com.android.quicksearchbox	1867		8611
com.android.settings	1891		8612
com.android.calendar	1912		8613
com.android.providers.calendar	1940		8614
com.android.messaging	1962		8615
com.android.keychain	2002		8616
com.android.dialer	2018		8617
com.android.managedprovisioning	2057		8618
com.android.email	2077		8619

Sirve para ver que hilo contiene qué tarea y cuanto RAM usa, es importante para gestionar los procesos que genera nuestra app.

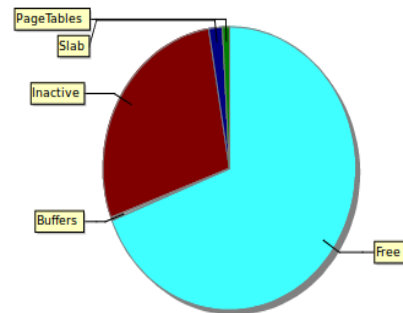
#### 4. Explique la importancia del icono para revisar el uso de memoria.

Para conocer que tanto de almacenamiento está siendo usado por nuestra máquina.

LogCat

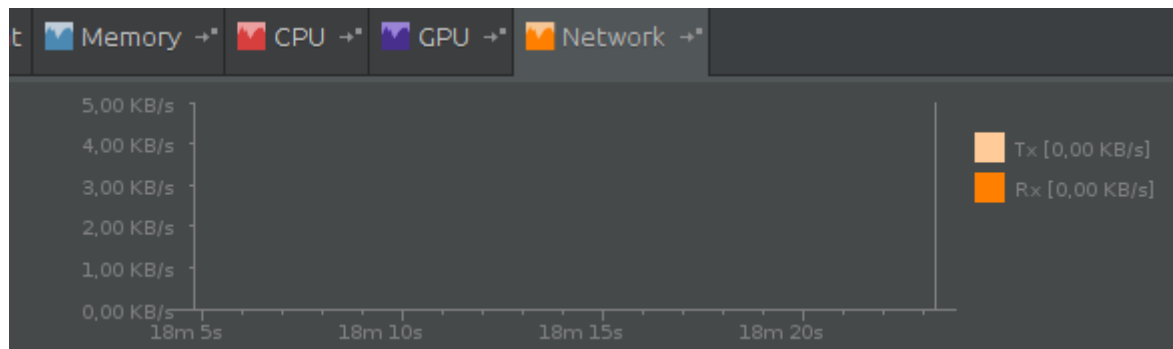


ADM



#### 5. Explique la importancia del icono para revisar el uso de internet.

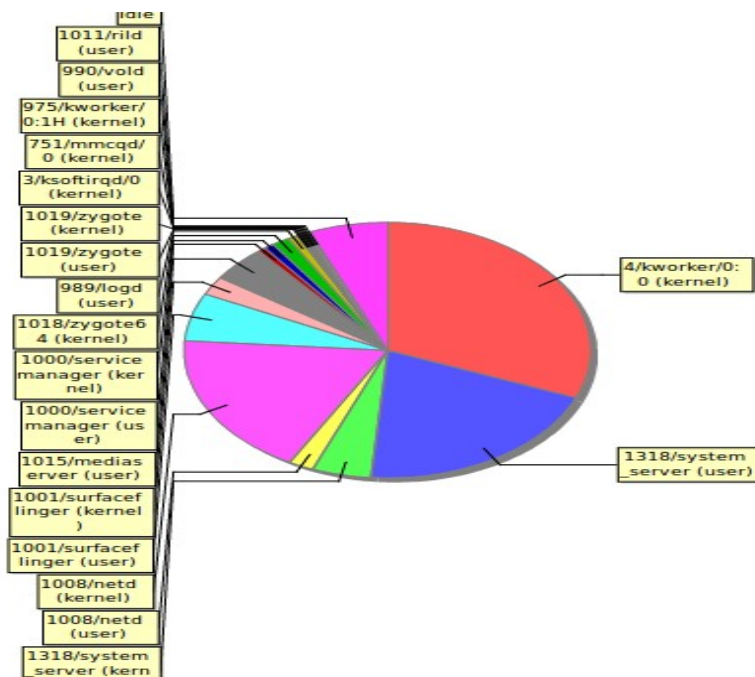
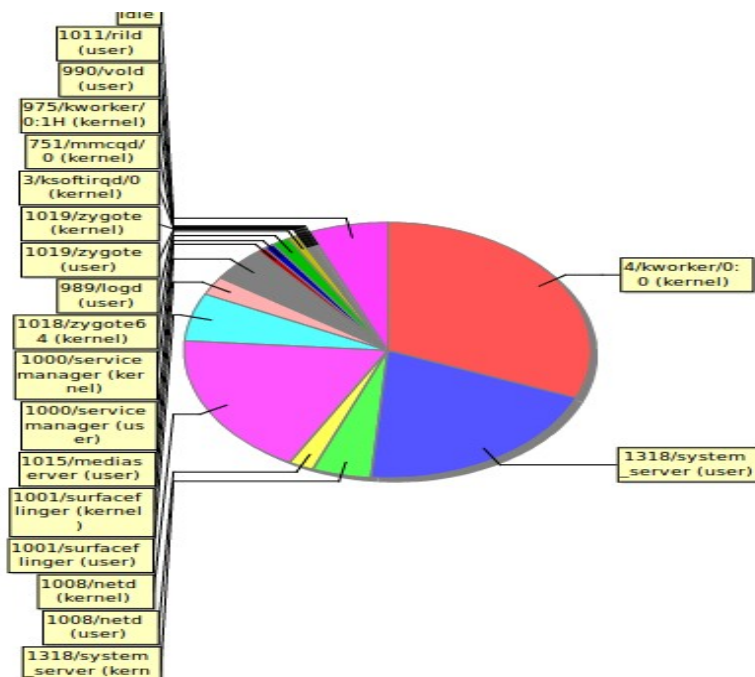
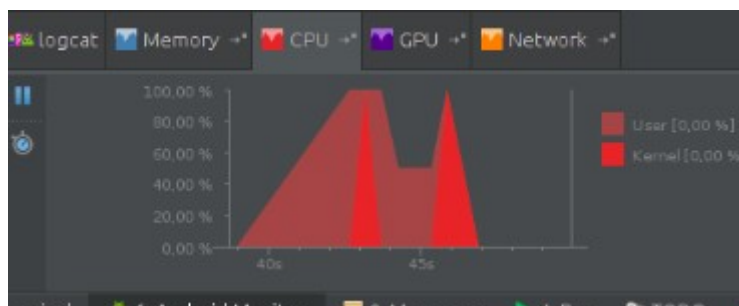
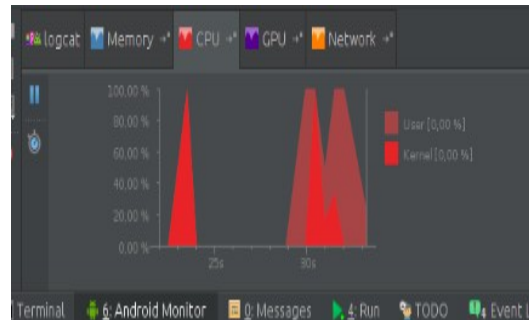
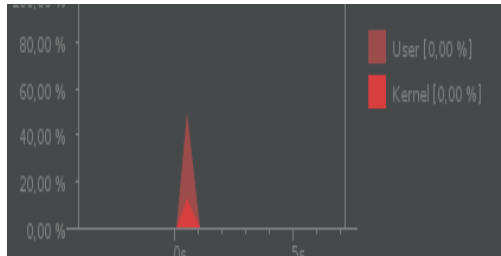
Nos indica que tanto recursos y servicios carga nuestra app, por ejemplo si fuera una app tipo face, podríamos ver cuantos Kb/s consume nuestra app al momento que correr el servicio, en nuestro caso, nuestra app no usa ningún servicio, así que esta en 0 siempre.





