



Manual Básico Android Studio



ÍNDICE

- 1 - Instalación de las herramientas necesarias para programar para Android Studio
- 2 - Pasos para crear el primer proyecto Android Studio
- 3 - Capturar el clic de un botón
- 4 - Controles RadioGroup y RadioButton
- 5 - Control CheckBox
- 6 - Control Spinner
- 7 - Control ListView (con una lista de String)
- 8 - Control ImageButton
- 9 - Notificaciones sencillas mediante la clase Toast
- 10 - Control EditText
- 11 - Lanzar un segundo "Activity"
- 12 - Lanzar un segundo "Activity" y pasar parámetros
- 13 - Almacenamiento de datos mediante la clase SharedPreferences
- 14 - Almacenamiento de datos en un archivo de texto en la memoria interna
- 15 - Almacenamiento de datos en un archivo de texto localizado en una tarjeta SD
- 16 - Almacenamiento en una base de datos SQLite
- 17 - Instalación del programa Android en un dispositivo real
- 18 - Layout (LinearLayout)
- 19 - Layout (TableLayout)
- 20 - Layout (FrameLayout)
- 21 - Layout (ScrollView y LinearLayout)
- 22 - Icono de la aplicación
- 23 - Reproducción de audio (archivo contenido en la aplicación)
- 24 - Reproducción, pausa, continuación y detención de un archivo de audio.
- 25 - Reproducción de audio (archivo contenido en una tarjeta SD)
- 26 - Reproducción de audio (archivo localizado en internet)
- 27 - Reproducción de audio utilizando el reproductor propio de Android (vía Intent)
- 28 - Grabación de audio mediante el grabador provisto por Android (via Intent)
- 29 - Captura de audio mediante la clase MediaRecorder

- 30 - Dibujar: graficar un píxel
- 31 - Dibujar: pintar fondo y dibujar líneas
- 32 - Dibujar: rectángulos
- 33 - Dibujar: círculos
- 34 - Dibujar: óvalos
- 35 - Dibujar: texto
- 36 - Dibujar: texto con fuentes externas
- 37 - Dibujar: texto sobre un camino
- 38 - Dibujar: una imagen
- 39 - Evento touch: dibujar un círculo
- 40 - Evento touch: juego del buscaminas
- 41 - Archivo strings.xml
- 42 - Internacionalización y archivo strings.xml
- 43 - Localización y archivo strings.xml
- 44 - Componente ActionBar (Básica)
- 45 - Componente ActionBar (Botones de acción)
- 46 - Componente ActionBar (Ocultarlo y mostrarlo)

1 - Instalación de las herramientas necesarias para programar para Android Studio

Descarga de herramientas necesarios para programar para Android con el Android Studio

1 - Primero debemos instalar el compilador de Java y la máquina virtual. Estas herramientas las podemos descargar de:

[Java SE Development Kit \(JDK\).](#)

2 - El segundo paso es la descarga del Android Studio (que contiene todo lo necesario para comenzar el desarrollo de aplicaciones en Android), lo hacemos del sitio :

[Android SDK.](#)

Ahora procedemos a su instalación en el equipo ejecutando el archivo que acabamos de descargar:



Dejamos por defecto para que instale el Android Studio, el Android SDK, Android Virtual Device etc.:



Dejamos con los valores por defecto en las siguientes pantallas de instalación.

3 - El tercer paso es ejecutar el Android Studio para poder empezar a desarrollar para Android:



La primera vez que lo ejecutamos puede salir un diálogo para actualizar el Android SDK para luego aparecer una interfaz similar a esta:

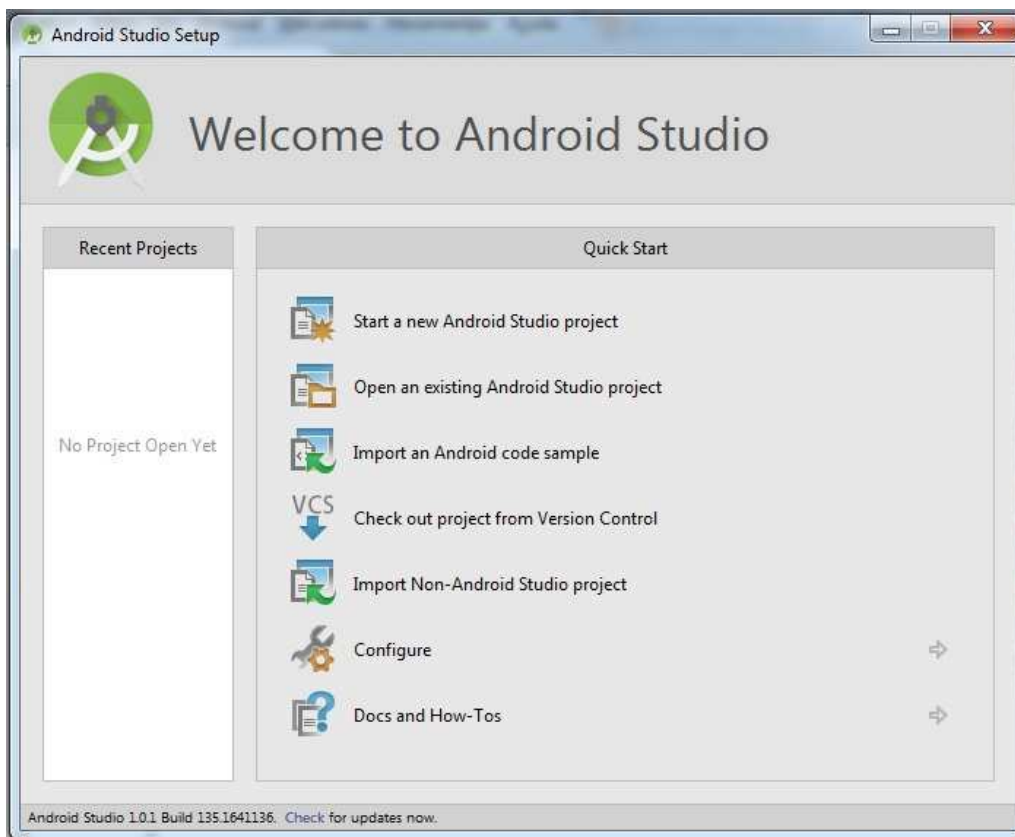


Desde esta interfaz siempre procedemos para crear un nuevo proyecto.

[Retornar](#)

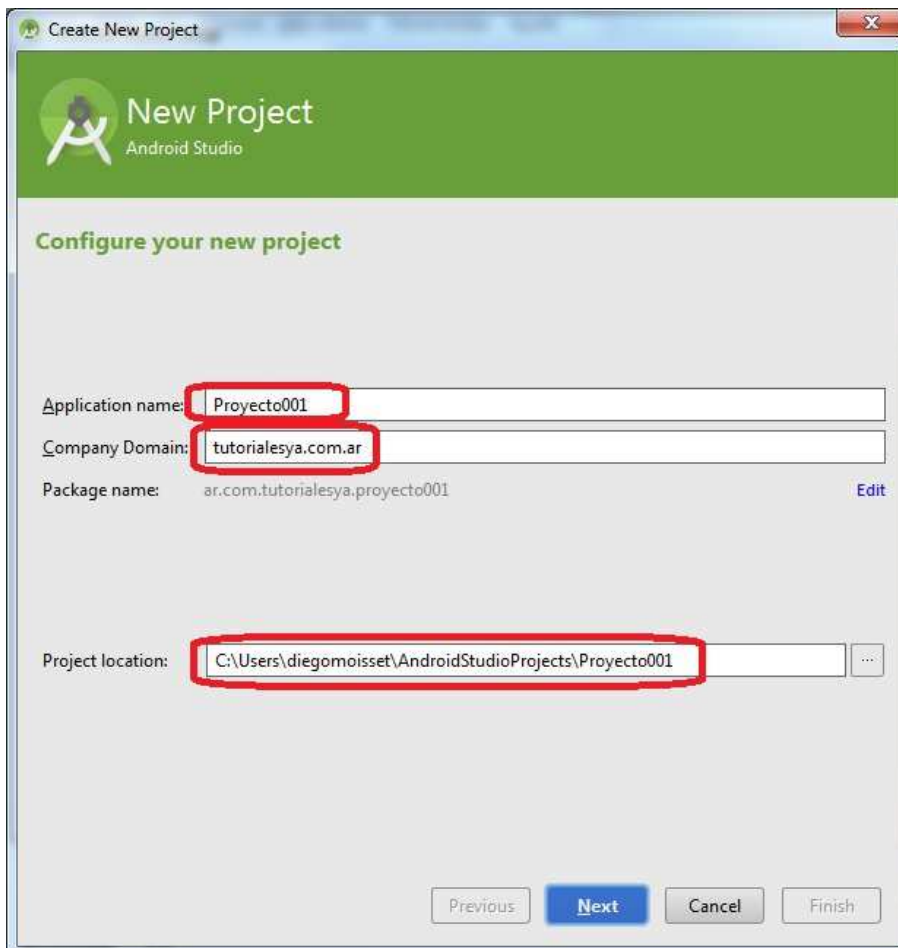
2 - Pasos para crear el primer proyecto Android Studio

Una vez que iniciamos el entorno del Android Studio aparece el diálogo principal:

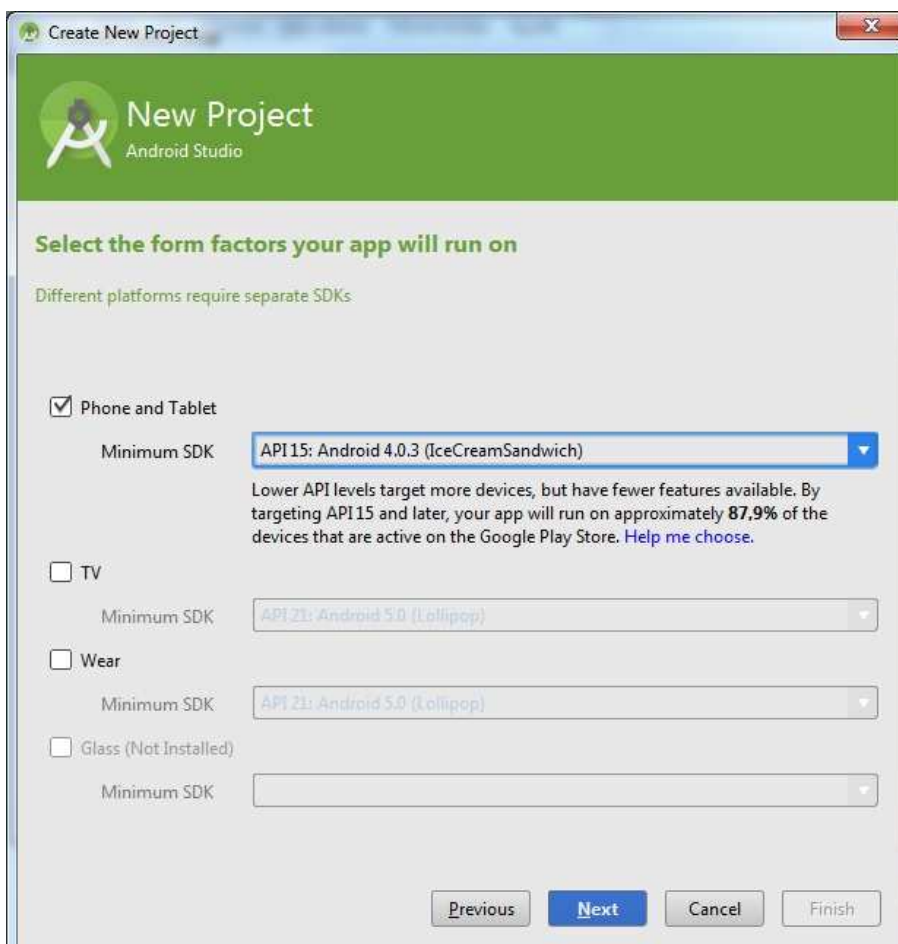


Elegimos la opción "Start a New Android Studio project"

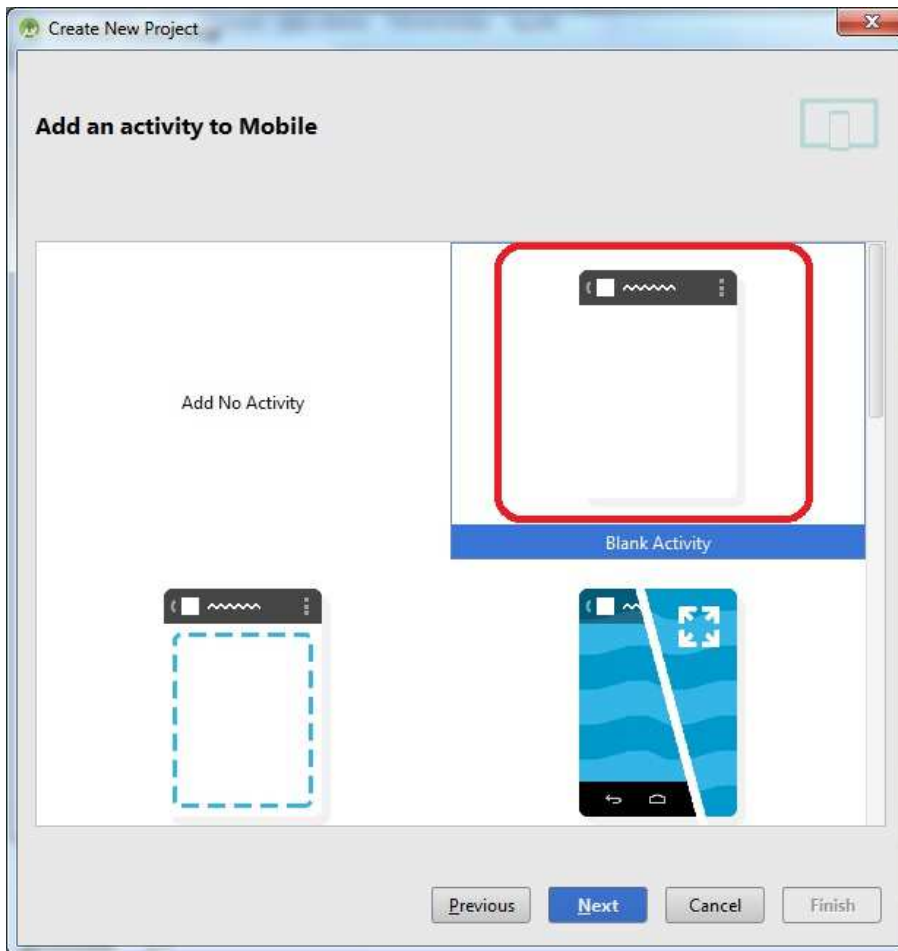
Ahora aparecerán una serie de ventanas para configurar el proyecto, el primer diálogo debemos especificar el Nombre de la aplicación, la url de nuestra empresa (que será el nombre del paquete que asigna java para los archivos fuentes) y la ubicación en el disco de nuestro proyecto:



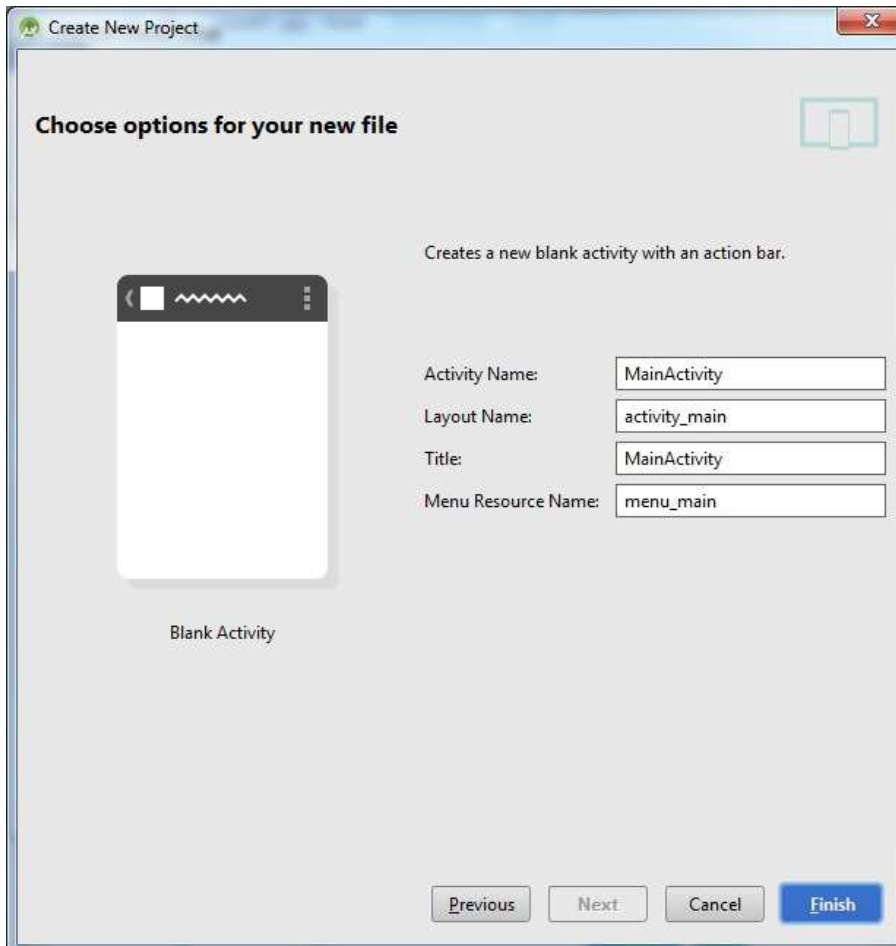
En el segundo diálogo procedemos a especificar la versión de Android mínima donde se ejecutará la aplicación que desarrollaremos (dejaremos la versión 4.0.3):



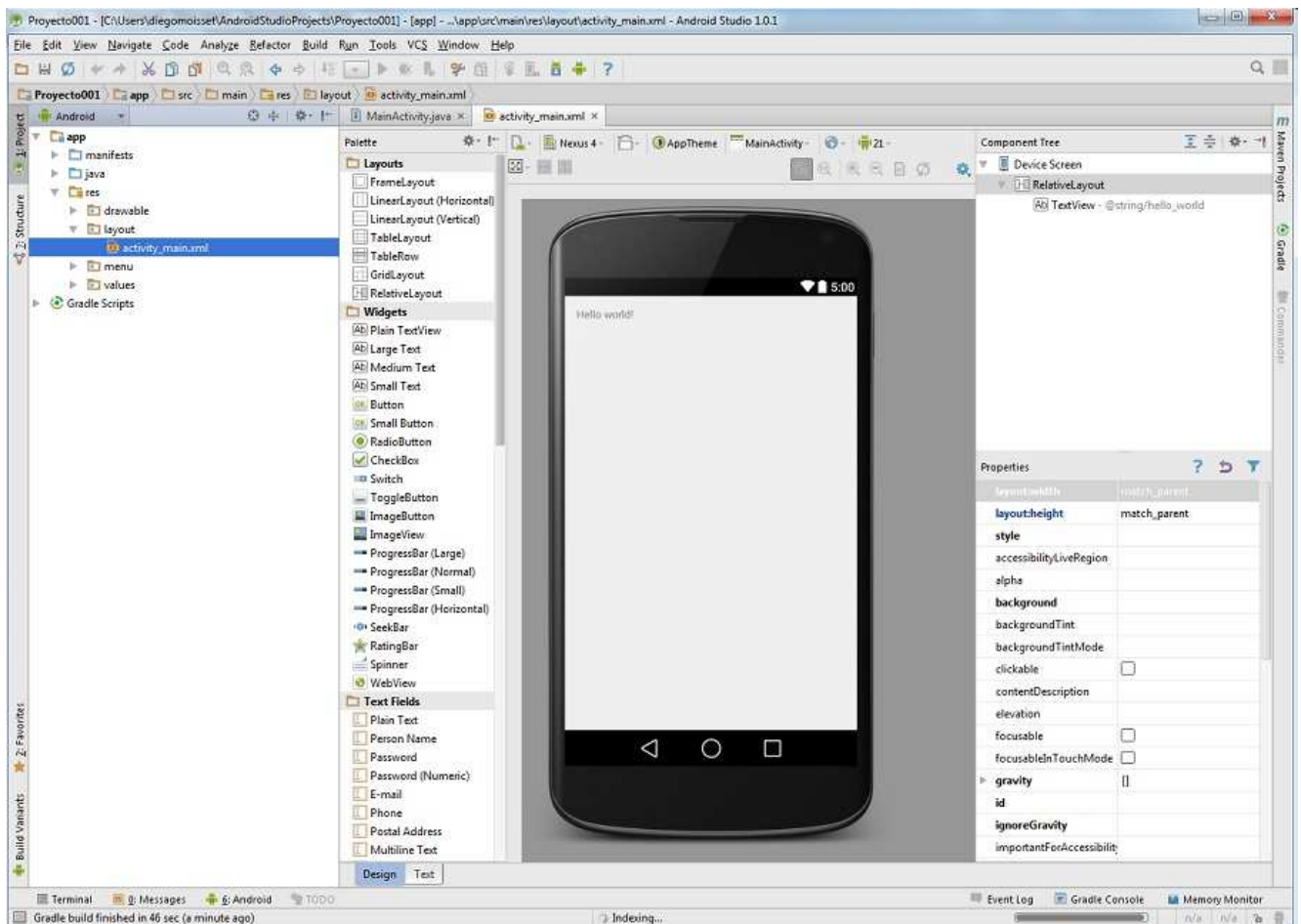
El tercer diálogo especificamos el esqueleto básico de nuestra aplicación, seleccionaremos "Blank Activity" (es decir se generará el código básico para que nuestra aplicación tenga una ventana):



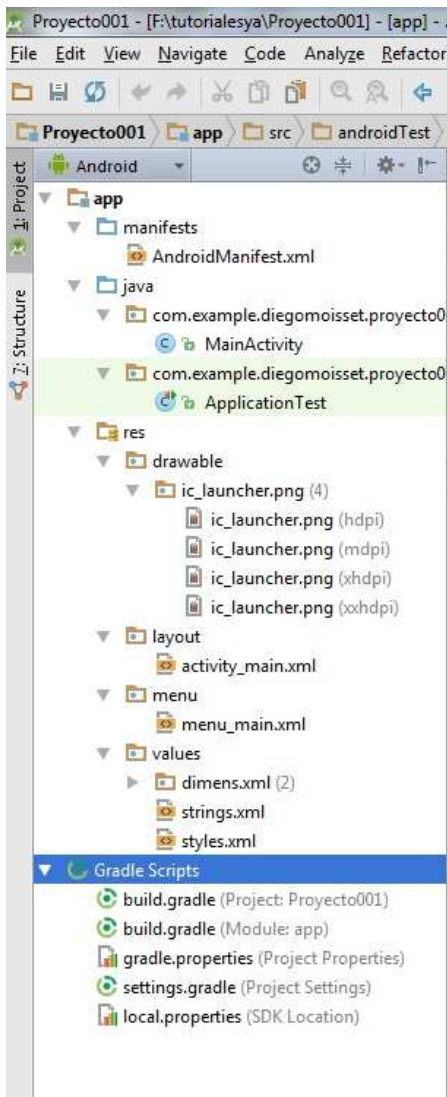
Finalmente el último diálogo tenemos que indicar el nombre de la ventana principal de la aplicación (Activity Name) y otros datos más que veremos a lo largo del curso (dejaremos con los nombres por defecto que propone Android Studio):



Tenemos finalmente creado nuestro primer proyecto en Android Studio y podemos ahora ver el entorno del Android Studio para codificar la aplicación:



El Android Studio nos genera todos los directorios y archivos básicos para iniciar nuestro proyecto, los podemos ver en el lado izquierdo del entorno de desarrollo:

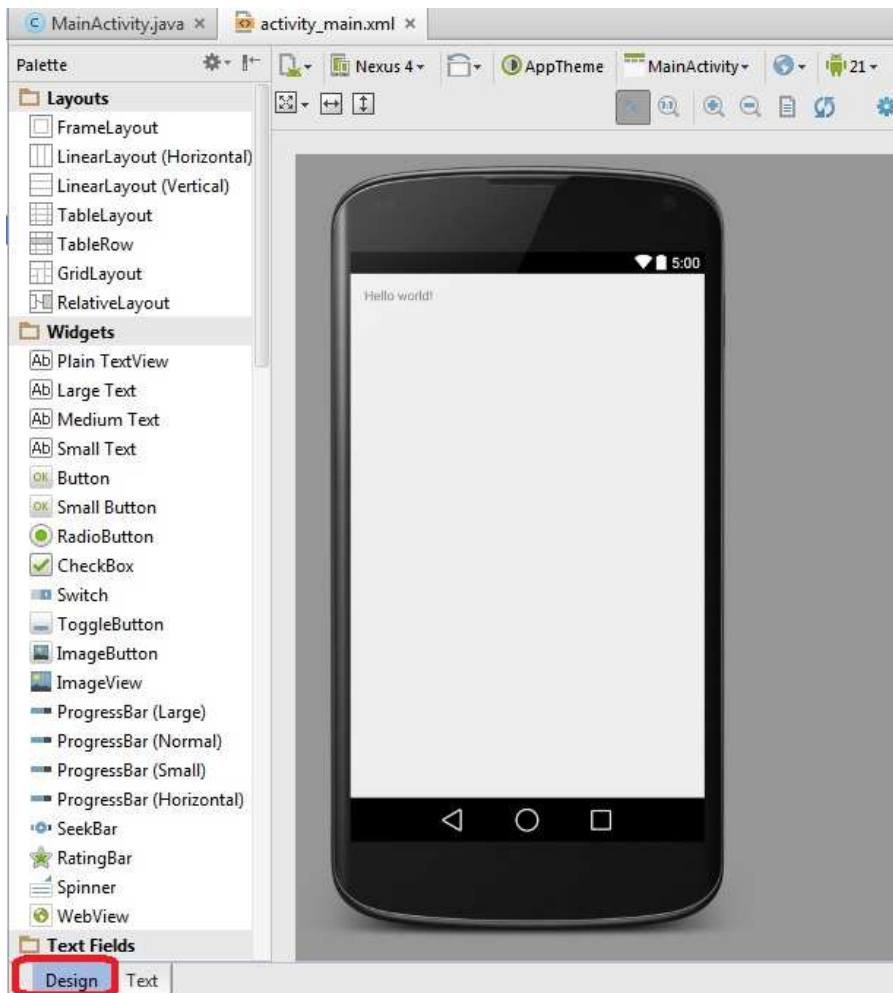


No haremos en este momento un análisis del significado y objetivo de cada uno de estas secciones y archivos generados, sino a medida que avancemos con este curso iremos viendo en forma puntual y profunda.

La interfaz visual de nuestro programa para Android se almacena en un archivo XML en la carpeta res, subcarpeta layout y el archivo se llama activity_main.xml. En esta carpeta tenemos creada nuestra primer pantalla.

Al seleccionar este archivo el Android Studio nos permite visualizar el contenido en "Design" o "Text" (es decir en vista de diseño o en vista de código):

Vista de diseño:



Vista de código:

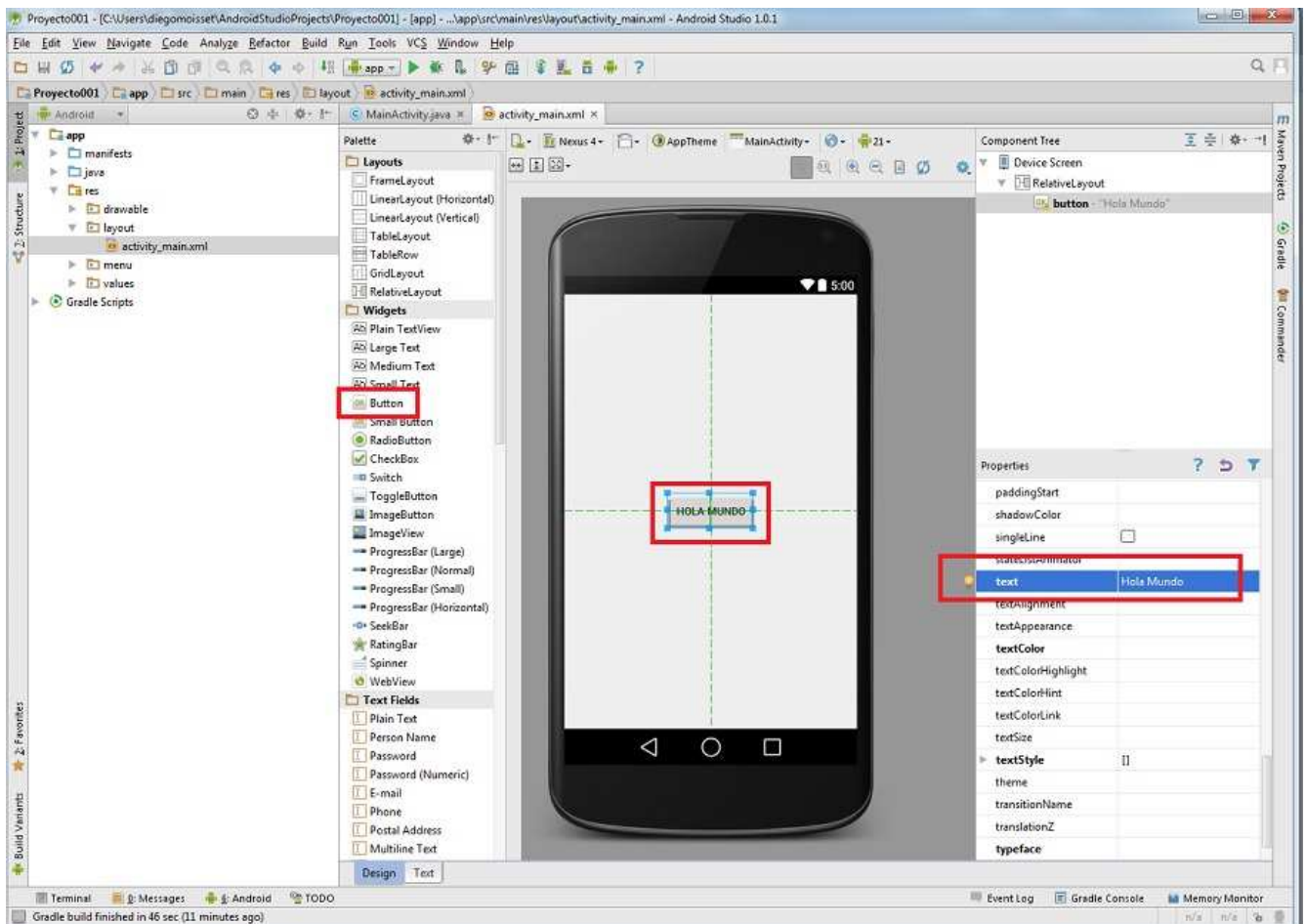


El Android Studio ya insertó un control de tipo RelativeLayout que permite ingresar controles visuales alineados a los bordes y a otros controles que haya en la ventana (más adelante analizaremos este layout)

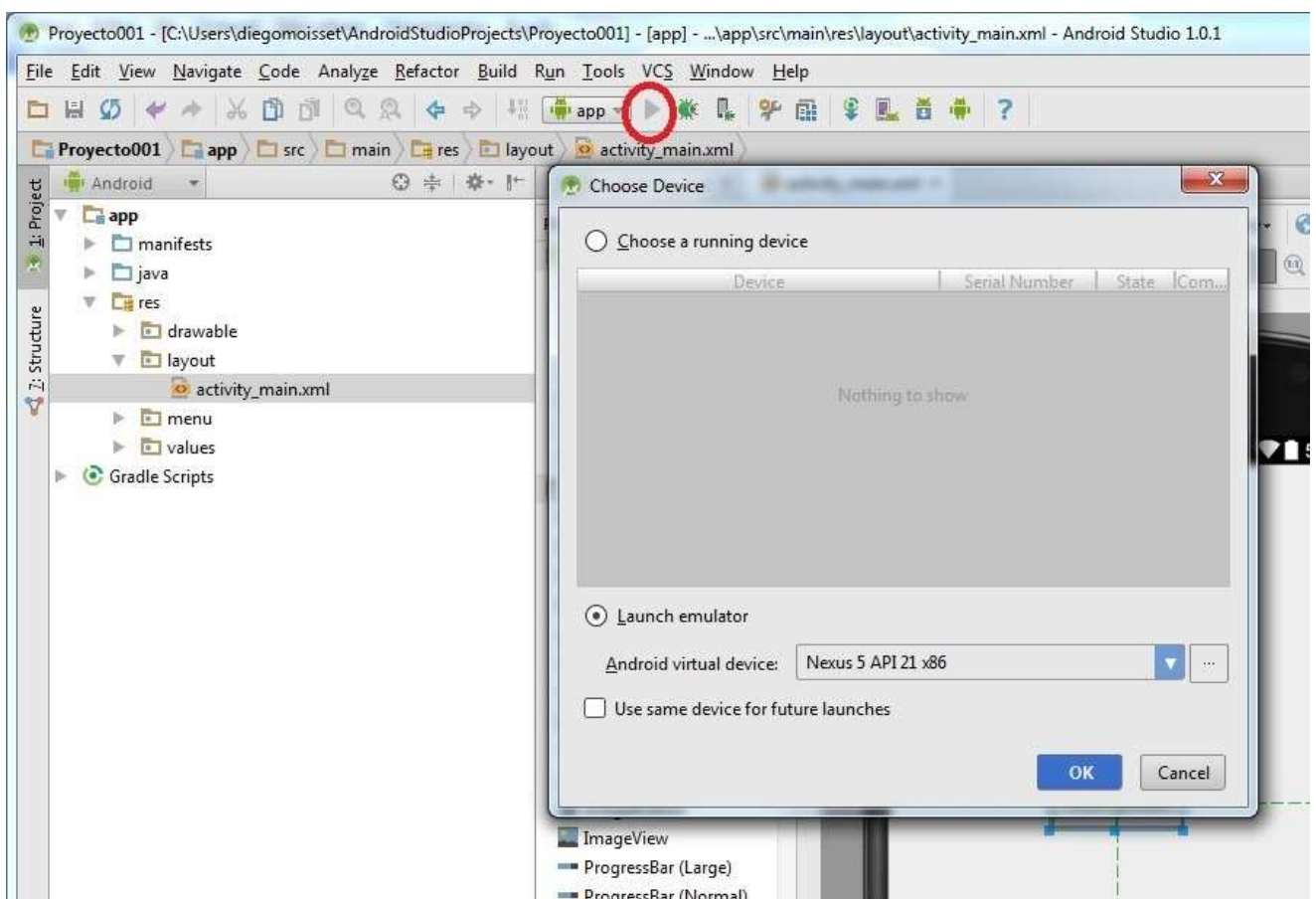
Ya veremos que podemos modificar todo este archivo para que se adapte a la aplicación que queremos desarrollar.

A lo largo de este curso iremos viendo los objetivos de cada una de las secciones que cuenta el Android Studio para implementar la interfaz, codificar en java las funcionalidades de la aplicación etc.

Antes de probar la aplicación en el emulador de un dispositivo Android procederemos a hacer un pequeño cambio a la interfaz que aparece en el celular: borraremos la label que dice "Hello World" (simplemente seleccionando con el mouse dicho elemento y presionando la tecla delete) y de la "Palette" arrastraremos un "Button" al centro del celular y en la ventana "Properties" estando seleccionado el "Button" cambiaremos la propiedad "text" por la cadena "Hola Mundo":



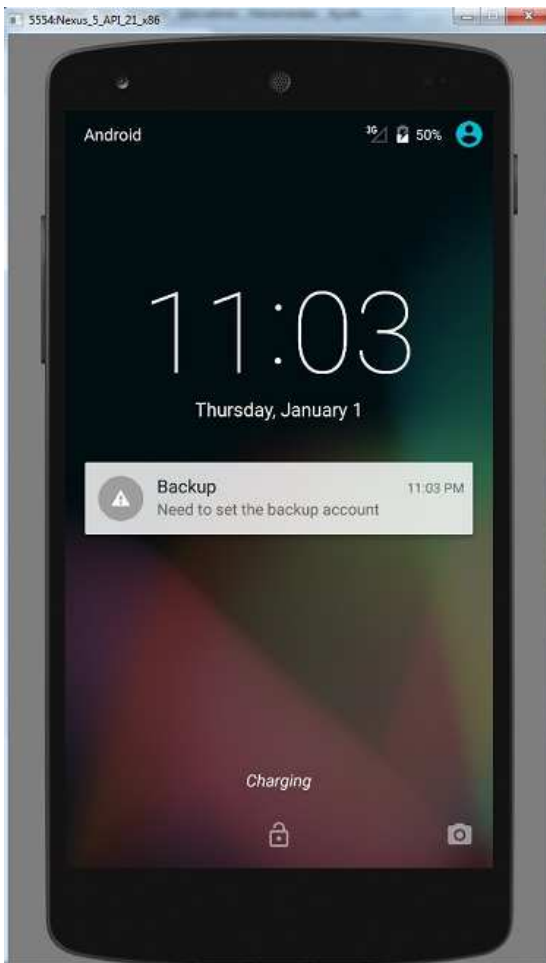
Para ejecutar la aplicación presionamos el triángulo verde o seleccionamos del menú de opciones "Run -> Run app" y en este diálogo procedemos a dejar seleccionado el emulador por defecto que aparece (Nexus 5) y presionamos el botón "OK":



Luego de un rato aparecerá el emulador de Android en pantalla (el arranque del emulador puede llevar más de un

minuto), es **IMPORTANTE** tener en cuenta que una vez que el emulador se ha arrancado no lo debemos cerrar cada vez que hacemos cambios en nuestra aplicación o codificamos otras aplicaciones, sino que volvemos a ejecutar la aplicación con los cambios y al estar el emulador corriendo el tiempo que tarda hasta que aparece nuestro programa en el emulador es muy reducido.

Cuando terminó de cargarse el emulador debe aparecer la interfaz del mismo:



Procedemos a desbloquear la pantalla de inicio y podremos observar la ejecución de nuestra primer aplicación en Android:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto001.zip](#)

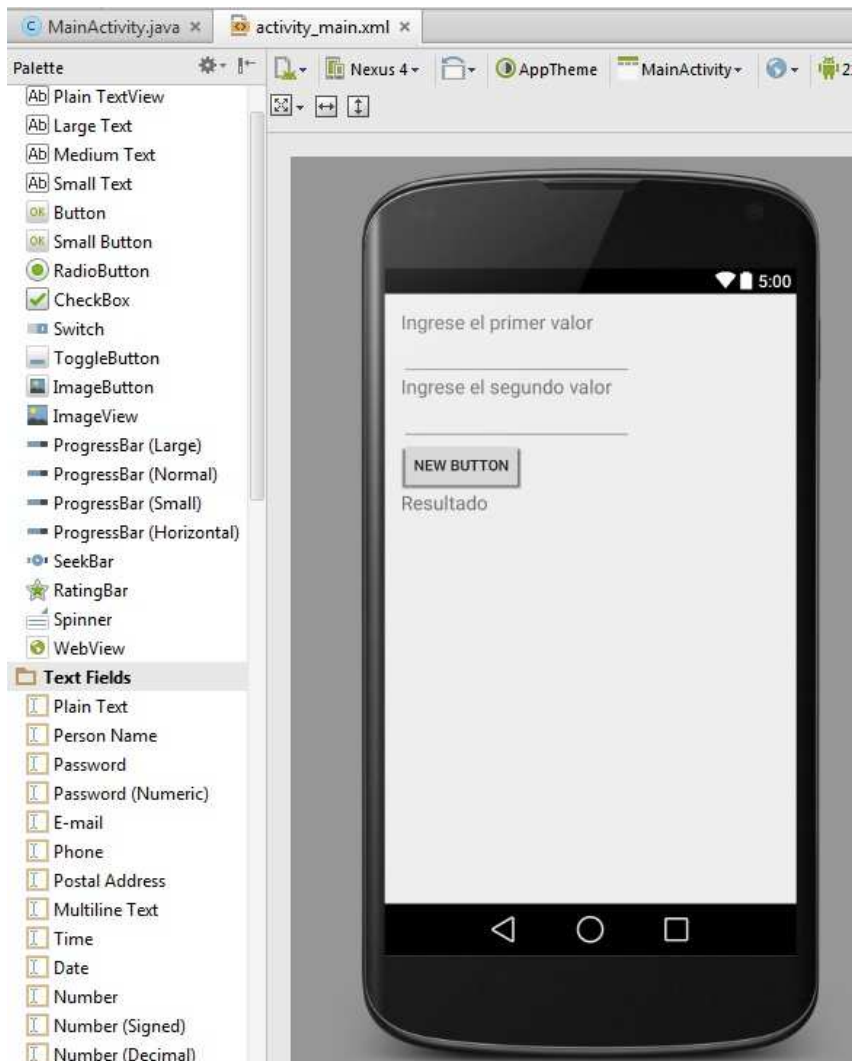
[Retornar](#)

3 - Capturar el clic de un botón

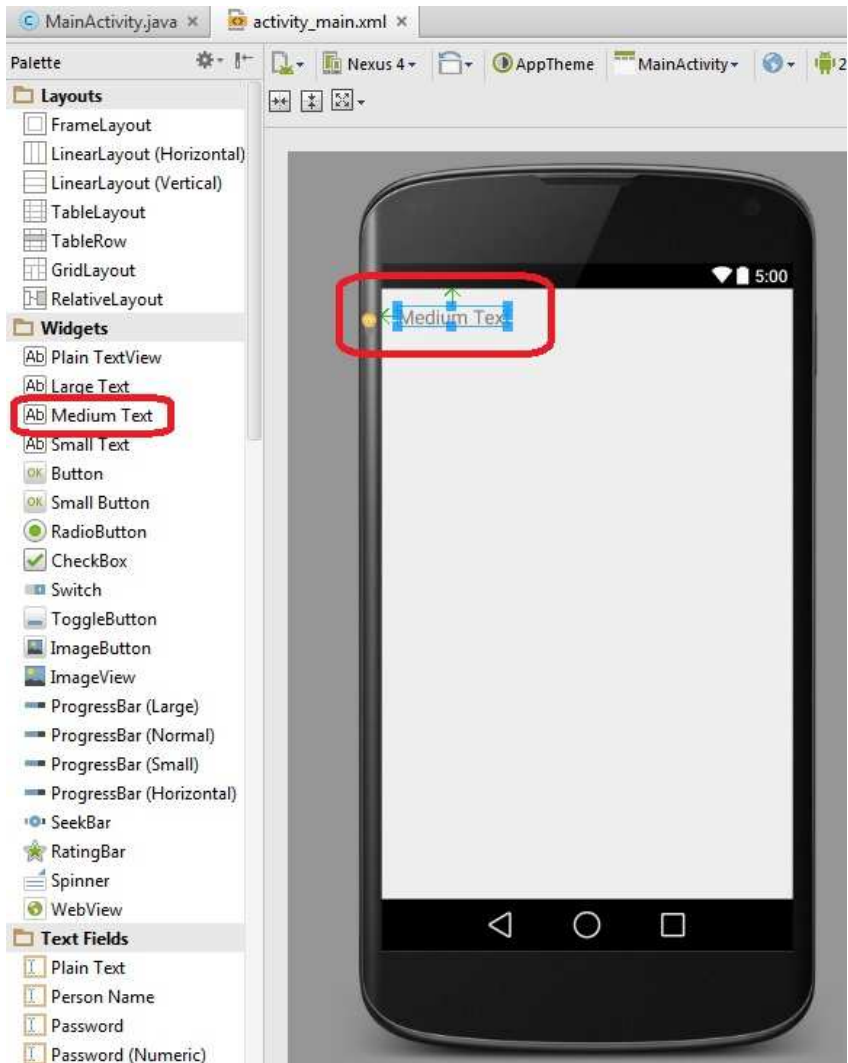
Problema:

Confeccionar un programa que permita la carga de dos números en controles de tipo EditText. Mostrar mensajes que soliciten la carga de los valores. Disponer un Button para sumar los dos valores ingresados. Mostrar el resultado en un tercer TextView.

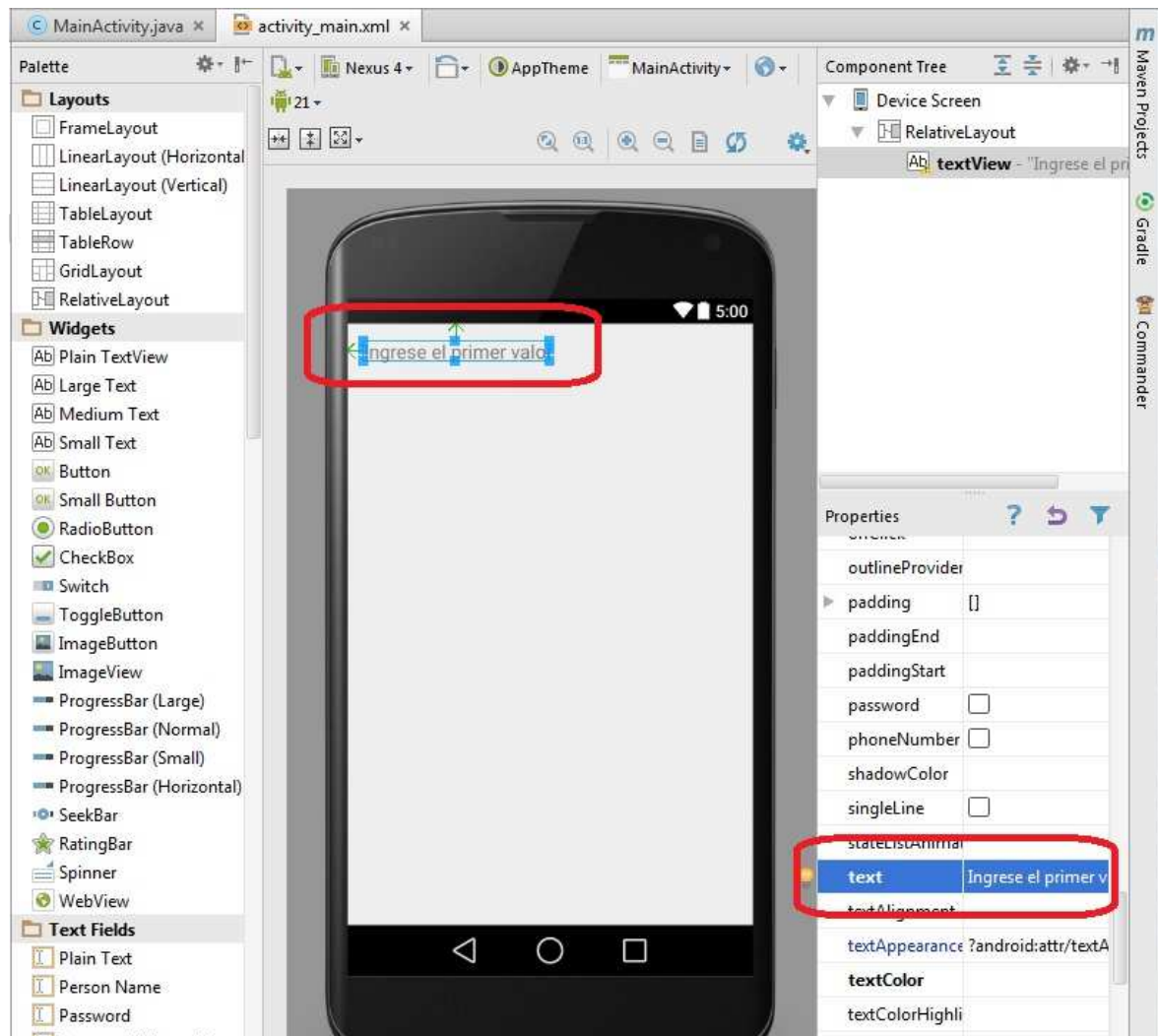
La interfaz visual debe quedar algo semejante a esto:



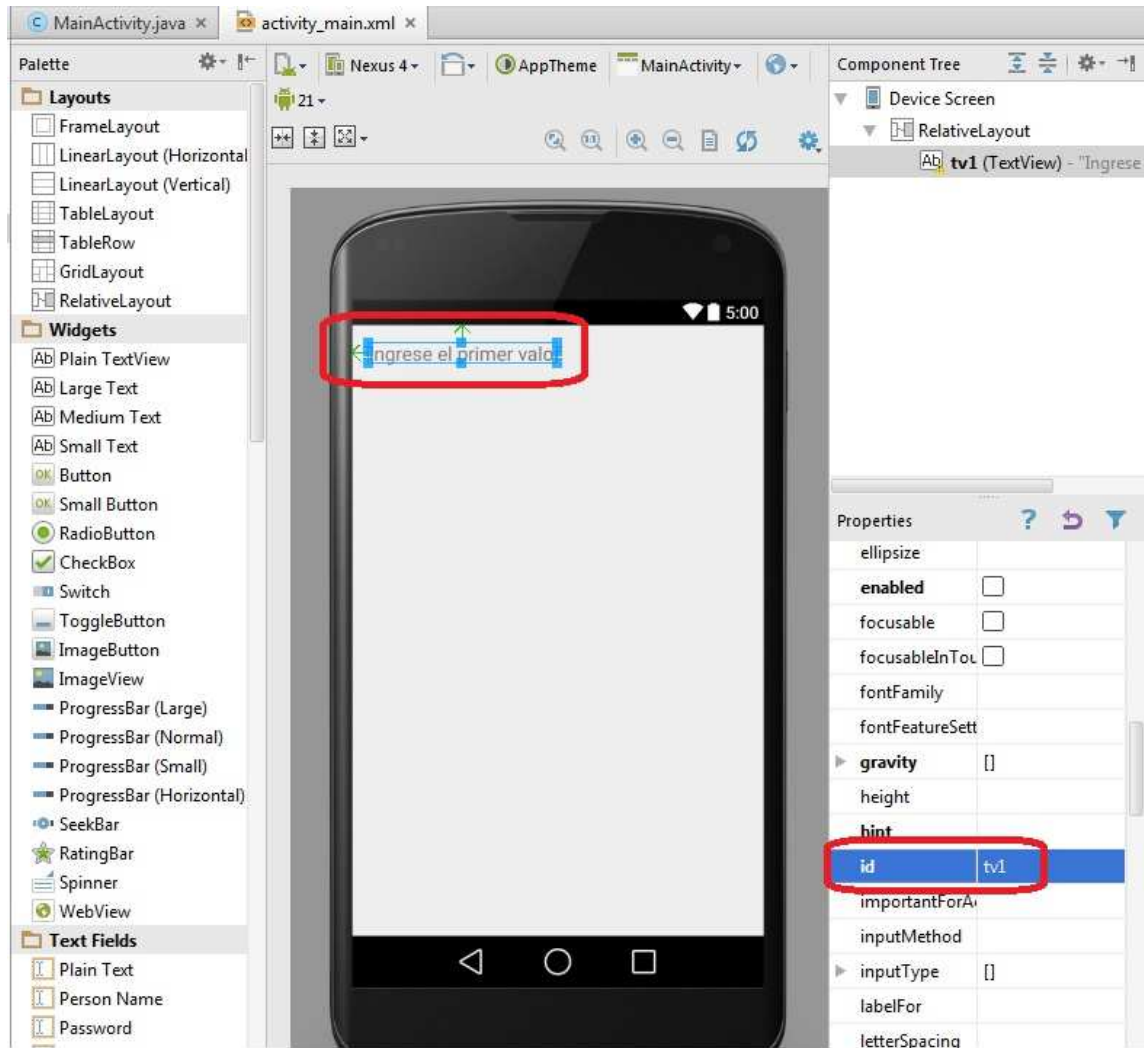
Veamos paso a paso como creamos la interfaz visual de nuestro programa. Primero borramos el TextView que aparece por defecto cuando se crea un proyecto con el Android Studio. Ahora desde la ventana "Palette" seleccionamos de la pestaña "Widgets" el control "Medium Text" (es de la clase TextView) y lo arrastramos a la ventana de diseño de nuestra interfaz a la parte superior izquierda:



Ahora lo seleccionamos y en la ventana de propiedades (Properties) especificamos la propiedad text (como habíamos visto anteriormente) disponemos el texto "Ingresa el primer valor:"

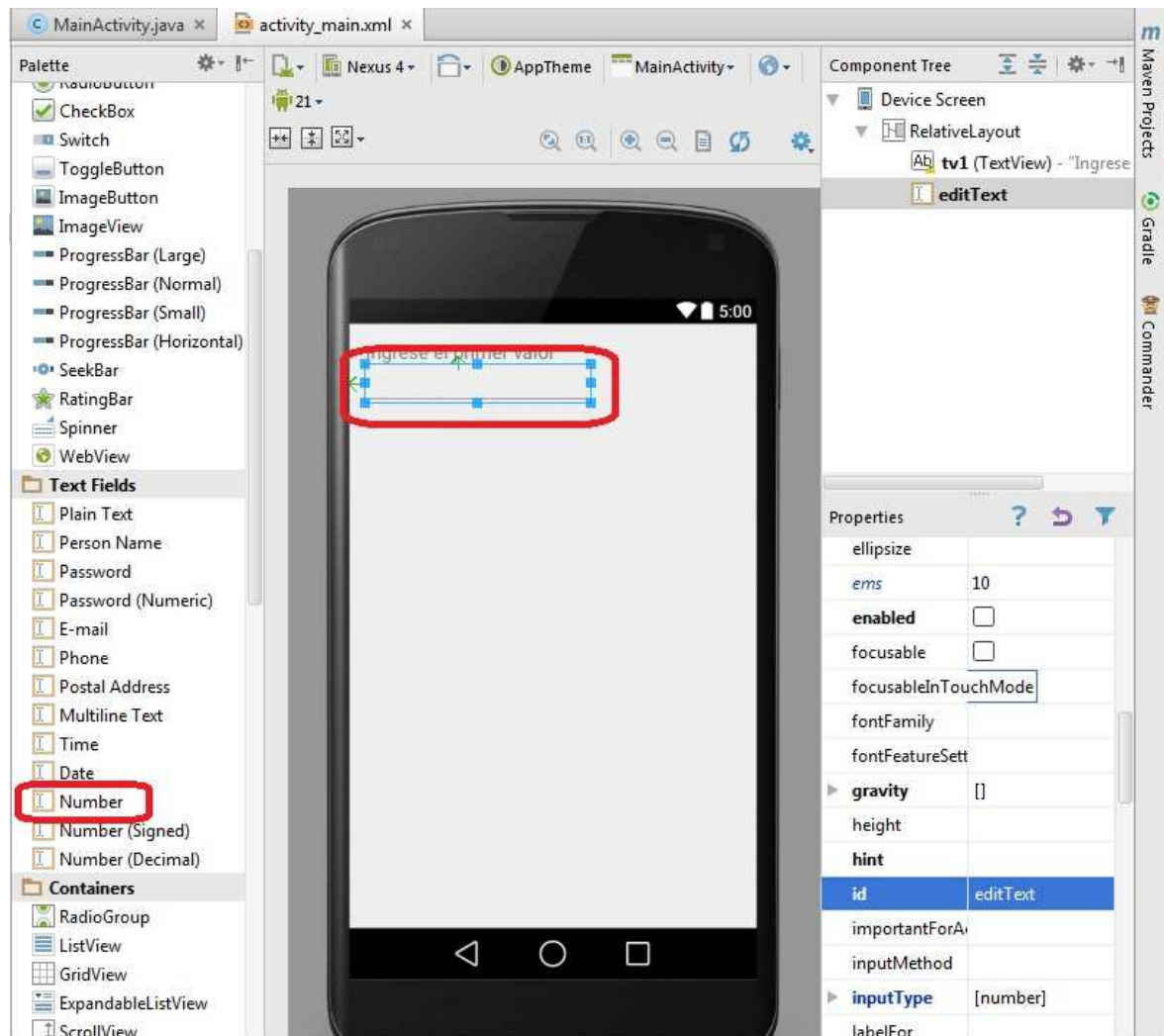


También vamos a especificar la propiedad "id", y le asignaremos el valor tv1



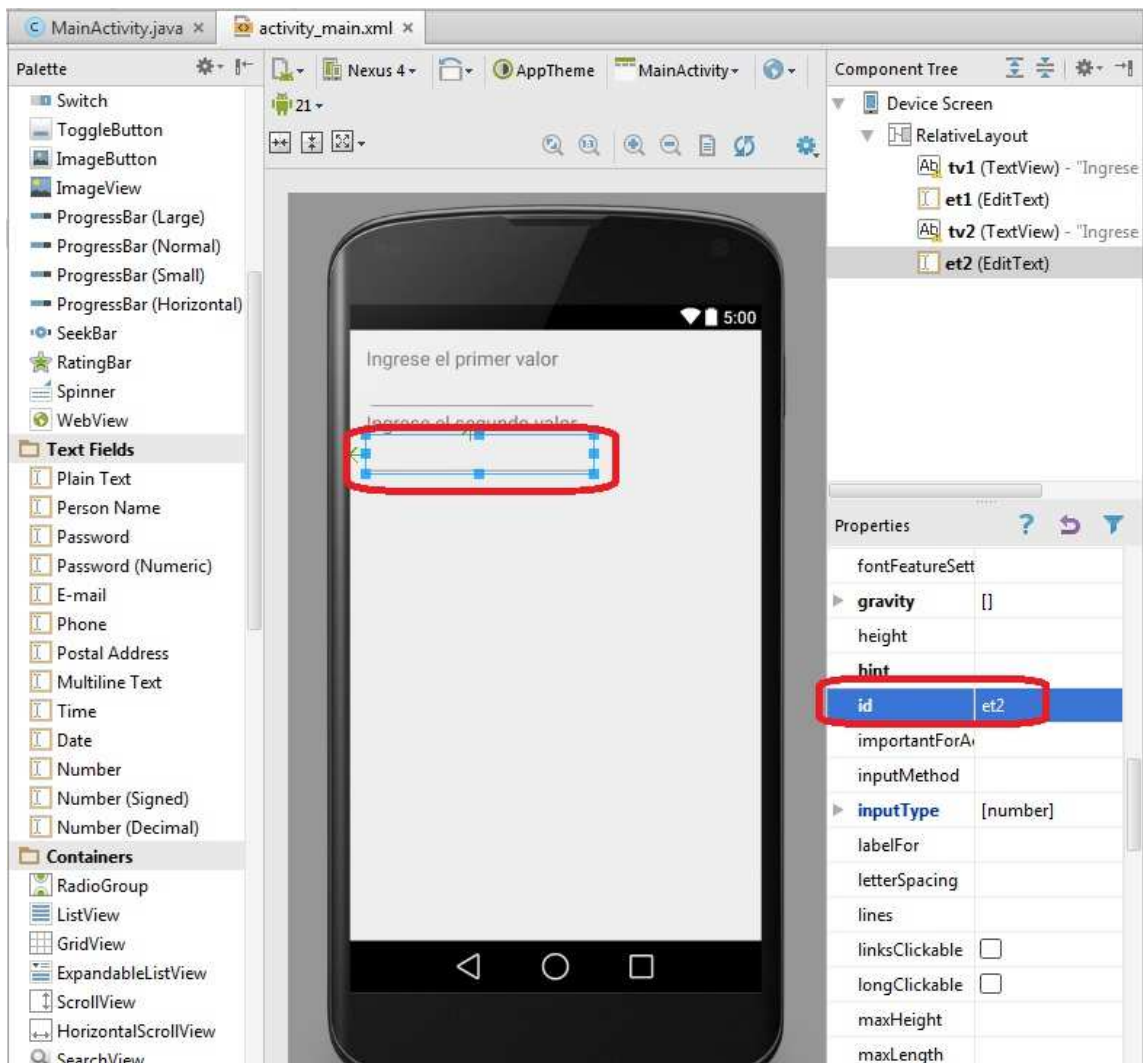
Hemos entonces asignado como nombre a este objeto: tv1 (recordemos que se trata de un objeto de la clase TextView)

Ahora de la paleta de componentes seleccionamos la pestaña "Text Fields" y arrastramos el "Number" (pertenece a la clase EditText)

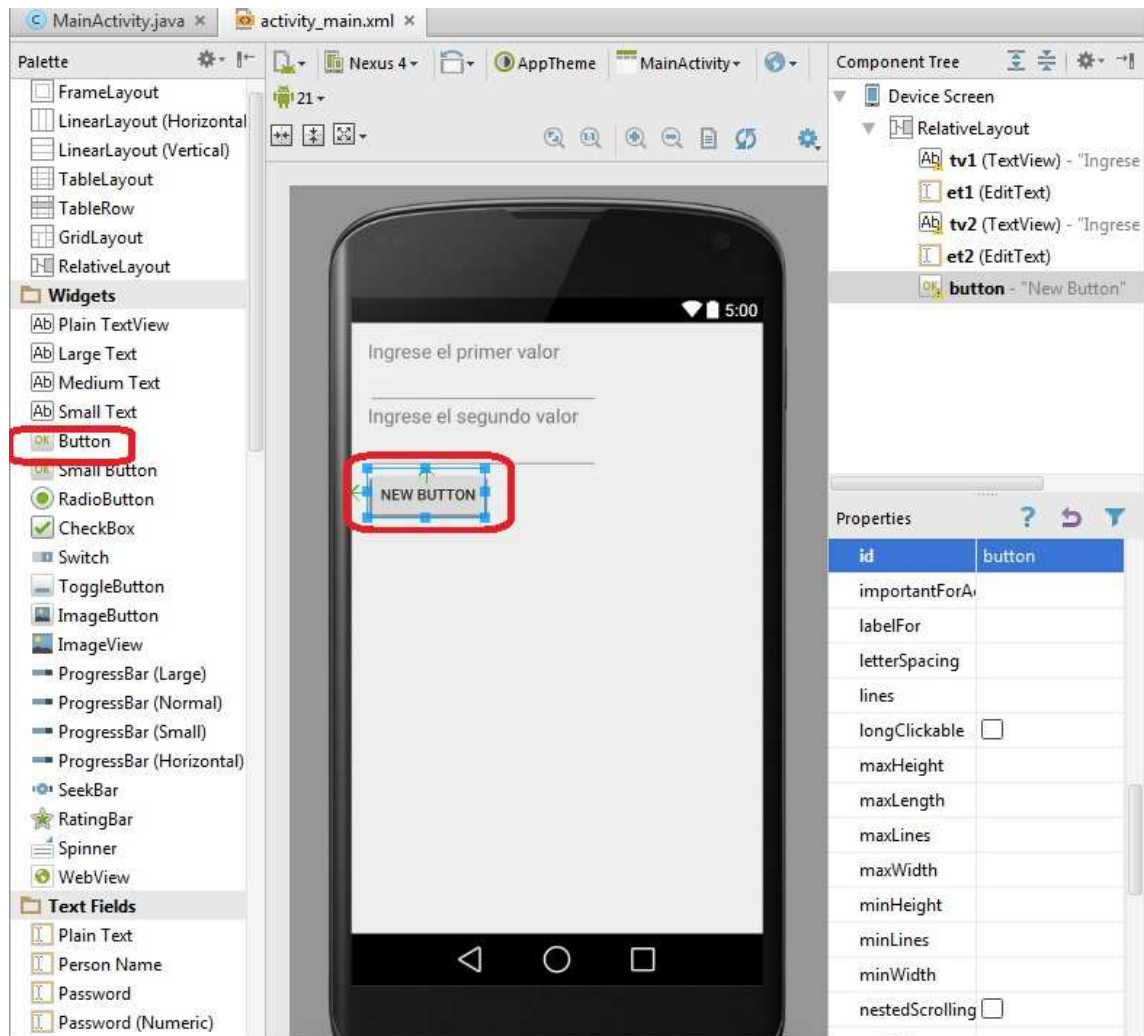


En la ventana de propiedades estando seleccionado el EditText configuramos la propiedad "id" con el nombre et1 (este nombre haremos referencia posteriormente desde Java)

Efectuamos los mismos pasos para crear el segundo "Medium Text" y "Number" (inicializamos las propiedades respectivas) Definimos los id con los nombres tv2 y et2, el resultado visual debe ser algo semejante a esto:

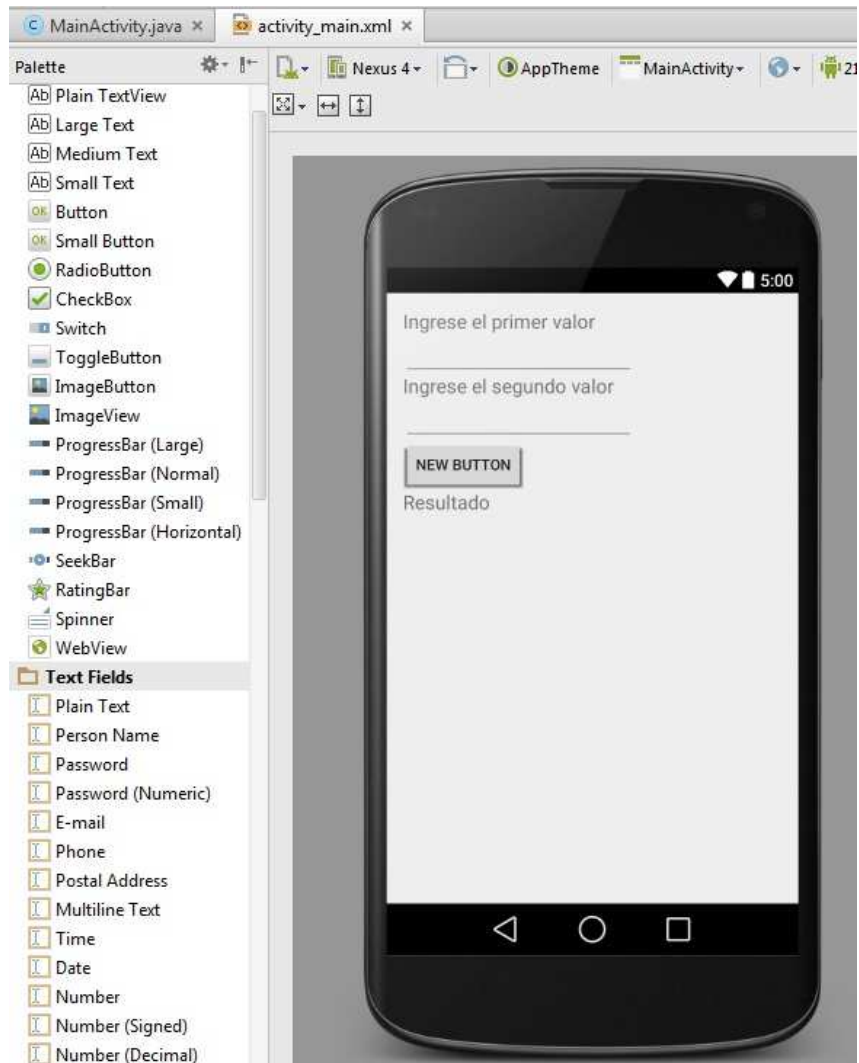


De la pestaña "Widgets" arrastramos un control de tipo "Button":



Inicializamos la propiedad text con el texto "Sumar" y la propiedad id la dejamos con el valor ya creado llamado "button".

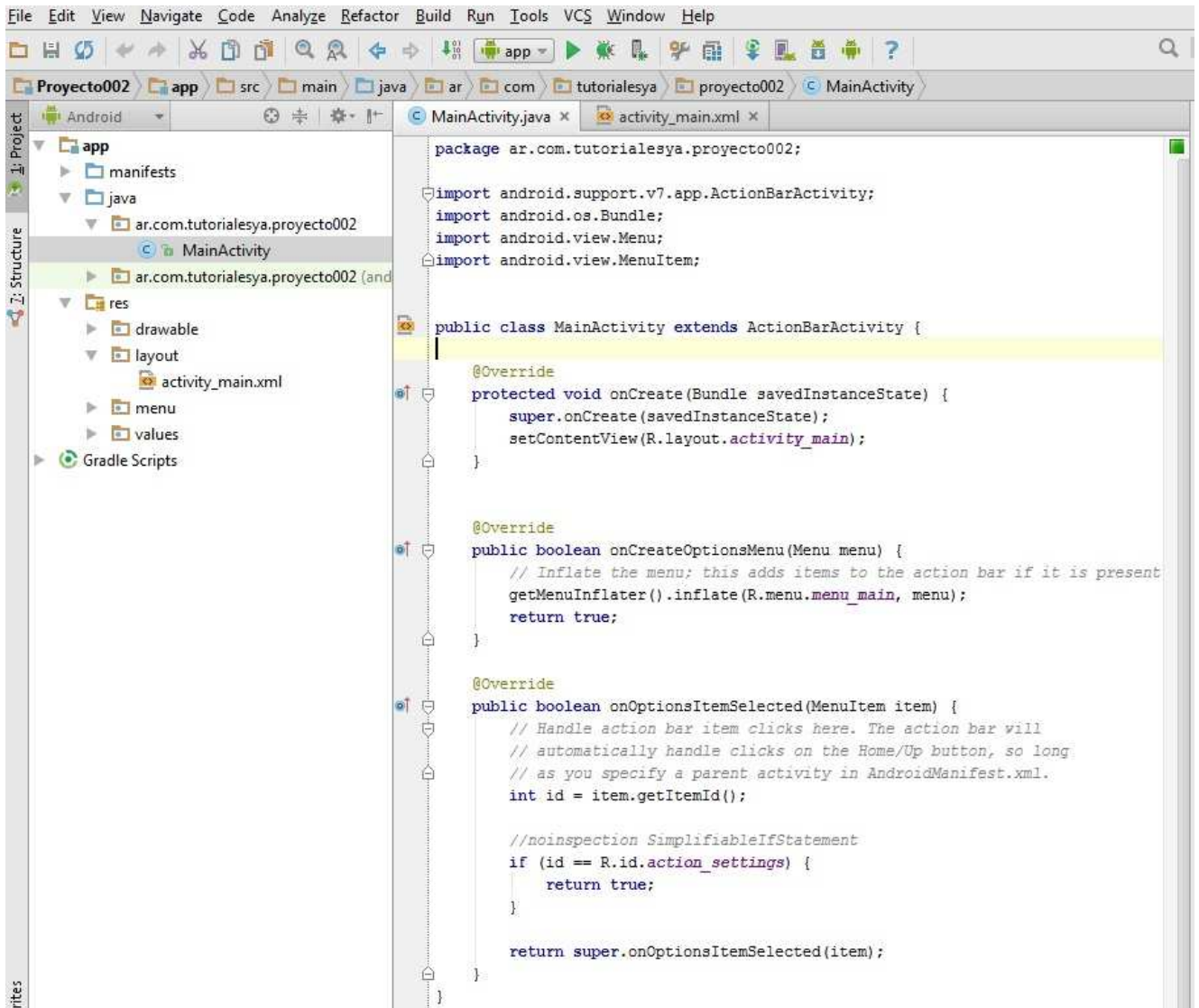
Para terminar con nuestra interfaz visual arrastramos un tercer objeto de tipo "Medium Text" (es de la clase TextView) y definimos la propiedad id con el valor "tv3" y la propiedad text con el texto "Resultado", recordemos que la interfaz final debe ser semejante a esta:



Si en este momento ejecutamos la aplicación aparece la interfaz visual correctamente pero cuando presionemos el botón no mostrará la suma.

Hasta ahora hemos trabajado solo con el archivo xml (activity_main.xml) donde se definen los controles visuales de la ventana que estamos creando.

Abrimos seguidamente el archivo MainActivity.java que lo podemos ubicar en la carpeta app\java\ar\com\tutorialesya\proyecto002\MainActivity:



La clase MainActivity hereda de la clase ActionBarActivity. La clase ActionBarActivity representa una ventana de Android y tiene todos los métodos necesarios para crear y mostrar los objetos que hemos dispuesto en el archivo xml.

El código fuente de la clase MainActivity.java es:

```

package ar.com.tutorialesya.proyecto002;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}

```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}
}

```

Como mínimo se debe sobrescribir el método onCreate heredado de la clase ActionBarActivity donde procedemos a llamar al método setContentView pasando como referencia una valor almacenado en una constante llamada activity_main contenida en una clase llamada layout que a su vez la contiene una clase llamada R (veremos más adelante que el Android Studio se encarga de crear la clase R en forma automática y sirve como puente entre el archivo xml y nuestra clase MainActivity)

Luego veremos los otros métodos onCreateOptionsMenu y onOptionsItemSelected.

Captura de eventos.

Ahora viene la parte donde definimos variables en java donde almacenamos las referencias a los objetos definidos en el archivo XML.

Definimos tres variables, dos de tipo EditText y finalmente una de tipo TextView (estas dos clases se declaran en el paquete android.widget, es necesario importar dichas clases para poder definir las variables de dichas clases, la forma más fácil de importar las clases es una vez que definimos el objeto por ejemplo private TextView tv3 veremos que aparece en rojo el nombre de la clase y si disponemos el cursor sobre el nombre de la clase y presionamos las teclas "Alt" y "Enter" luego el Android Studio codifica automáticamente la línea que importa la clase: import android.widget.TextView;):

```

package ar.com.tutorialesya.proyecto002;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2;
    private TextView tv3;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.

```



```

        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Recordar que la forma más fácil de importar las clases EditText y TextView es tipear las dos líneas:

```

private EditText et1,et2;
private TextView tv3;

```

y luego presionar las teclas "Alt" y "Enter" en cada nombre de clase que se debe importar
Esto hace que se escriban automáticamente los import:

```

import android.widget.EditText;
import android.widget.TextView;

```

Los nombres que le dí a los objetos en este caso coinciden con la propiedad id (no es obligatorio):

```

private EditText et1,et2;
private TextView tv3;

```

No definimos TextView para los dos mensajes "Ingrese el primer valor" e "Ingrese el segundo valor" ya que no necesitaremos interactuar con ellos. También veremos que el objeto de la clase Button no es necesario definir un atributo sino que desde el archivo XML inicializaremos la propiedad onClick.

En el método onCreate debemos enlazar estas variables con los objetos definidos en el archivo XML, esto se hace llamando al método findViewById:

```

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

    et1=(EditText)findViewById(R.id.et1);
    et2=(EditText)findViewById(R.id.et2);
    tv3=(TextView)findViewById(R.id.tv3);
}

```

Al método findViewById debemos pasar la constante creada en la clase R (recordemos que se crea automáticamente esta clase) el nombre de la constante si debe ser igual con el nombre de la propiedad del objeto creado en el archivo XML. Como la clase findViewById retorna una clase de tipo View luego debemos utilizar el operador cast (es decir le antecedemos entre paréntesis el nombre de la clase)

Ya tenemos almacenados en las variables las referencias a los tres objetos que se crean al llamar al método: setContentView(R.layout.main); .

Ahora planteamos el método que se ejecutará cuando se presione el botón (el método debe recibir como parámetro un objeto de la clase View) En nuestro ejemplo lo llamé sumar:

```

package ar.com.tutorialesya.proyecto002;

```

```

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2;
    private TextView tv3;

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

        et1=(EditText)findViewById(R.id.et1);
        et2=(EditText)findViewById(R.id.et2);
        tv3=(TextView)findViewById(R.id.tv3);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //Este método se ejecutará cuando se presione el botón
    public void sumar(View view) {

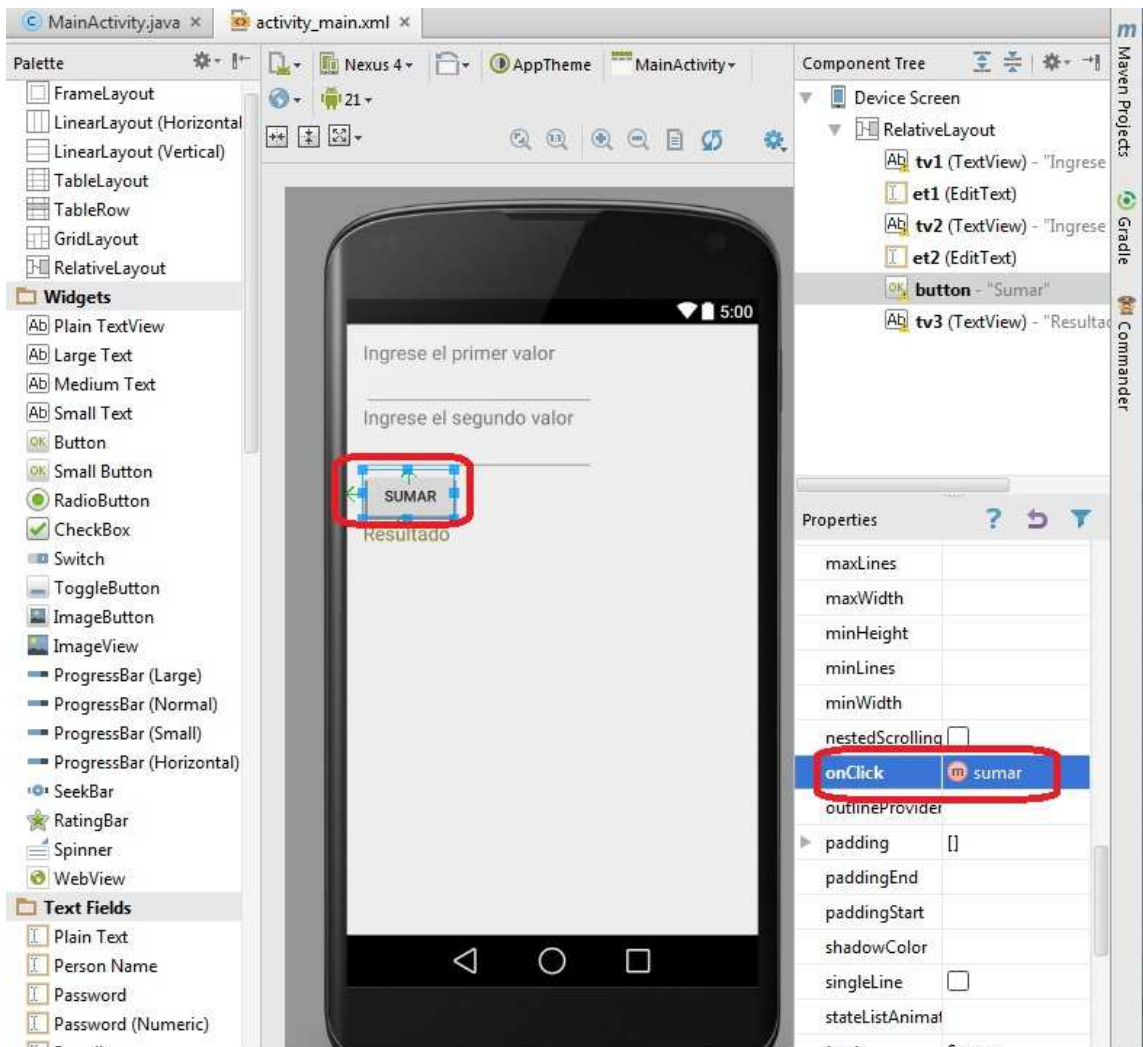
    }

}

```

Debemos importar la clase View ("Alt" y "Enter" estando el cursor sobre el nombre de la clase "View")

Ahora debemos ir al archivo XML e inicializar la propiedad onClick del objeto button con el nombre del método que acabamos de crear (este paso es fundamental para que el objeto de la clase Button pueda llamar al método sumar que acabamos de crear):



Finalmente implementaremos la lógica para sumar los dos valores ingresados en los controles EditText:

```
public void sumar(View view) {
    String valor1=et1.getText().toString();
    String valor2=et2.getText().toString();
    int nro1=Integer.parseInt(valor1);
    int nro2=Integer.parseInt(valor2);
    int suma=nro1+nro2;
    String resu=String.valueOf(suma);
    tv3.setText(resu);
}
```

Extraemos el texto de los dos controles de tipo EditText y los almacenamos en dos variables locales de tipo String. Convertimos los String a tipo entero, los sumamos y el resultado lo enviamos al TextView donde se muestra la suma (previo a convertir la suma a String)

La clase completa queda entonces como:

```
package ar.com.tutorialesya.proyecto002;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {
```

```

private EditText et1,et2;
private TextView tv3;

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

    et1=(EditText)findViewById(R.id.et1);
    et2=(EditText)findViewById(R.id.et2);
    tv3=(TextView)findViewById(R.id.tv3);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

//Este método se ejecutará cuando se presione el botón
public void sumar(View view) {
    String valor1=et1.getText().toString();
    String valor2=et2.getText().toString();
    int nro1=Integer.parseInt(valor1);
    int nro2=Integer.parseInt(valor2);
    int suma=nro1+nro2;
    String resu=String.valueOf(suma);
    tv3.setText(resu);
}
}

```

Si ejecutamos nuestro programa podemos ver ahora que luego de cargar dos valores al presionar el botón aparece en el tercer TextView el resultado de la suma de los dos EditText:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto002.zip](#)

[Retornar](#)

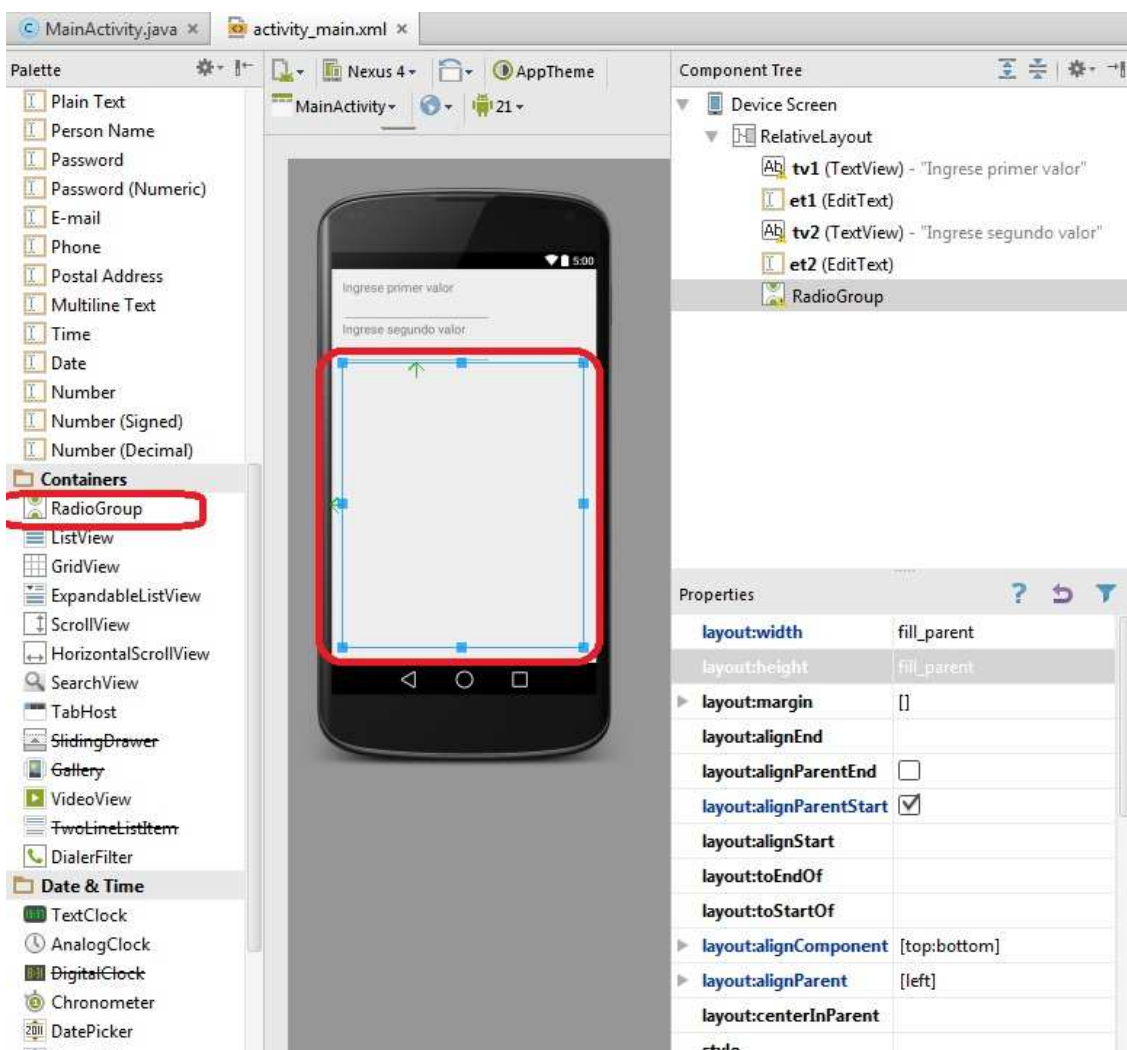
4 - Controles RadioGroup y RadioButton

El objetivo de este concepto es practicar la implementación de un programa que requiera controles de tipo RadioButton para seleccionar una actividad. Aprenderemos como agrupar un conjunto de RadioButton y verificar cual está seleccionado.

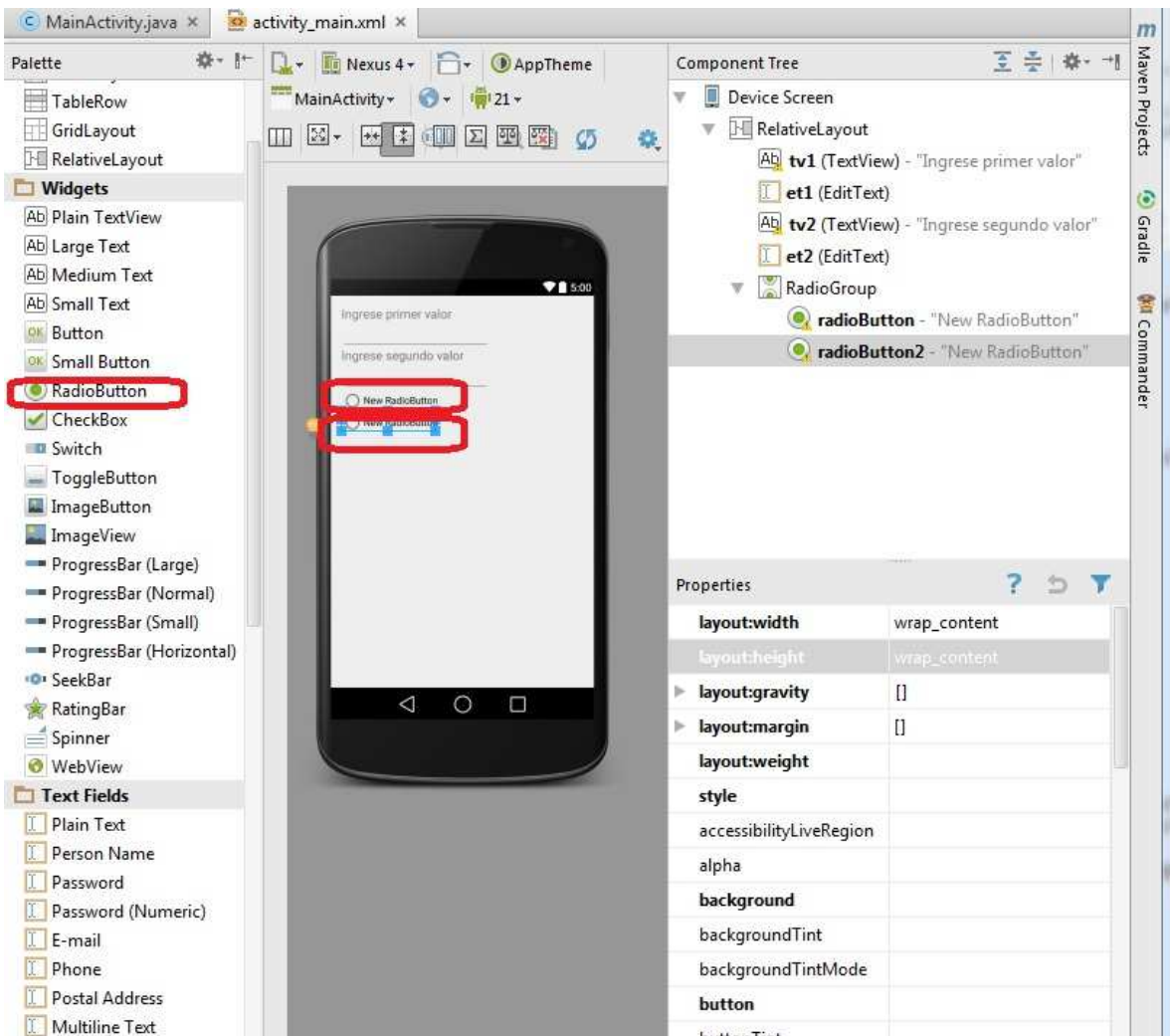
Problema:

Realizar la carga de dos números en controles de tipo EditText. Mostrar un mensaje que solicite la carga de los valores. Disponer dos controles de tipo RadioButton para seleccionar si queremos sumar o restar dichos valores. Finalmente mediante un control de tipo Button efectuamos la operación respectiva. Mostramos el resultado en un TextView.

El problema es similar al anterior. Para disponer los controles de tipo RadioButton debemos en realidad primero insertar un control de tipo RadioGroup (este control se encuentra en la paleta de componentes en la pestaña Containers de la "Palette"):

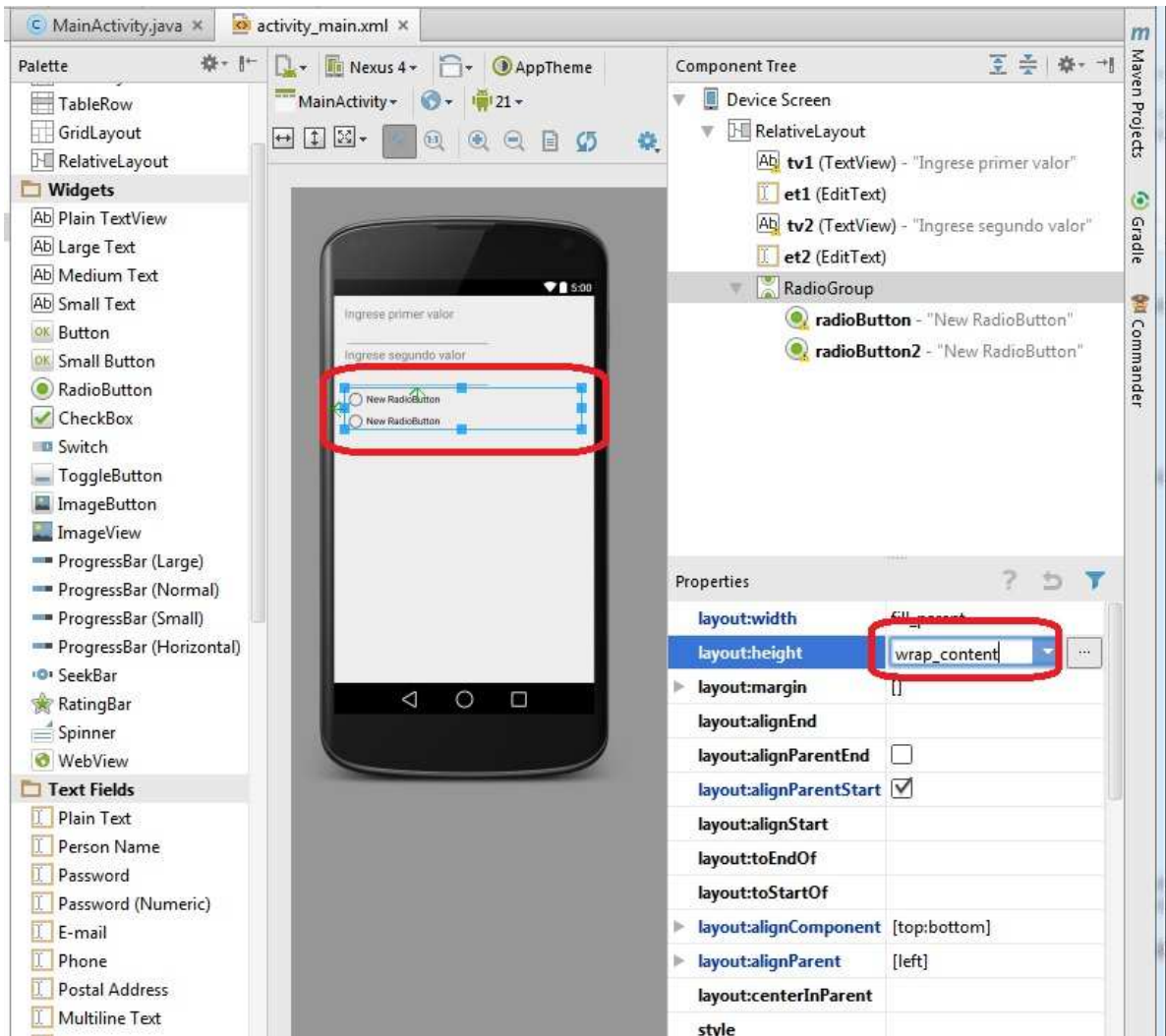


Ahora debemos arrastrar dos controles de la clase RadioButton de la pestaña "Widgets" dentro del RadioGroup



Nuestro problema solo requiere dos controles de tipo RadioButton.

Otra cosa muy importante es seleccionar nuevamente el control RadioGroup y cambiar la propiedad layout:height con el valor wrap_parent (para que el control RadioGroup solo ocupe el espacio de los dos controles RadioButton):



Ahora a los dos controles de tipo RadioButton definimos sus id (los llamaremos r1 y r2 respectivamente)
Cambiamos sus propiedades text por los textos "sumar" y "restar".

No olvidemos también cambiar los id de los controles EditText por et1 y et2 (igual que en el problema anterior)

Por último agreguemos un botón y un TextView para mostrar el resultado
Inicializamos las propiedades del botón con los valores:

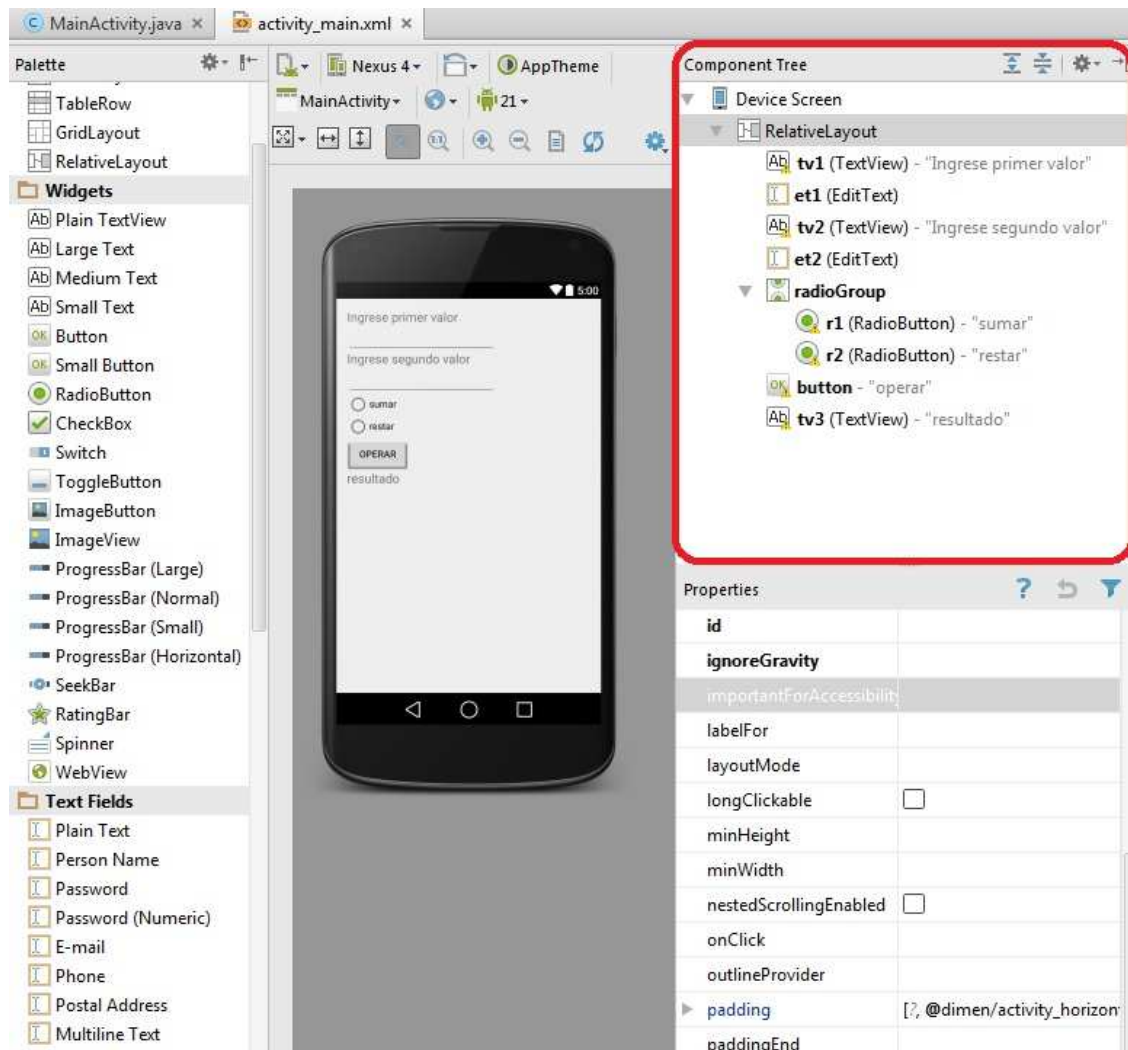
```
id : button
text : operar
```

Y el tercer "Medium Text" (TextView) con los valores:

```
id : tv3
text : resultado
```

Podemos controlar en la ventana "Component Tree" el id definido para cada control (tv1, et1, tv2, et2, radioGroup, r1, r2, button, tv3)

También podemos observar de que clase es cada control visual y el texto de la propiedad text para aquellos controles que tienen sentido su inicialización.



Captura del evento clic del button e identificación del RadioButton seleccionado.