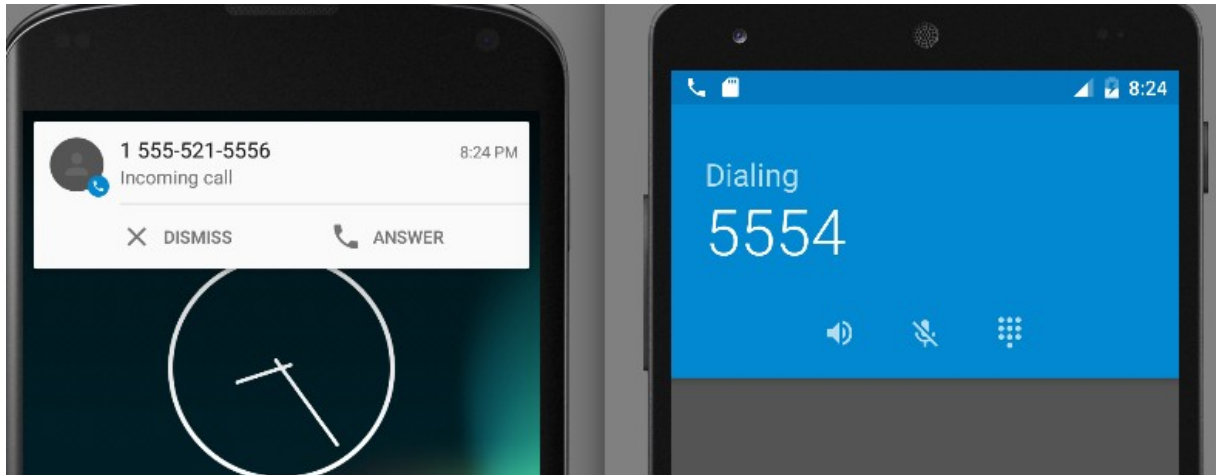
## 6. Explique la importancia del icono para revisar la carga de trabajo.

Las gráficas nos muestra que tanto de la RAM se está usando, es muy importante para saber si nuestra aplicación es óptima o necesita mejorar en su procesamiento y su algoritmia.



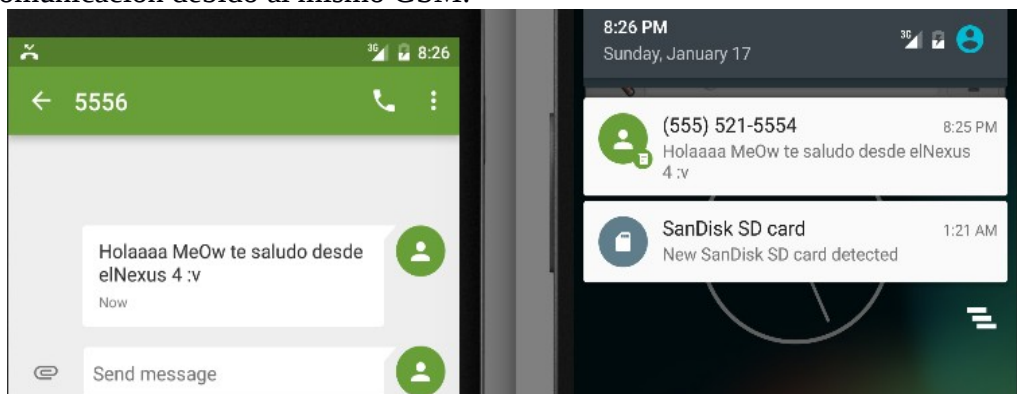
### 7. Realice una llamada telefónica al equipo virtual.

Para llamar entre máquinas emuladas, no podemos hacerlo con un celular debido a que los GSM simulados no son el mismo que el de un móvil, es por eso que para llamar o enviar mensaje use 2 máquinas virtuales, un Nexus 4 Android API 19 y un Nexus 6 API 23.