El código fuente de la clase MainActivity es:

```
package ar.com.tutorialesya.proyecto003;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private EditText et1, et2;
    private TextView tv3;
    private RadioButton r1, r2;

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

        et1=(EditText) findViewById(R.id.et1);
        et2=(EditText) findViewById(R.id.et2);
```

```

        tv3=(TextView)findViewById(R.id.tv3);
        r1=(RadioButton)findViewById(R.id.r1);
        r2=(RadioButton)findViewById(R.id.r2);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //Este método se ejecutará cuando se presione el botón
    public void operar(View view) {
        String valor1=et1.getText().toString();
        String valor2=et2.getText().toString();
        int nro1=Integer.parseInt(valor1);
        int nro2=Integer.parseInt(valor2);
        if (r1.isChecked()==true) {
            int suma=nro1+nro2;
            String resu=String.valueOf(suma);
            tv3.setText(resu);
        } else
        if (r2.isChecked()==true) {
            int resta=nro1-nro2;
            String resu=String.valueOf(resta);
            tv3.setText(resu);
        }
    }
}

```

Primero debemos enlazar el objeto button con el método operar. Para esto similar al problema anterior seleccionamos el control button y cambiamos la propiedad onClick por el valor operar (si no hacemos esto nunca se ejecutará el método operar de la clase MainActivity)

Como podemos ver el código fuente es igual al problema anterior. Tenemos dos objetos más que debemos inicializar en el método onCreate:

```

r1=(RadioButton)findViewById(R.id.r1);
r2=(RadioButton)findViewById(R.id.r2);

```

Las variables r1 y r2 son de la clase RadioButton y son necesarios en el método operar para verificar cual de los dos RadioButton están seleccionados. La clase RadioButton tiene un método llamado isChecked que retorna true si dicho elemento está seleccionado:


```
public void operar(View view) {  
    String valor1=et1.getText().toString();  
    String valor2=et2.getText().toString();  
    int nro1=Integer.parseInt(valor1);  
    int nro2=Integer.parseInt(valor2);  
    if (r1.isChecked()==true) {  
        int suma=nro1+nro2;  
        String resu=String.valueOf(suma);  
        tv3.setText(resu);  
    } else  
        if (r2.isChecked()==true) {  
            int resta=nro1-nro2;  
            String resu=String.valueOf(resta);  
            tv3.setText(resu);  
        }  
}
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto003.zip](#)

[Retornar](#)

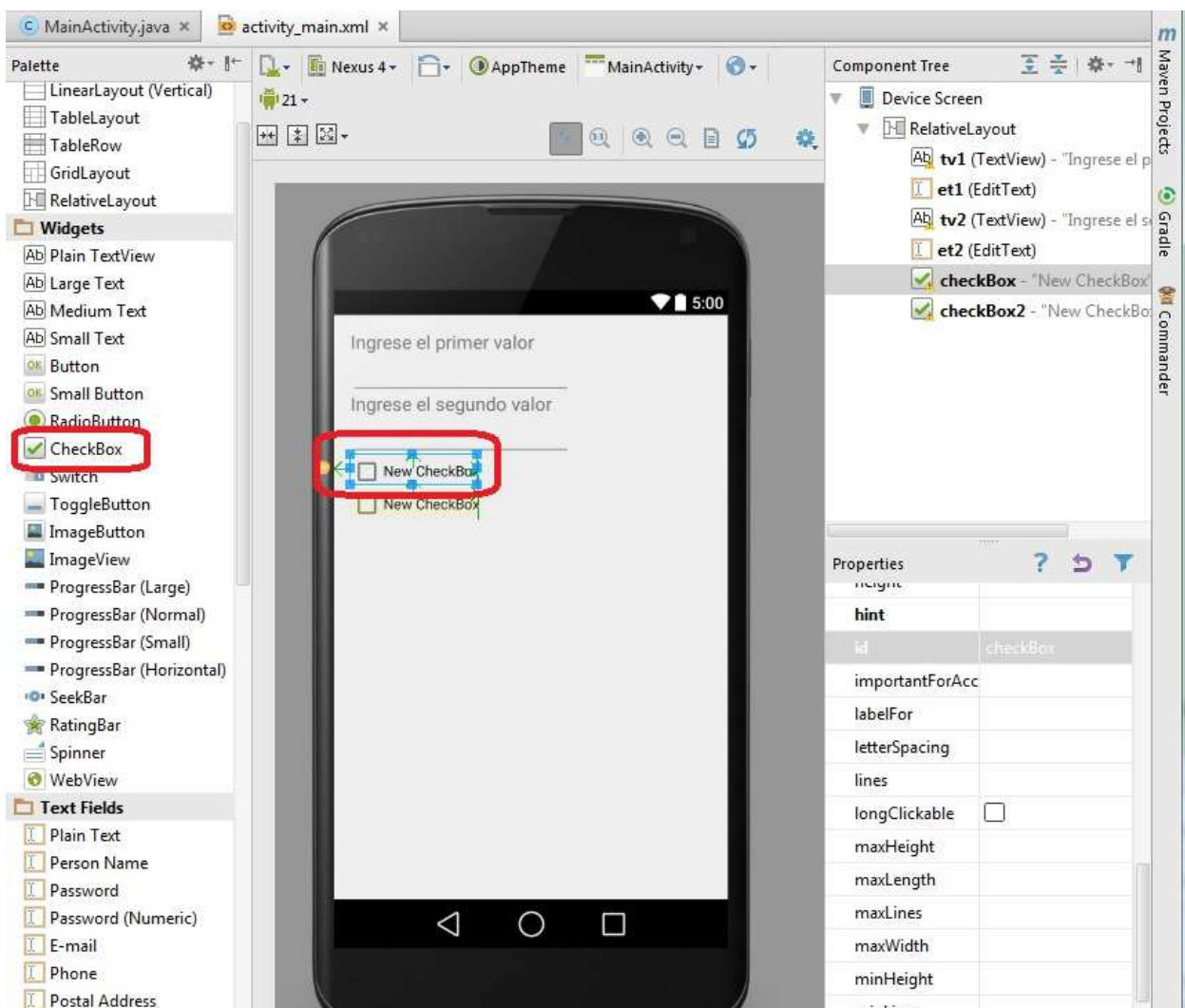
5 - Control CheckBox

El objetivo de este concepto es seguir practicando lo visto hasta ahora para la creación de un proyecto con Android Studio e incorporar el control visual CheckBox

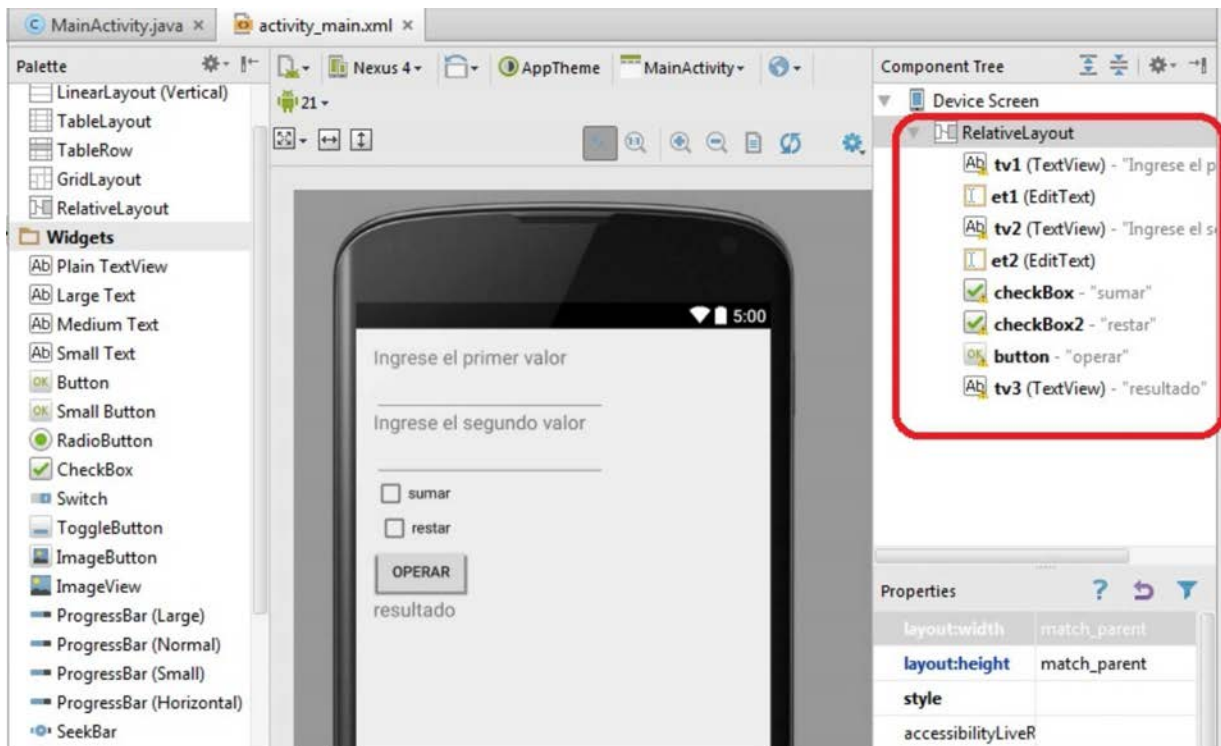
Problema:

Realizar la carga de dos números en controles de tipo EditText ("Number"). Mostrar un mensaje que solicite la carga de los valores. Disponer dos controles de tipo CheckBox para seleccionar si queremos sumar y/o restar dichos valores. Finalmente mediante un control de tipo Button efectuamos la operación respectiva. Mostramos el o los resultados en un TextView ("Medium Text").

Lo nuevo en este problema es la inserción de dos objetos de la clase CheckBox que se encuentra en la pestaña "Widgets":



Luego la interfaz gráfica final para este problema y los nombres de los controles los podemos ver en la ventana "Component Tree":



No olvidemos inicializar la propiedad `onClick` del objeto `button` con el valor "operar" (es el nombre del método a ejecutarse cuando se presione el botón y lo implementa la clase que hacemos)

Código fuente:

```
package ar.com.tutorialesya.proyecto004;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2;
    private TextView tv3;
    private CheckBox checkBox1,checkBox2;

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

        et1=(EditText) findViewById(R.id.et1);
        et2=(EditText) findViewById(R.id.et2);
```

```

        tv3=(TextView) findViewById(R.id.tv3);
        checkBox1=(CheckBox) findViewById(R.id.checkBox);
        checkBox2=(CheckBox) findViewById(R.id.checkBox2);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //Este método se ejecutará cuando se presione el botón
    public void operar(View view) {
        String valor1=et1.getText().toString();
        String valor2=et2.getText().toString();
        int nro1=Integer.parseInt(valor1);
        int nro2=Integer.parseInt(valor2);
        String resu="";
        if (checkBox1.isChecked()==true) {
            int suma=nro1+nro2;
            resu="La suma es: "+ suma;
        }
        if (checkBox2.isChecked()==true) {
            int resta=nro1-nro2;
            resu=resu + " La resta es: "+ resta;
        }
        tv3.setText(resu);
    }
}

```

Definimos dos objetos de la clase CheckBox como atributos de la clase:

```
private CheckBox checkBox1,checkBox2;
```

En el método onCreate los inicializamos con los objetos definidos en el archivo XML:

```
checkBox1=(CheckBox)findViewById(R.id.checkBox);
checkBox2=(CheckBox)findViewById(R.id.checkBox2);
```

En el método operar debemos definir dos if a la misma altura ya que los dos controles de tipo CheckBox pueden estar seleccionados simultáneamente. Definimos una variable de tipo String y la inicializamos con cadena vacía para el caso en que los dos CheckBox no estén seleccionados:

```
String resu="";
if (checkBox1.isChecked()==true) {
    int suma=nro1+nro2;
```

```
        resu="La suma es: "+ suma;
    }
    if (checkBox2.isChecked()==true) {
        int resta=nro1-nro2;
        resu=resu + " La resta es: "+ resta;
    }
    tv3.setText(resu);
```

Cuando ejecutamos el programa en el emulador tenemos como resultado:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto004.zip](#)

[Retornar](#)

6 - Control Spinner

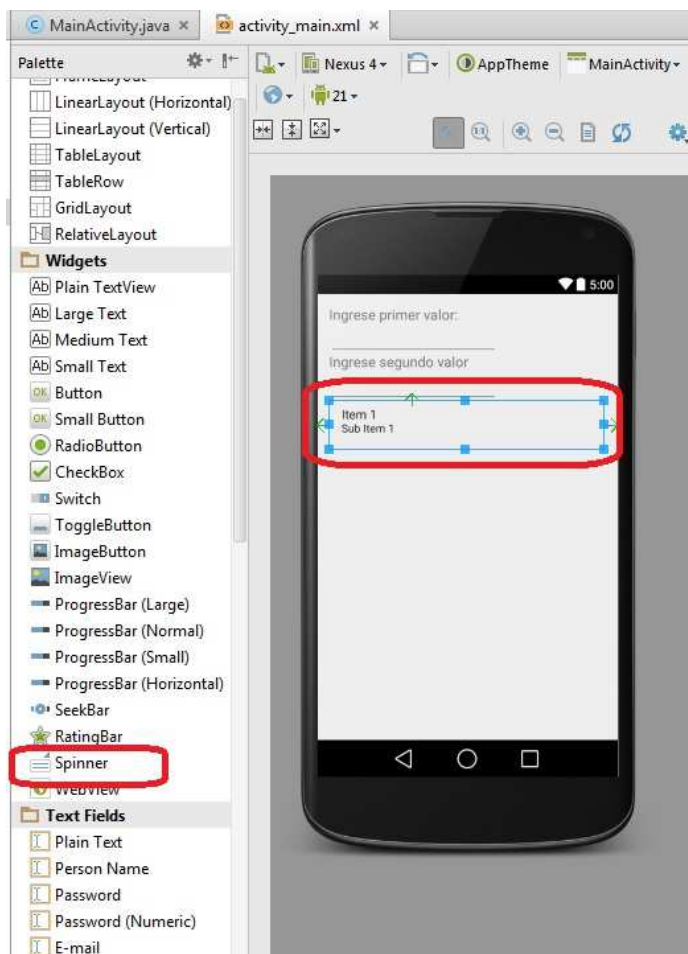
El objetivo de este concepto es seguir practicando lo visto hasta ahora e incorporar el control visual Spinner.

El control Spinner muestra una lista de String y nos permite seleccionar uno de ellos. Cuando se lo selecciona se abre y muestra todos sus elementos para permitir seleccionar uno de ellos.

Problema:

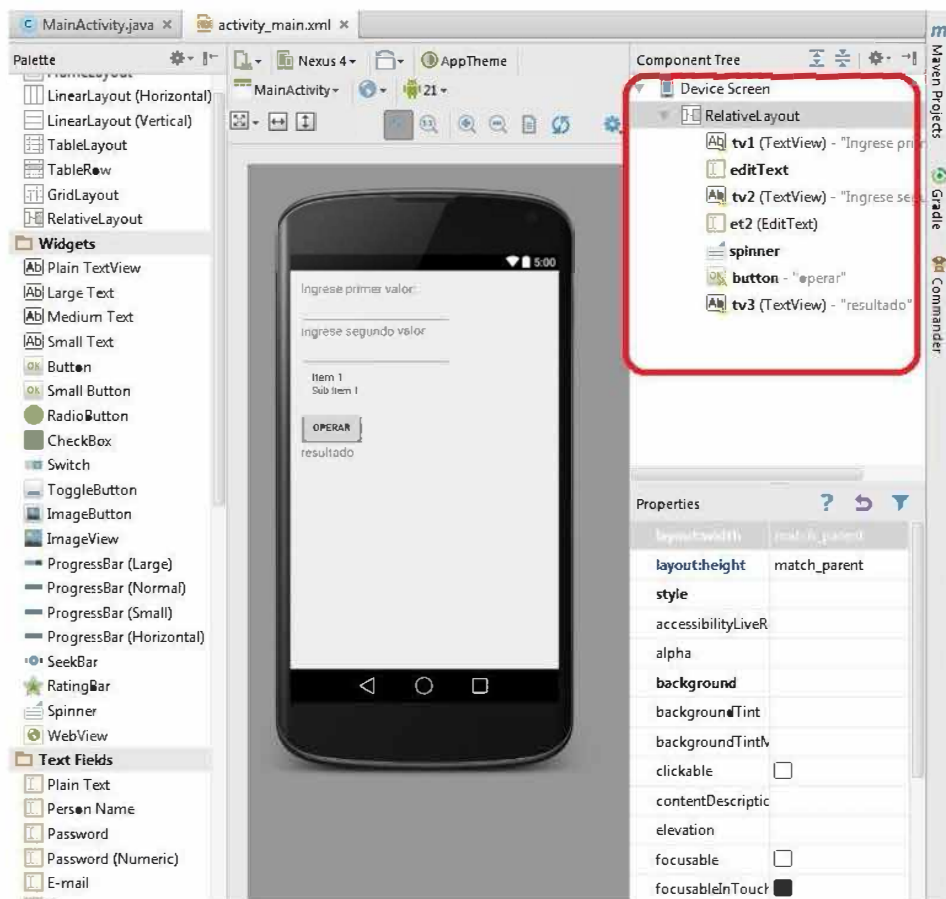
Realizar la carga de dos números en controles de tipo EditText. Mostrar un mensaje que solicite la carga de los valores. Disponer un control de tipo Spinner que permita seleccionar si queremos sumar, restar, multiplicar o dividir dichos valores. Finalmente mediante un control de tipo Button efectuamos la operación respectiva. Mostramos el resultado en un TextView.

Lo nuevo en este problema es la inserción de un control de tipo Spinner que se encuentra en la pestaña "Widgets" (cuando lo arrastremos podemos redimensionarlo con el mouse para que ocupe todo el ancho del dispositivo):



Dejamos la propiedad id con el valor spinner (valor por defecto que crea el Android Studio al insertar un objeto).

En la siguiente imagen en la ventana "Component Tree" del Android Studio podemos observar los objetos dispuestos en el formulario, sus id, sus textos y de que clase son cada uno:



No olvidemos inicializar la propiedad onClick del objeto button con el valor "operar" (dicho nombre es el método que debemos implementar)

Código fuente:

```
package ar.com.tutorialesya.proyecto005;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private Spinner spinner1;
    private EditText et1, et2;
    private TextView tv3;

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

        et1=(EditText) findViewById(R.id.et1);
        et2=(EditText) findViewById(R.id.et2);
        tv3=(TextView) findViewById(R.id.tv3);

        spinner1 = (Spinner) findViewById(R.id.spinner);
        String []opciones={"sumar","restar","multiplicar","dividir"};
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.layout
        .simple_spinner_item, opciones);
```



```

        spinner1.setAdapter(adapter);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //Este método se ejecutará cuando se presione el botón
    public void operar(View view) {
        String valor1=et1.getText().toString();
        String valor2=et2.getText().toString();
        int nro1=Integer.parseInt(valor1);
        int nro2=Integer.parseInt(valor2);
        String selec=spinner1.getSelectedItem().toString();
        if (selec.equals("sumar")) {
            int suma=nro1+nro2;
            String resu=String.valueOf(suma);
            tv3.setText(resu);
        } else
        if (selec.equals("restar")) {
            int resta=nro1-nro2;
            String resu=String.valueOf(resta);
            tv3.setText(resu);
        }
        else
        if (selec.equals("multiplicar")) {
            int multi=nro1*nro2;
            String resu=String.valueOf(multi);
            tv3.setText(resu);
        }
        else
        if (selec.equals("dividir")) {
            int divi=nro1/nro2;
            String resu=String.valueOf(divi);
            tv3.setText(resu);
        }
    }
}

```

Definimos un objeto de la clase Spinner:

```
private Spinner spinner1;
```

En el método onCreate obtenemos la referencia al control visual declarado en el archivo XML:

```
spinner1 = (Spinner) findViewById(R.id.spinner);
```

Definimos un vector con la lista de String que mostrará el Spinner:

```
String []opciones={"sumar","restar","multiplicar","dividir"};
```

Definimos y creamos un objeto de la clase ArrayAdapter:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, opciones);
```

Al constructor le pasamos como primer parámetro la referencia de nuestro Activity (this), el segundo parámetro indica el tipo de Spinner, pudiendo ser las constantes:

```
android.R.layout.simple_spinner_item  
android.R.layout.simple_spinner_dropdown_item
```

El tercer parámetro es la referencia del vector que se mostrará.

Luego llamamos al método setAdapter de la clase Spinner pasando la referencia del objeto de la clase ArrayAdapter que acabamos de crear:

```
spinner1.setAdapter(adapter);
```

En el método operar que se ejecuta cuando presionamos el botón procedemos a extraer el contenido seleccionado del control Spinner:

```
String selec=spinner1.getSelectedItem().toString();
```

Luego mediante una serie de if anidados verificamos si debemos sumar, restar, multiplicar o dividir:

```
if (selec.equals("sumar")) {  
    int suma=nro1+nro2;  
    String resu=String.valueOf(suma);  
    tv3.setText(resu);  
} else  
    if (selec.equals("restar")) {  
        int resta=nro1-nro2;  
        String resu=String.valueOf(resta);  
        tv3.setText(resu);  
    }  
    else  
        if (selec.equals("multiplicar")) {  
            int multi=nro1*nro2;  
            String resu=String.valueOf(multi);  
            tv3.setText(resu);  
        }  
        else  
            if (selec.equals("dividir")) {  
                int divi=nro1/nro2;  
                String resu=String.valueOf(divi);  
                tv3.setText(resu);  
            }  
            }
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto005.zip](#)

[Retornar](#)

7 - Control ListView (con una lista de String)

El control ListView a diferencia del Spinner que se cierra luego de seleccionar un elemento permanecen visibles varios elementos (se lo utiliza cuando hay que mostrar muchos elementos)

En este primer ejemplo mostraremos una lista de String (es decir cada elemento de la lista es un String, veremos más adelante que podemos tener listas de objetos de otro tipo: imágenes, íconos, varios String por elemento etc.)

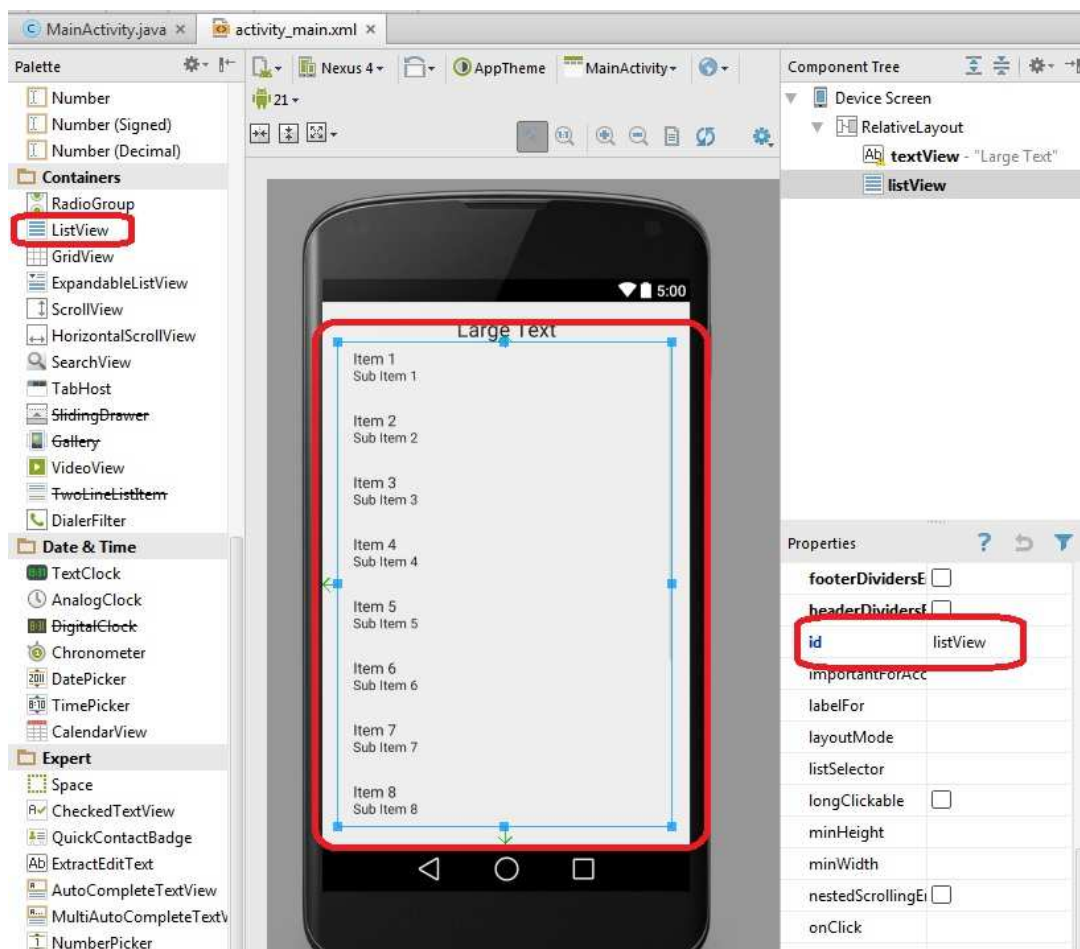
Si la lista no entra en el espacio que hemos fijado para el ListView nos permite hacer scroll de los mismos.

El control ListView se encuentra en la pestaña "Containers".

Problema:

Disponer un ListView con los nombres de países de sudamérica. Cuando se seleccione un país mostrar en un TextView ("Large Text") la cantidad de habitantes del país seleccionado.

La interfaz visual a implementar es la siguiente (primero disponemos un TextView ("Large Text") (llamado tv1) y un ListView (llamado listView por defecto)):



Código fuente:

```
package ar.com.tutorialesya.proyecto006;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
```

```

public class MainActivity extends ActionBarActivity {

    private String[] paises = { "Argentina", "Chile", "Paraguay", "Bolivia",
                                "Peru", "Ecuador", "Brasil", "Colombia", "Venezuela", "Uruguay" };
    private String[] habitantes = { "40000000", "17000000", "6500000",
                                    "10000000", "30000000", "14000000", "183000000", "44000000",
                                    "29000000", "3500000" };

    private TextView tv1;
    private ListView lv1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tv1=(TextView)findViewById(R.id.tv1);
        lv1 = (ListView)findViewById(R.id.listView);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.layout
        .simple_list_item_1,paises);
        lv1.setAdapter(adapter);
        lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView parent, View view, int position, long id){
                tv1.setText("Población de "+ lv1.getItemAtPosition(position) + " es "+
                                                                    habitantes[position]);
            }
        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}

```

Primero definimos dos vectores paralelos donde almacenamos en uno los nombres de países y en el otro almacenamos la cantidad de habitantes de dichos países:

```

private String[] paises={"Argentina","Chile","Paraguay","Bolivia","Peru",
                        "Ecuador","Brasil","Colombia","Venezuela","Uruguay"};
private String[] habitantes={"40000000","17000000","6500000","10000000","30000000",
                             "14000000","183000000","44000000","29000000","3500000"};

```

Definimos un objeto de tipo TextView y otro de tipo ListView donde almacenaremos las referencias a los objetos que definimos en el archivo XML:

```

private TextView tv1;
private ListView lv1;

```

En el método onCreate obtenemos la referencia a los dos objetos:

```

tv1=(TextView)findViewById(R.id.tv1);
lv1 =(ListView)findViewById(R.id.listView);

```

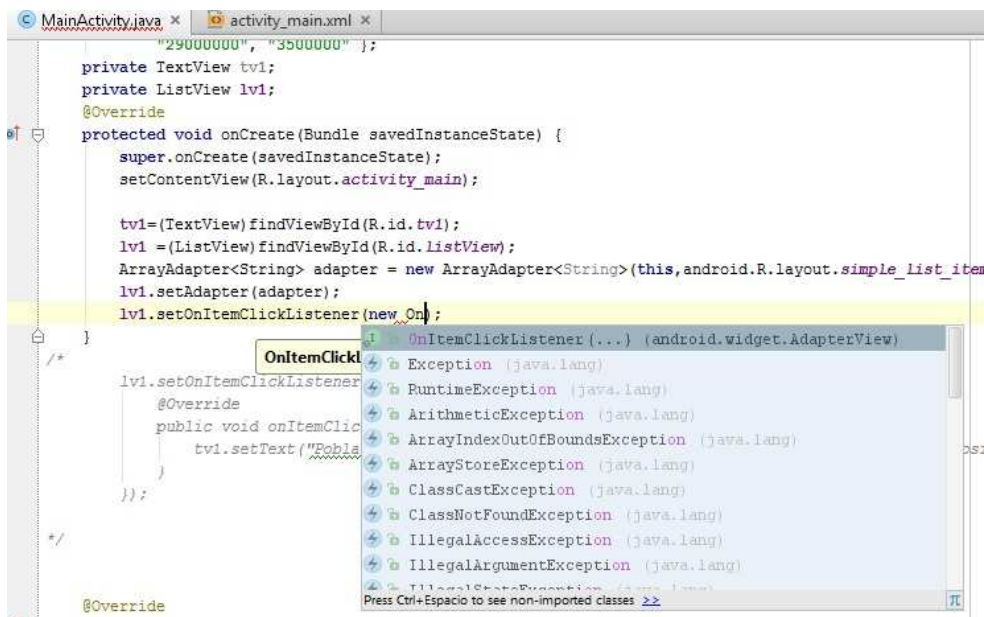
Creamos un objeto de la clase ArrayAdapter de forma similar a como lo hicimos cuando vimos la clase Spinner:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, paises);
lv1.setAdapter(adapter);
```

Llamamos al método `setOnItemClickListener` de la clase `ListView` y le pasamos como parámetro una clase anónima que implementa la interfaz `OnItemClickListener` (dicha interfaz define el método `onItemClick` que se dispara cuando seleccionamos un elemento del `ListView`):

```
lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView parent, View view, int position, long id) {
        tv1.setText("Población de " + lv1.getItemAtPosition(position) + " es " + habitantes[position]);
    }
});
```

Para codificar esta clase anónima podemos utilizar la facilidad que nos brinda el Android Studio para generarla automáticamente, para esto tipeamos:



Luego presionamos la tecla "Enter" y el Android Studio automáticamente nos genera la clase anónima implementando la interfaz `onItemClick`.

Dentro del método `onItemClick` modificamos el contenido del `TextView` con el nombre del país y la cantidad de habitantes de dicho país. Este método recibe en el tercer parámetro la posición del ítem seleccionado del `ListView`.

Cuando ejecutamos el proyecto podemos ver una interfaz en el emulador similar a esta:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto006.zip](#)

[Retornar](#)

8 - Control ImageButton

Hemos visto la creación de objetos de la clase Button, ahora veremos otra clase muy similar a la anterior llamada ImageButton que tiene la misma filosofía de manejo con la diferencia que puede mostrar una imagen en su superficie.

Problema:

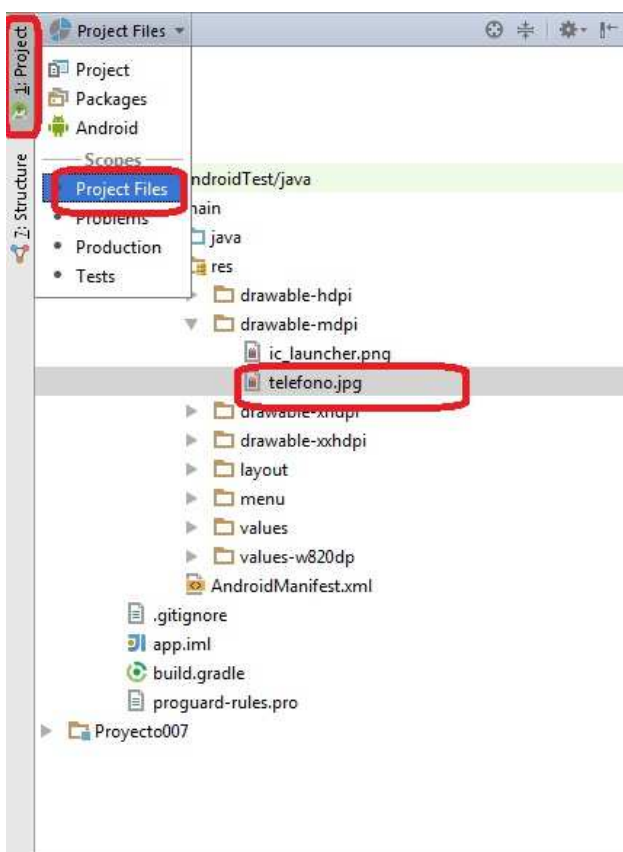
Disponer un objeto de la clase ImageButton que muestre una imagen de un teléfono. Cuando se presione mostrar en un control TextView el mensaje "Llamando".

Primero crearemos un proyecto llamado proyecto007 y luego debemos buscar una imagen en formato png que represente un telefono de 50*50 píxeles.

Nombre del archivo: telefono.png

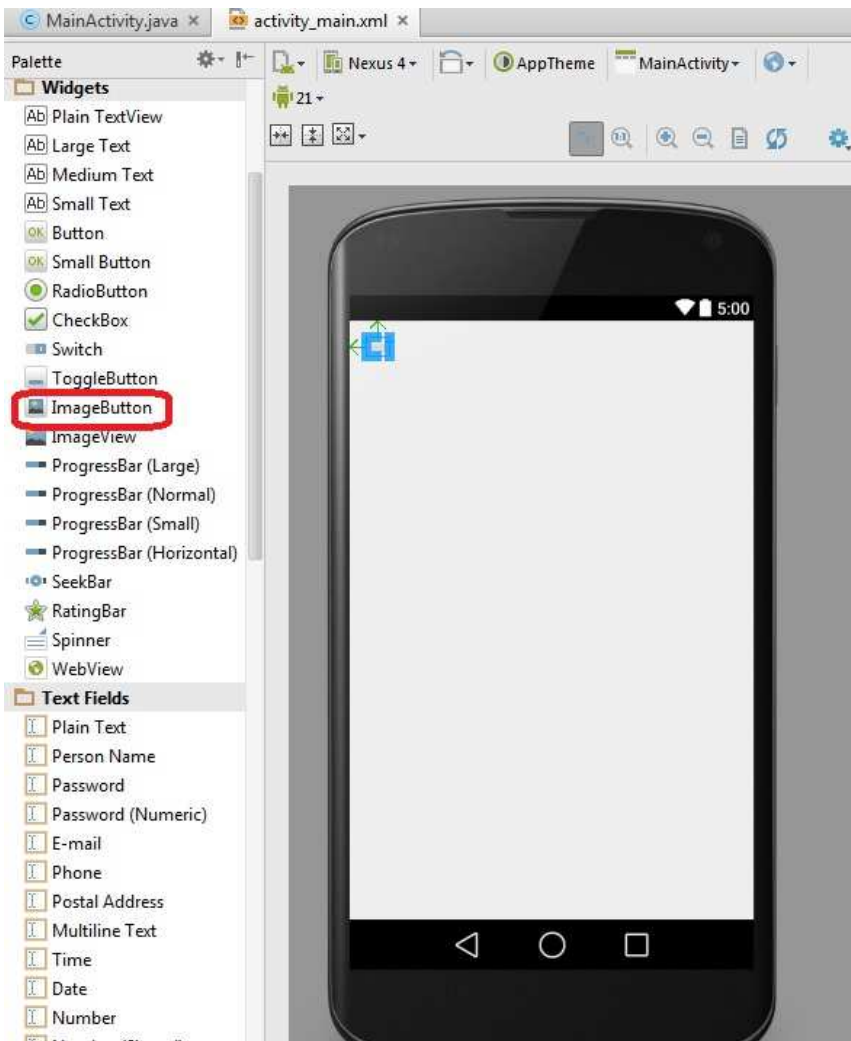
Ahora copiamos el archivo a la carpeta de recursos de nuestro proyecto : Proyecto007/app/src/main/res/drawable-mdpi
Esto lo podemos hacer desde el administrador de archivos del sistema operativo.

En la ventana "Project" cambiamos a vista "Project Files" y navegamos hasta la carpeta donde copiamos el archivo, vemos que el archivo está en dicha carpeta:

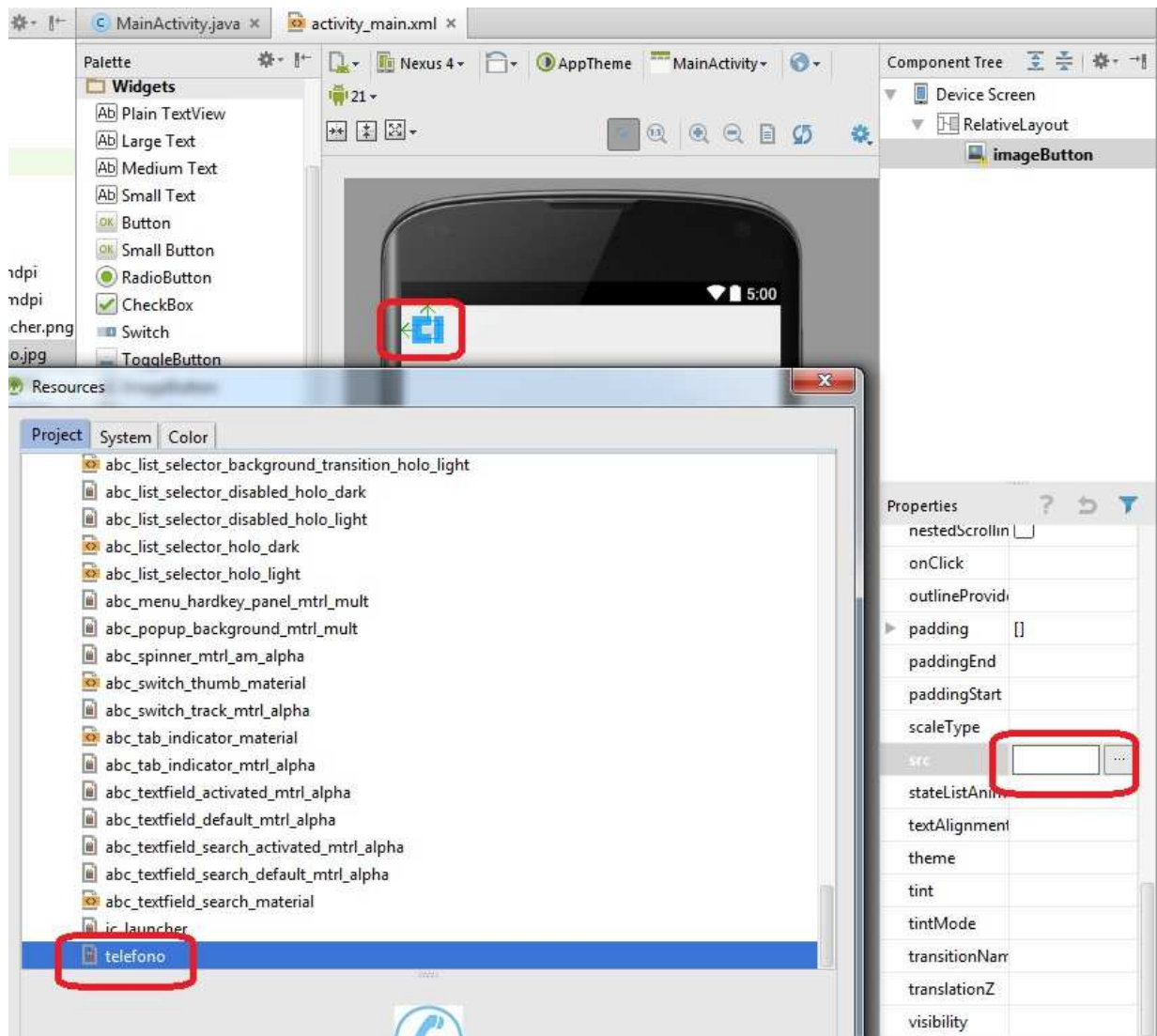


Esto solo a efectos de ver que el proyecto se actualiza con los archivos de recursos que copiamos a las carpetas drawable.

Ahora insertaremos el objeto de la clase ImageButton en el formulario de nuestra aplicación. La clase ImageButton se encuentra en la pestaña "Widgets":

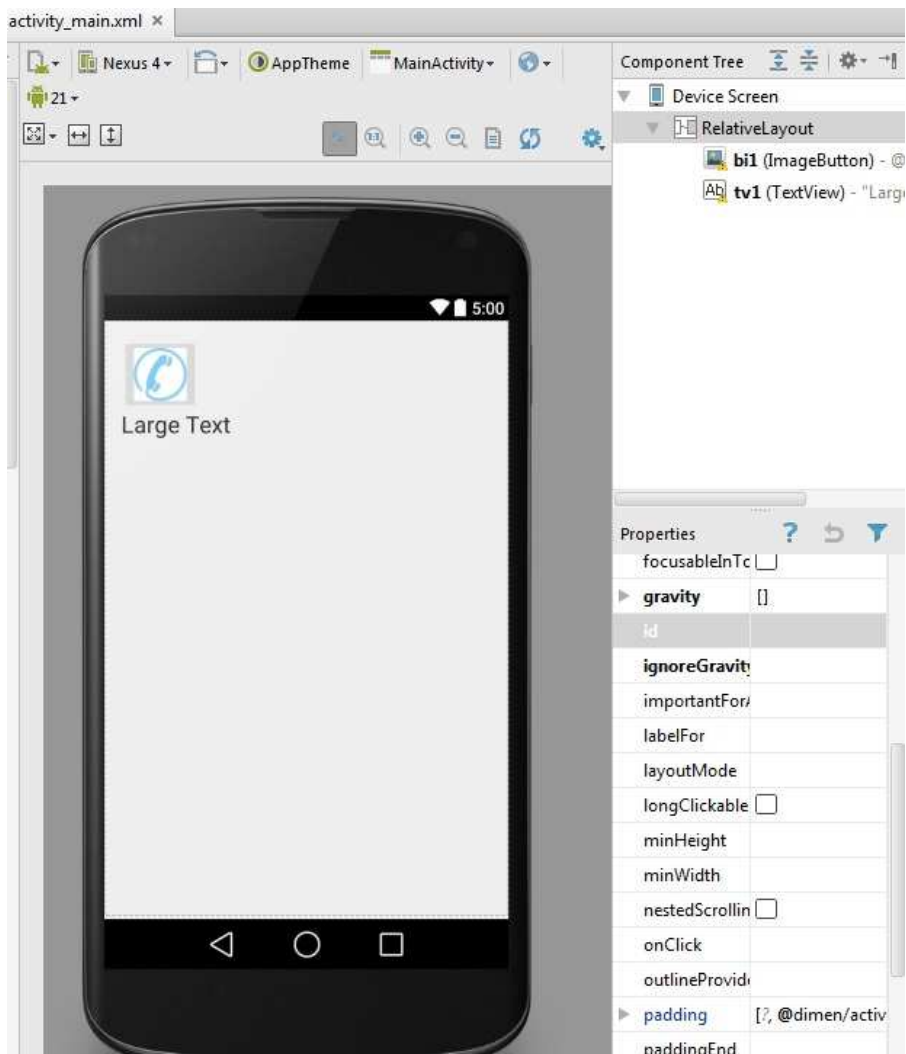


Seguidamente seleccionamos la propiedad "src" del objeto ImageButton que acabamos de disponer y en dicho diálogo buscamos el nombre de la imagen que copiamos en la carpeta drawable-mdpi:



Inicializamos la propiedad ID con el nombre bi1

Agreguemos un TextView a nuestra aplicación y configuremos sus propiedades ID (con tv1) y text. Luego la interfaz visual debe ser similar a:



Código fuente:

```
package ar.com.tutorialesya.proyecto007;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private TextView tv1;

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

        tv1=(TextView) findViewById(R.id.tv1);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
    }
}
```

```

        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //Este método se ejecutará cuando se presione el ImageButton
    public void llamar(View view) {
        tv1.setText("Llamando");
    }
}

```

Definimos el atributo de tipo TextView:

```
TextView tv1;
```

Enlazamos el control definido en el archivo XML y la variable de java:

```
tv1=(TextView)findViewById(R.id.tv1);
```

Implementamos el método que se ejecutará cuando se presione el el objeto de la clase ImageButton:

```

public void llamar(View view) {
    tv1.setText("Llamando");
}

```

No olvidemos inicializar la propiedad onClick del objeto ib1 con el nombre del método "llamar" (recordemos que esto lo hacemos accediendo a la propiedad On Click en la ventana de "Properties")

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto007.zip](#)

Comentarios extras de este control.

Cuando copiamos el archivo lo hicimos a la carpeta:

drawable-mdpi

Pero vimos que hay otras carpetas con nombres similares:

```

drawable-ldpi
drawable-hdpi
drawable-xhdpi
drawable-xxhdpi

```

El objetivo de estas otras carpetas es guardar las mismas imágenes pero con una resolución mayor si la guardamos en drawable-hdpi y con una resolución menor al guardarla en drawable-ldpi.

Esto se hace si queremos que nuestra aplicación sea más flexible si se ejecuta en un celular, en un tablet, en un televisor etc.

Debemos tener en cuenta estos tamaños:

En la carpeta res/drawable-mdpi/

(guardamos la imagen con el tamaño original)

En la carpeta res/drawable-ldpi/
(guardamos la imagen con el tamaño del 75% con respecto al de la carpeta drawable-mdpi)

En la carpeta res/drawable-hdpi/
(guardamos la imagen con el tamaño del 150% con respecto al de la carpeta drawable-mdpi)

En la carpeta res/drawable-xhdpi/
(guardamos la imagen con el tamaño del 200% con respecto al de la carpeta drawable-mdpi)

En la carpeta res/drawable-xxhdpi/
(guardamos la imagen con el tamaño del 300% con respecto al de la carpeta drawable-mdpi)

[Retornar](#)

9 - Notificaciones sencillas mediante la clase Toast

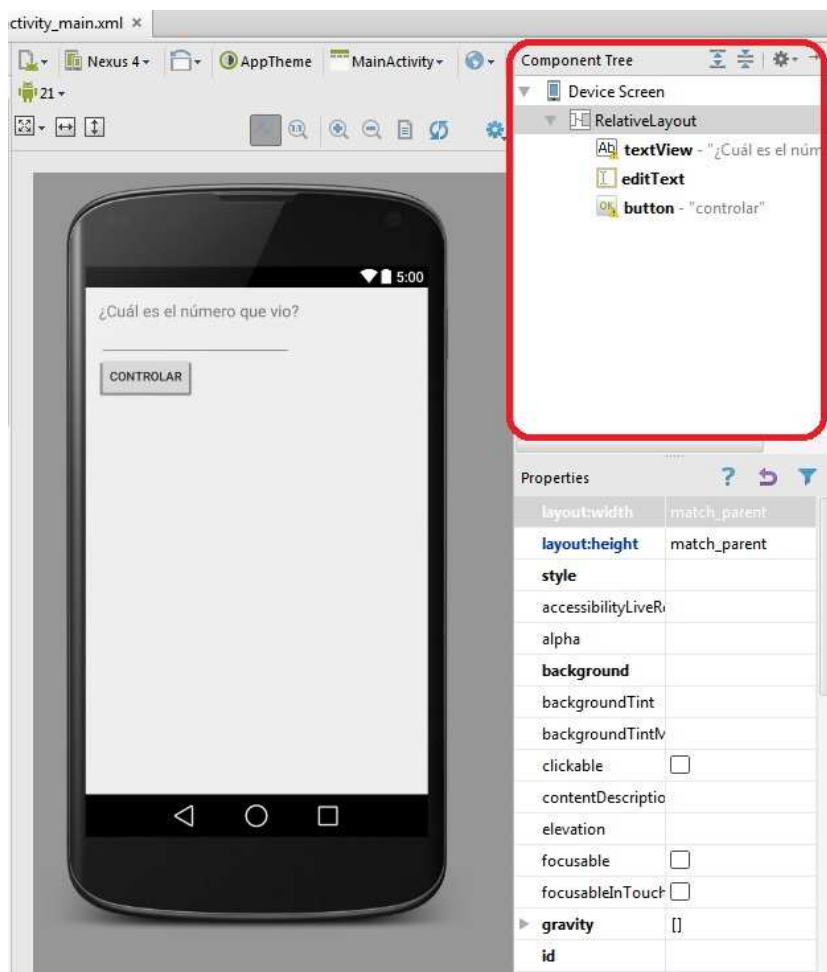
Android permite mostrar una ventana emergente temporal que informa al usuario mediante un mensaje que aparece en la pantalla por un lapso pequeño de tiempo (luego de pasado un tiempo la ventana desaparece).

Esta ventana que se superpone a la interfaz que se está mostrando en ese momento se administra mediante una clase llamada Toast.

Problema:

Generar un número aleatorio entre 1 y 100000. Mostrar mediante una ventana emergente dicho número por un lapso de tiempo. Luego mediante un control EditText ("Number") pedir al operador que ingrese el número que vio en la pantalla. Cuando se presione un botón controlar el número generado aleatoriamente con el que ingresó el usuario y mediante otro Toast informar si acertó o no.

Lo primero que hacemos es crear la interfaz siguiente:



Es decir que disponemos un TextView ("Medium Text"), EditText ("Number") y un Button. Dejamos por defecto los nombres asignados a las id de cada uno de estos objetos: textView, editText y button.

Luego seleccionamos el botón de la interfaz visual e inicializamos la propiedad onClick con el nombre del método que se ejecutará al ser presionado, dicho nombre debe ser: "controlar" (recordemos que los nombre de métodos en Java por convención empiezan en minúsculas)

Código fuente:

```
package ar.com.tutorialesya.proyecto008;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
```

```

import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

    private EditText et1;
    private int num;

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

        et1=(EditText)findViewById(R.id.editText);
        num=(int) (Math.random()*100001);
        String cadena=String.valueOf(num);
        Toast notificacion=Toast.makeText(this,cadena,Toast.LENGTH_LONG);
        notificacion.show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void controlar(View v) {
        String valorIngresado=et1.getText().toString();
        int valor=Integer.parseInt(valorIngresado);
        if (valor==num) {
            Toast notificacion=Toast.makeText(this,"Muy bien recordaste el número mos
            notificacion.show();
        }
        else {
            Toast notificacion=Toast.makeText(this,"Lo siento pero no es el número qu
            notificacion.show();
        }
    }
}

```

Analicemos el código fuente de la aplicación, lo primero es definir un objeto de la clase EditText y una variable entera donde almacenaremos el valor aleatorio que generamos y mostramos en un principio:

```

private EditText et1;
private int num;

```

En el método onCreate que es el primero que se ejecuta al inicializar la aplicación es donde inicializamos la referencia del EditText:

```

et1=(EditText)findViewById(R.id.editText);

```

Generamos un valor aleatorio comprendido entre 0 y 100000 y lo almacenamos en la variable num:

```
num=(int)(Math.random()*100001);
```

Convertimos el valor entero a String ya que la clase Toast siempre hay que pasarte un String para mostrar:

```
String cadena=String.valueOf(num);
```

Veamos ahora lo nuevo que es la clase Toast, tenemos que crear un objeto de la clase Toast, para ello definimos una variable que la llamamos notificacion y mediante el método estático makeText de la clase Toast creamos el objeto.

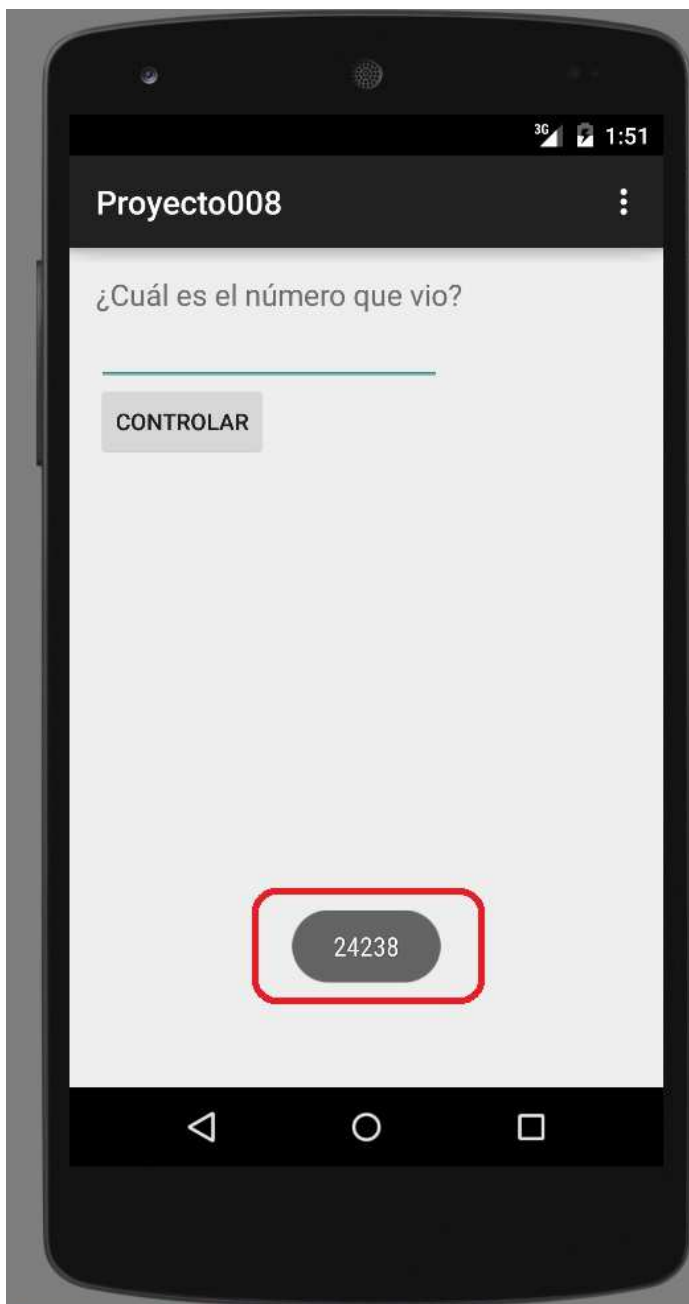
El método makeText tiene tres parámetros: el primero hace referencia a la ventana o Activity donde aparece (this), el segundo el la variable de tipo String que se mostrará en pantalla y por último es una constante que indica que la notificación se mostrará por un tiempo largo o corto:

```
Toast notificacion=Toast.makeText(this,cadena,Toast.LENGTH_LONG);
```

Finalmente cuando queremos que se muestre la notificación en pantalla procedemos a llamar al método show de la clase Toast:

```
notificacion.show();
```

Esto hace que inmediatamente arranquemos la aplicación se mostrará la notificación con el número que deberá memorizar el usuario:



Luego de unos segundo desaparece la notificación de pantalla (es decir en nuestro programa desaparece de pantalla el

número aleatorio)

Cuando el operador termina de cargar el número y procede a ejecutar el botón "controlar" se procede a ejecutar el código que dispusimos en el método "controlar":

```
public void controlar(View v) {
    String valorIngresado=et1.getText().toString();
    int valor=Integer.parseInt(valorIngresado);
    if (valor==num) {
        Toast notificacion=Toast.makeText(this,"Muy bien recordaste el número mostrado.",Toast.LENGTH_LONG);
        notificacion.show();
    }
    else {
        Toast notificacion=Toast.makeText(this,"Lo siento pero no es el número que mostré.",Toast.LENGTH_LONG);
        notificacion.show();
    }
}
```

En este método extraemos del control EditText el número que ingreso el usuario:

```
String valorIngresado=et1.getText().toString();
```

Lo convertimos a entero:

```
int valor=Integer.parseInt(valorIngresado);
```

Y mediante un if verificamos si coincide con el número aleatorio ingresado por el operador, si coincide inmediatamente creamos una nueva notificación y procedemos a mostrar que acertó:

```
if (valor==num) {
    Toast notificacion=Toast.makeText(this,"Muy bien recordaste el número mostrado.",Toast.LENGTH_LONG);
    notificacion.show();
}
```

Por el falso mediante otra notificación mostramos que no ingresó correctamente el número:

```
else {
    Toast notificacion=Toast.makeText(this,"Lo siento pero no es el número que mostré.",Toast.LENGTH_LONG);
    notificacion.show();
}
```

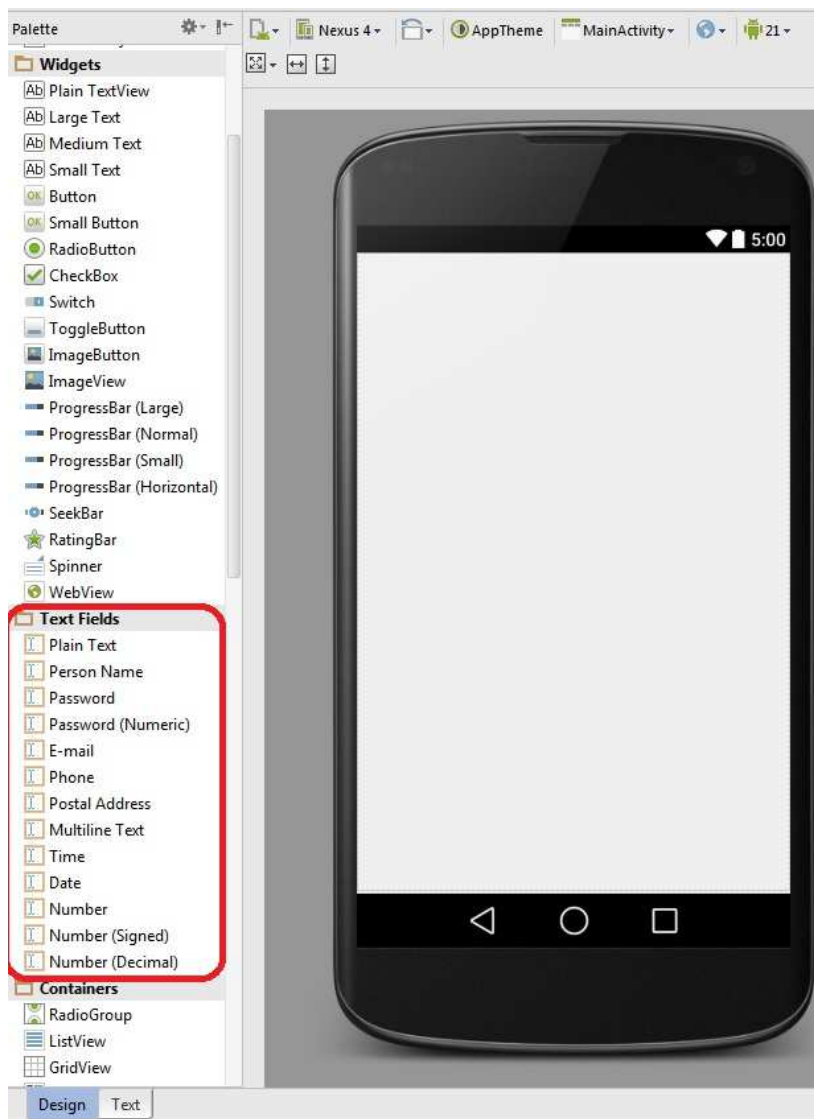
Este proyecto lo puede descargar en un zip desde este enlace: [proyecto008.zip](#)

[Retornar](#)

10 - Control EditText

Desde el primer problema hemos estado utilizando el control que permite en Android ingresar valores por teclado. La clase que administra la entrada de cadenas de datos por teclado es EditText.