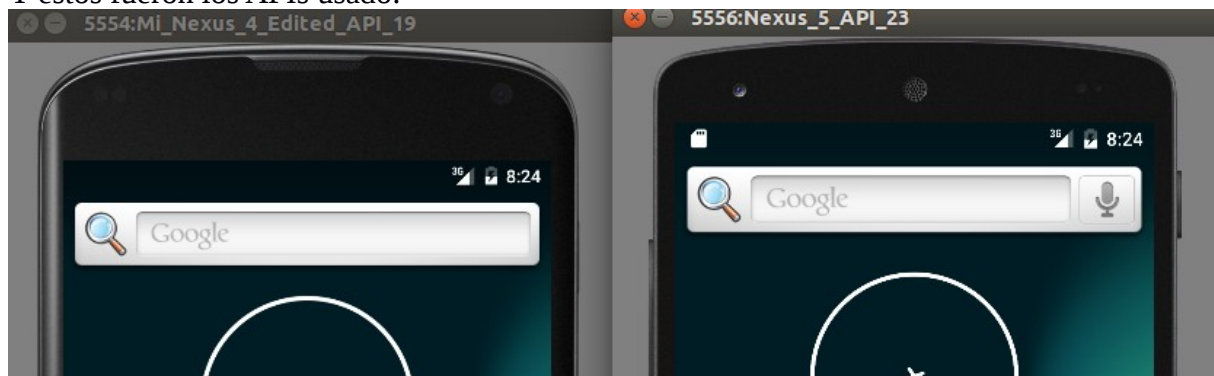


### 8. Envíe un mensaje al dispositivo.

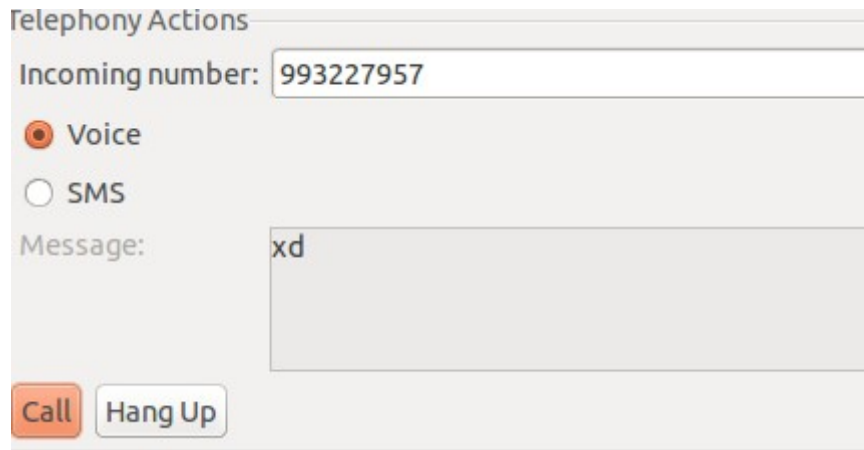
Para el mensaje es muy similar, entre 2 máquinas virtuales que corren sobre AVD, si hay comunicación debido al mismo GSM.



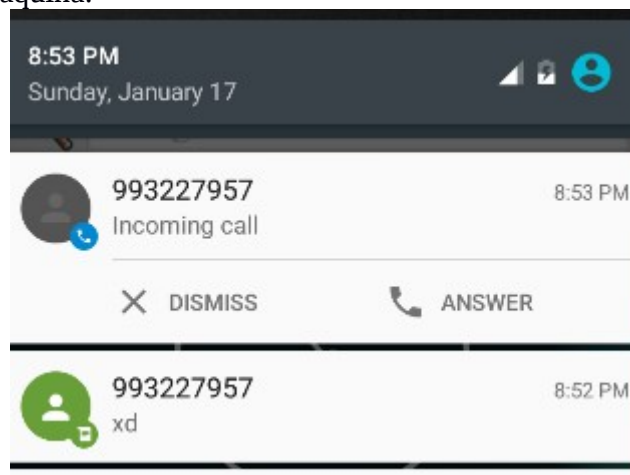
Y estos fueron los APIs usado:



Aparte de esto, se puede simular las llamadas en Android Device Monitor, en la parte de Emulated Control , siguiendo en Telephony Actions, se envia y llama a la máquina virtual.



Y como se ve en la máquina.



**Link del github con los códigos del laboratorio:**

[https://github.com/Jenazad/PDM/tree/master/Laboratorio\\_3](https://github.com/Jenazad/PDM/tree/master/Laboratorio_3)

## Referencias

<http://developer.android.com/intl/es/guide/topics/ui/layout/listview.html>  
<http://developer.android.com/intl/es/reference/android/content/Intent.html#Intent%28android.content.Context,%20java.lang.Class%3C?%3E%29>  
<http://developer.android.com/intl/es/guide/topics/ui/layout/listview.html>  
<http://developer.android.com/intl/es/reference/android/support/v7/app/AppCompatActivity.html>  
<http://www.aprendeandroid.com/13/fundamentos3.htm>  
<http://www.101apps.co.za/index.php/articles/using-android-s-log-class-api-to-debug-android-application-code.html>  
<http://developer.android.com/intl/es/tools/debugging/debugging-log.html>  
<http://developer.android.com/intl/es/tools/debugging/debugging-log.html#startingLogcat>  
<http://blog.desdelinux.net/guia-rapida-para-utilizar-github/>  
<http://android.stackexchange.com/questions/29709/how-to-make-phonecalls-on-an-android-avd>