Pero en la palette de componentes vemos hay que muchos tipos de EditText:



Como podemos ver en la pestaña "Text Fields" se encuentran todos los tipos de EditText que nos ofrece Android para utilizar en nuestras aplicaciones: Password, E-mail, Number, etc.

Dependiendo del tipo de entrada de datos que necesitemos utilizaremos un tipo específico de EditText.

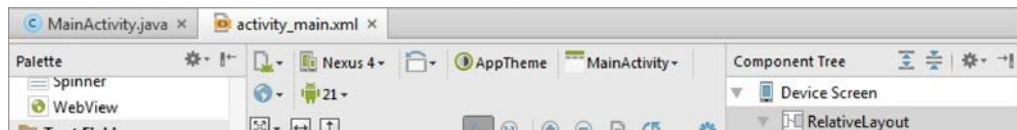
Problema

Confeccionar una aplicación para android que permita ingresar el nombre de usuario y su clave en dos controles de tipo EditText.

Verificar al presionar un botón si se ingresó algún texto en la clave. Si no se ingresó texto informar mediante una notificación dicha situación.

(utilizar el método `length()` de la clase `String` para ver cuantos caracteres se ingresaron.

La interfaz visual debe ser similar a esta:



Código fuente:

```
package ar.com.tutorialesya.proyecto009;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2;

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

        et1=(EditText) findViewById(R.id.editText);
        et2=(EditText) findViewById(R.id.editText2);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}
```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void verificar(View v) {
    String clave=et2.getText().toString();
    if (clave.length()==0) {
        Toast notificacion= Toast.makeText(this,"La clave no puede quedar vacía",
        notificacion.show();
    }
}
}

```

Como podemos ver cuando se presiona el botón "verificar" se procede a extraer el contenido del EditText de la clave y mediante el método length() controlamos si tiene cero caracteres, en caso afirmativo mostramos la notificación en pantalla:

```

public void verificar(View v) {
    String clave=et2.getText().toString();
    if (clave.length()==0) {
        Toast notificacion= Toast.makeText(this,"La clave no puede quedar vacía",Toast.LENGTH_LONG);
        notificacion.show();
    }
}

```

En pantalla tendremos un resultado similar a esto:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto009.zip](#)

[Retornar](#)

11 - Lanzar un segundo "Activity"

Hasta ahora todos los programas han tenido una sola ventana (Activity)

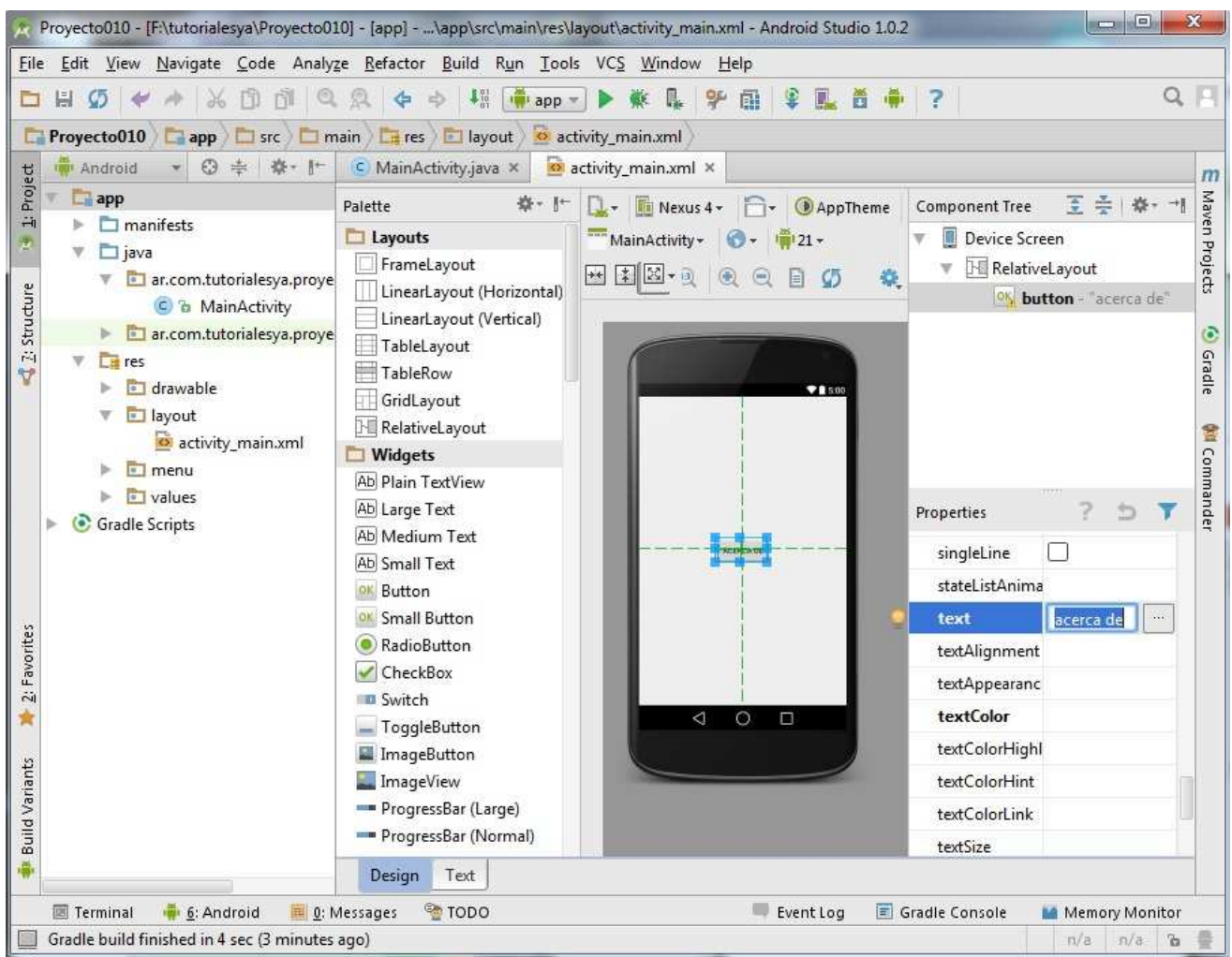
Es muy común que una aplicación tenga más de una ventana. Para implementar esto en Android debemos plantear otros dos archivos uno xml con la interfaz y otro java con la lógica (tengamos en cuenta que cuando utilizamos Android Studio automáticamente cuando creamos un proyecto nos crea el archivo XML y el código java del primer Activity)

Vamos a ver en este concepto los pasos que debemos dar para crear otro Activity y como activarlo desde el Activity principal.

Problema:

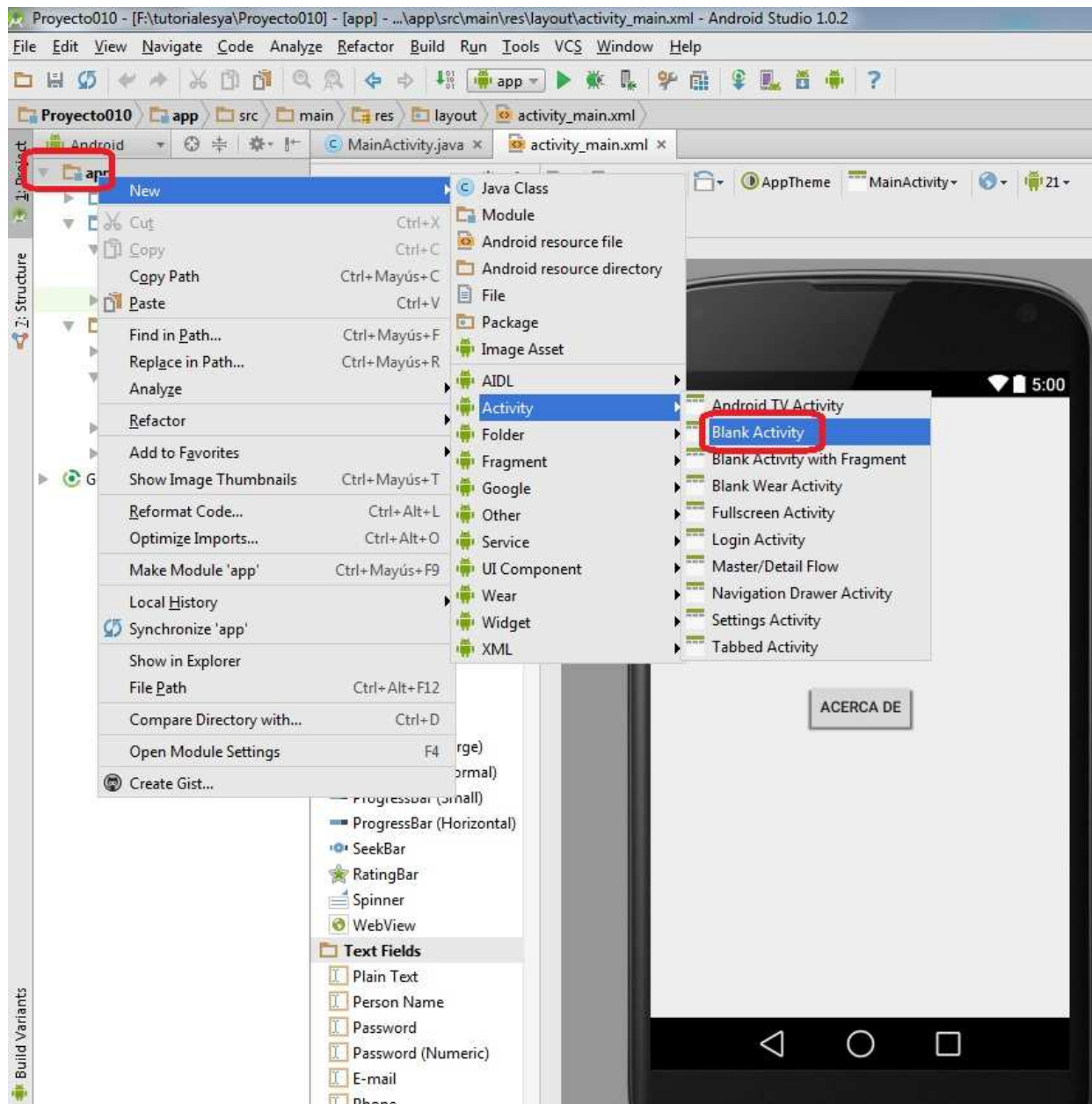
Confeccionar un programa que muestre en la ventana principal un botón que al ser presionado muestre otra ventana (Activity) mostrando un TextView con el nombre del programador de la aplicación y un botón para cerrar la ventana o actividad y que vuelva al primer Activity.

1 - Primero creamos un nuevo proyecto que lo llamaremos Proyecto010 y en la ventana principal creamos la siguiente interfaz:



Es decir que nosotros queremos que cuando se presione el botón "ACERCA DE" nos abra otra ventana (Activity) y nos muestre el nombre del programador y un botón para cerrar dicha ventana.

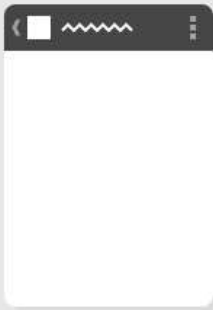
2 - Ahora tenemos que crear el segundo Activity. Para esto hacemos clic con el botón derecho del mouse en la ventana de Project donde dice "app" y seleccionamos New -> Activity -> Blank Activity



Aparece un diálogo donde debemos definir el nombre del Activity "Activity Name" y le asignaremos como nombre "AcercaDe" (se crearán dos archivos AcercaDe.java y activity_acerca_de.xml):

Choose options for your new file

Creates a new blank activity with an action bar.

 Blank Activity

Activity Name: **AcercaDe**

Layout Name: activity_acerca_de

Title: AcercaDe

Menu Resource Name: menu_acerca_de

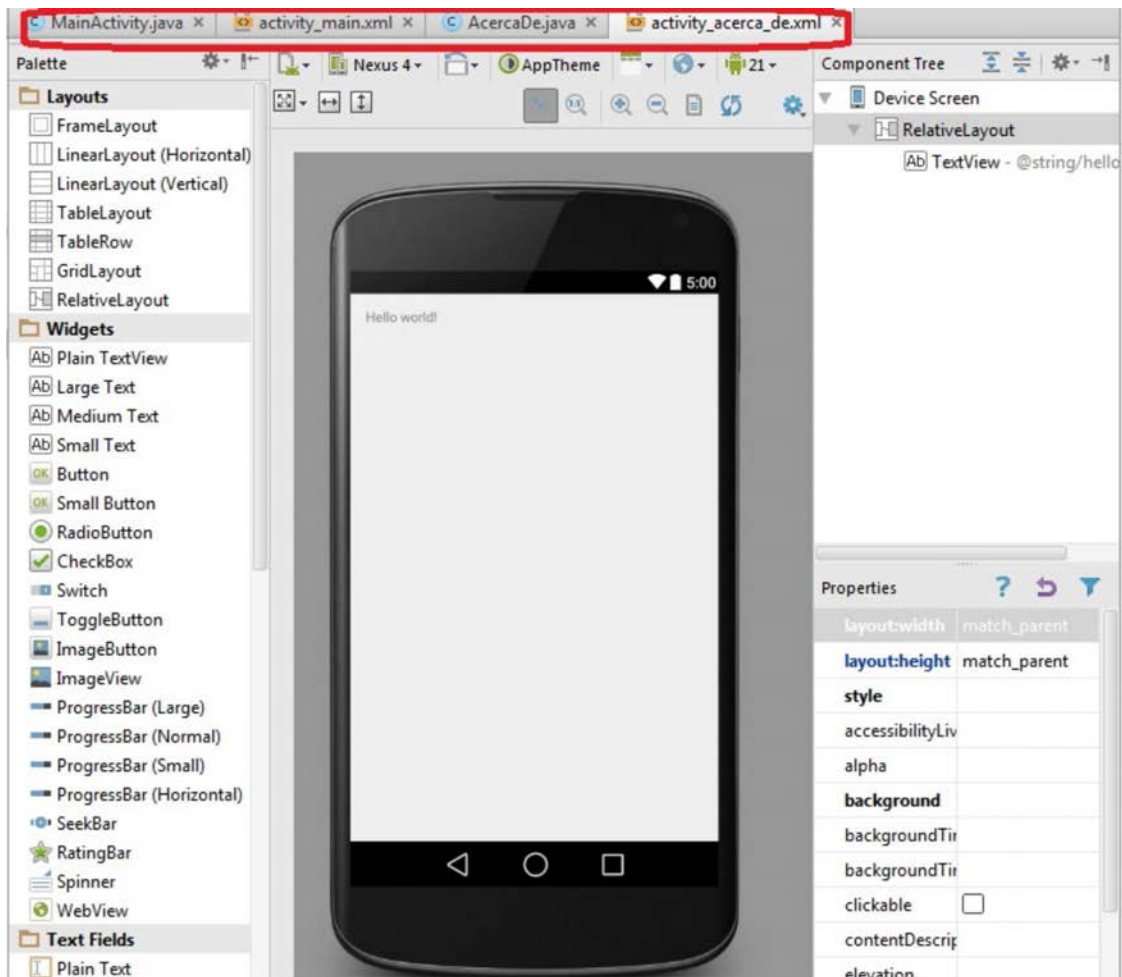
☐ Launcher Activity

Hierarchical Parent: ...

Package name: ar.com.tutorialesya.proyecto010

Target Source Set: main

Ya tenemos los cuatro archivos necesarios para implementar la aplicación:



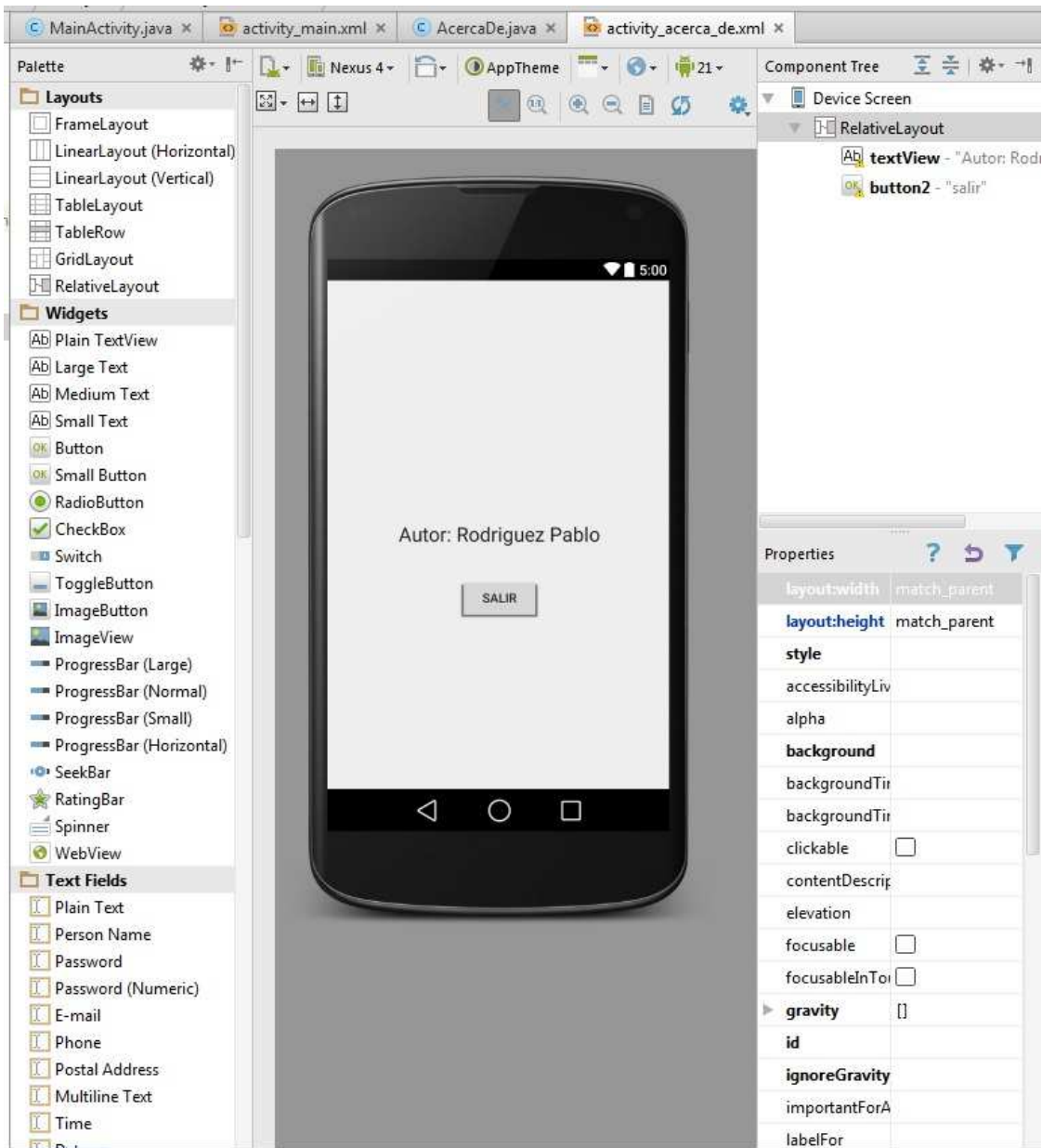
Los dos primeros que se crean cuando iniciamos el proyecto:

```
activity_main.xml
MainActivity.java
```

Y estos dos nuevos archivos para la segunda ventana:

```
activity_acerca_de.xml
AcercaDe.java
```

Implementamos la interfaz visual del segundo Activity es decir del archivo activity_acerca_de.xml con los siguientes controles:



3 - Ahora implementaremos la funcionalidad en la actividad (Activity) principal para que se active la segunda ventana.

Inicializamos la propiedad `onClick` del objeto de la clase `Button` con el valor "acercade", este es el método que se ejecutará cuando se presione.

El código fuente de la actividad principal queda:

```
package ar.com.tutorialesya.proyecto010;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void acercade(View view) {
    Intent i = new Intent(this, AcercaDe.class );
    startActivity(i);
}
}

```

En el método acercade creamos un objeto de la clase Intent y le pasamos como parámetros la referencia del objeto de esta clase (this) y la referencia del otro Activity (AcercaDe.class)

Llamamos posteriormente al método startActivity pasando el objeto de la clase Intent (con esto ya tenemos en pantalla la ventana del segundo Activity):

```

public void acercade(View view) {
    Intent i = new Intent(this, AcercaDe.class );
    startActivity(i);
}

```

Si ejecutamos el programa podemos ver que cuando presionamos el botón "Acerca De" aparece la segunda ventana donde se muestra el TextView con el valor "Autor : Rodriguez Pablo" y un botón con el texto "salir" (si presionamos este botón no sucede nada, esto debido a que no hemos asociado ningún evento a dicho botón)

4 - Debemos codificar el evento onClick de la segunda actividad. Seleccionemos el botón "salir" y definamos en la propiedad onClick el nombre de método que se ejecutará al presionarse el botón (en nuestro caso lo llamaremos "salir") :

El código fuente de la actividad AcercaDe queda:

```

package ar.com.tutorialesya.proyecto010;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class AcercaDe extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```

        setContentView(R.layout.activity_acerca_de);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_acerca_de, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void salir(View v) {
        finish();
    }
}

```

Cuando se presiona el botón salir se ejecuta el método "salir" llamando al método finish() que tiene por objetivo liberar el espacio de memoria de esta actividad y pedir que se active la actividad anterior.

Ahora nuestro programa está funcionando completamente:

Primer Activity:



Segundo Activity:

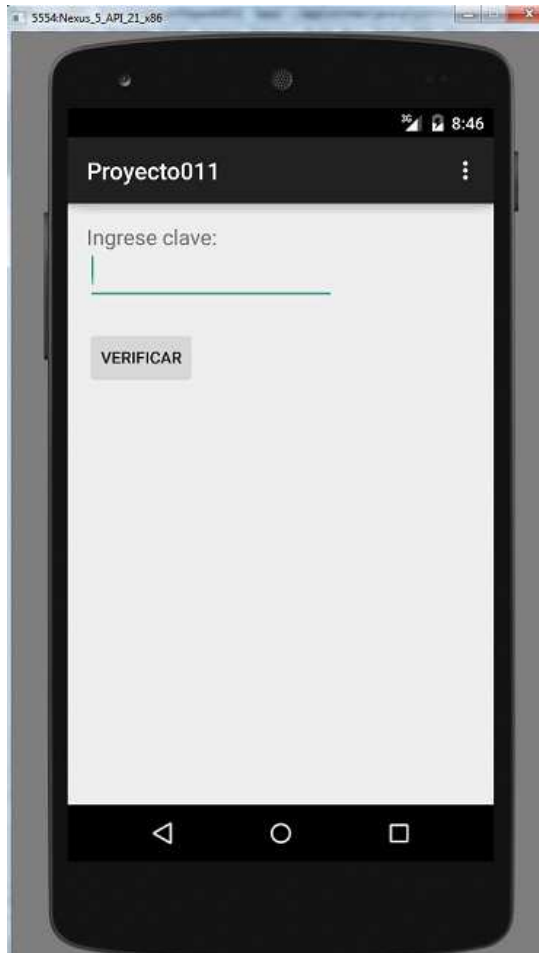


Este proyecto lo puede descargar en un zip desde este enlace: proyecto010.zip

Problema propuesto

1. Realizar un programa que contenga dos Activity. En el primero que solicite el ingreso de una clave (control Password) Si ingresa la clave "abc123" activar el segundo Activity mostrando en un TextView un mensaje de bienvenida (mostrar en Toast si se ingresa la clave incorrecta en el primer Activity).
Llamar al proyecto: Proyecto011.

En tiempo de ejecución los dos Activity deben mostrarse algo similar a esto:





Este proyecto lo puede descargar en un zip desde este enlace: [proyecto011.zip](#)

[Retornar](#)

12 - Lanzar un segundo "Activity" y pasar parámetros

Hemos visto en el concepto anterior que un programa puede tener más de una ventana representando cada ventana con una clase que hereda de `ActionBarActivity`.

Una situación muy común es que la primer ventana necesite enviar datos a la segunda para que a partir de estos proceda a efectuar una acción.

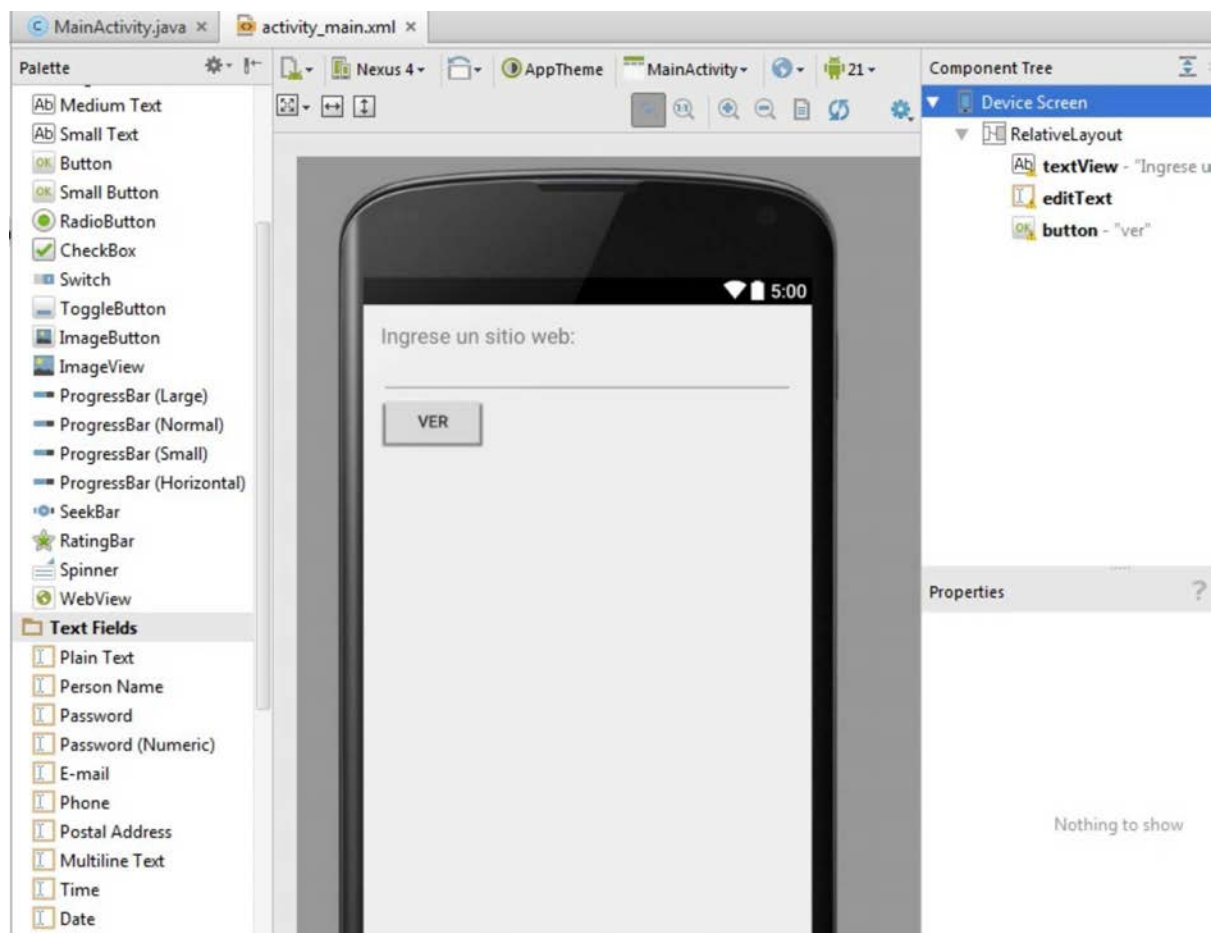
Problema:

Confeccionar un programa que solicite el ingreso de una dirección de un sitio web y seguidamente abrir una segunda ventana que muestre dicha página.

Para resolver este problema utilizaremos el control visual `WebView` que nos permite mostrar el contenido de un sitio web.

Crearemos un nuevo proyecto llamado proyecto012.

1 - Nuestro primer Activity tendrá la siguiente interfaz visual (ver controles):



Tenemos un control de tipo `TextView`, otro de tipo `EditText` y finalmente otro de tipo `Button` (inicializar la propiedad `onClick` con el nombre de método llamado "ver").

El código fuente de esta Activity es:


```

package ar.com.tutorialesya.proyecto12;

import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends ActionBarActivity {

    private EditText et1;

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

        et1=(EditText)findViewById(R.id.editText);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void ver(View v) {
        Intent i=new Intent(this,Actividad2.class);
        i.putExtra("direccion", et1.getText().toString());
        startActivity(i);
    }
}

```

Como podemos ver la diferencia con el concepto anterior es que llamamos al método putExtra de la clase Intent. Tiene dos parámetros de tipo String, en el primero indicamos el nombre del dato y en el segundo el valor del dato:

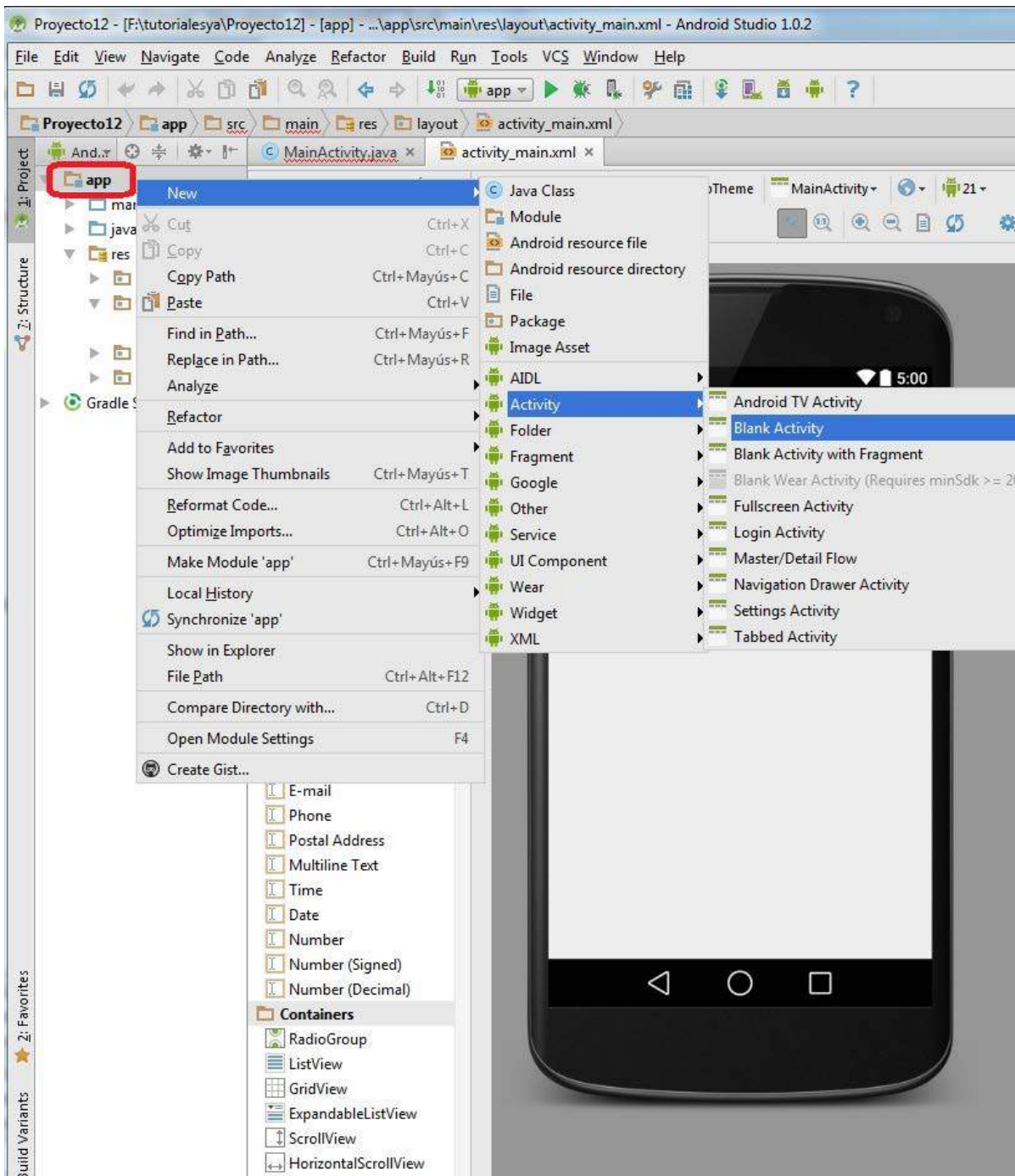
```

public void ver(View v) {
    Intent i=new Intent(this,Actividad2.class);
    i.putExtra("direccion", et1.getText().toString());
    startActivity(i);
}

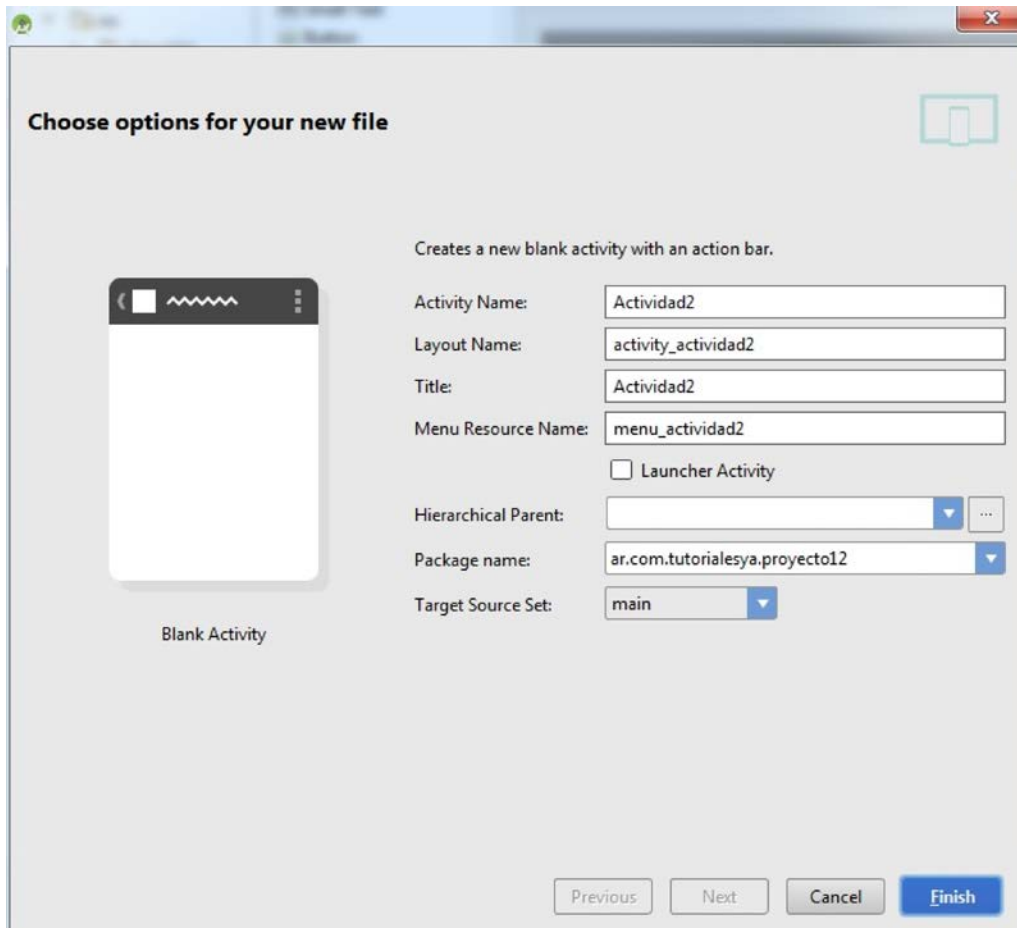
```

La segunda interfaz visual (recordemos que debemos presionar el botón derecho en la ventana Project (sobre app) y

seleccionar la opción New -> Activity -> Blank Activity:



En el diálogo que aparece especificamos el nombre de nuestra segundo Activity y lo llamaremos Actividad2:



Codificamos la funcionalidad de la segunda actividad

```
package ar.com.tutorialesya.proyecto12;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.webkit.WebView;

public class Actividad2 extends ActionBarActivity {

    private WebView webView1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_actividad2);

        webView1 = (WebView) findViewById(R.id.webView);

        Bundle bundle = getIntent().getExtras();
        String dato=bundle.getString("direccion");
        webView1.loadUrl("http://" + dato);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_actividad2, menu);
    }
}
```

```

        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void finalizar(View v) {
        finish();
    }
}

```

En esta clase definimos una variable de tipo Bundle y la inicializamos llamando al método `getExtras()` de la clase `Intent` (esto lo hacemos para recuperar el o los parámetros que envió la otra actividad (`Activity`)):

```

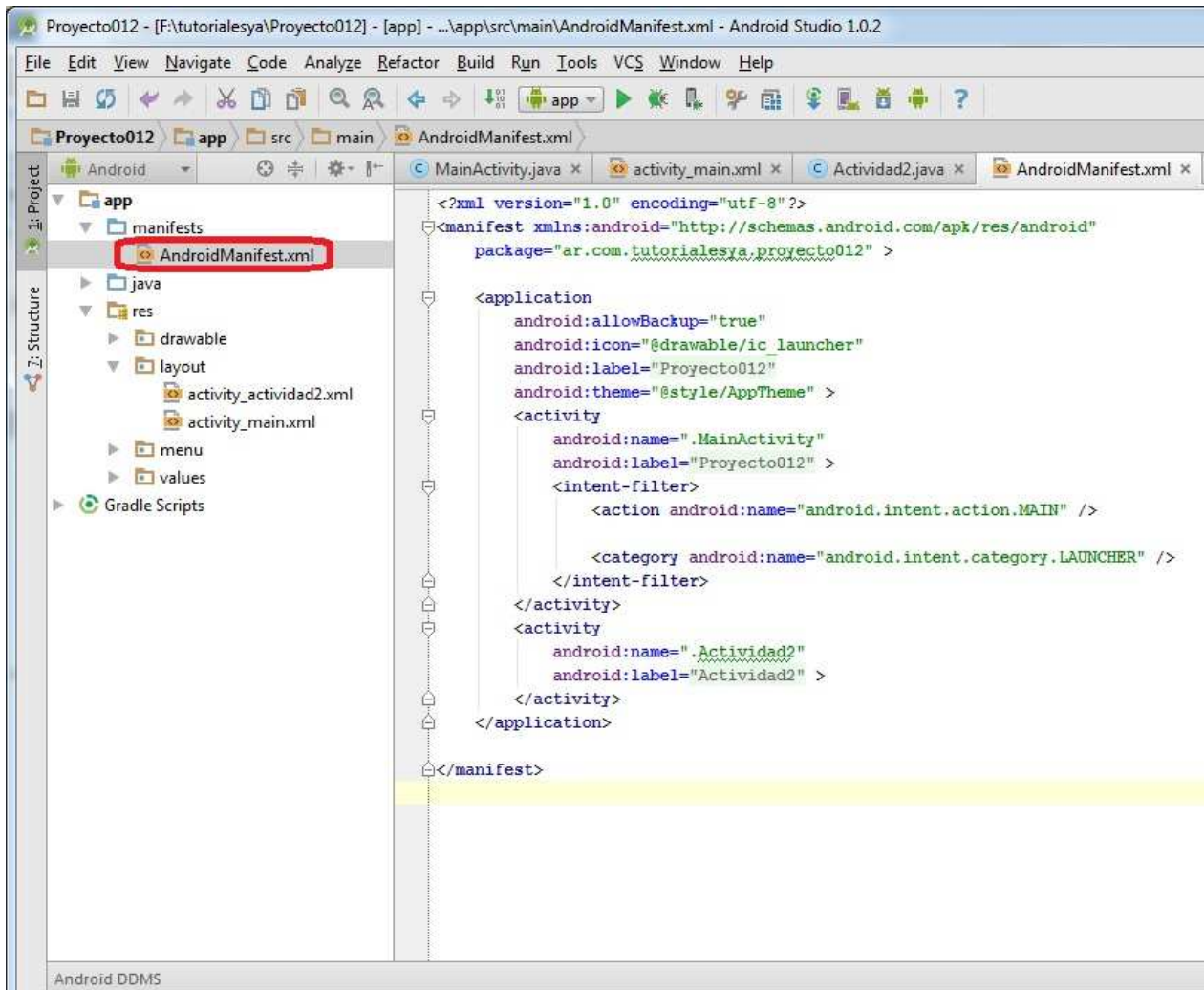
Bundle bundle = getIntent().getExtras();
String dato=bundle.getString("direccion");
webView1.loadUrl("http://" + dato);

```

El método `loadUrl` de la clase `WebView` permite visualizar el contenido de un sitio web.

Importante

Como nuestra aplicación debe acceder a internet debemos hacer una configuración en el archivo "`AndroidManifest.xml`", podemos ubicar este archivo:



Agregamos el permiso tipeando lo siguiente en este archivo:

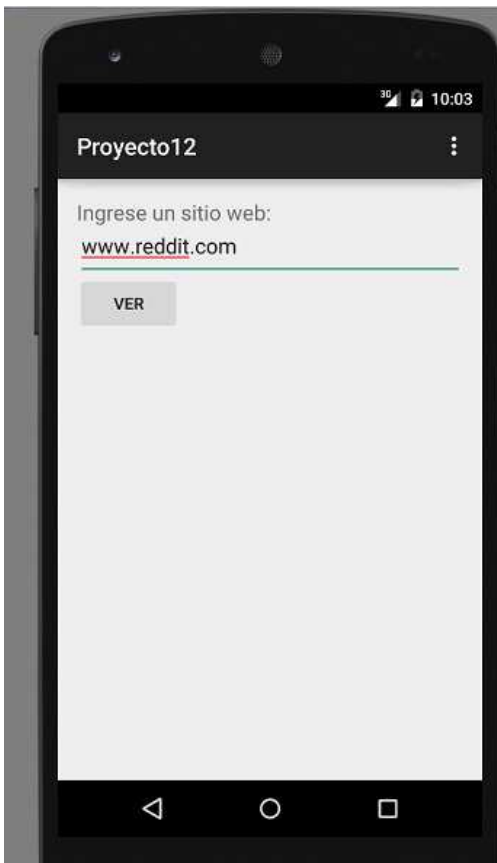
```
<uses-permission android:name="android.permission.INTERNET" />
```

Luego el archivo AndroidManifest.xml queda con el permiso agregado:

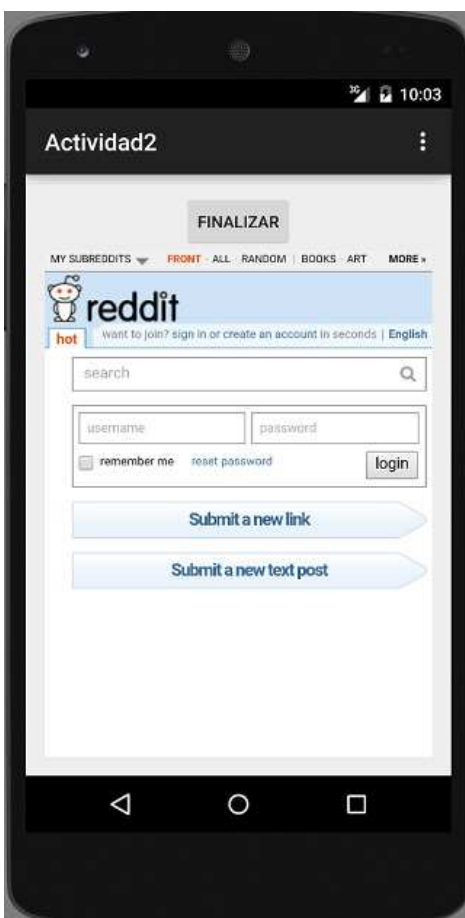


Ahora nuestro programa debería estar funcionando completamente.

La primer ventana debería ser algo similar a esto:



La segunda ventana debería ser algo similar a esto otro:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto012.zip](#)

[Retornar](#)

13 - Almacenamiento de datos mediante la clase SharedPreferences

La plataforma de Android nos da varias facilidades para el almacenamiento permanente de datos (es decir que los mismos no se borran cuando se apaga o cierra la aplicación)

Según el tipo de necesidades utilizaremos alguno de estos métodos:

1. Mediante la clase SharedPreferences.
2. Mediante archivos de Texto.
3. En una base de datos con acceso a SQL.

No será raro que una aplicación utilice más de uno de estos métodos para el almacenamiento de datos.

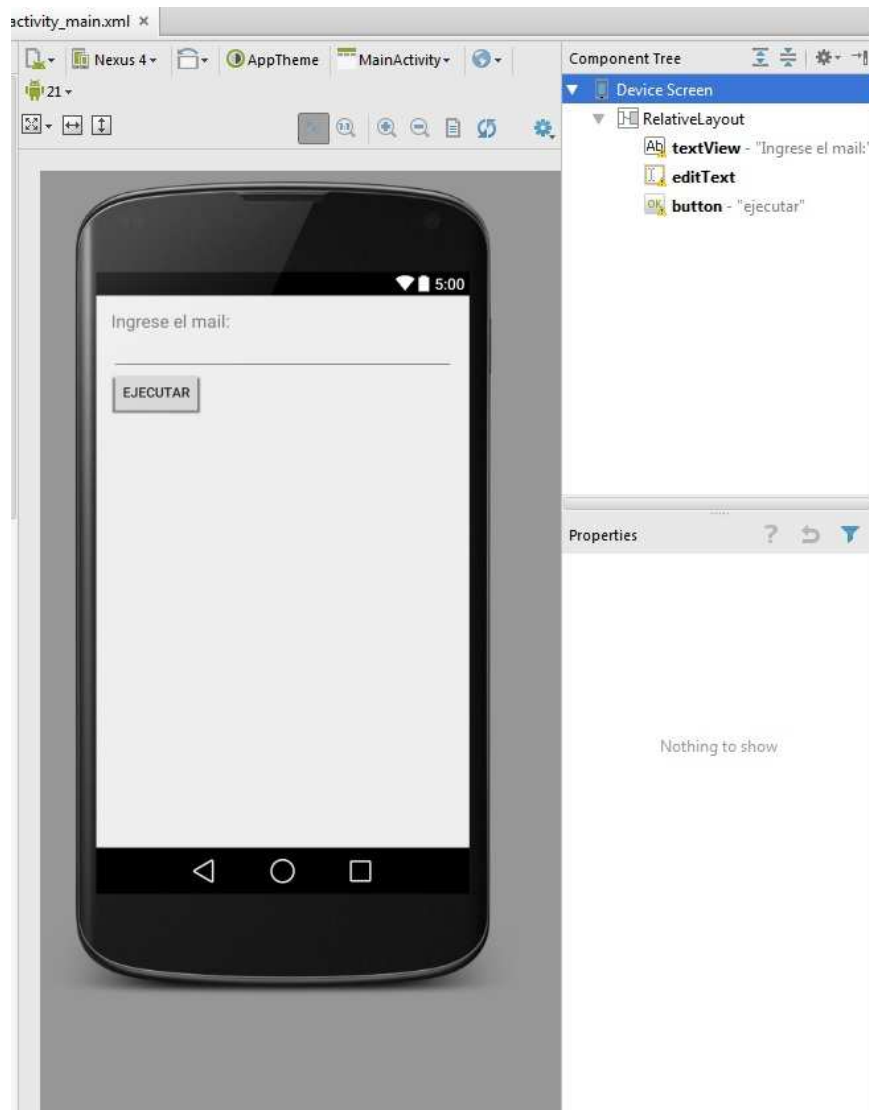
Cuando tenemos que almacenar una cantidad limitada de datos es adecuado utilizar la clase SharedPreferences. Por ejemplo configuraciones de la aplicación como pueden ser colores de pantalla, nivel actual en un juego, datos iniciales de controles de entrada de dato etc.

Problema 1:

Confeccionar un programa que solicite el ingreso del mail de una persona. Guardar el mail ingresado utilizando la clase SharedPreferences. Cada vez que se inicie la aplicación almacenar en el control EditText el último mail ingresado. Disponer un botón para almacenar el mail ingresado y finalizar el programa.

Crearemos un nuevo proyecto llamado proyecto013.

La interfaz visual a implementar es:



Es decir:

Disponemos un TextView la propiedad Text con "Ingrese el mail:".

Disponemos un EditText

Disponemos un Button su propiedad Text con el valor confirmar
 su propiedad onClick con el valor ejecutar

El código java es:

```
package ar.com.tutorialesya.proyecto013;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends ActionBarActivity {

    private EditText et1;

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

```

        et1=(EditText)findViewById(R.id.editText);
        SharedPreferences prefe=getSharedPreferences("datos", Context.MODE_PRIVATE);
        et1.setText(prefe.getString("mail",""));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void ejecutar(View v) {
        SharedPreferences preferencias=getSharedPreferences("datos",Context.MODE_PRIVATE);
        SharedPreferences.Editor editor=preferencias.edit();
        editor.putString("mail", et1.getText().toString());
        editor.commit();
        finish();
    }
}

```

Obtenemos la referencia del EditText:

```
et1=(EditText)findViewById(R.id.editText);
```

Obtenemos una referencia de un objeto de la clase SharedPreferences a través del método getSharedPreferences. El primer parámetro es el nombre del archivo de preferencias y el segundo la forma de creación del archivo (MODE_PRIVATE indica que solo esta aplicación puede consultar el archivo XML que se crea)

```
SharedPreferences prefe=getSharedPreferences("datos",Context.MODE_PRIVATE);
```

Para extraer los datos del archivo de preferencias debemos indicar el nombre a extraer y un valor de retorno si dicho nombre no existe en el archivo de preferencias (en nuestro ejemplo la primera vez que se ejecute nuestro programa como es lógico no existe el archivo de preferencias lo que hace que Android lo cree, si tratamos de extraer el valor de mail retornará el segundo parámetro es decir el String con una cadena vacía:

```
et1.setText(prefe.getString("mail",""));
```

Cuando se presiona el botón "Confirmar" lo que hacemos es grabar en el archivo de preferencias el contenido del EditText en una variable llamada "mail":

```

public void ejecutar(View v) {
    SharedPreferences preferencias=getSharedPreferences("datos",Context.MODE_PRIVATE);
    Editor editor=preferencias.edit();
    editor.putString("mail", et1.getText().toString());
    editor.commit();
    finish();
}

```

Debemos crear un objeto de la clase Editor y obtener la referencia del objeto de la clase SharedPreferences que acabamos de crear. Mediante el método putString almacenamos en mail el valor del String cargado en el EditText. Luego debemos llamar al método commit de la clase Editor para que el dato quede almacenado en forma permanente en el archivo de preferencias. Esto hace que cuando volvamos a arrancar la aplicación se recupere el último mail ingresado.

Recordemos que el método finish de la clase Activity finaliza la actividad actual (como tenemos una aplicación con una sola actividad finalizará completamente nuestro programa).

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto013.zip](#)

Comentarios extras.

El modo de operación del archivo puede ser:

MODE_PRIVATE solo la aplicación puede acceder al archivo de preferencias.

MODE_WORLD_READABLE otras aplicaciones pueden consultar el archivo de preferencias

MODE_WORLD_WRITEABLE otras aplicaciones pueden consultar y modificar el archivo.

MODE_MULTI_PROCESS varios procesos pueden acceder (Requiere Android 2.3)

Cuando guardamos datos en el archivo de preferencias podemos almacenar distintos tipos de datos según el método que llamemos en el momento de grabar:

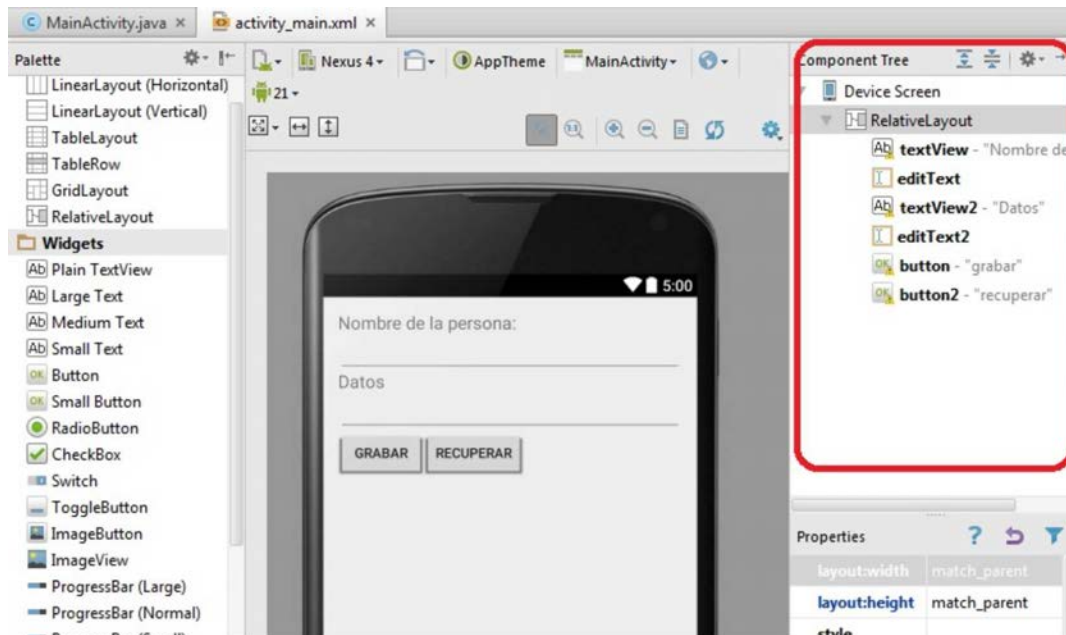
```
editor.putInt("edad",3);
editor.putBoolean("activo", true);
editor.putFloat("altura", 2.3f);
```

Cuando los recuperamos debemos indicar también que tipo de datos extraemos:

```
int e=prefe.getInt("edad", 0);
boolean acti=prefe.getBoolean("activo", false);
float alt=prefe.getFloat("altura", 0f);
```

Problema 2:

Confeccionar un programa que permita administrar una agenda personal. Nuestra clave será el nombre de la persona. La interfaz visual a implementar será similar a esta:



El código fuente en java es:

```
package ar.com.tutorialesya.proyecto014;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        et1=(EditText) findViewById(R.id.editText);
        et2=(EditText) findViewById(R.id.editText2);
    }

    @Override
```

```

public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void grabar(View v) {
    String nombre=et1.getText().toString();
    String datos=et2.getText().toString();
    SharedPreferences preferencias=getSharedPreferences("agenda", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor=preferencias.edit();
    editor.putString(nombre, datos);
    editor.commit();
    Toast.makeText(this,"Datos grabados",Toast.LENGTH_LONG).show();
}

public void recuperar(View v) {
    String nombre=et1.getText().toString();
    SharedPreferences prefe=getSharedPreferences("agenda", Context.MODE_PRIVATE);
    String d=prefe.getString(nombre, "");
    if (d.length()==0) {
        Toast.makeText(this,"No existe dicho nombre en la agenda",Toast.LENGTH_LONG).show();
    }
    else {
        et2.setText(d);
    }
}
}

```

Definimos dos objetos de la clase EditText donde se ingresan el nombre de la persona y los datos de dicha persona:

```
private EditText et1,et2;
```

Cuando se presiona el botón grabar:

```

public void grabar(View v) {
    String nombre=et1.getText().toString();
    String datos=et2.getText().toString();
    SharedPreferences preferencias=getSharedPreferences("agenda", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor=preferencias.edit();
    editor.putString(nombre, datos);
    editor.commit();
    Toast.makeText(this,"Datos grabados",Toast.LENGTH_LONG).show();
}

```

Extraemos los dos datos de los EditText, creamos un objeto de la clase SharedPreferences con el nombre de "agenda". Creamos un objeto de la clase Editor y procedemos a grabar en el archivo de preferencias mediante putString:

```
editor.putString(nombre, datos);
```

Significa que en el archivo de preferencias se almacena una entrada con el nombre de la persona y los datos de dicha persona.

Por otro lado tenemos la lógica para recuperar los datos de una persona de la agenda:

```

public void recuperar(View v) {
    String nombre=et1.getText().toString();

```

```

    SharedPreferences prefe=getSharedPreferences("agenda", Context.MODE_PRIVATE);
    String d=prefe.getString(nombre, "");
    if (d.length()==0) {
        Toast.makeText(this,"No existe dicho nombre en la agenda",Toast.LENGTH_LONG).show();
    }
    else {
        et2.setText(d);
    }
}

```

Abrimos el archivo de preferencias y llamamos al método getString buscando el nombre ingresado en el et1. En el caso que lo encuentre retorna el dato asociado a dicha clave.

En el emulador podemos ver como ingresamos y recuperamos datos de la agenda:

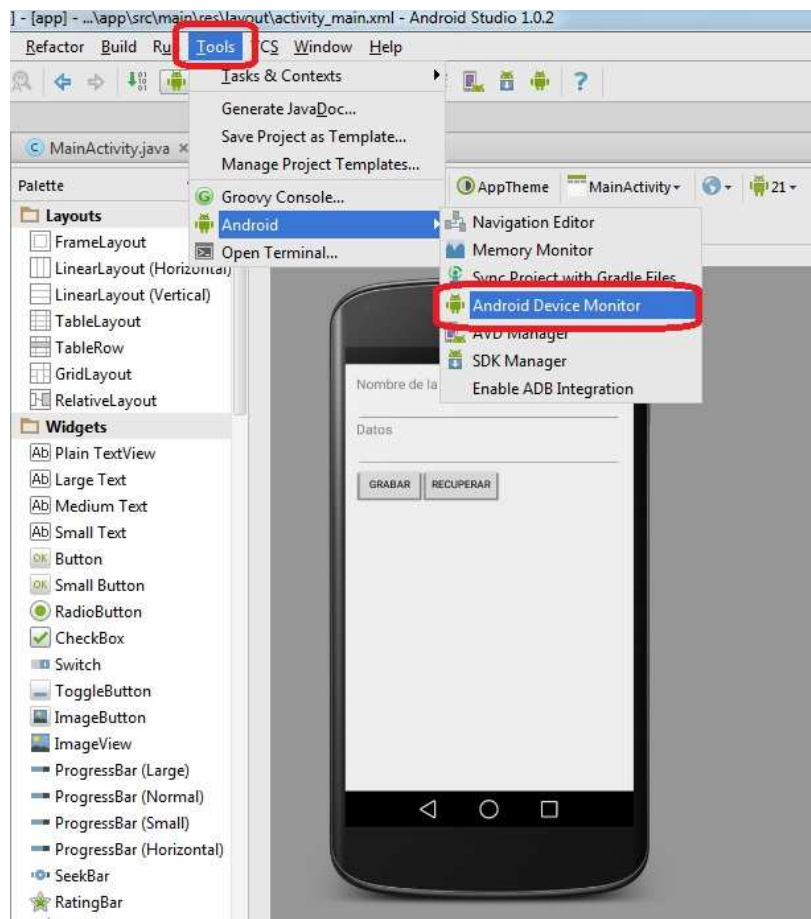


Este proyecto lo puede descargar en un zip desde este enlace: [proyecto014.zip](#)

Android Device Monitor

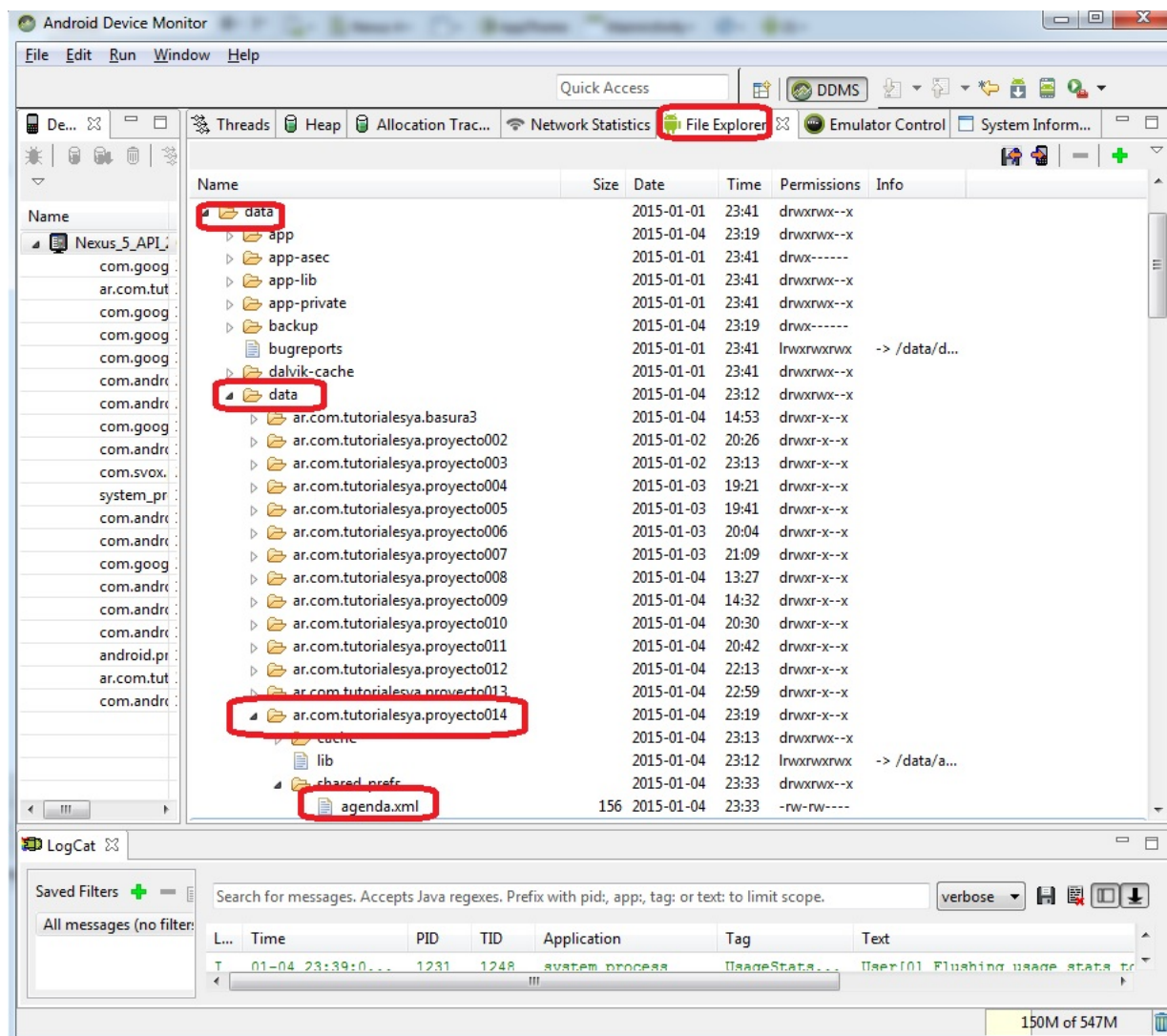
Veamos una herramienta para poder entre otras cosas acceder al sistema de archivos del emulador de Android.

Vamos a arrancar el "Android Device Monitor", lo podemos ubicar en el menú de opciones:

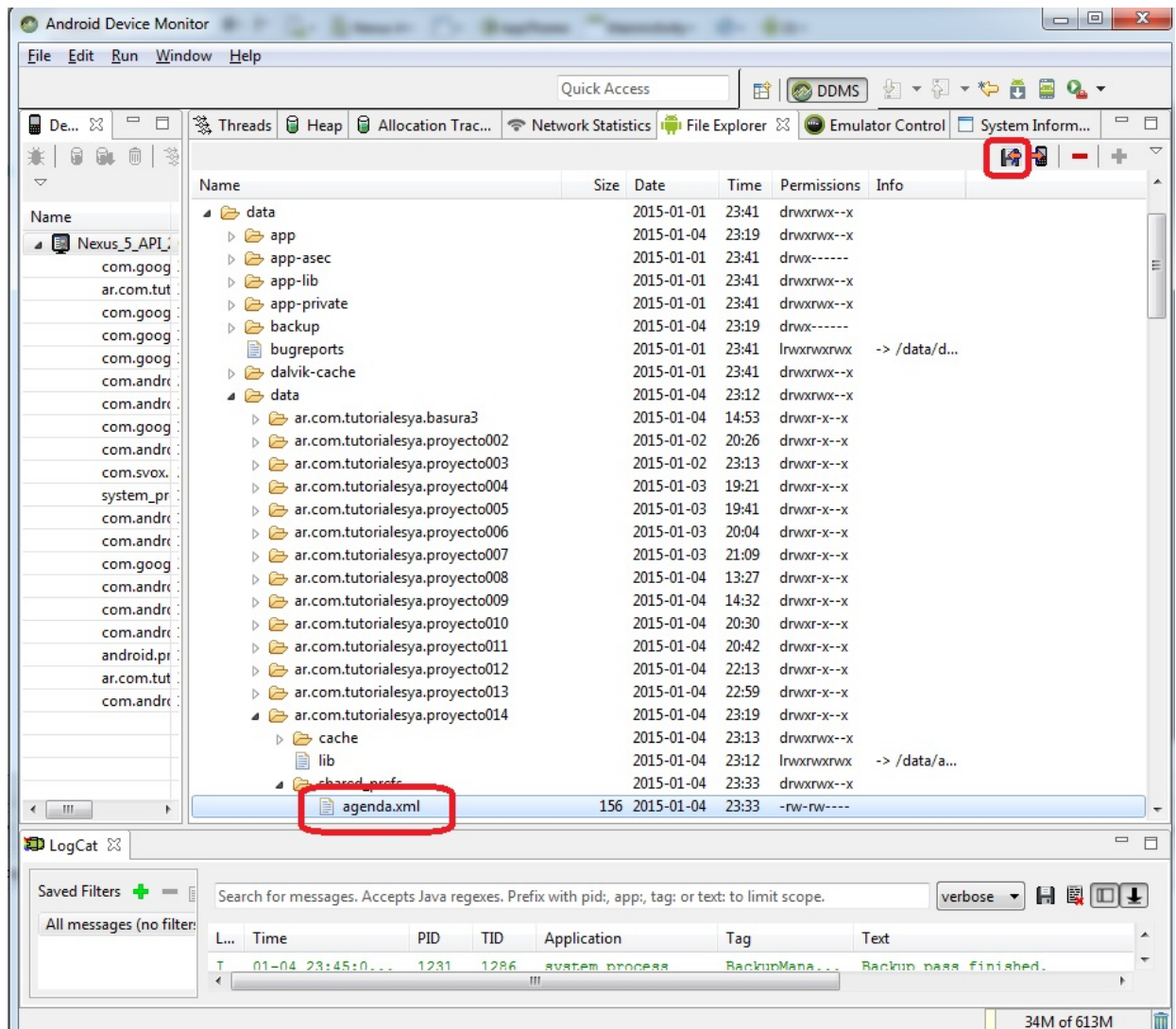


Mediante este programa vamos a acceder al "File Explorer" del emulador para localizar el archivo de preferencias donde se ha guardado la agenda.

Seleccionamos la pestaña "File Explorer" y luego navegamos hasta la carpeta que contiene los datos de la aplicación, podemos ver que hay un archivo llamado agenda.xml (siempre y cuando ya hayamos ejecutado la aplicación y cargado datos):



Podemos extraer este archivo del dispositivo para analizarlo, para ello presionamos el ícono de la parte superior derecha estando seleccionado el archivo agenda.xml:



Luego podemos utilizar cualquier editor de texto para ver el contenido del archivo agenda.xml y veremos que tiene algo similar a esto (depende que datos usted cargó):

```

1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <string name="ana">tel:56120</string>
4      <string name="diego">tel:12345</string>
5  </map>
6

```

Como podemos ver el archivo xml tiene una marca map y dentro un conjunto de marcas string donde define una propiedad llamada name donde se almacena el nombre de la persona y en el interior de la marca sus datos.

La clase SharedPreferences se utiliza siempre que tenemos que almacenar un valor como clave (en este caso el nombre de la persona) y los datos asociados a dicha clave.

Problema propuesto

1. Realizar un programa que genere un número aleatorio entre 1 y 50, pedir que el operador lo adivine, informar si ganó o si el número es mayor o menor al ingresado. Cuando el operador lo adivine incrementar en uno el puntaje de juego. Cada vez que se ingrese a la aplicación mostrar el puntaje actual, es decir recordar el puntaje en un archivo de preferencias.

La interfaz visual de la aplicación a desarrollar es:

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto015.zip](#)

[Retornar](#)

14 - Almacenamiento de datos en un archivo de texto en la memoria interna

Otra posibilidad de almacenar datos en nuestro dispositivo Android es el empleo de un archivo de texto que se guardará en el almacenamiento interno del equipo (la otra posibilidad es almacenarlo en una tarjeta SD Card)

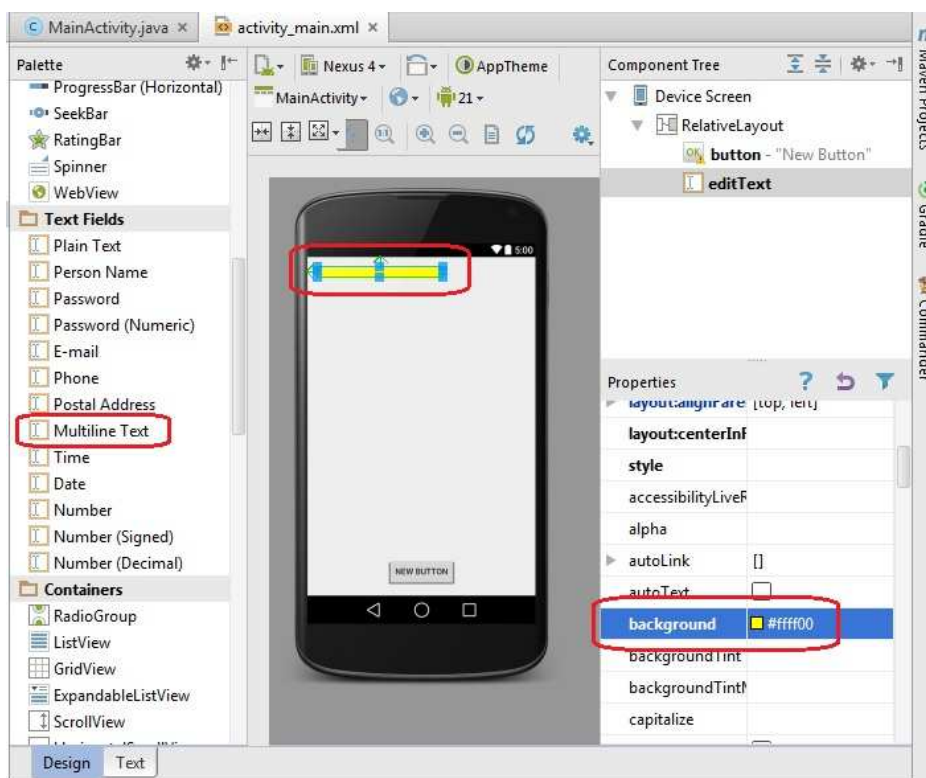
Problema 1:

Confeccionar un programa que permita almacenar notas en un control EditText y cuando se presione un botón almacenar los datos del EditText en un archivo de texto llamado "notas.txt".

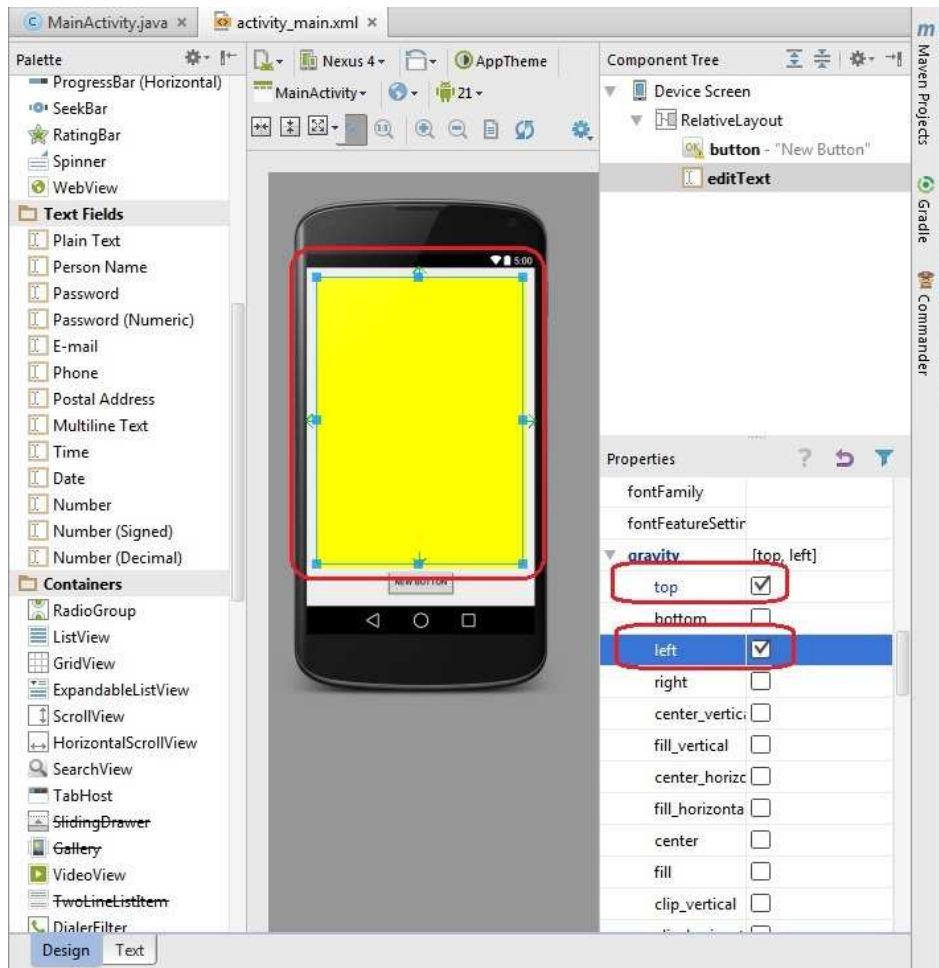
Cada vez que se ingrese al programa verificar si existe el archivo de textos "notas.txt", proceder a su lectura y almacenamiento de datos en el EditText.

Crear un proyecto en Android Studio y definir como nombre: proyecto016.

Para crear la interfaz visual primero disponemos un botón alineado en la parte inferior del celular y luego de la pestaña "Text Fields" seleccionamos un objeto de la clase EditText ("Multiline Text") y lo disponemos en la parte superior de la pantalla e inicializamos la propiedad background del EditText con el valor #ffff00 (que corresponde con el color amarillo):



Finalmente procedemos a redimensionar el EditText por medio del mouse y configuramos la propiedad gravity tildando los valores top y left para que los datos que ingresa el operador aparezcan en la parte superior izquierda y no centrados:



El código fuente de la aplicación:

```
package ar.com.tutorialesya.proyecto016;

import android.app.Activity;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends ActionBarActivity {
    private EditText et1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        et1=(EditText) findViewById(R.id.editText);
        String[] archivos = fileList();

        if (existe(archivos, "notas.txt"))
            try {
                InputStreamReader archivo = new InputStreamReader(
                    openFileInput("notas.txt"));
                BufferedReader br = new BufferedReader(archivo);
                String linea = br.readLine();
                String todo = "";
            }
    }
}
```

```

        while (linea != null) {
            todo = todo + linea + "\n";
            linea = br.readLine();
        }
        br.close();
        archivo.close();
        et1.setText(todo);
    } catch (IOException e) {
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

private boolean existe(String[] archivos, String archbusca) {
    for (int f = 0; f < archivos.length; f++)
        if (archbusca.equals(archivos[f]))
            return true;
    return false;
}

public void grabar(View v) {
    try {
        OutputStreamWriter archivo = new OutputStreamWriter(openFileOutput(
            "notas.txt", Activity.MODE_PRIVATE));
        archivo.write(et1.getText().toString());
        archivo.flush();
        archivo.close();
    } catch (IOException e) {
    }
    Toast t = Toast.makeText(this, "Los datos fueron grabados",
        Toast.LENGTH_SHORT);
    t.show();
    finish();
}
}

```

Veamos primero como grabamos datos en un archivo de texto. Esto se hace en el método grabar que se ejecuta cuando presionamos el botón "grabar" (recordemos de inicializar la propiedad "onClick" del botón con el valor "grabar"):

```

public void grabar(View v) {
    try {
        OutputStreamWriter archivo = new OutputStreamWriter(openFileOutput("notas.txt",Activity.MODE_PRIVATE)

```

Creamos un objeto de la clase OutputStreamWriter y al constructor de dicha clase le enviamos el dato que retorna el método openFileOutput propio de la clase ActionBarActivity que le pasamos como parámetro el nombre del archivo de texto y el modo de apertura.

Seguidamente si el archivo se creó correctamente procedemos a llamar al método write y le pasamos el String a grabar,

en este caso extraemos los datos del EditText:

```
archivo.write(et1.getText().toString());
```

Luego de grabar con el método write llamamos al método flush para que vuelque todos los datos que pueden haber quedado en el buffer y procedemos al cerrado del archivo:

```
archivo.flush();
archivo.close();
```

Todo esto está cerrado en un try/catch para verificar si sucede algún error en la apertura del archivo.

Finalmente mostramos un mensaje temporal en pantalla utilizando la clase Toast:

```
Toast t=Toast.makeText(this,"Los datos fueron grabados", Toast.LENGTH_SHORT);
t.show();
finish();
```

Para crear un objeto de la clase Toast llamamos al método makeText de la clase Toast y le pasamos la referencia del ActionBarActivity actual, el String a mostrar y el tiempo de duración del mensaje. Con el objeto devuelto por el método makeText procedemos a llamar seguidamente al método show para que se muestre el mensaje.

Es común llamar al método show de la clase Toast en una sola línea como esta:

```
Toast.makeText(this,"Los datos fueron grabados", Toast.LENGTH_SHORT).show();
```

El método onCreate tiene por objetivo verificar si existe el archivo de texto, proceder a su lectura y mostrar su contenido en el EditText.

Primero obtenemos la lista de todos los archivos creados por la Activity. En nuestro ejemplo puede ser cero o uno:

```
String []archivos=fileList();
```

Llamamos a un método que verifica si en el vector de tipo String existe el archivo "notas.txt":

```
if (existe(archivos,"notas.txt"))
```

En el caso que me retorne true procedemos a crear un objeto de la clase InputStreamReader y al constructor de dicha clase le pasamos el dato devuelto por el método openFileInput:

```
InputStreamReader archivo=new InputStreamReader(openFileInput("notas.txt"));
```

Creamos un objeto de la clase BufferedReader y le pasamos al constructor la referencia del objeto de la clase InputStreamReader:

```
BufferedReader br=new BufferedReader(archivo);
```

Leemos la primer línea del archivo de texto:

```
String linea=br.readLine();
```

Inicializamos un String vacío:

```
String todo="";
```

Mientras el método readLine de la clase BufferedReader devuelva un String:

```
while (linea!=null)
{
```

Lo concatenamos al String junto a un salto de línea:

```
todo=todo+linea+"\n";
```

Leemos la próxima línea:

```
    linea=br.readLine();
}
```

Llamamos al método close de la clase BufferedReader y al del InputStreamReader:

```
br.close();
```



```
archivo.close();
```

Cargamos el EditText con el contenido del String que contiene todos los datos del archivo de texto:

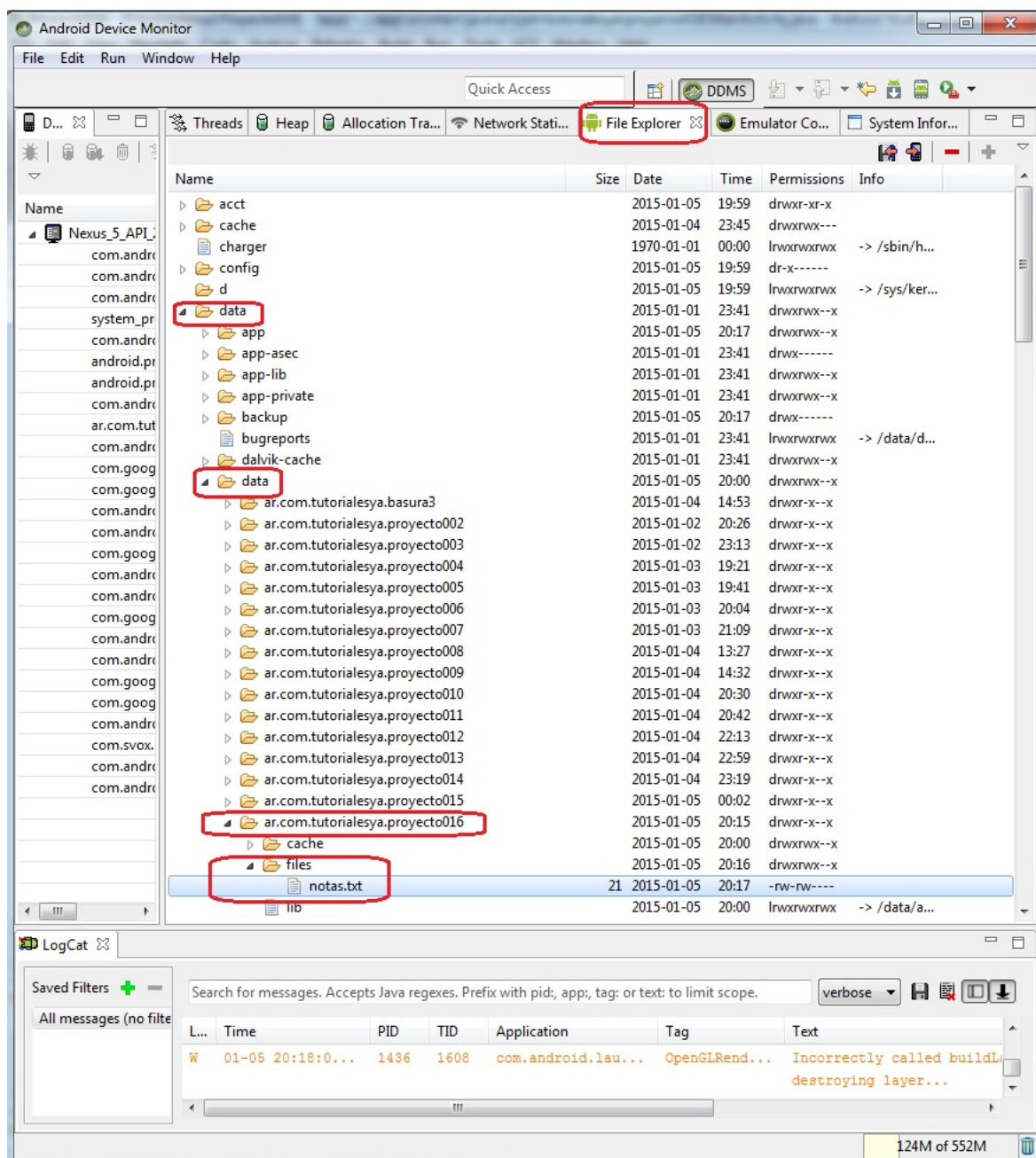
```
et1.setText(todo);
```

El método existe llega un vector de tipo String y otro String a buscar. Dentro de un for verificamos el String a buscar con cada uno de los String del vector, si lo encontramos retornamos true. Si recorre todo el for sin encontrarlo fuera del for retornamos false:

```
private boolean existe(String[] archivos,String archbusca)
{
    for(int f=0;f<archivos.length;f++)
        if (archbusca.equals(archivos[f]))
            return true;
    return false;
}
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto016.zip](#)

Luego de ejecutar el programa podemos entrar al "Android Device Monitor" (podemos ejecutarlo desde el menú del Android Studio: Tools->Android->Android Device Monitor) y localizar en el dispositivo donde se guarda el archivo de texto creado:

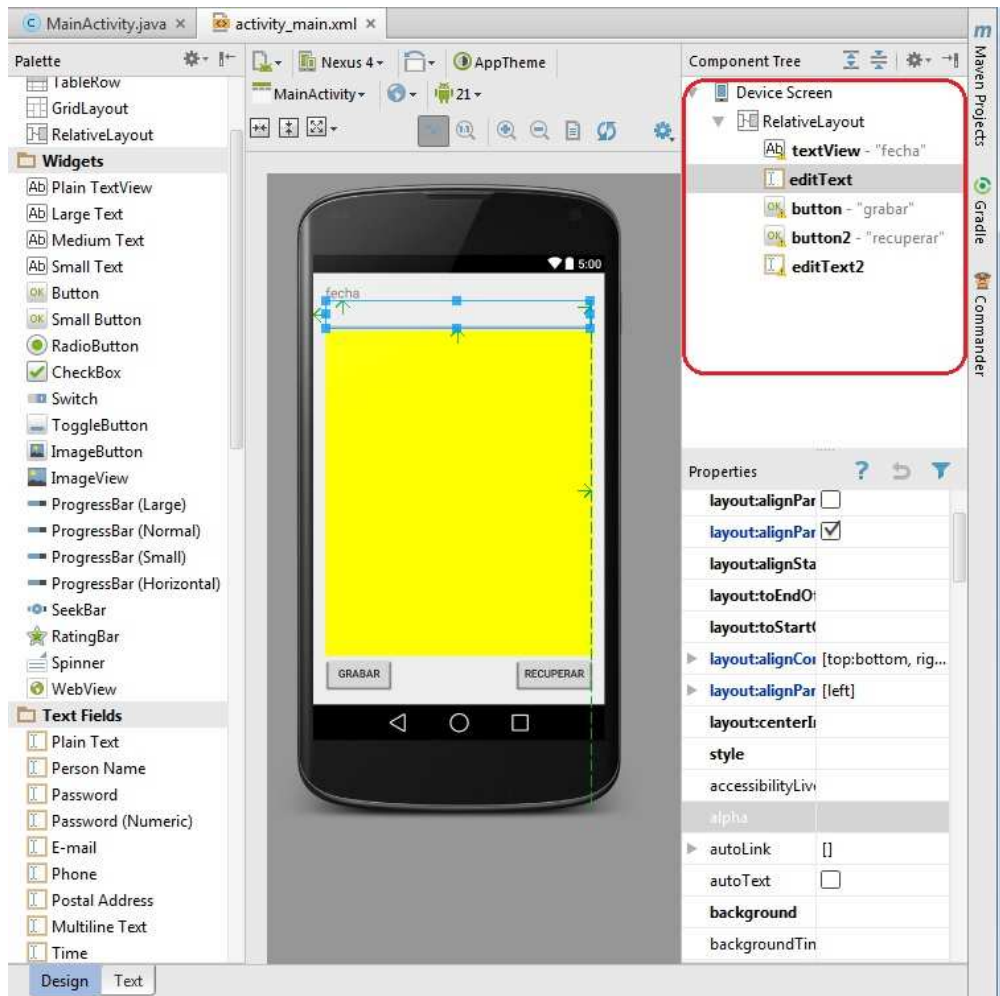


Como vemos se crea una carpeta llamada "files" que contiene el archivo creado (si creamos más archivos de texto se almacenarán en esta carpeta)

Problema 2:

Confeccionar un programa para administrar un calendario de actividades diarias. Los nombres de archivos corresponderán a las fechas que ingresamos. Luego cuando consultamos una fecha verificamos si hay un archivo de texto que coincida con dicha fecha.

La interfaz visual a crear será la siguiente:



Es decir disponemos un "Medium Text" donde dice fecha. Un EditText de tipo "Date" donde se carga la fecha. Luego otro EditText de tipo "Multiline Text" y finalmente un botón.

El código fuente de la aplicación:

```
package ar.com.tutorialesya.proyecto017;

import android.app.Activity;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    private EditText et1,et2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```

        setContentView(R.layout.activity_main);
        et1=(EditText)findViewById(R.id.editText);
        et2=(EditText)findViewById(R.id.editText2);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void grabar(View v) {
        String nomarchivo=et1.getText().toString();
        nomarchivo=nomarchivo.replace('/', '-');
        try {
            OutputStreamWriter archivo = new OutputStreamWriter(openFileOutput(
                nomarchivo, Activity.MODE_PRIVATE));
            archivo.write(et2.getText().toString());
            archivo.flush();
            archivo.close();
        } catch (IOException e) {
        }
        Toast t = Toast.makeText(this, "Los datos fueron grabados",
            Toast.LENGTH_SHORT);
        t.show();
        et1.setText("");
        et2.setText("");
    }

    public void recuperar(View v) {
        String nomarchivo=et1.getText().toString();
        nomarchivo=nomarchivo.replace('/', '-');
        boolean enco=false;
        String[] archivos = fileList();
        for (int f = 0; f < archivos.length; f++)
            if (nomarchivo.equals(archivos[f]))
                enco= true;
        if (enco==true) {
            try {
                InputStreamReader archivo = new InputStreamReader(
                    openFileInput(nomarchivo));
                BufferedReader br = new BufferedReader(archivo);
                String linea = br.readLine();
                String todo = "";
                while (linea != null) {
                    todo = todo + linea + "\n";
                    linea = br.readLine();
                }
                br.close();
                archivo.close();
                et2.setText(todo);
            } catch (IOException e) {
            }
        } else
    }

```

```

        {
            Toast.makeText(this,"No hay datos grabados para dicha fecha",Toast.LENGTH
            et2.setText("");
        }
    }
}

```

Definimos dos variables que hacen referencia a los EditText donde se cargan la fecha en uno y las actividades de dicho día en el otro:

```
private EditText et1,et2;
```

En el método onCreate obtenemos las referencias de los dos EditText:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    et1=(EditText)findViewById(R.id.editText);
    et2=(EditText)findViewById(R.id.editText2);
}

```

El método grabar que se ejecuta cuando presionamos el botón grabar (no olvidar de inicializar la propiedad onClick de cada botón con el nombre del método respectivo) tenemos primero que extraer la fecha ingresada en el primer EditText y remplazar las barras de la fecha por guiones ya que no se puede utilizar este caracter dentro de un nombre de archivo en Android:

```

String nomarchivo=et1.getText().toString();
nomarchivo=nomarchivo.replace('/', '-');

```

Creamos un objeto de la clase OutputStreamWriter y al constructor de dicha clase le enviamos el dato que retorna el método openFileOutput propio de la clase ActionBarActivity que le pasamos como parámetro el nombre del archivo de texto y el modo de apertura.

```

try {
    OutputStreamWriter archivo = new OutputStreamWriter(openFileOutput(
        nomarchivo, Activity.MODE_PRIVATE));
}

```

Seguidamente si el archivo se creó correctamente procedemos a llamar al método write y le pasamos el String a grabar, en este caso extraemos los datos del EditText:

```
archivo.write(et2.getText().toString());
```

Luego de grabar con el método write llamamos al método flush para que vuelque todos los datos que pueden haber quedado en el buffer y procedemos al cerrado del archivo:

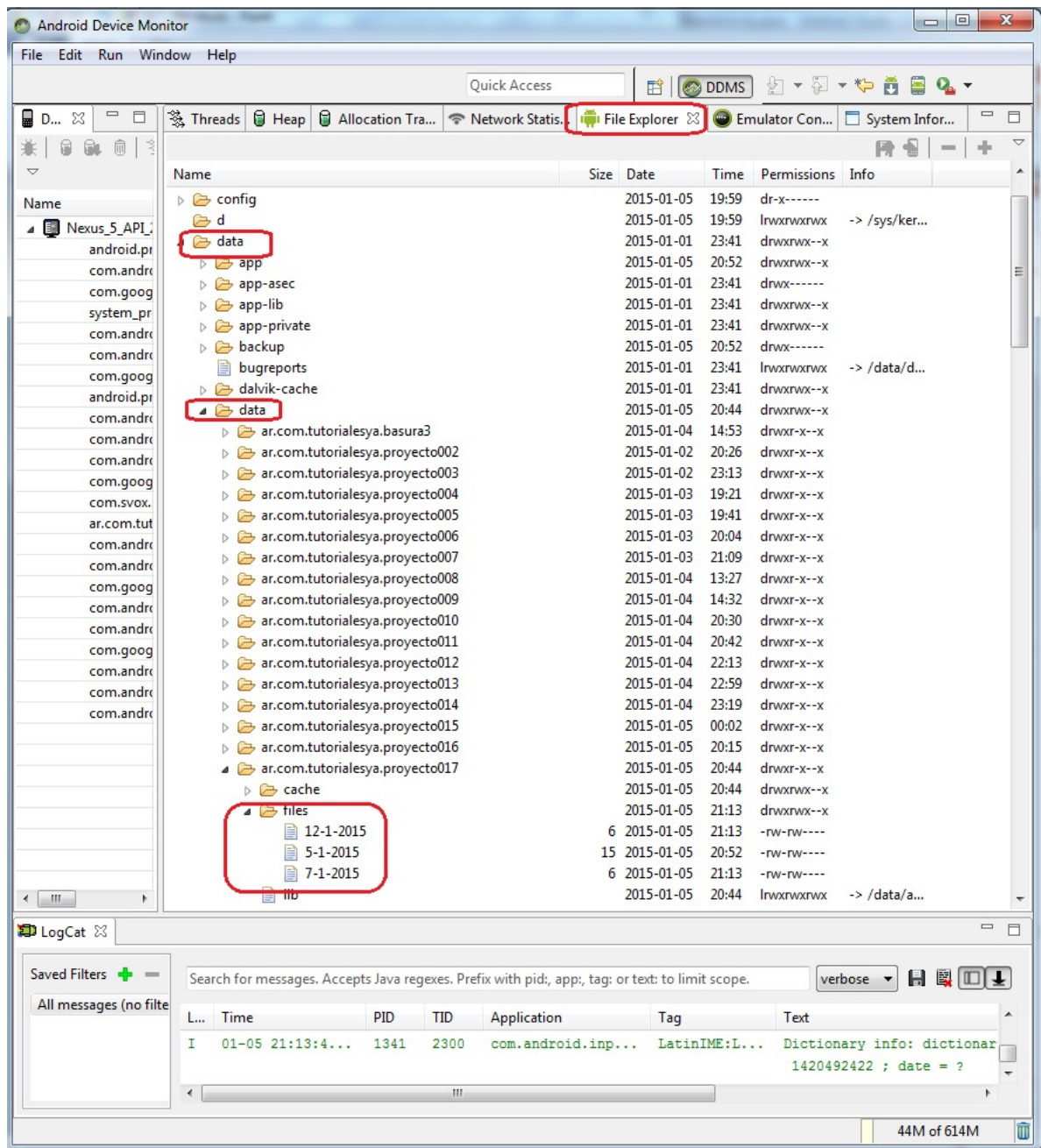
```

archivo.flush();
archivo.close();

```

Cada vez que se graba un dato se genera un archivo de texto para dicha fecha, si ya había un archivo con el mismo nombre (es decir la misma fecha) se pisa el anterior.

Desde el "Android Device Monitor" podemos observar todos los archivos de texto que se van creando a medida que registramos actividades en la aplicación que desarrollamos:



Para recuperar los datos de una determinada fecha se ejecuta el método "recuperar":

```
public void recuperar(View v) {
    String nomarchivo=et1.getText().toString();
    nomarchivo=nomarchivo.replace('/', '-');
    boolean enco=false;
    String[] archivos = fileList();
    for (int f = 0; f < archivos.length; f++)
        if (nomarchivo.equals(archivos[f]))
            enco= true;
    if (enco=true) {
        try {
            InputStreamReader archivo = new InputStreamReader(
                openFileInput(nomarchivo));
            BufferedReader br = new BufferedReader(archivo);
            String linea = br.readLine();
            String todo = "";
            while (linea != null) {
                todo = todo + linea + "\n";
                linea = br.readLine();
            }
            br.close();
            archivo.close();
            et2.setText(todo);
        } catch (IOException e) {
        }
    } else
    {
        Toast.makeText(this,"No hay datos grabados para dicha fecha",Toast.LENGTH_LONG).show();
    }
}
```

```

        et2.setText("");
    }
}

```

Lo primero que hacemos es recuperar del EditText la fecha que ingresó el operador para buscar:

```
String nomarchivo=et1.getText().toString();
```

Remplazamos las barras por guiones (recordemos que grabamos guiones en la carga ya que la barra de una fecha no es un caracter válido en un archivo en Android):

```
nomarchivo=nomarchivo.replace('/', '-');
```

Obtenemos la lista de todos los archivos que almacena la aplicación mediante la llamada al método `fileList()`:

```
String[] archivos = fileList();
```

Mediante un for recorremos el vector con los nombres de archivos y los comparamos con la fecha ingresada, en el caso de encontrarlo procedemos a cambiar el estado de una bandera:

```

for (int f = 0; f < archivos.length; f++)
    if (nomarchivo.equals(archivos[f]))
        enco= true;

```

Si la bandera está en true significa que existe el archivo por lo que procedemos a abrirlo, leerlo y cargar el et2 con los datos del archivo:

```

if (enco==true) {
    try {
        InputStreamReader archivo = new InputStreamReader(
            openFileInput(nomarchivo));
        BufferedReader br = new BufferedReader(archivo);
        String linea = br.readLine();
        String todo = "";
        while (linea != null) {
            todo = todo + linea + "\n";
            linea = br.readLine();
        }
        br.close();
        archivo.close();
        et2.setText(todo);
    } catch (IOException e) {
    }
}

```

La interfaz visual en tiempo de ejecución debe ser similar a esta:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto017.zip](#)

[Retornar](#)

15 - Almacenamiento de datos en un archivo de texto localizado en una tarjeta SD

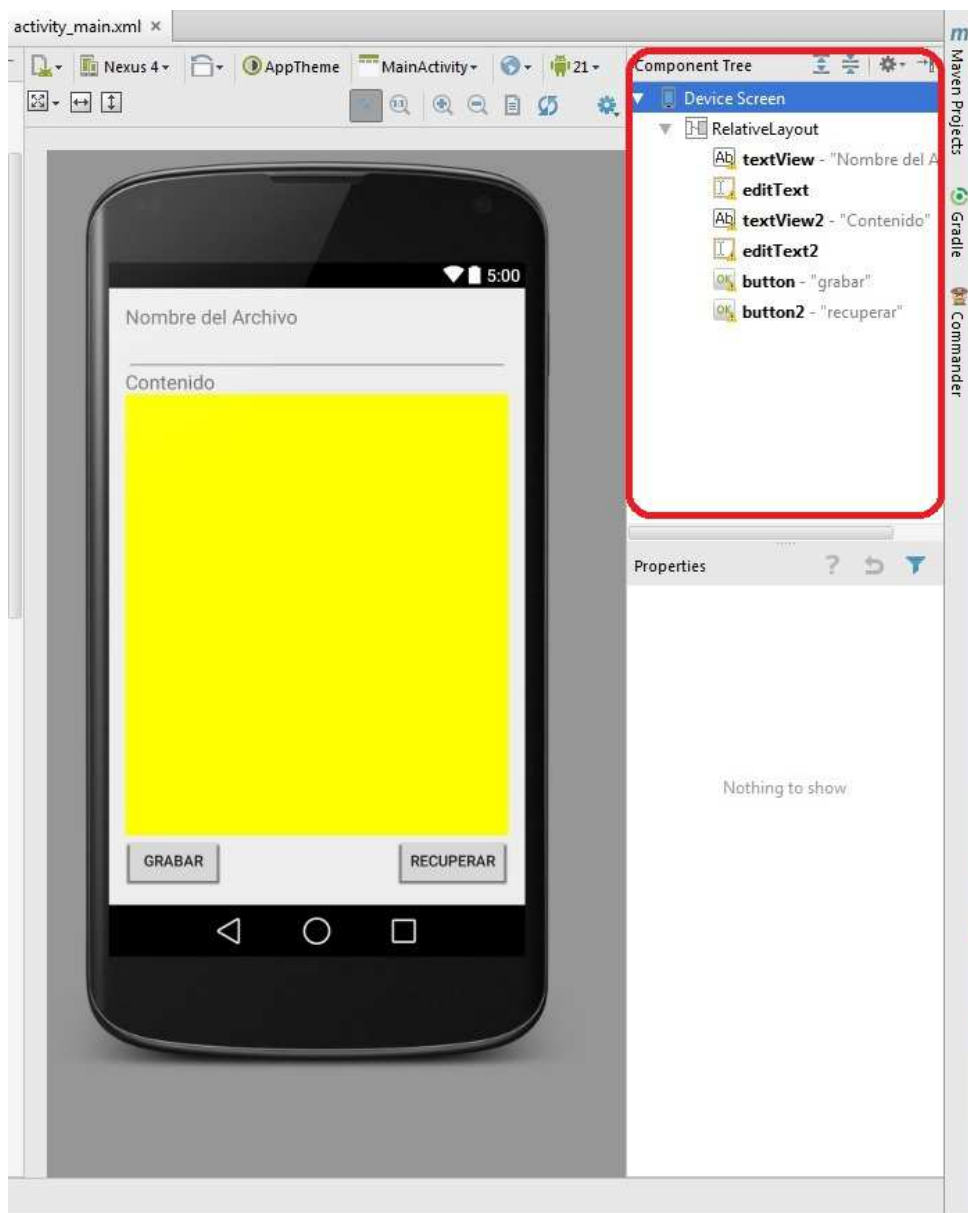
En el concepto anterior vimos como crear y leer un archivo de texto en la memoria interna del equipo Android. En algunas situaciones podría ser útil almacenar los datos en una tarjeta SD (tener en cuenta que no todos los dispositivos Android cuentan con esta característica), esto debido a su mayor capacidad o la facilidad de compartir los archivos con otras personas entregando la tarjeta SD.

Problema:

Confeccionar un programa que permita ingresar el nombre de un archivo y el contenido. Permitir grabar los datos ingresados al presionar un botón. Disponer un segundo botón que permita recuperar los datos del archivo de texto.

Hacer que los archivos se graben en una tarjeta SD.

La interfaz visual a implementar es la siguiente:

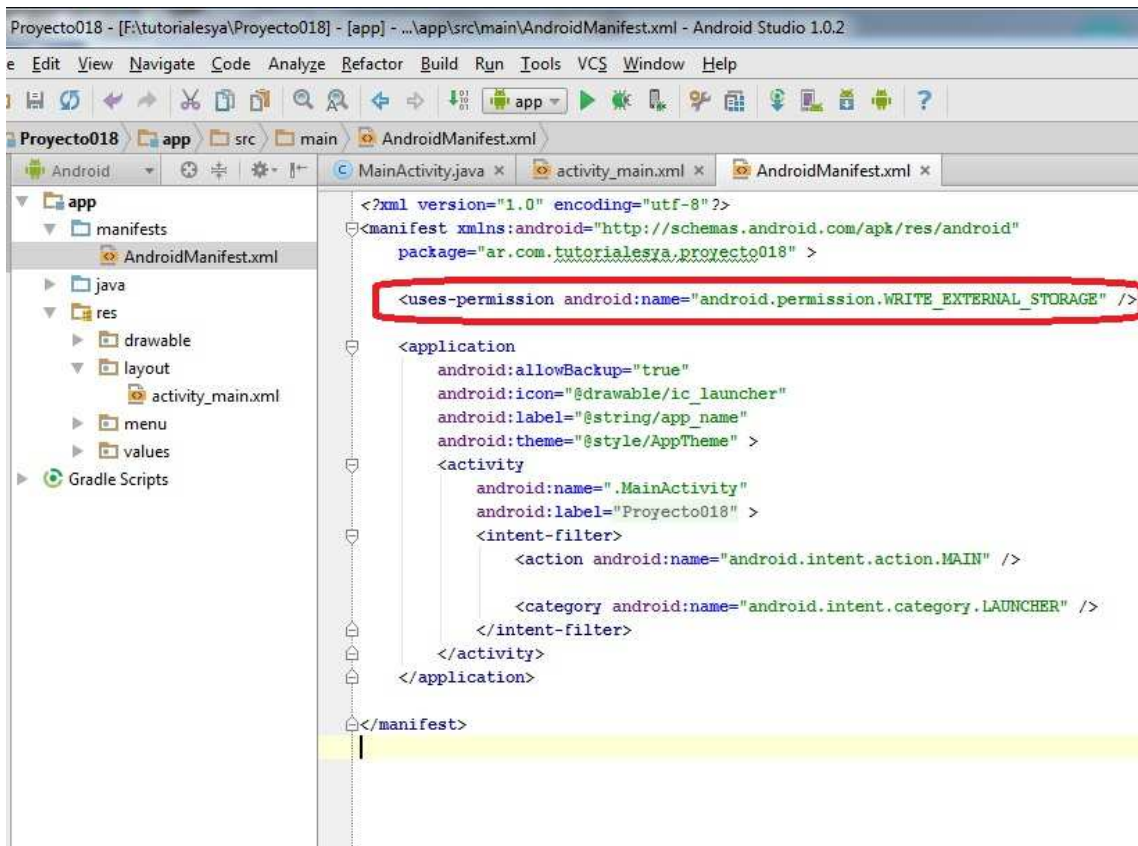


Podemos ver en la ventana "Component Tree" que la interfaz contiene dos TextView, dos EditText y dos Button.

No olvidar inicializar las propiedades onClick de cada botón con los nombres de métodos "grabar" y "recuperar".

El primer paso es modificar el archivo AndroidManifest.xml para permitir el acceso a la tarjeta SD desde nuestra aplicación esto lo hacemos desde el editor de texto del Android Studio

En la carpeta app/manifests podemos abrir el archivo "AndroidManifest.xml" y agregar la línea de permiso de acceso a la memoria externa del dispositivo:



Tenemos que agregar el permiso siguiente:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Tener cuidado que debe estar fuera del elemento application pero dentro del elemento manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ar.com.tutorialesya.proyecto018" >

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            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>
</manifest>
```

El código fuente es:

```
package ar.com.tutorialesya.proyecto018;

import android.os.Environment;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
```

```

import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2;

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

        et1=(EditText)findViewById(R.id.editText);
        et2=(EditText)findViewById(R.id.editText2);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void grabar(View v) {
        String nomarchivo = et1.getText().toString();
        String contenido = et2.getText().toString();
        try {

```



```

File tarjeta = Environment.getExternalStorageDirectory();
Toast.makeText(this,tarjeta.getAbsolutePath(),Toast.LENGTH_LONG).show();
File file = new File(tarjeta.getAbsolutePath(), nomarchivo);
OutputStreamWriter osw = new OutputStreamWriter(
    new FileOutputStream(file));
osw.write(contenido);
osw.flush();
osw.close();
Toast.makeText(this, "Los datos fueron grabados correctamente",
    Toast.LENGTH_SHORT).show();
et1.setText("");
et2.setText("");
} catch (IOException ioe) {
    Toast.makeText(this, "No se pudo grabar",
        Toast.LENGTH_SHORT).show();
}
}

public void recuperar(View v) {
    String nomarchivo = et1.getText().toString();
    File tarjeta = Environment.getExternalStorageDirectory();
    File file = new File(tarjeta.getAbsolutePath(), nomarchivo);
    try {
        FileInputStream fln = new FileInputStream(file);
        InputStreamReader archivo = new InputStreamReader(fln);
        BufferedReader br = new BufferedReader(archivo);
        String linea = br.readLine();
        String todo = "";
        while (linea != null) {
            todo = todo + linea + " ";
            linea = br.readLine();
        }
        br.close();
        archivo.close();
        et2.setText(todo);

    } catch (IOException e) {
        Toast.makeText(this, "No se pudo leer",
            Toast.LENGTH_SHORT).show();
    }
}
}
}

```

El método para grabar los datos en un archivo de texto localizado en una tarjeta SD comienza obteniendo el directorio raíz de la tarjeta a través del método `getExternalStorageDirectory()`, el mismo retorna un objeto de la clase `File`.

```

public void grabar(View v) {
    String nomarchivo = et1.getText().toString();
    String contenido=et2.getText().toString();
    try
    {
        File tarjeta = Environment.getExternalStorageDirectory();

```

Creamos un nuevo objeto de la clase File indicando el camino de la unidad SD y el nombre del archivo a crear:

```
File file = new File(tarjeta.getAbsolutePath(), nomarchivo);
```

Por último similar al acceso de un archivo interno creamos un objeto de la clase OutputStreamWriter:

```
OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream(file));
```

Grabamos el contenido del EditText:

```
osw.write(contenido);
```

Cerramos el archivo:

```
osw.flush();
```

```
osw.close();
```

```
Toast.makeText(this, "Los datos fueron grabados  
correctamente", Toast.LENGTH_SHORT).show();
```

```
et1.setText("");
```

```
et2.setText("");
```

```
}
```

```
catch (IOException ioe)
```

```
{
```

```
    Toast.makeText(this, "No se pudo grabar",
```

```
        Toast.LENGTH_SHORT).show();
```

```
}
```

```
}
```

Para la lectura del archivo nuevamente obtenemos la referencia de la tarjeta SD para obtener el path de la unidad de almacenamiento, el resto del algoritmo es similar al visto con un archivo interno:

```
public void recuperar(View v) {
```

```
    String nomarchivo = et1.getText().toString();
```

```
    File tarjeta = Environment.getExternalStorageDirectory();
```

```
    File file = new File(tarjeta.getAbsolutePath(), nomarchivo);
```

```
    try {
```

```
        FileInputStream fIn = new FileInputStream(file);
```

```
        InputStreamReader archivo = new InputStreamReader(fIn);
```

```
        BufferedReader br = new BufferedReader(archivo);
```

```
        String linea = br.readLine();
```

```
        String todo = "";
```

```
        while (linea != null)
```

```
        {
```

```
            todo = todo + linea + "\n";
```

```
            linea = br.readLine();
```

```
        }
```

```
        br.close();
```

```
        archivo.close();
```

```
        et2.setText(todo);
```

```
    } catch (IOException e)
```

```
    {
```

```
        Toast.makeText(this, "No se pudo leer",
```

```
            Toast.LENGTH_SHORT).show();
```

```
    }
```

```
}
```

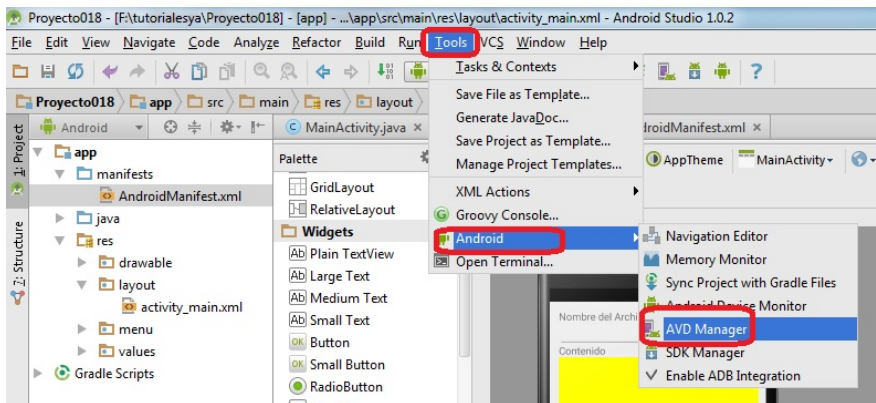
Este proyecto lo puede descargar en un zip desde este enlace: [proyecto018.zip](#)

Importante.

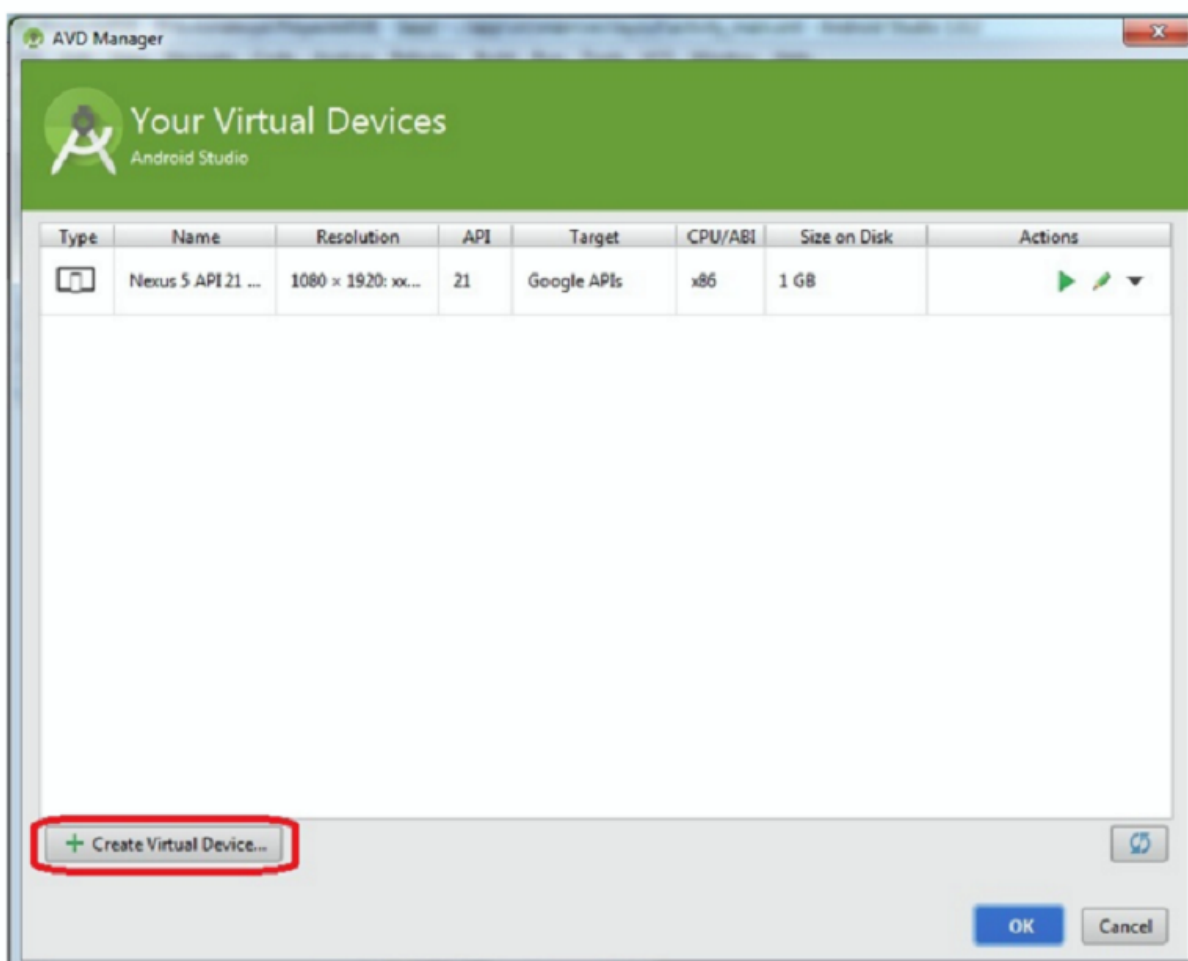
Si lo probamos con el emulador del Nexus 5 en el Android Studio cuando tratemos de grabar nos mostrará la notificación "No se pudo grabar", esto debido a que dicho celular no permite extender la memoria mediante tarjetas sd.

La solución para probar es crear otro dispositivo virtual. Los pasos para crear otro dispositivo virtual en Android Studio son los siguientes:

1 - Desde el menú de opciones del Android Studio accedemos a Tools->Android->AVD Manager.

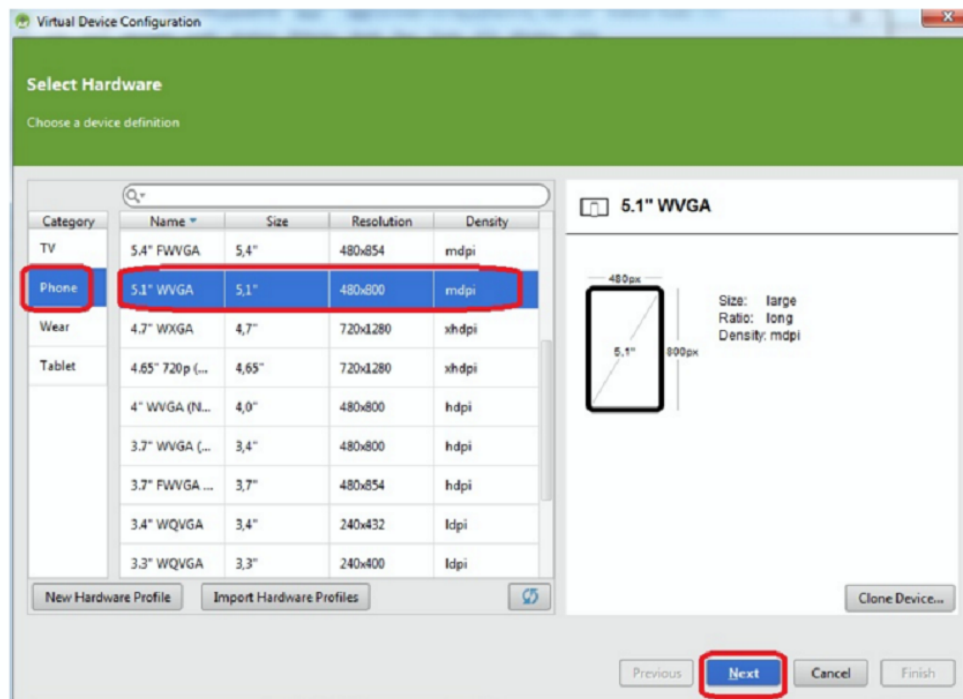


2. Aparece un diálogo con todas las máquinas virtuales creadas hasta el momento (en las primeras versiones de Android Studio crea una máquina virtual para el Nexus 5)

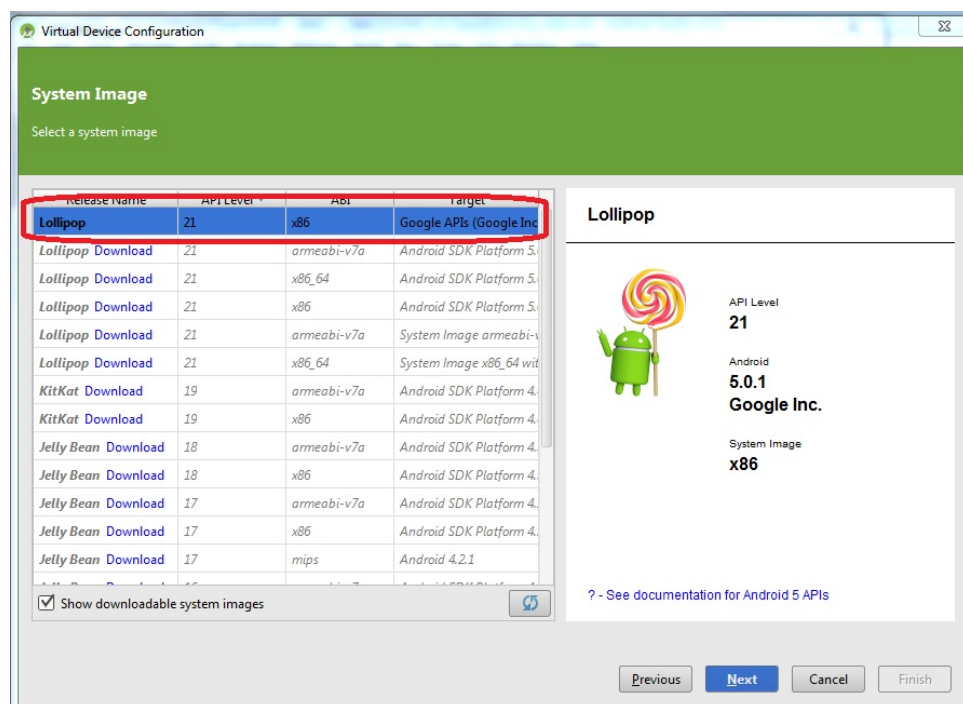


Presionamos el botón "Create Virtual Device".

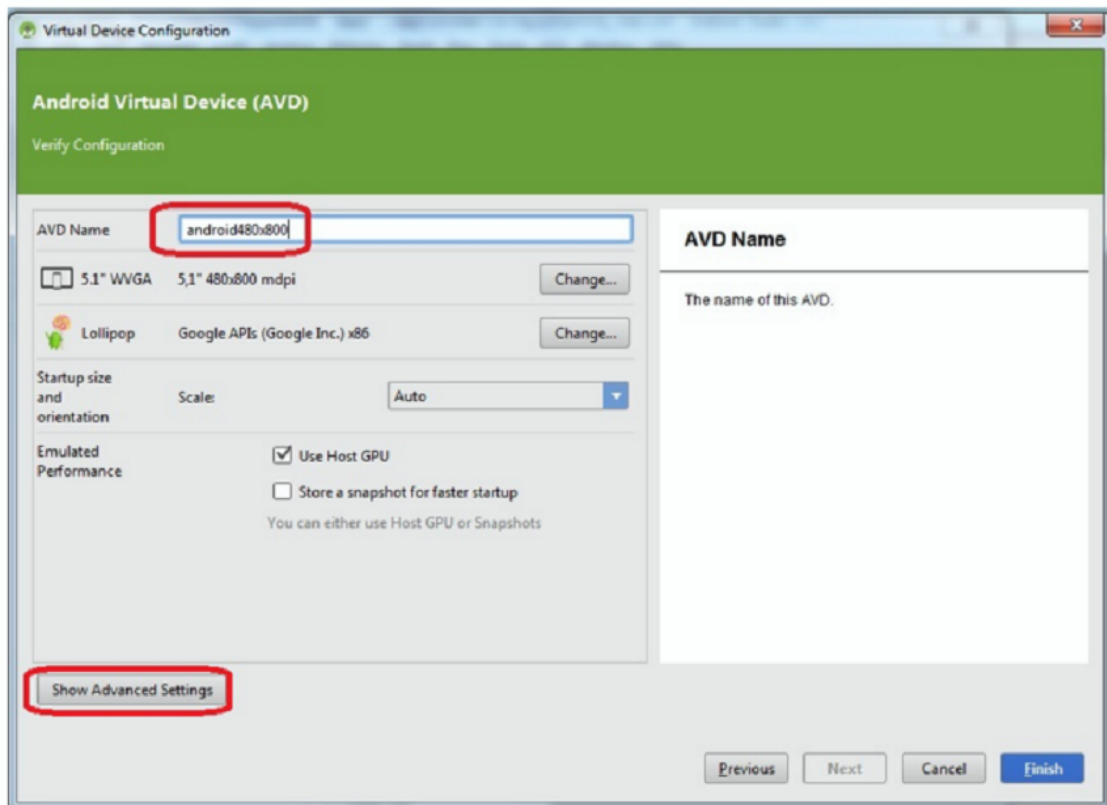
3. En este nuevo diálogo debemos seleccionar que crearemos un dispositivo virtual de tipo "Phone" y por ejemplo elegiremos uno genérico de 5.1 pulgadas:



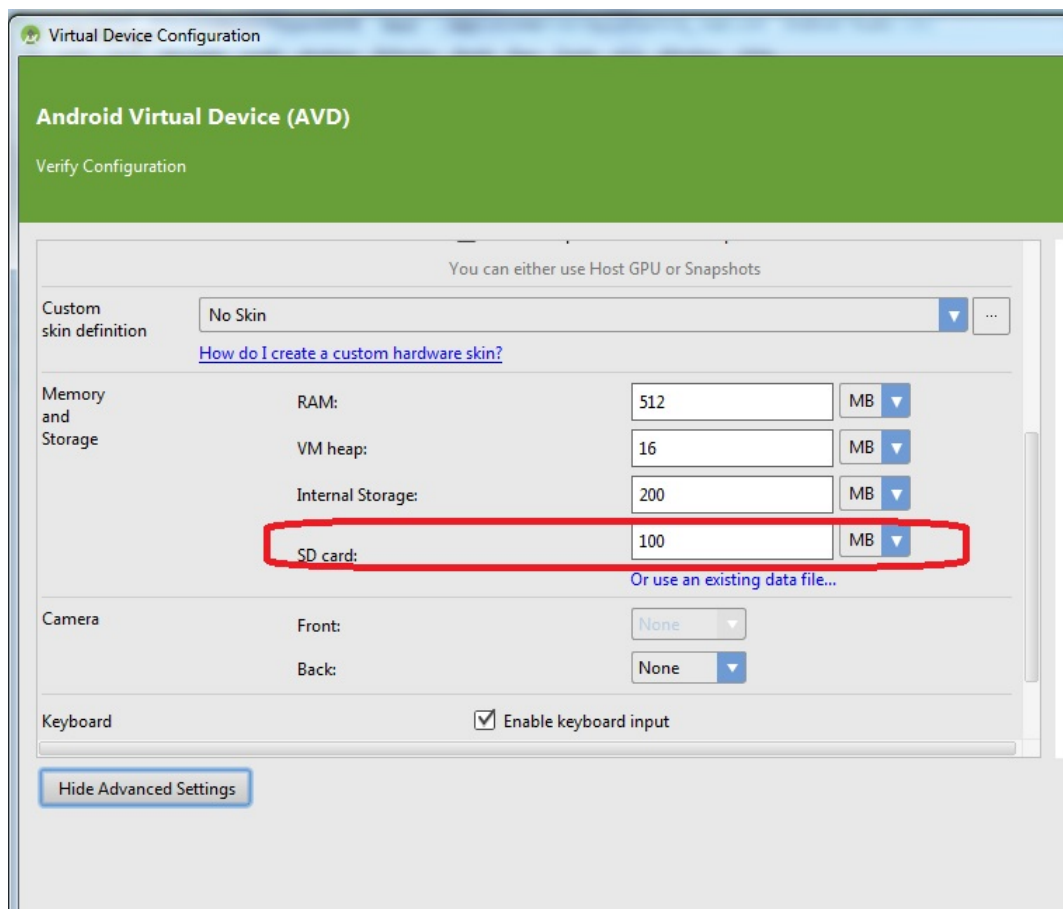
4. El siguiente diálogo seleccionamos la imagen de máquina virtual que disponemos:



5. En el nuevo diálogo asignamos un nombre al AVD, por ejemplo: Android480x800:

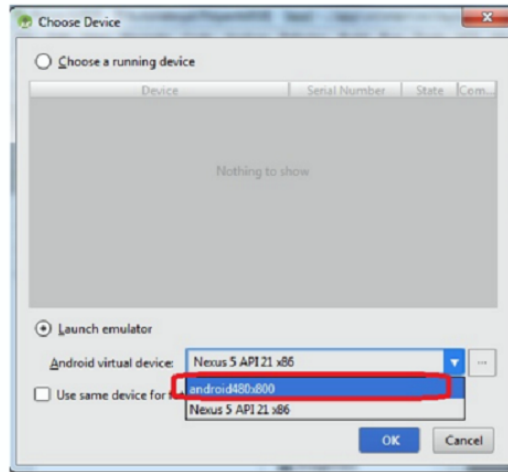


6. Presionamos el botón "Show Advanced Settings"
Controlamos que tenga configurado la propiedad de SD card con un valor de 100 o más:



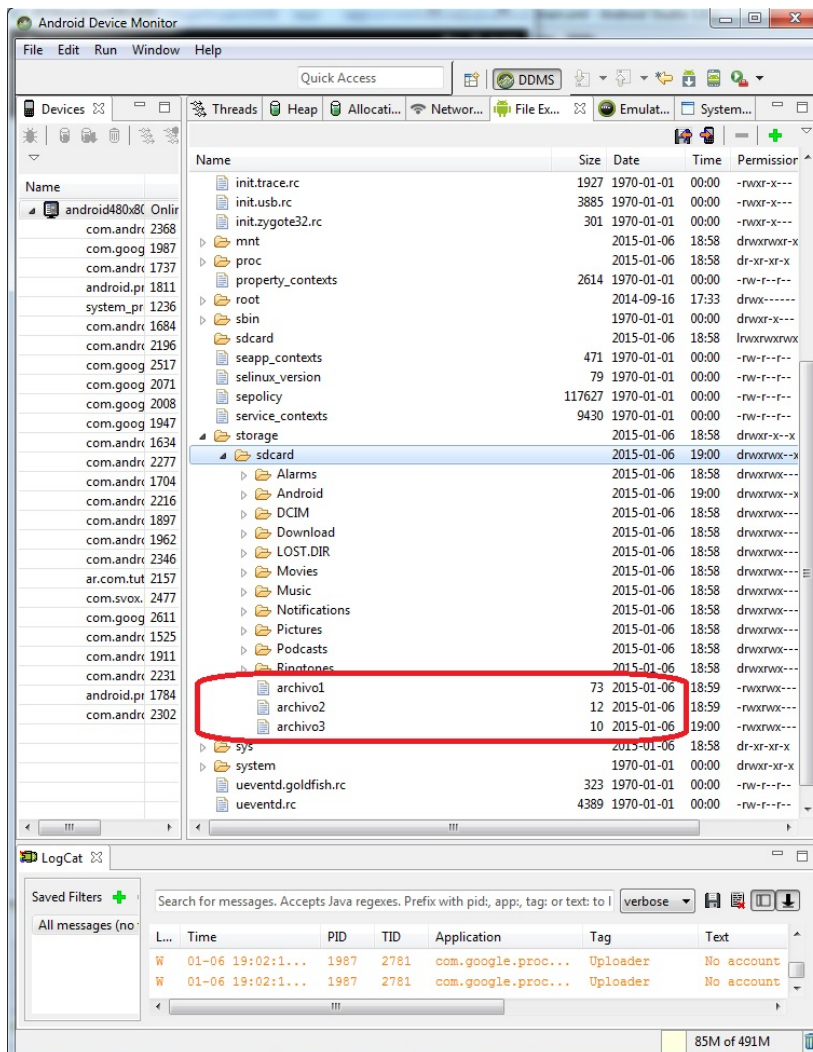
Finalmente ya tenemos configurado nuestra nueva máquina virtual que permite almacenar datos en una tarjeta sd.

Cuando ejecutemos nuevamente un proyecto tenemos que seleccionar esta nueva máquina virtual para que arranque:



Ubicación de los archivos desde el Android Device Monitor.

Si luego de ejecutar la aplicación y almacenar un par de archivos de texto queremos ver donde se almacenan en nuestro dispositivo virtual podemos inicializar el "Android Device Monitor" desde el menú del Android Studio Tools->Android->Android Device Monitor:



Como podemos ver se almacenan en una carpeta que no tiene nada que ver con la carpeta donde se almacenan los archivos de texto internos.

16 - Almacenamiento en una base de datos SQLite

Hemos visto hasta ahora dos modos de almacenar datos en forma permanente (archivos de texto y la clase `SharedPreferences`), ahora veremos otra herramienta nativa de Android para almacenar datos en una base de datos llamada SQLite.

SQLite es una base de datos Open Source, es muy popular en muchos dispositivos pequeños, como Android.

Las ventajas que presenta utilizar SQLite es que no requiere configuración, no tiene un servidor de base de datos ejecutándose en un proceso separado y es relativamente simple su empleo.

Problema:

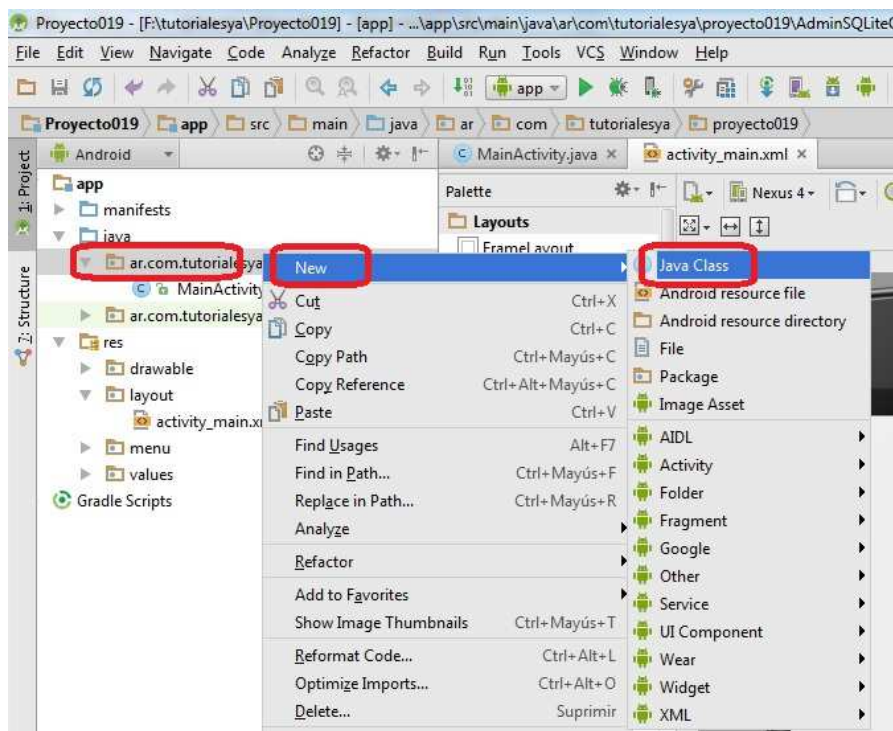
Confeccionar un programa que permita almacenar los datos de artículos. Crear la tabla artículos y definir los campos código, descripción del artículo y precio.

El programa debe permitir:

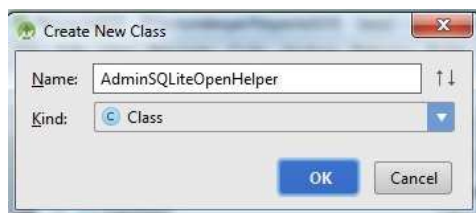
- 1 - Carga de artículo.
- 2 - Consulta por el código.
- 3 - Consulta por la descripción.
- 4 - Borrado de un artículo ingresando su código.
- 4 - Modificación de la descripción y el precio.

Lo primero que haremos es crear una clase que herede de `SQLiteOpenHelper`. Esta clase nos permite crear la base de datos y actualizar la estructura de tablas y datos iniciales.

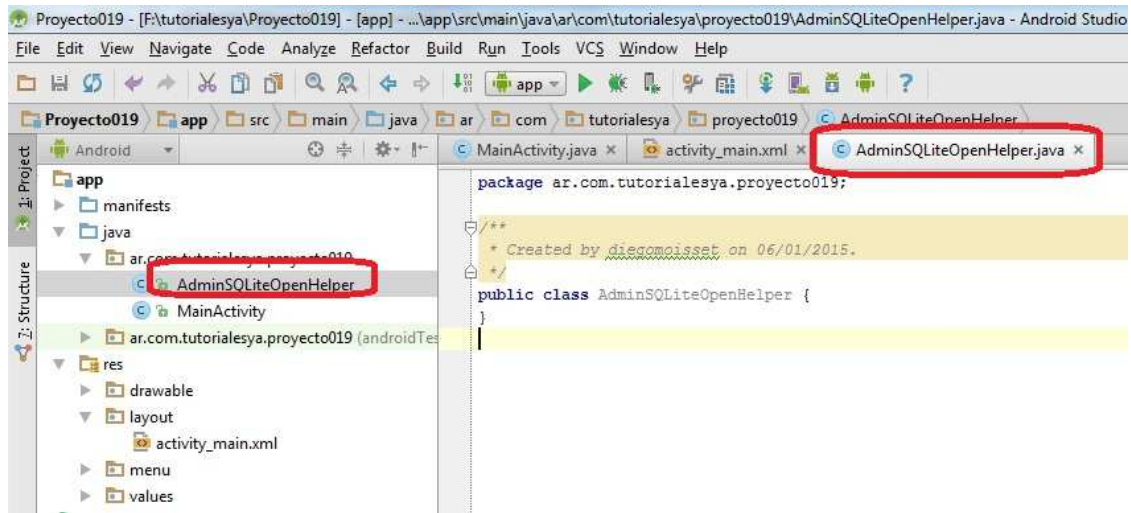
Para crear una nueva clase desde Android Studio procedemos a presionar el botón derecho del mouse sobre la carpeta que contienen todos los archivos java del proyecto y seleccionamos `New -> Java Class`:



En este diálogo ingresamos el nombre de nuestra clase, en nuestro ejemplo la llamaremos `AdminSQLiteOpenHelper`:



Ya tenemos un nuevo archivo en nuestro proyecto:



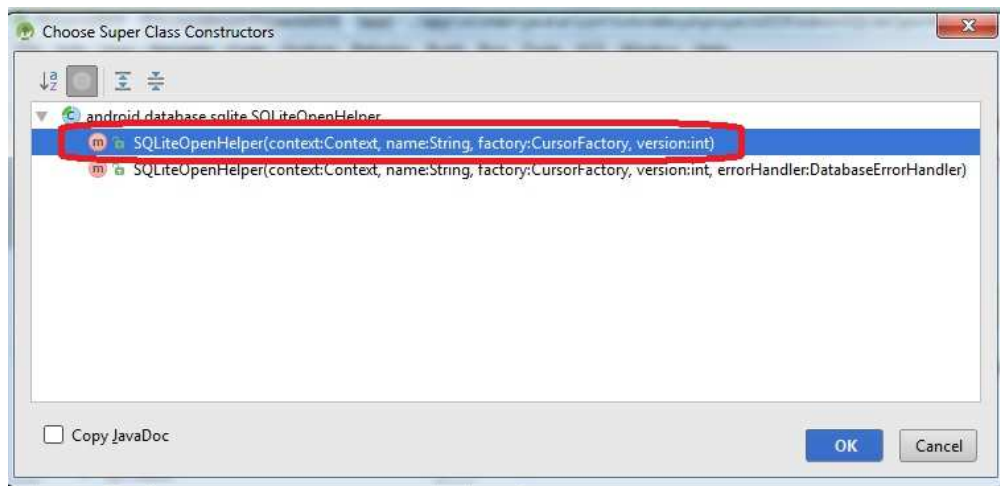
Ahora tenemos que codificar esta clase que tiene por objetivo administrar la base de datos que crearemos. Primero hacemos que nuestra clase herede de la clase SQLiteOpenHelper:

```
package ar.com.tutorialesya.proyecto019;

import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by diegomoisset on 06/01/2015.
 */
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper{
}
```

La clase SQLiteOpenHelper requiere que se implementen dos métodos obligatoriamente onCreate y onUpgrade, podemos hacer que el Android Studio nos ayude en su codificación así no tenemos que tipearlos nosotros: presionamos la teclas "ALT" y "ENTER" teniendo el cursor sobre el nombre de la clase "AdminSQLiteOpenHelper" y nos aparece un menú donde seleccionamos "Implement Methods" y luego un diálogo donde seleccionamos los métodos a implementar:



Ya tenemos la clase con los dos métodos:

```
package ar.com.tutorialesya.proyecto019;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by diegomoisset on 06/01/2015.
 */
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper{
    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```


}

Pero todavía nos marca un error ya que la clase padre SQLiteOpenHelper implementa constructores que tienen parámetros, entonces tenemos que definir el constructor en esta clase, para ello estando el cursor sobre el nombre de la clase "AdminSQLiteOpenHelper" presionamos nuevamente las teclas "ALT" y "Enter" y en el menú contextual que aparece seleccionamos "Create constructor matching super". Aparece un diálogo con todos los constructores que implementa la clase padre (seleccionamos el que tiene tres parámetros):



Ahora la clase no muestra ningún error y pasaremos a implementar la creación de la tabla artículos dentro del método onCreate:

```
package ar.com.tutorialesya.proyecto019;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

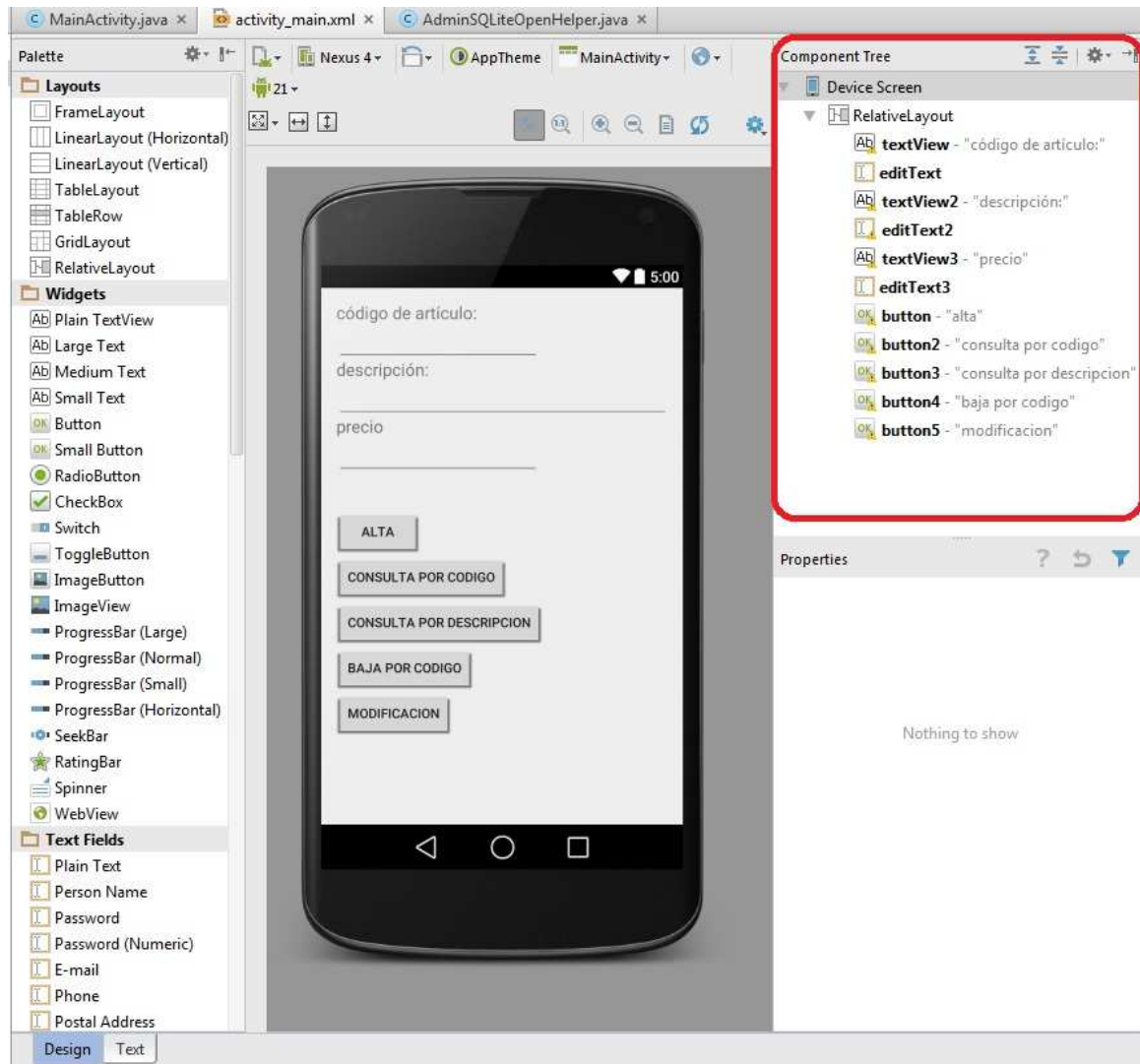
/**
 * Created by diegomoisset on 06/01/2015.
 */
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper{
    public AdminSQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table articulos(codigo int primary key,descripcion text,precio real)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

Hemos codificado en el método onCreate la creación de la tabla artículos con los campos codigo (que es entero y clave primaria), descripcion que es de tipo texto y precio es un valor real. El método onCreate se ejecutará una única vez (Eventualmente si uno quiere modificar la estructura de la tabla debemos hacerlo en el método onUpgrade).

Ahora implementemos la interfaz visual para resolver nuestro problema. Debemos crear en nuestro archivo activity_main.xml la siguiente interfaz:



Como vemos disponemos tres TextView, tres EditText y cinco Button.

El código fuente de nuestro Activity es el siguiente:

```
package ar.com.tutorialesya.proyecto019;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

    private EditText et1,et2,et3;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        et1=(EditText) findViewById(R.id.editText);
        et2=(EditText) findViewById(R.id.editText2);
        et3=(EditText) findViewById(R.id.editText3);
    }
}
```

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void alta(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String cod = et1.getText().toString();
    String descri = et2.getText().toString();
    String pre = et3.getText().toString();
    ContentValues registro = new ContentValues();
    registro.put("codigo", cod);
    registro.put("descripcion", descri);
    registro.put("precio", pre);
    bd.insert("articulos", null, registro);
    bd.close();
    et1.setText("");
    et2.setText("");
    et3.setText("");
    Toast.makeText(this, "Se cargaron los datos del artículo",
        Toast.LENGTH_SHORT).show();
}

public void consultaporcodigo(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String cod = et1.getText().toString();
    Cursor fila = bd.rawQuery(
        "select descripcion,precio from articulos where codigo=" + cod, null);
    if (fila.moveToFirst()) {
        et2.setText(fila.getString(0));
        et3.setText(fila.getString(1));
    } else
        Toast.makeText(this, "No existe un artículo con dicho código",
            Toast.LENGTH_SHORT).show();
    bd.close();
}

public void consultapordescripcion(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String descri = et2.getText().toString();
    Cursor fila = bd.rawQuery(
        "select codigo,precio from articulos where descripcion='" + descri +
        + "'", null);
    if (fila.moveToFirst()) {
        et1.setText(fila.getString(0));
        et3.setText(fila.getString(1));
    } else
        Toast.makeText(this, "No existe un artículo con dicha descripción",
            Toast.LENGTH_SHORT).show();
}

```

```

        bd.close();
    }

    public void bajaporcodigo(View v) {
        AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
            "administracion", null, 1);
        SQLiteDatabase bd = admin.getWritableDatabase();
        String cod= et1.getText().toString();
        int cant = bd.delete("articulos", "codigo=" + cod, null);
        bd.close();
        et1.setText("");
        et2.setText("");
        et3.setText("");
        if (cant == 1)
            Toast.makeText(this, "Se borró el artículo con dicho código",
                Toast.LENGTH_SHORT).show();
        else
            Toast.makeText(this, "No existe un artículo con dicho código",
                Toast.LENGTH_SHORT).show();
    }

    public void modificacion(View v) {
        AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
            "administracion", null, 1);
        SQLiteDatabase bd = admin.getWritableDatabase();
        String cod = et1.getText().toString();
        String descri = et2.getText().toString();
        String pre = et3.getText().toString();
        ContentValues registro = new ContentValues();
        registro.put("codigo", cod);
        registro.put("descripcion", descri);
        registro.put("precio", pre);
        int cant = bd.update("articulos", registro, "codigo=" + cod, null);
        bd.close();
        if (cant == 1)
            Toast.makeText(this, "se modificaron los datos", Toast.LENGTH_SHORT)
                .show();
        else
            Toast.makeText(this, "no existe un artículo con el código ingresado",
                Toast.LENGTH_SHORT).show();
    }
}

```

Recordar de inicializar la propiedad onClick de cada botón para enlazarlo con el método respectivo: "alta","consultaporcodigo","consultapordescripcion","bajaporcodigo" y "modificacion".

1 - Alta de datos.

Cuando se presiona el botón "ALTA" se ejecuta el método "alta" recordemos inicializar la propiedad "onClick" del botón desde la ventana de visualización del archivo XML.

Lo primero que hacemos en este método es crear un objeto de la clase que planteamos anteriormente y le pasamos al constructor this (referencia del Activity actual), "administracion" (es el nombre de la base de datos que crearemos en el caso que no exista) luego pasamos null y un uno indicando que es la primer versión de la base de datos (en caso que cambiemos la estructura o agreguemos tablas por ejemplo podemos pasar un dos en lugar de un uno para que se ejecute el método onUpgrade donde indicamos la nuestra estructura de la base de datos)

Luego de crear un objeto de la clase AdminSQLiteOpenHelper procedemos a crear un objeto de la clase SQLiteDatabase llamando al método getWritableDatabase (la base de datos se abre en modo lectura y escritura).

Creamos un objeto de la clase ContentValues y mediante el método put inicializamos todos los campos a cargar.

Seguidamente llamamos al método insert de la clase SQLiteDatabase pasando en el primer parámetro el nombre de la tabla, como segundo parámetro un null y por último el objeto de la clase ContentValues ya inicializado (este método es el que provoca que se inserte una nueva fila en la tabla articulos en la base de datos llamada administracion)

Borramos seguidamente los EditText y mostramos un mensaje para que conozca el operador que el alta de datos se efectuó en forma correcta:

```

public void alta(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();

```

```

String cod = et1.getText().toString();
String descri = et2.getText().toString();
String pre = et3.getText().toString();
ContentValues registro = new ContentValues();
registro.put("codigo", cod);
registro.put("descripcion", descri);
registro.put("precio", pre);
bd.insert("articulos", null, registro);
bd.close();
et1.setText("");
et2.setText("");
et3.setText("");
Toast.makeText(this, "Se cargaron los datos del artículo",
    Toast.LENGTH_SHORT).show();
}

```

2 - Consulta de artículo por código.

Cuando se presiona el botón "CONSULTA POR CODIGO" se ejecuta el método `consultaporcodigo`:

```

public void consultaporcodigo(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String cod = et1.getText().toString();
    Cursor fila = bd.rawQuery(
        "select descripcion,precio from articulos where codigo=" + cod, null);
    if (fila.moveToFirst()) {
        et2.setText(fila.getString(0));
        et3.setText(fila.getString(1));
    } else
        Toast.makeText(this, "No existe un artículo con dicho código",
            Toast.LENGTH_SHORT).show();
    bd.close();
}

```

En el método `consultaporcodigo` lo primero que hacemos es crear un objeto de la clase `AdminSQLiteOpenHelper` y obtener una referencia de la base de datos llamando al método `getWritableDatabase`.

Seguidamente definimos una variable de la clase `Cursor` y la inicializamos con el valor devuelto por el método llamado `rawQuery`.

La clase `Cursos` almacena en este caso una fila o cero filas (una en caso que hayamos ingresado un código existente en la tabla `articulos`), llamamos al método `moveToFirst()` de la clase `Cursos` y retorna `true` en caso de existir un artículo con el código ingresado, en caso contrario retorna `cero`.

Para recuperar los datos propiamente dichos que queremos consultar llamamos al método `getString` y le pasamos la posición del campo a recuperar (comienza a numerarse en `cero`, en este ejemplo la columna `cero` representa el campo `descripcion` y la columna `1` representa el campo `precio`)

3 - Consulta de artículo por descripción.

Cuando se presiona el botón "CONSULTA POR DESCRIPCION" se ejecuta el método `consultapordescripcion`:

```

public void consultapordescripcion(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String descri = et2.getText().toString();
    Cursor fila = bd.rawQuery(
        "select codigo,precio from articulos where descripcion='" + descri + "'", null);
    if (fila.moveToFirst()) {
        et1.setText(fila.getString(0));
        et3.setText(fila.getString(1));
    } else
        Toast.makeText(this, "No existe un artículo con dicha descripción",
            Toast.LENGTH_SHORT).show();
    bd.close();
}

```

En el método `consultapordescripcion` lo primero que hacemos es crear un objeto de la clase `AdminSQLiteOpenHelper` y obtener una referencia de la base de datos llamando al método `getWritableDatabase`.

Seguidamente definimos una variable de la clase `Cursor` y la inicializamos con el valor devuelto por el método llamado `rawQuery`.

Es importante notar en el `where` de la cláusula SQL hemos dispuesto comillas simples entre el contenido de la variable `descri`:

```

"select codigo,precio from articulos where descripcion='" + descri + "'", null);

```

Esto es obligatorio para los campos de tipo text (en este caso descripcion es de tipo text)

4 - Baja o borrado de datos.

Para borrar uno o más registros la clase SQLiteDatabase tiene un método que le pasamos en el primer parámetro el nombre de la tabla y en el segundo la condición que debe cumplirse para que se borre la fila de la tabla. El método delete retorna un entero que indica la cantidad de registros borrados:

```
public void bajaporcodigo(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String cod= et1.getText().toString();
    int cant = bd.delete("articulos", "codigo=" + cod, null);
    bd.close();
    et1.setText("");
    et2.setText("");
    et3.setText("");
    if (cant == 1)
        Toast.makeText(this, "Se borró el artículo con dicho código",
            Toast.LENGTH_SHORT).show();
    else
        Toast.makeText(this, "No existe un artículo con dicho código",
            Toast.LENGTH_SHORT).show();
}
```

5 - Modificación de datos.

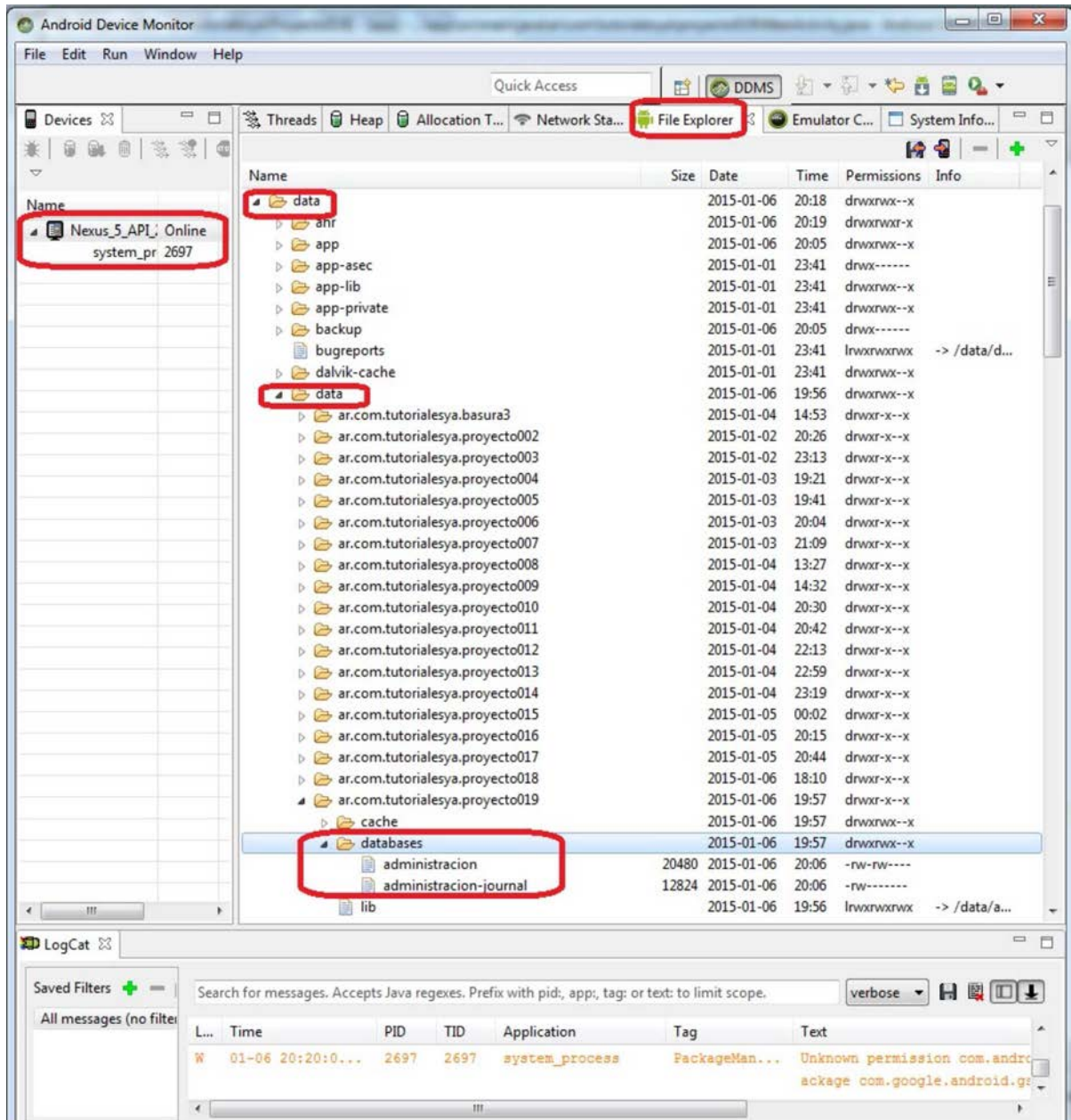
En la modificación de datos debemos crear un objeto de la clase ContentValues y mediante el método put almacenar los valores para cada campo que será modificado. Luego se llama al método update de la clase SQLiteDatabase pasando el nombre de la tabla, el objeto de la clase ContentValues y la condición del where (el cuanto parámetro en este ejemplo no se lo emplea)

```
public void modificacion(View v) {
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    String cod = et1.getText().toString();
    String descri = et2.getText().toString();
    String pre = et3.getText().toString();
    ContentValues registro = new ContentValues();
    registro.put("codigo", cod);
    registro.put("descripcion", descri);
    registro.put("precio", pre);
    int cant = bd.update("articulos", registro, "codigo=" + cod, null);
    bd.close();
    if (cant == 1)
        Toast.makeText(this, "se modificaron los datos", Toast.LENGTH_SHORT)
            .show();
    else
        Toast.makeText(this, "no existe un artículo con el código ingresado",
            Toast.LENGTH_SHORT).show();
}
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto019.zip](#)

Ubicación de la base de datos en el dispositivo

Para conocer donde se almacena físicamente la base de datos en el dispositivo podemos abrir el programa "Android Device Monitor" desde el menú del Android Studio Tools->Android->Android Device Monitor:



[Retornar](#)

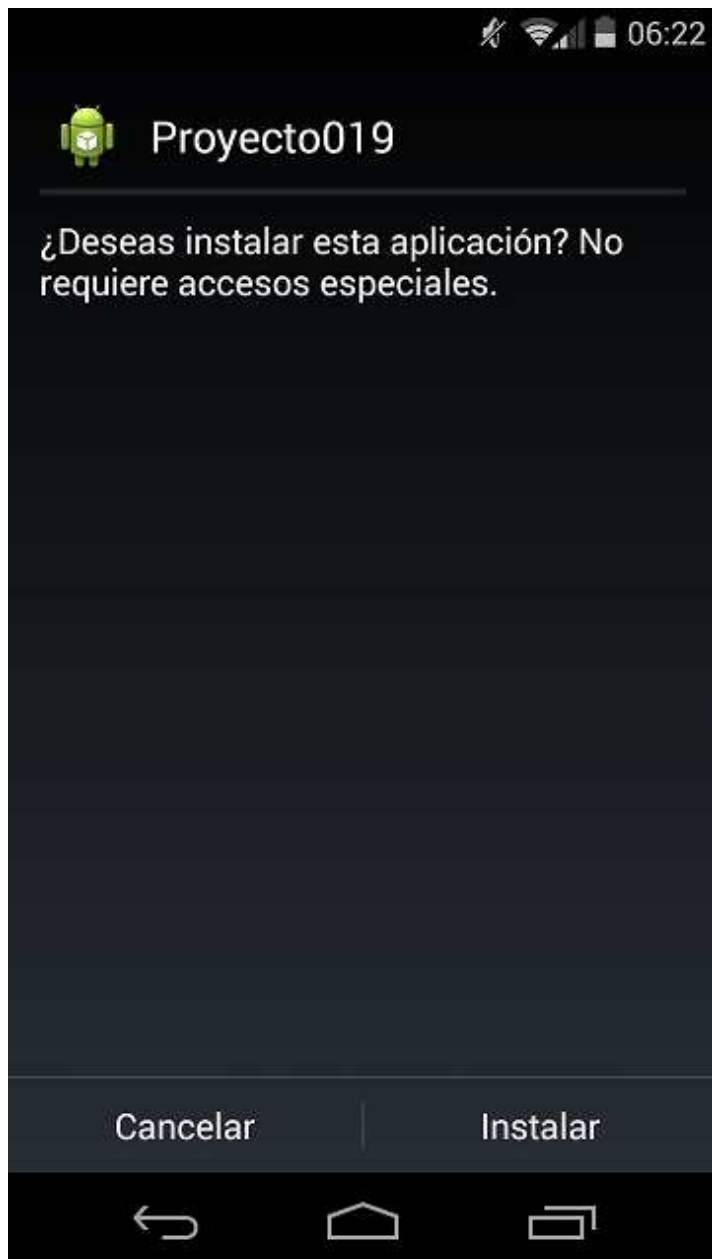
17 - Instalación del programa Android en un dispositivo real

Normalmente uno cuando desarrolla aplicaciones en Android hace toda la programación, depuración y pruebas en un dispositivo virtual en la pc.

Ahora vamos a ver los pasos para probar la aplicación en un dispositivo Android real.

La primera forma que veremos de probar nuestra aplicación es copiar el archivo con extensión APK a nuestro dispositivo.

1. Primero en el ícono de configuración de nuestro teléfono o tablet android seleccionamos la opción "Aplicaciones" -> y marcamos la opción "Origen desconocido (Permitir la instalación de aplicaciones no pertenecientes al mercado)"
2. Desde nuestro equipo de escritorio enviamos un mail a nuestro celular adjuntando el archivo con extensión apk que se encuentra en el directorio `app/build/outputs/apk` de nuestro proyecto.
3. Abramos el mail desde nuestro celular y seleccionemos el archivo adjunto. Confirmamos que queremos instalarlo.



Ya podemos probar nuestra aplicación:

Proyecto019

código de artículo:
|

descripción:
|

precio
|

alta

consulta por codigo

consulta por descripcion

baja por codigo

modificacion

[Retornar](#)

18 - Layout (LinearLayout)

Android organiza las componentes visuales (Button, EditText, TextView etc.) en pantalla mediante contenedores llamados Layout.

Hasta ahora no nos preocupada como organizar una pantalla, sino nos hemos centrado en la funcionalidad de cada programa que implementamos.

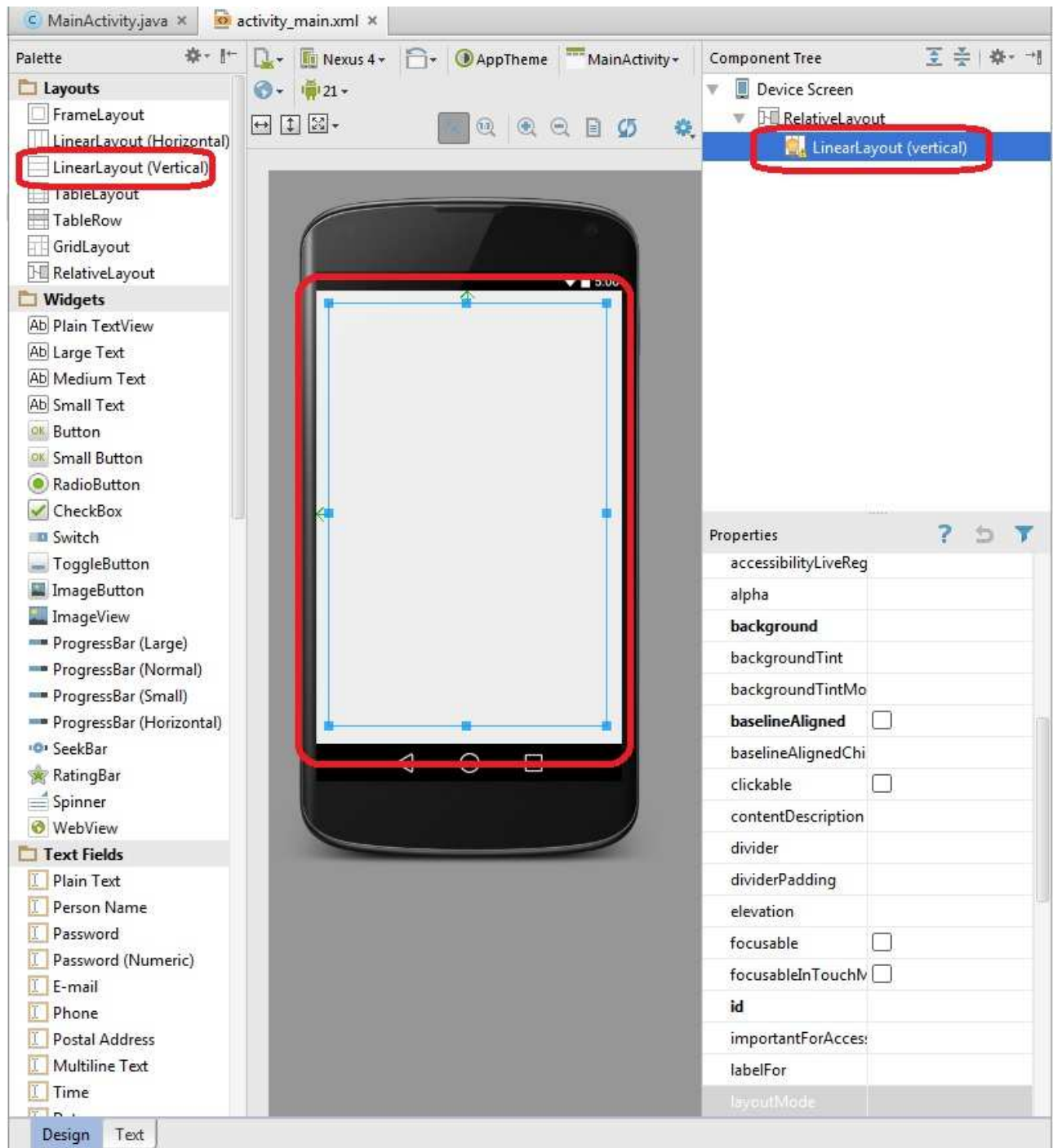
Ahora comenzaremos a preocuparnos como organizar y disponer las componentes dentro de la pantalla.

- LinearLayout.

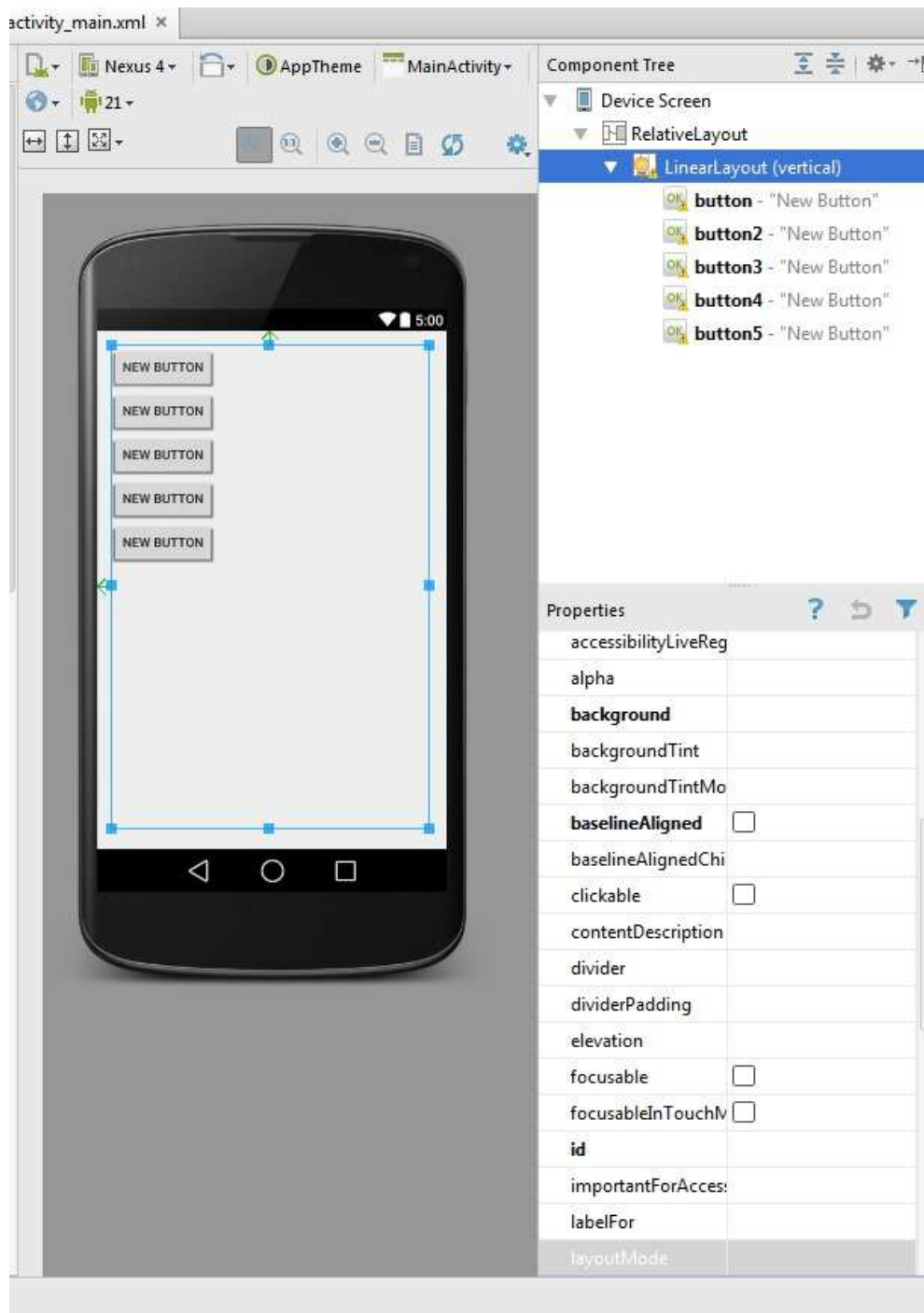
LinearLayout es uno de los diseños más simples y más empleado. Simplemente establece los componentes visuales uno junto al otro, ya sea horizontal o verticalmente.

Creemos un proyecto llamado Proyecto020, borremos el TextView que agrega por defecto el Android Studio.

De la "Palette" de componentes de Android Studio arrastraremos de la pestaña "Layouts" el objeto de la clase "Linear Layout (Vertical)":

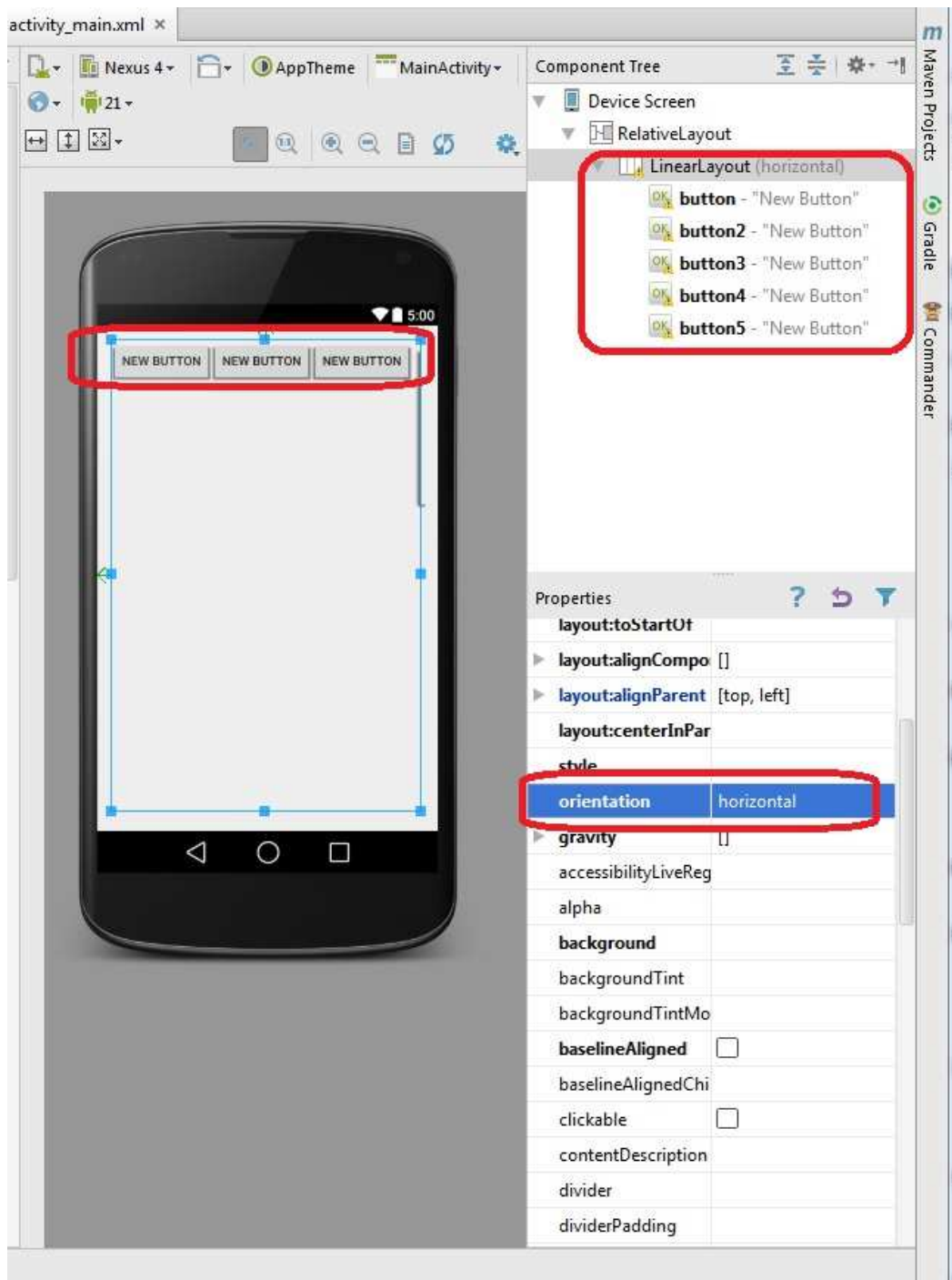


Ahora podemos ver que sucede cuando disponemos cinco botones dentro del contenedor LinearLayout (Todos los botones se disponen obligatoriamente uno debajo del otro y no hay posibilidad de ubicarlos con el mouse):



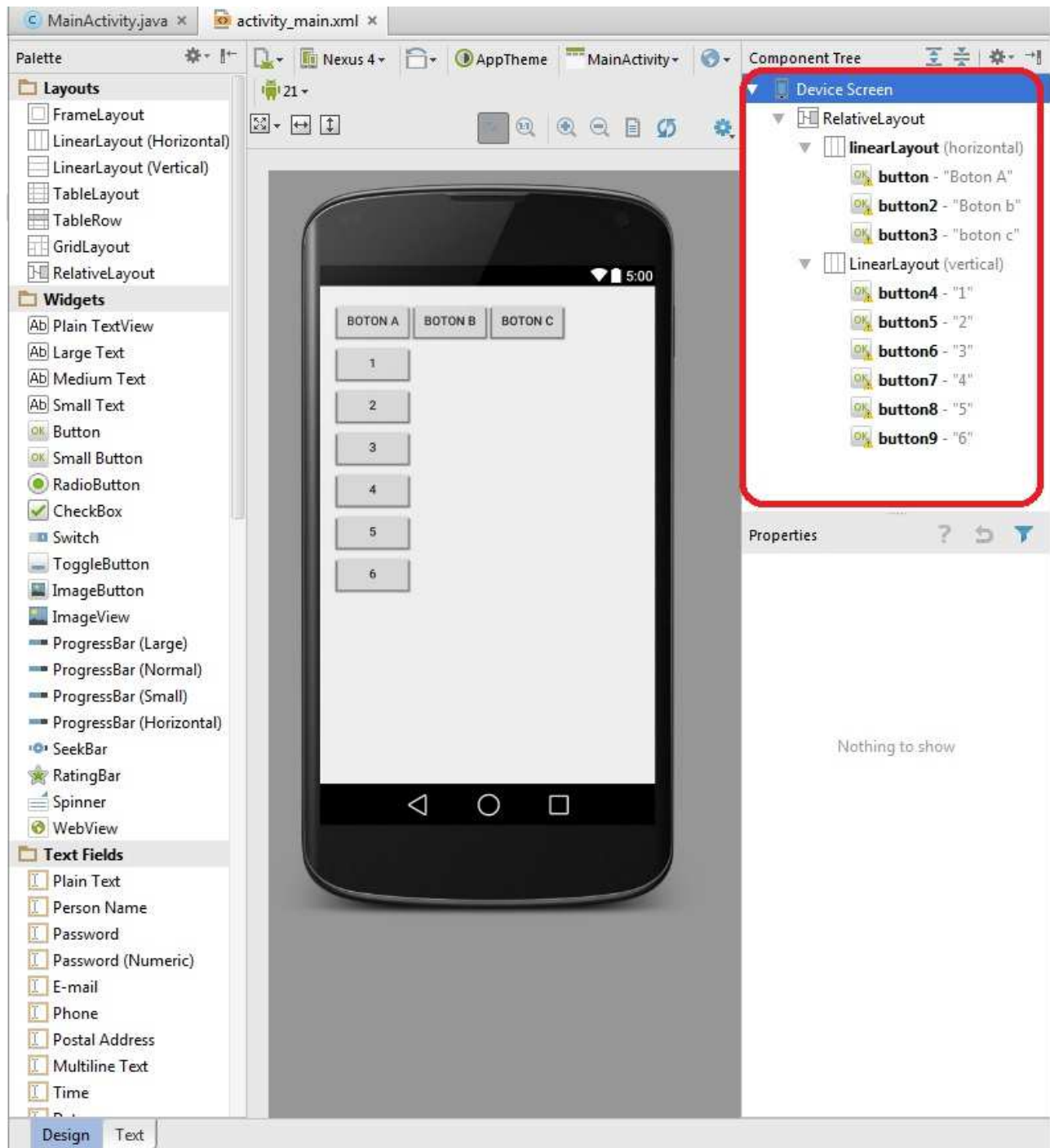
El primer cambio que podemos hacer a esta tipo de Layout es cambiar la propiedad "orientation" por el valor horizontal (esto hace que los botones se dispongan uno al lado del otro y no hay posibilidad de disponer controles uno debajo de otro)

Hay que tener en cuenta que si los controles no entran en pantalla los mismos no se ven y si son botones como en este caso es imposible hacer clic sobre los mismos:

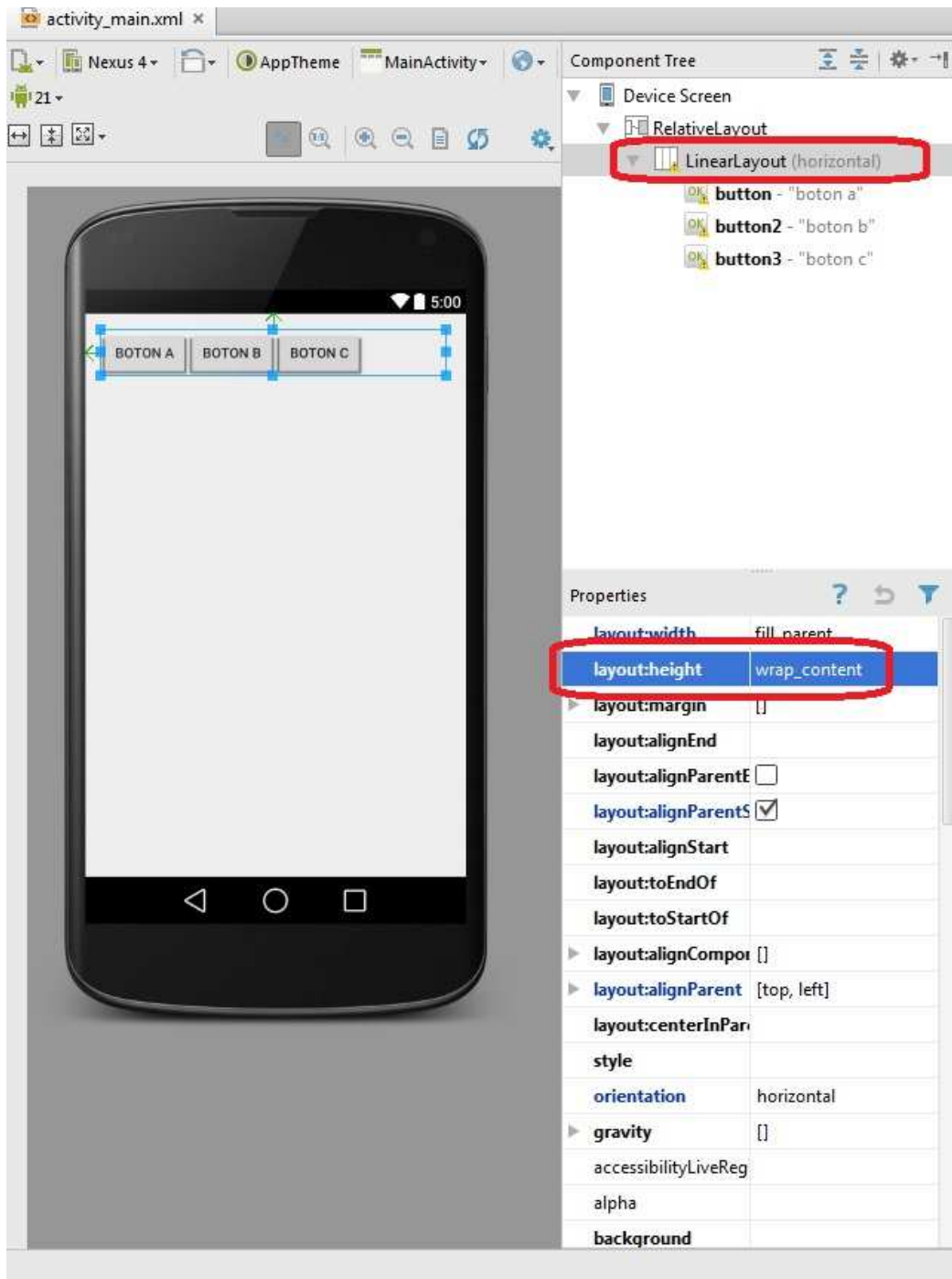


Tengamos en cuenta que en la "Palette" de Android Studio aparecen dos opciones para disponer "Linear Layout(Horizontal)" o "Linear Layout(Vertical)" la única diferencia es que se inicializa la propiedad "orientation" con el valor: horizontal o vertical.

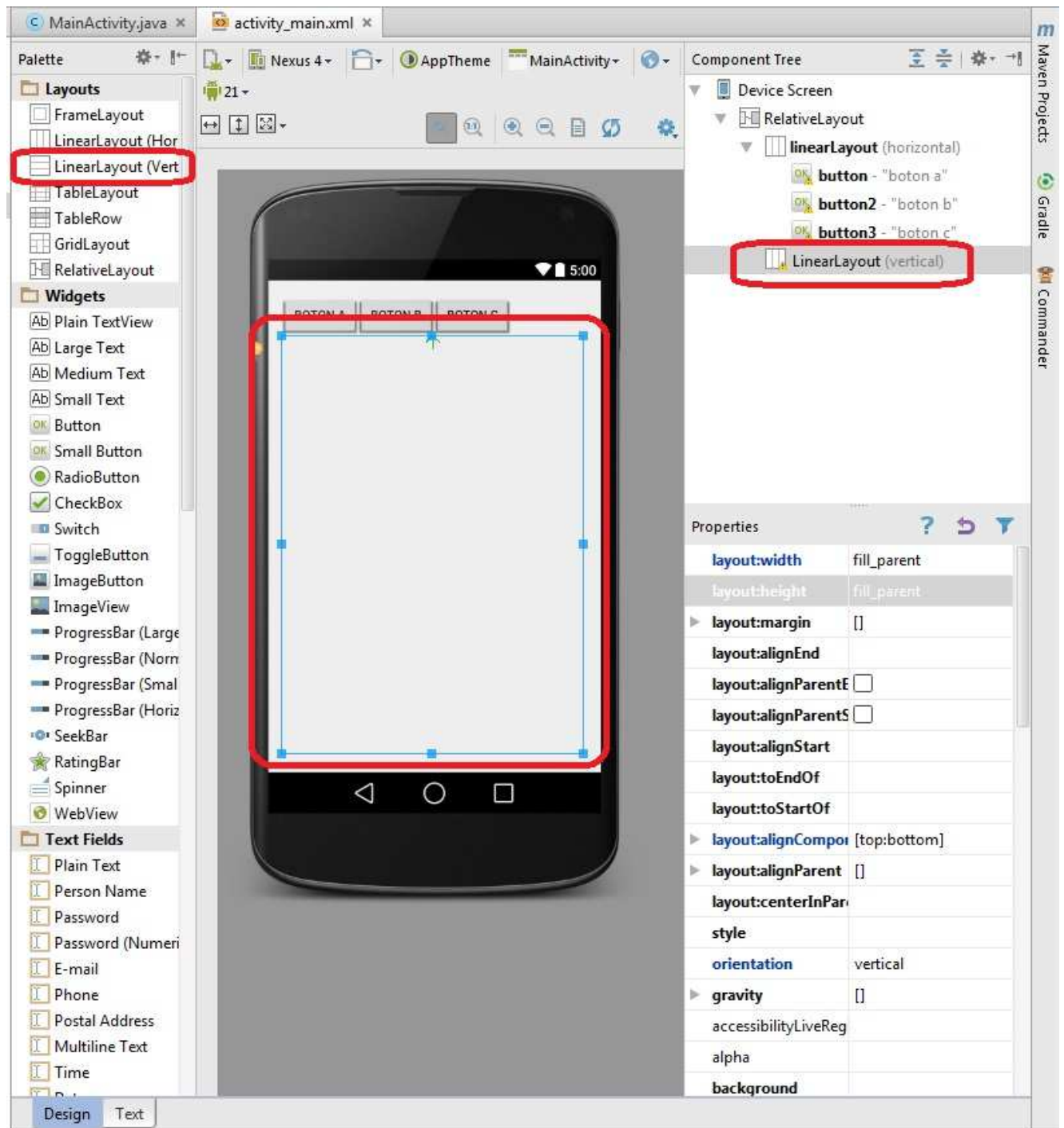
Podemos disponer más de una componente de tipo LinearLayout para implementar nuestra interfaz visual. Veamos los objetos y propiedades a configurar si queremos la siguiente interfaz:



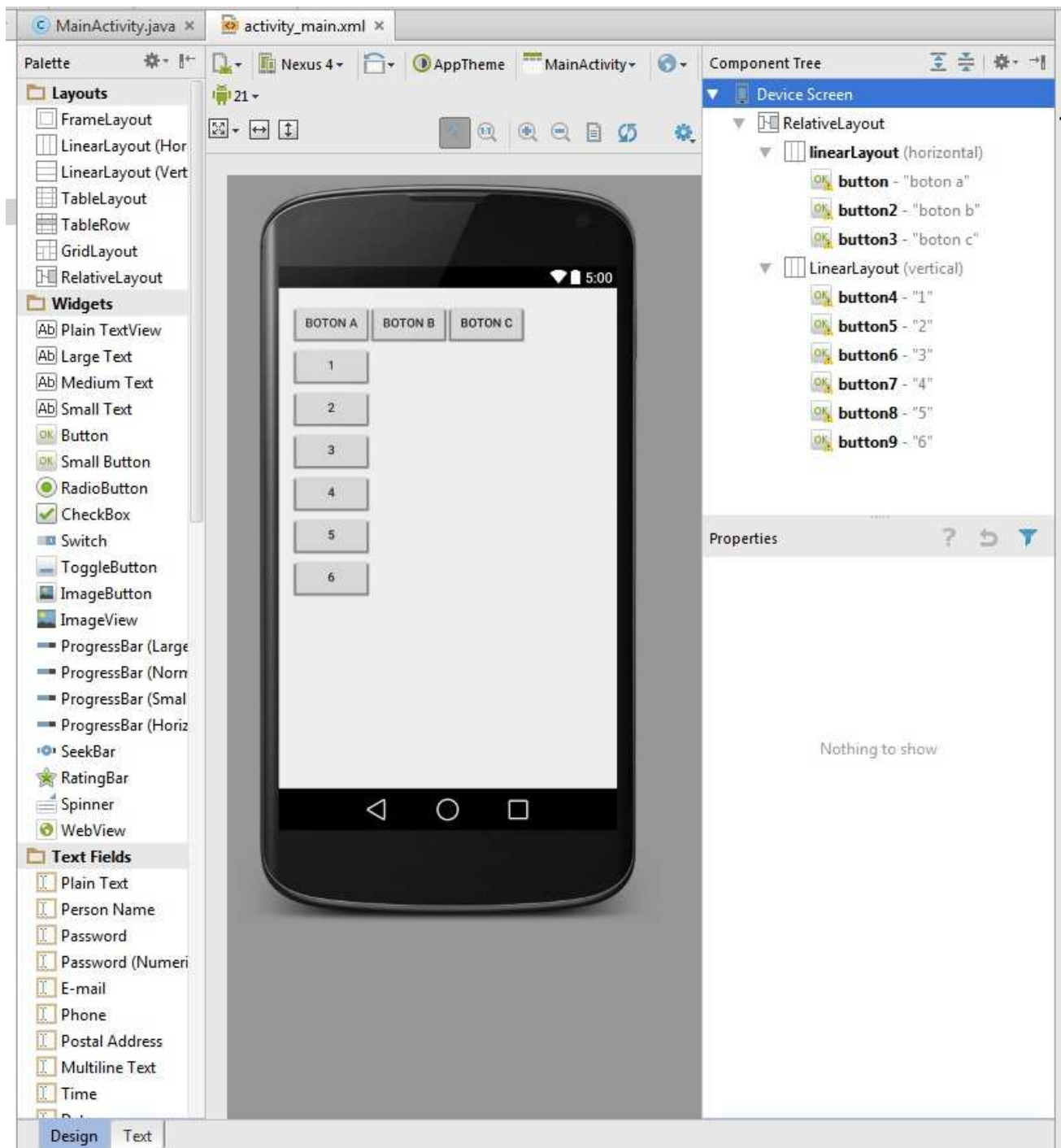
Como vemos hemos dispuesto dos LinearLayout el primero con orientación horizontal con los botones con las etiquetas "BOTON A", "BOTON B" y "BOTON C". Luego de disponer los botones podemos observar que el LinearLayout ocupa todo el alto de la pantalla (para que ocupe solo el espacio de la altura de los botones debemos seleccionar el Linear Layout y cambiar la propiedad "layout:height" por el valor: "wrap_content" asegurarse que está seleccionado el LinearLayout y no alguno de los botones):



Arrastremos el segundo LinearLayout vertical dentro del RelativeLayout (podemos arrastrarlo sin problemas a la interfaz visual o arrastrarlo a la ventana "Component Tree", pero lo importante que quede al mismo nivel que el primer Layout y no dentro del otro LinearLayout y lo redimensionamos para que quede debajo del otro en pantalla):



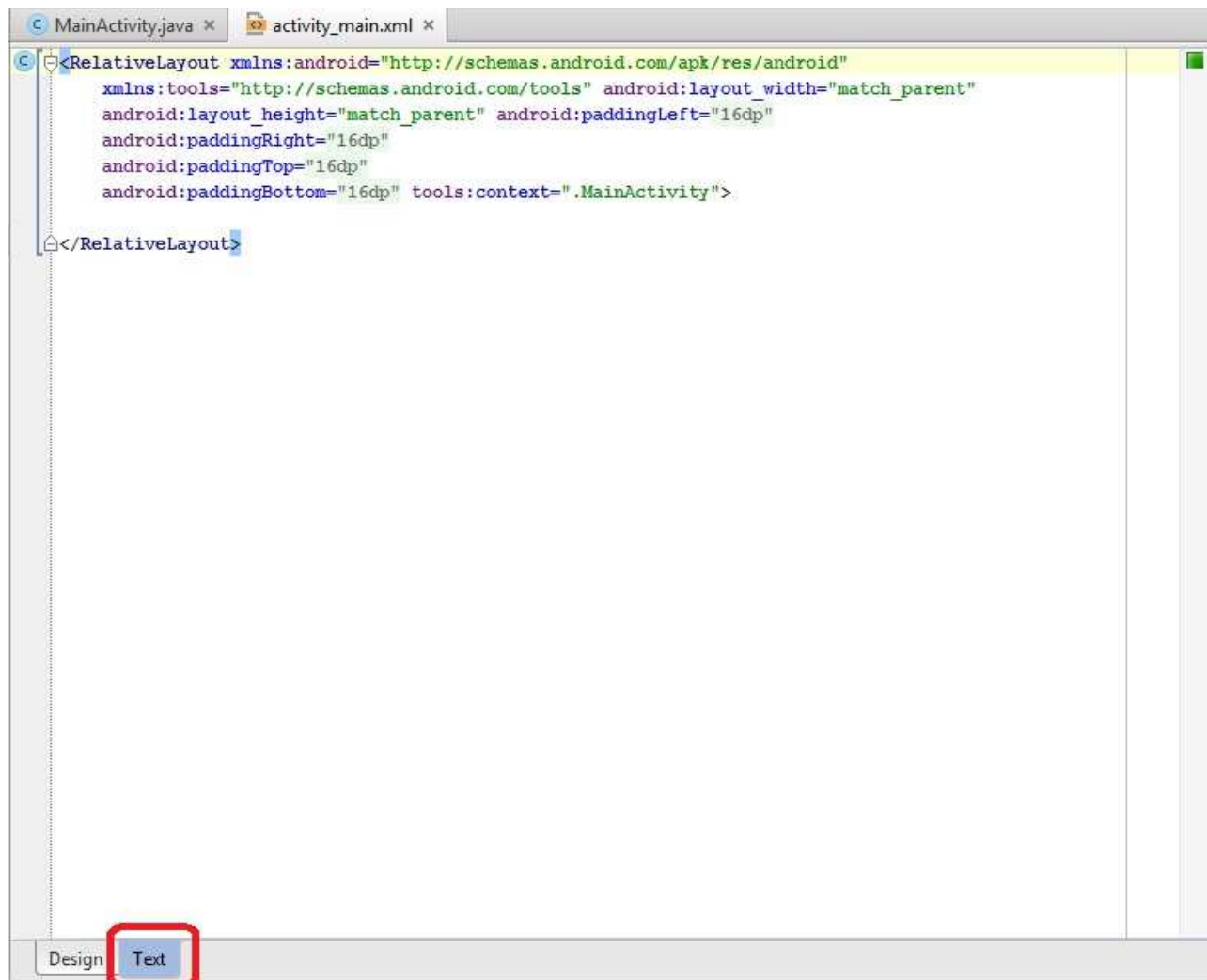
Ahora podemos arrastrar un conjunto de botones dentro del segundo Layout:



Eliminando el RelativeLayout de la raíz de la aplicación.

Si no necesitamos que la raíz de la interfaz defina un RelativeLayout podemos disponer en su lugar por ejemplo un LinearLayout. Modificaremos el programa hecho hasta ahora de tal forma que la interfaz sea solo un LinearLayout con orientación vertical y con 5 botones en su interior.

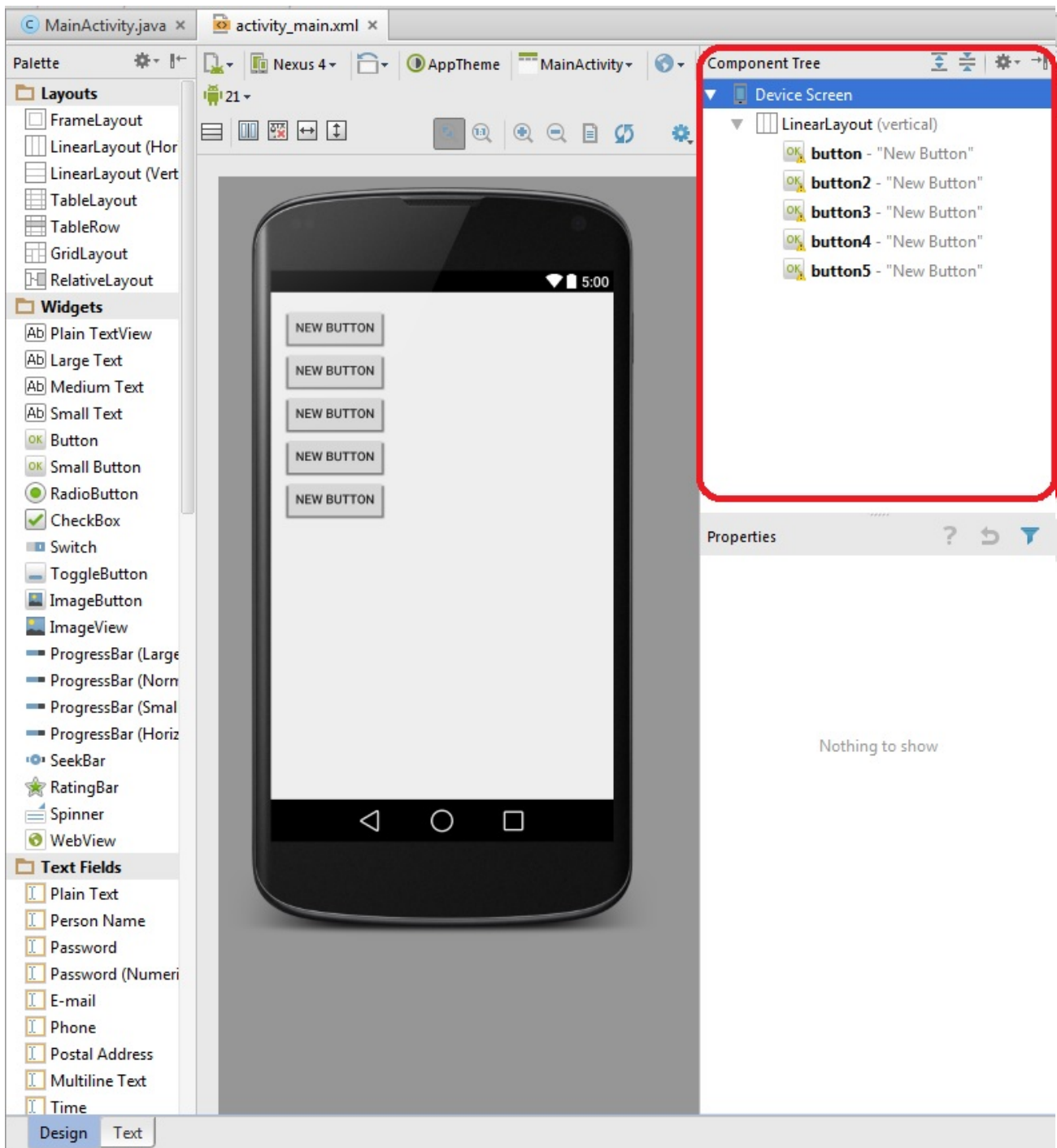
No lo podemos hacer en forma visual con Android Studio, debemos modificar manualmente el código del archivo XML, cambiamos la vista "Design" por "Text":



Procedemos a modificar el nombre del Layout y definir e inicializar la propiedad "orientation" (Reemplazamos el nombre RelativeLayout por LinearLayout y agregamos la propiedad android:orientation="vertical"):



Ahora podemos disponer los botones en la interfaz y aparecerán uno debajo del otro similar a como empezamos pero será mas eficiente ya que no hay un RelativeLayout en la raíz de la interfaz:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto020.zip](#)

[Retornar](#)

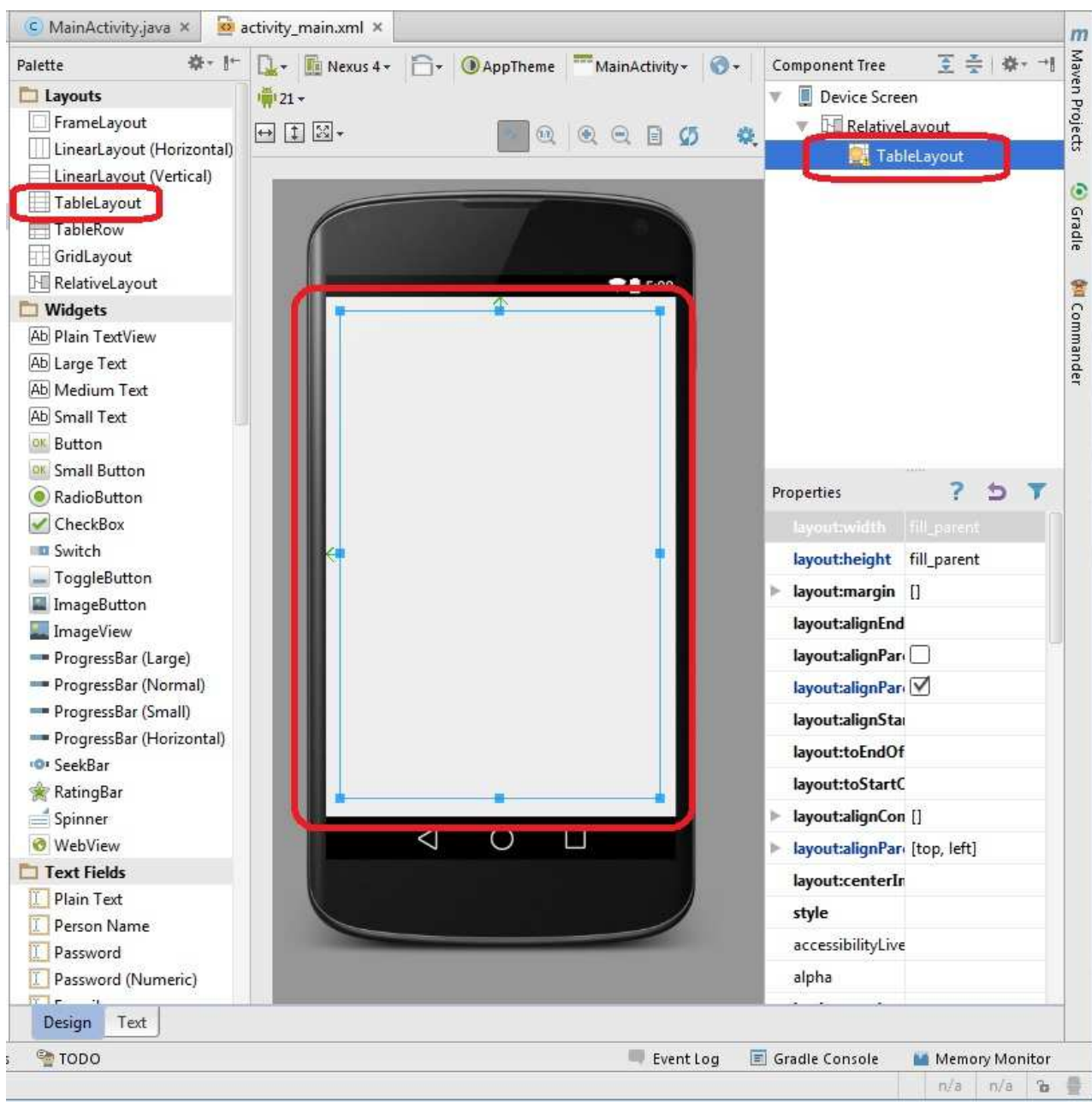
19 - Layout (TableLayout)

El Layout de tipo TableLayout agrupa componentes en filas y columnas. Un TableLayout contiene un conjunto de componentes de tipo TableRow que es el que agrupa componentes visuales por cada fila (cada fila puede tener distinta cantidad de componentes visuales)

Problema

Disponer 9 botones en forma de un tablero de TaTeTi. Utilizar un TableLayout, tres TableRow y nueve botones.

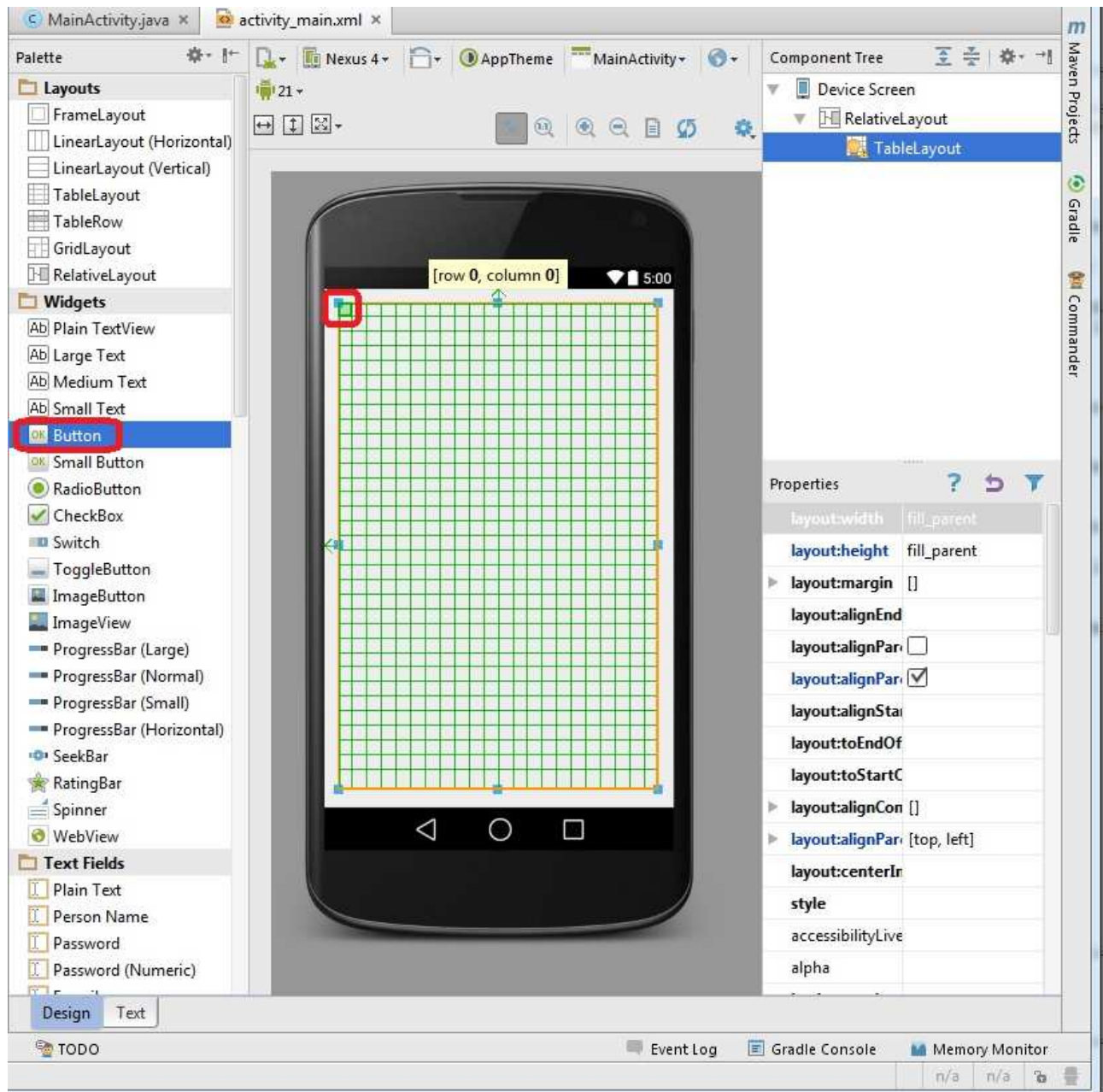
Primero creemos el Proyecto021 y vamos a la pestaña Layout, identifiquemos la componente "TableLayout" y la arrastramos al interior de la interfaz visual:



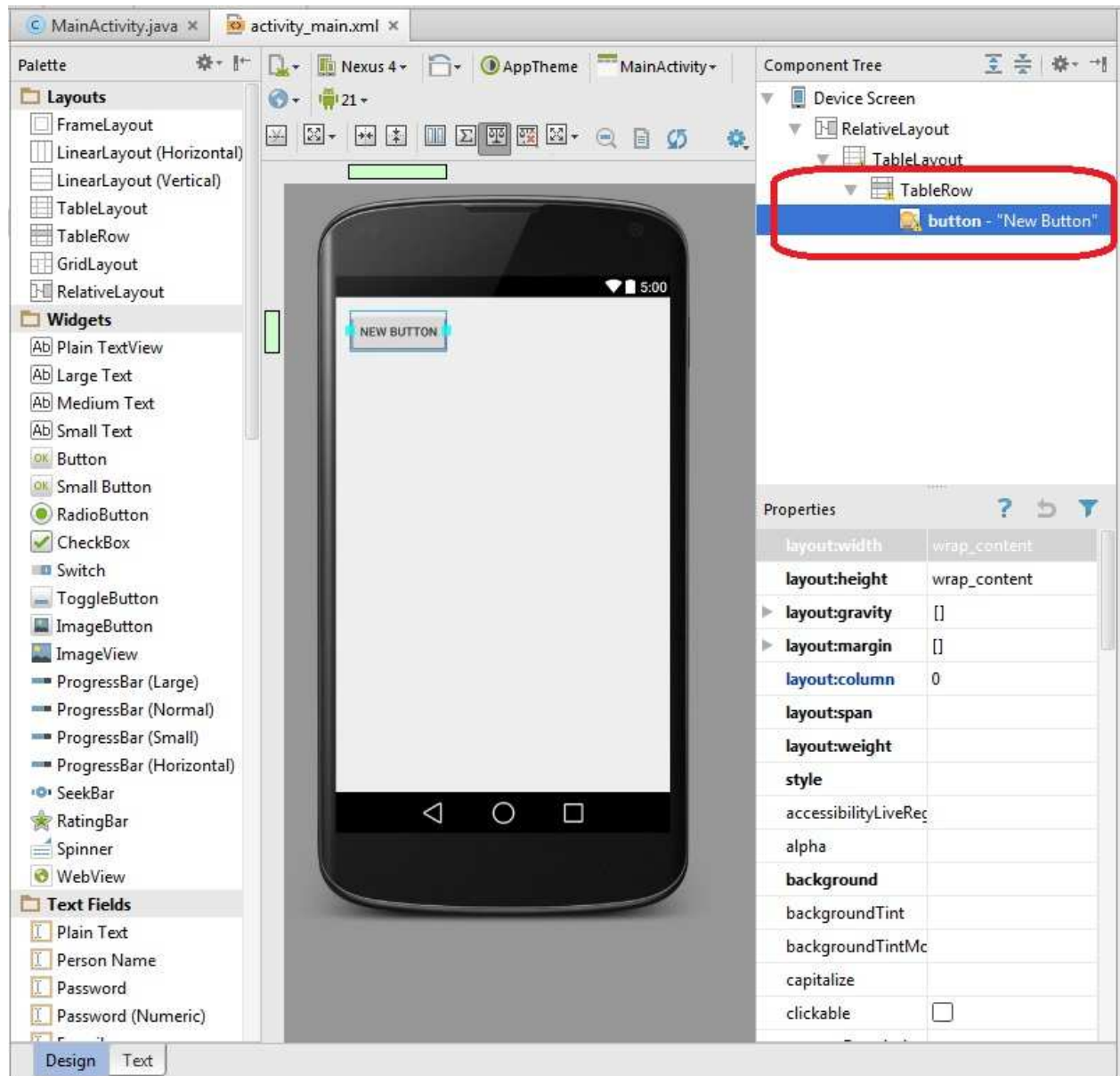
Luego de esto tenemos toda la pantalla cubierta con una componente de tipo "TableLayout", ahora tenemos que empezar a disponer cada uno de los botones e ir creando las filas.

El entorno del Android Studio nos facilita la creación de las filas del TableLayout, debemos arrastrar las

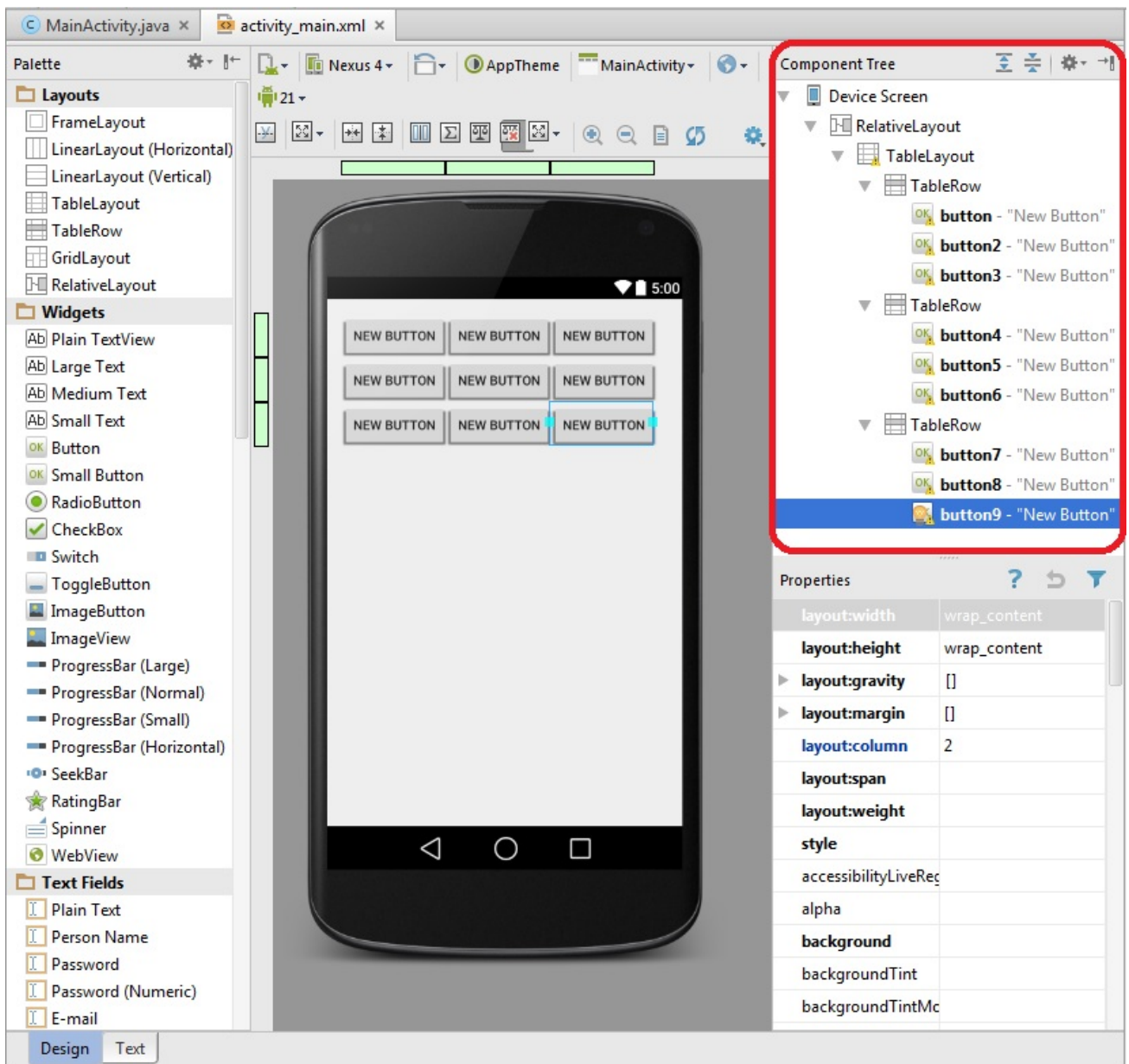
componentes que tendrá cada fila y automáticamente irá creando cada "TableRow". Probemos de arrastrar el primer botón dentro del "TableLayout", lo ubicaremos en la primera celda de la primera fila:



Una vez que lo dejamos caer dentro de la celda de la primera fila y columna que nos muestra el Android Studio podemos ver que se han creado un objeto de la clase "TableRow" y dentro de éste está el botón:

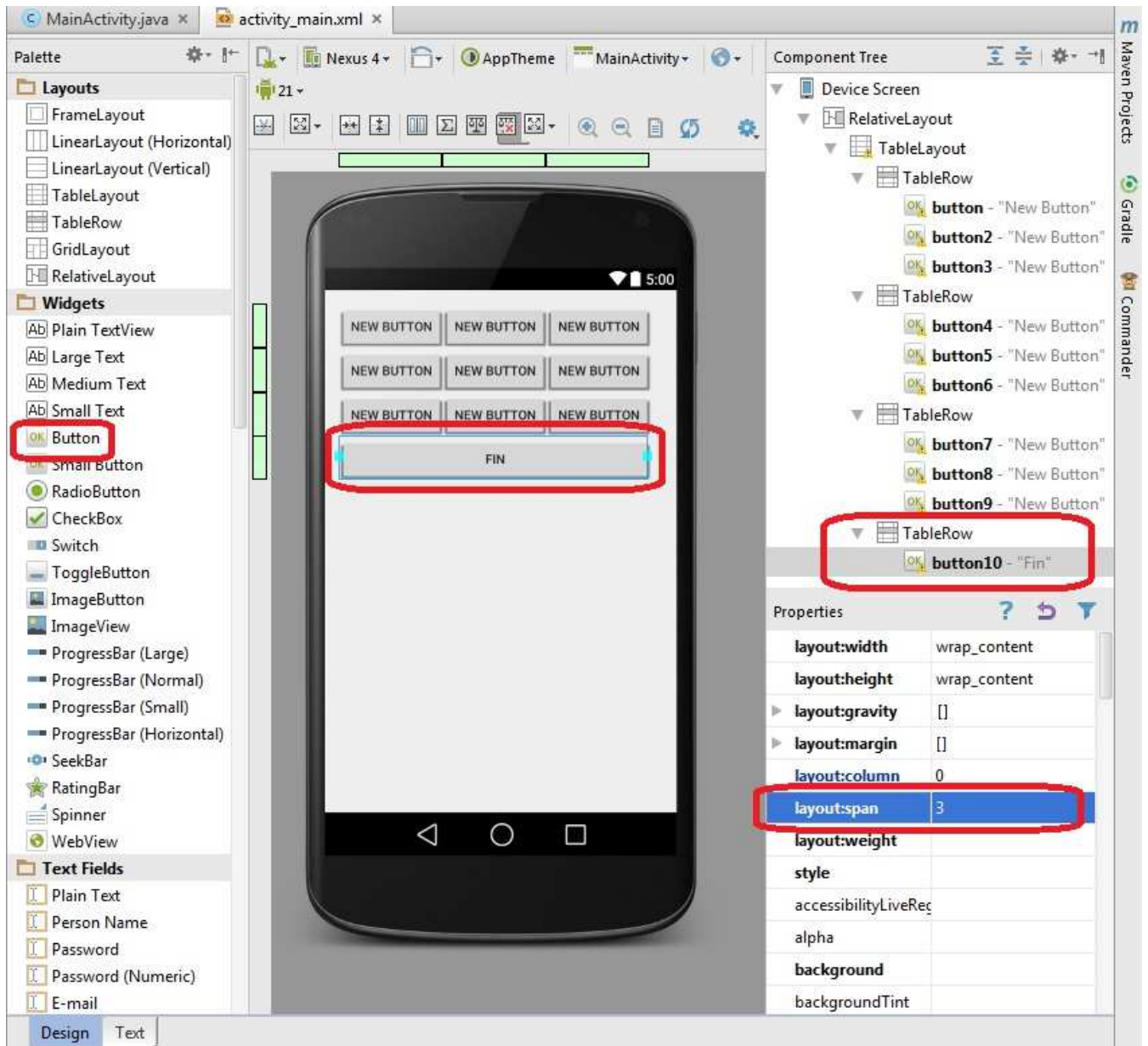


Lo mismo hacemos para agregar los dos botones que faltan en la primer fila, seguidamente agregamos los tres botones de la segunda fila y los tres botones de la tercer fila:



Otra posibilidad que nos da este Layout es que un control se expanda más de una celda.

Disponer un botón en la cuarta fila del TableLayout y luego con el mouse expandirlo para que ocupe tres celdas (cuando hacemos esto la propiedad `layout_span` del objeto se inicializa con la cantidad de celdas que ocupará:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto021.zip](#)

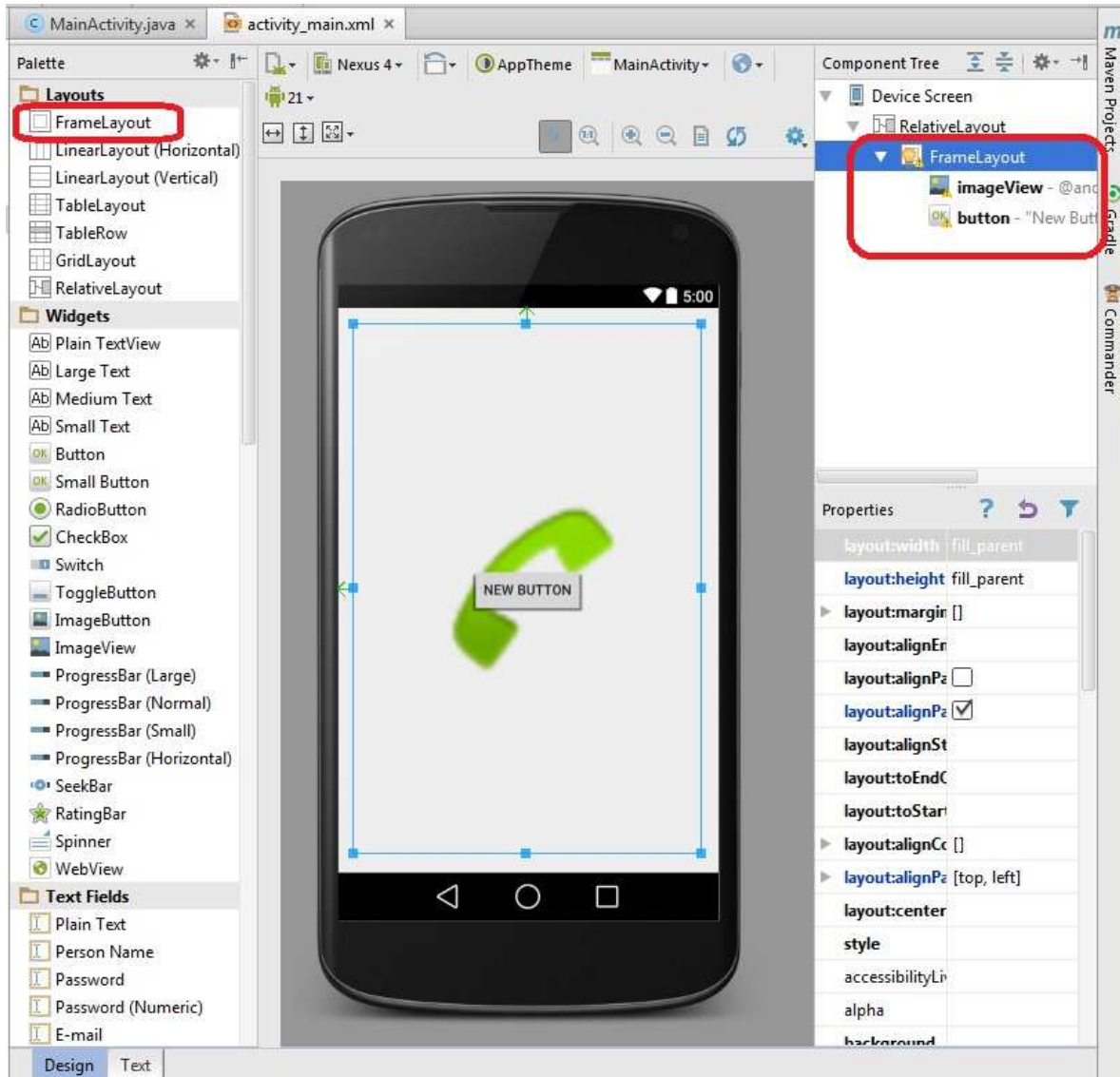
[Retornar](#)

20 - Layout (FrameLayout)

El control de tipo FrameLayout dispone dentro del contenedor todos los controles visuales alineados al vértice superior izquierdo, centrado, vértice inferior derecho etc. (tiene nueve posiciones posibles).

Si disponemos dos o más controles los mismos se apilan.

Por ejemplo si disponemos dentro de un FrameLayout un ImageView y un Button luego el botón se superpone a la imagen:

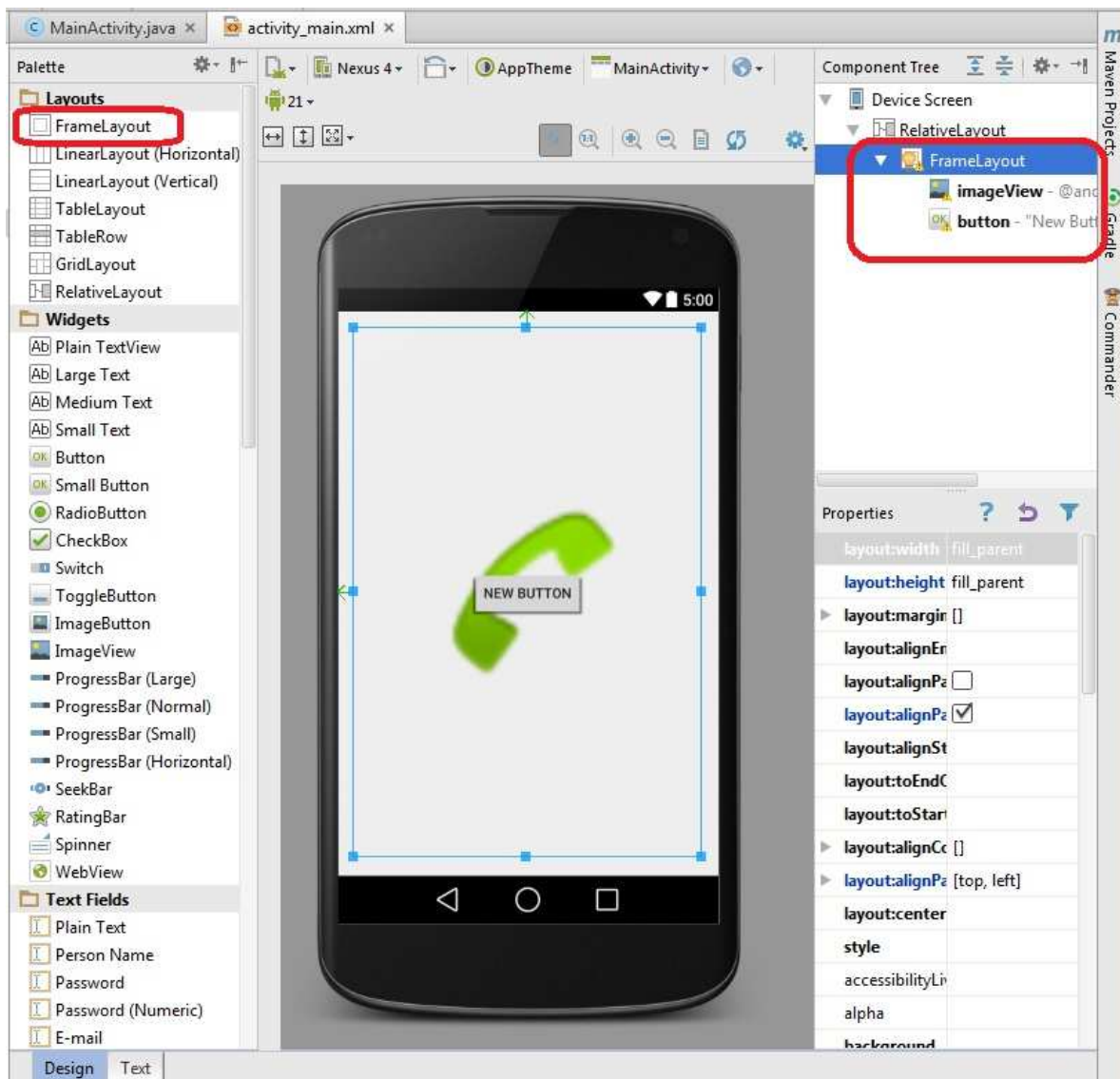


Una actividad posible del control FrameLayout es disponer una serie de controles visuales no visibles e ir alternando cual se hace visible.

Problema:

Disponer un ImageView y un Button dentro de un layout FrameLayout. Cuando se inicia la aplicación mostrar solo el botón y al ser presionado ocultar el botón y hacer visible la imagen que muestra el ImageView.

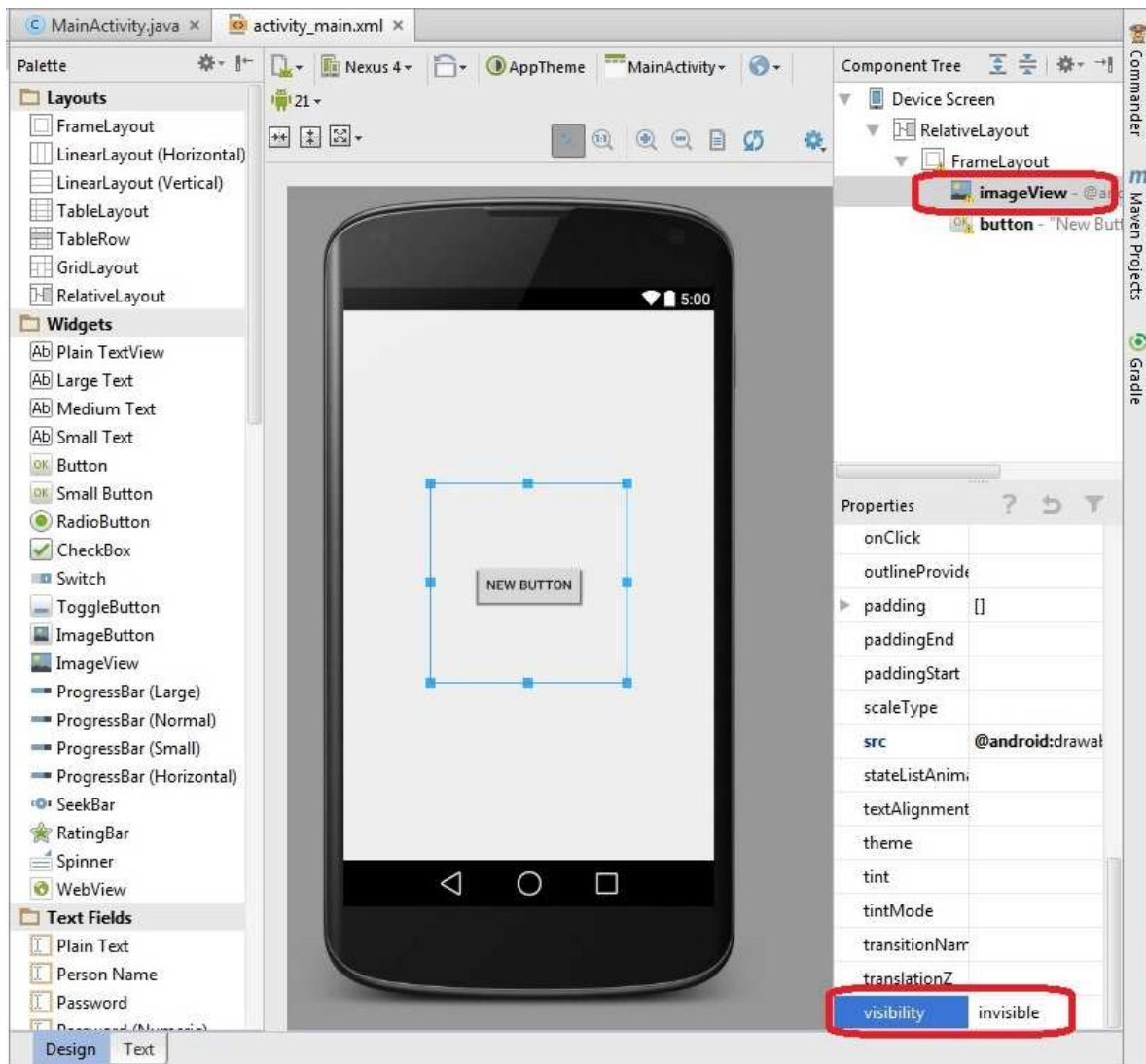
Creamos una interfaz visual similar a la siguiente:



Inicializar la propiedad src del objeto de la clase ImageView eligiendo una imagen en el diálogo "Resources" en la pestaña "System" donde se encuentran todas las imágenes que tiene el sistema operativo Android por defecto.

Seleccionamos el control ImageView y fijamos la propiedad visibility con el valor invisible (esto hace que la imagen no se muestre en pantalla)

Luego inicializamos la propiedad onClick del control Button indicando el nombre del método que se ejecutará al ser presionado (onClick: ocultar):



El código fuente de la clase es:

```
package ar.com.tutorialesya.proyecto022;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

    private ImageView iv1;
    private Button b1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        iv1=(ImageView) findViewById(R.id.imageView);
        b1=(Button) findViewById(R.id.button);
    }

    @Override
```

```

public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void ocultar(View v) {
    b1.setVisibility(View.INVISIBLE);
    iv1.setVisibility(View.VISIBLE);
}
}

```

Cuando se presiona el botón procedemos a ocultar el botón y hacer visible la imagen:

```

public void ocultar(View v) {
    b1.setVisibility(View.INVISIBLE);
    iv1.setVisibility(View.VISIBLE);
}

```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto022.zip](#)

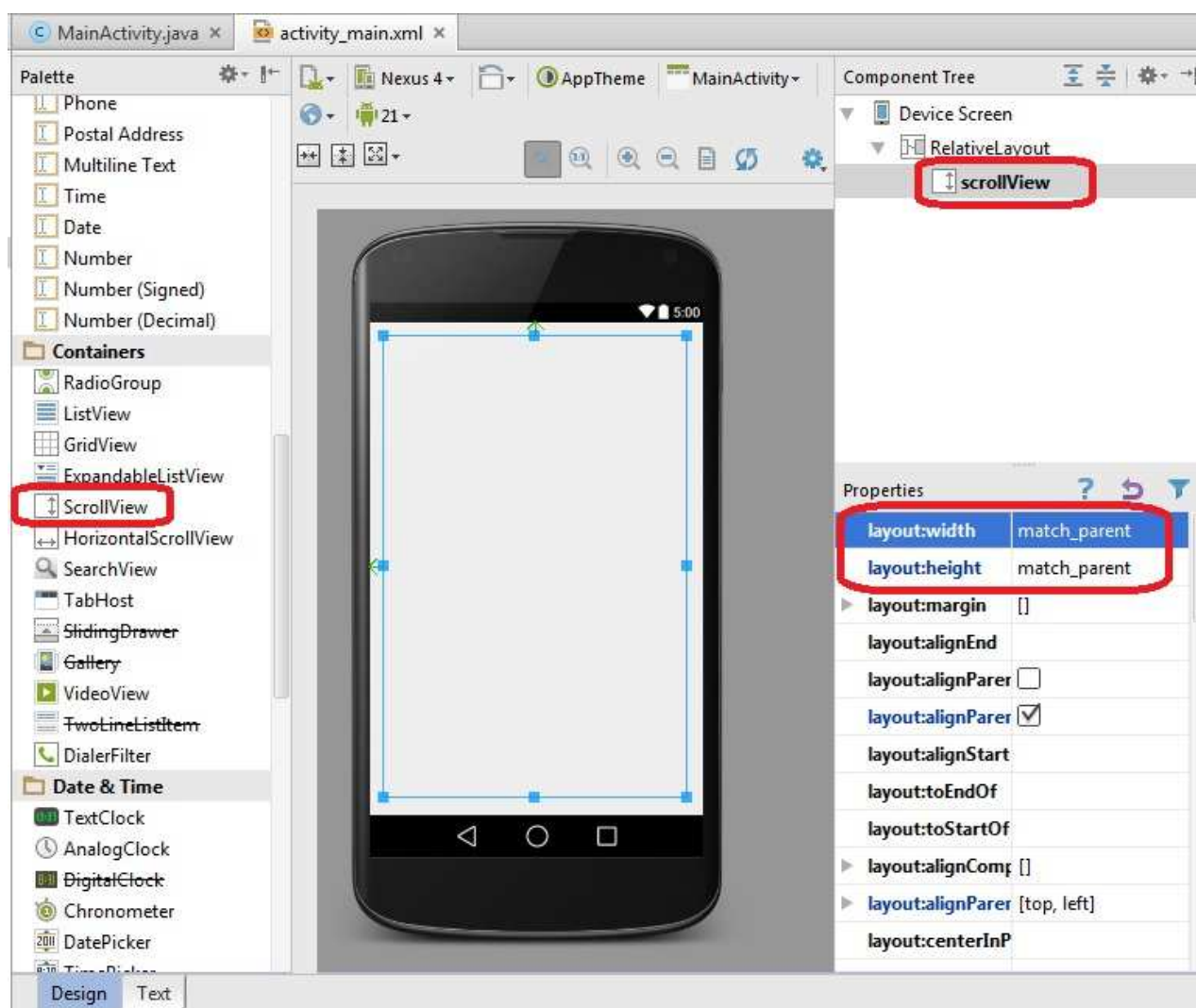
[Retornar](#)

21 - Layout (ScrollView y LinearLayout)

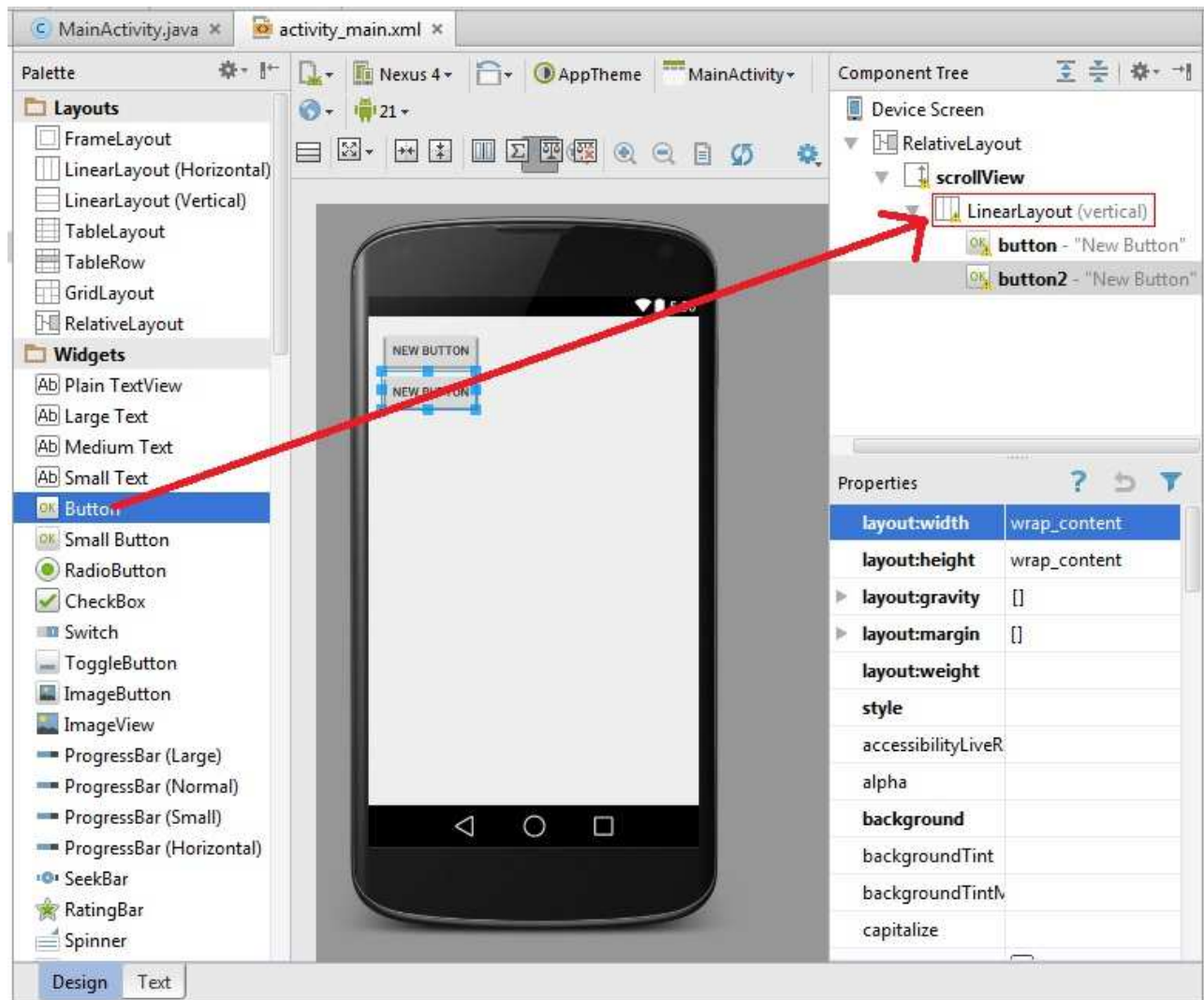
El ScrollView junto con un LinearLayout nos permite disponer una cantidad de componentes visuales que superan la cantidad de espacio del visor del celular o tablet. Luego el usuario puede desplazar con el dedo la interfaz creada.

Problema:

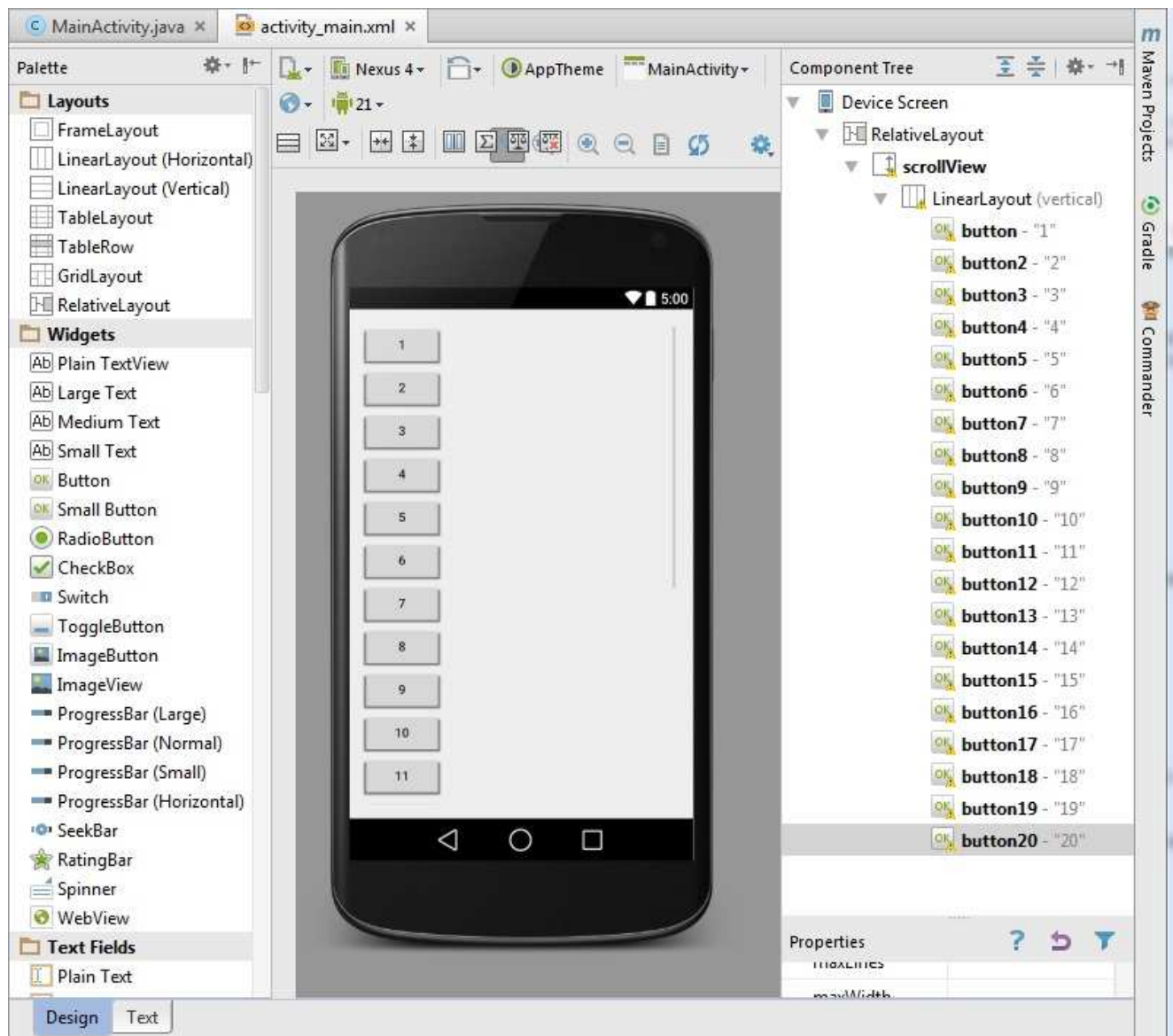
Crear un proyecto llamado: Proyecto023 y disponer un control de tipo ScrollView (que se encuentra en la pestaña "Containers") modificar las propiedades layout:width y layout:height con el valor: match_parent, con esto tenemos que el ScrollView ocupa todo el contenedor (en nuestro caso ocupa todo el espacio del RelativeLayout que lo contiene):



Seguidamente arrastramos un objeto de la clase "LinearLayout Vertical" de la pestaña "Layouts" y lo disponemos dentro del ScrollView. Ahora vamos a disponer 20 objetos de la clase Button dentro del LinearLayout (como no van a entrar en pantalla iremos arrastrando los botones a la ventana "Component Tree" sobre el objeto "LinearLayout"):



Seguimos arrastrando botones dentro del LinearLayout hasta completar los 20 (luego cambiamos la propiedad text), como podemos observar hay más botones dentro del LinearLayout que los que puede mostrarse en la interfaz del dispositivo (por eso tuvimos que arrastrarlos a la ventana del "Component Tree"):



Gracias a la funcionalidad del ScrollView junto al LinearLayout ahora en tiempo de ejecución podemos hacer scroll:



Este proyecto lo puede descargar en un zip desde este enlace: proyecto023.zip

[Retornar](#)

22 - Icono de la aplicación

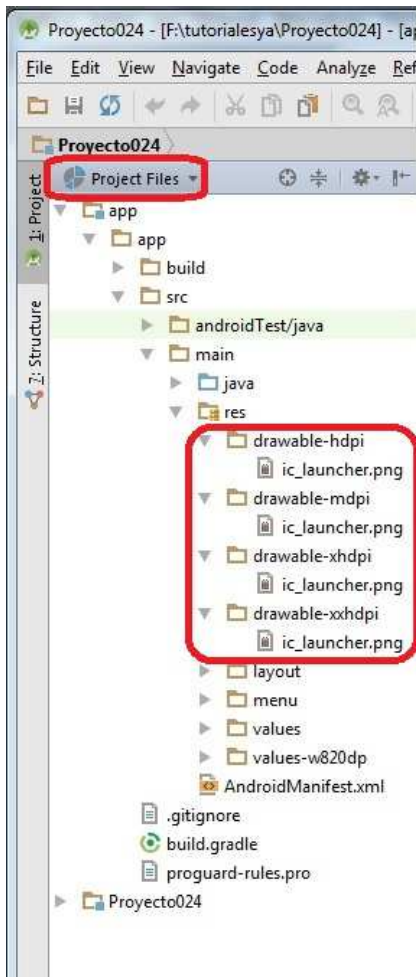
Cuando creamos un proyecto para implementar una aplicación con el entorno de desarrollo Android Studio, éste nos crea un ícono por defecto:



Los íconos e imágenes se almacenan en la carpeta res y se encuentra en "Proyecto024\app\src\main\res\drawable-mdpi" res (resources) y en esta hay cuatro carpetas llamadas:

drawable-ldpi
drawable-mdpi
drawable-hdpi
drawable-xhdpi

Y en cada una de estas hay un archivo llamado ic_launcher.png (para ver las carpetas podemos seleccionar "Project Files"):



Como las resoluciones de los dispositivos Android pueden ser muy distintos (un celular, una tablet, un televisor etc.) se recomienda proporcionar múltiples copias de cada imagen de recursos a diferentes resoluciones y almacenarlos en las carpetas nombradas respetando las siguientes reglas:

res/drawable-mdpi/
El ícono debe ser de 48*48 píxeles.
res/drawable-hdpi/
150% del tamaño de las imágenes almacenadas en la carpeta drawable-mdpi
El ícono debe ser de 72*72 píxeles.
res/drawable-xhdpi/
200% del tamaño de las imágenes almacenadas en la carpeta drawable-mdpi
El ícono debe ser de 96*96 píxeles.

res/drawable-xxhdpi/
300% del tamaño de las imágenes almacenadas en la carpeta drawable-mdpi
El ícono debe ser de 144*144 píxeles.

Problema:

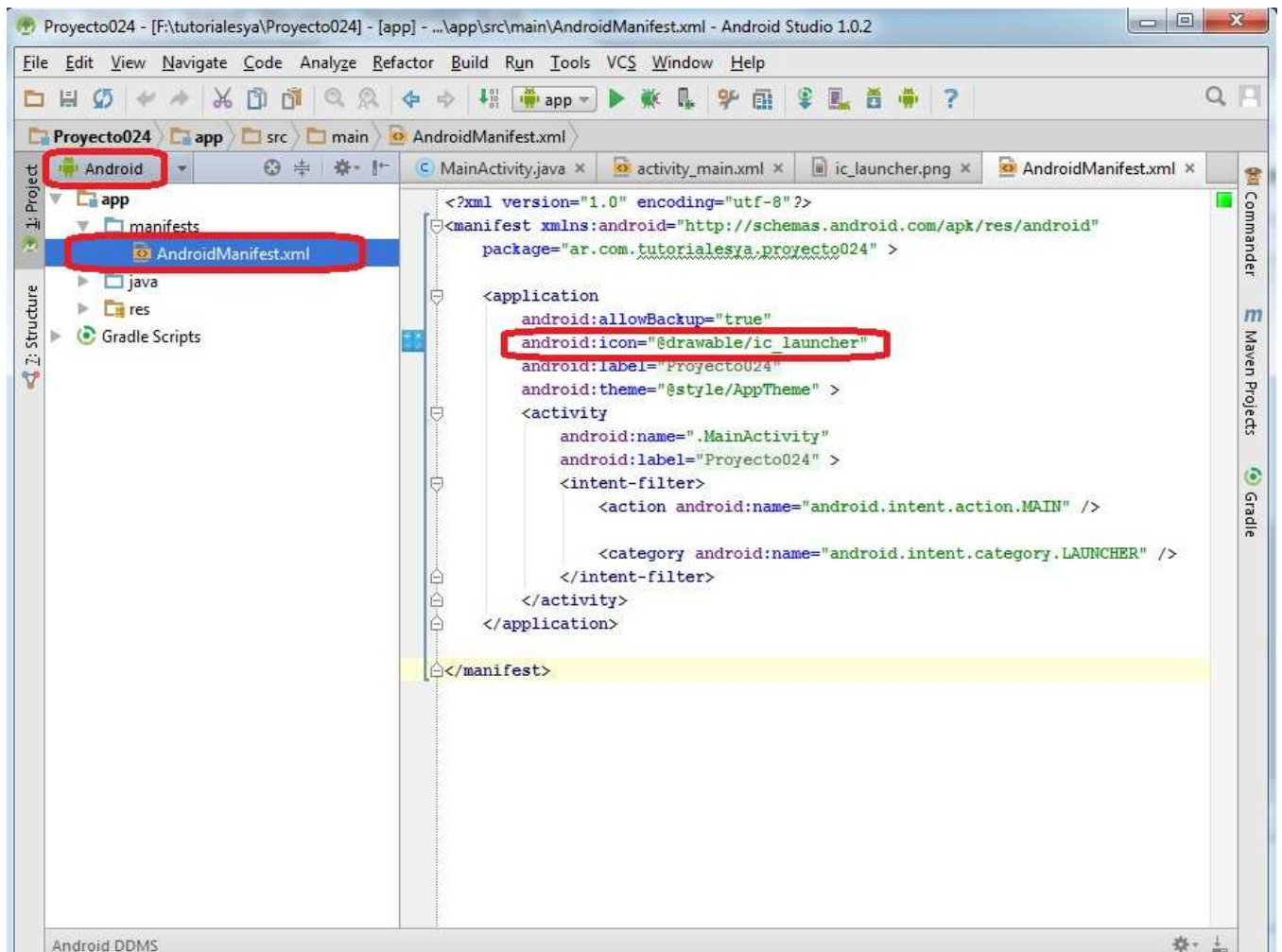
Crear una aplicación, dibujar y almacenar cuatro archivos llamados ic_launcher.png (borrar los actuales). Tener en cuenta que el archivo que se almacena en la carpeta drawable-mdpi debe ser de 48 píxeles, el de la carpeta drawable-hdpi debe ser de 72 píxeles de ancho y alto, el de la carpeta drawable-xhdpi debe ser de 96 píxeles y finalmente el de la carpeta xxhdpi debe ser de 144*144 píxeles.

Ejecutar la aplicación y ver el ícono nuevo.



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto024.zip](#)

En el archivo AndroidManifest.xml es donde indicamos el nombre del ícono de la aplicación:



Como vemos tenemos el nombre del archivo ic_launcher (no debemos indicar extensión)

Importante

Los nombres de archivos solo pueden tener caracteres en minúsculas, números y el guión bajo, cualquier otro carácter generará un error cuando tratemos de ejecutar la aplicación. Si bien podemos utilizar números dentro del nombre del archivo no puede ubicarse un número como primer carácter del nombre del archivo (esto es debido a que el Android Studio genera un archivo de recursos y define variable con dicho nombre de archivo)

[Retornar](#)

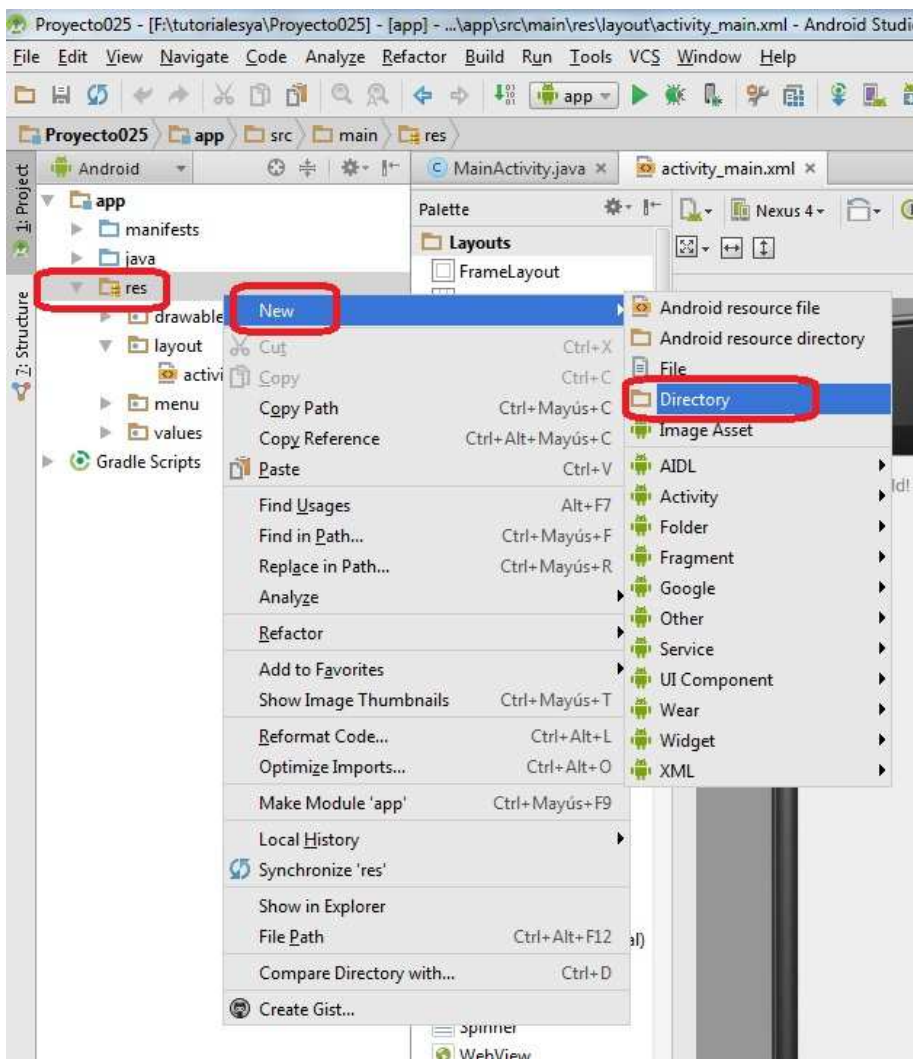
23 - Reproducción de audio (archivo contenido en la aplicación)

Veremos los pasos para reproducir un archivo mp3 (otros formatos soportados por Android son: Ogg, Wav)

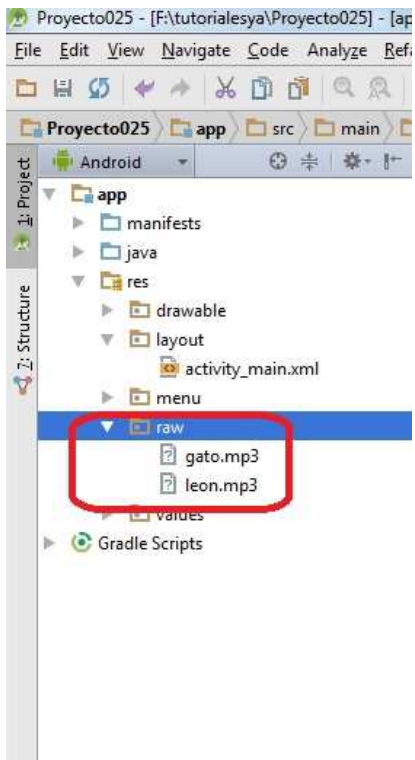
Problema:

Disponer dos botones con las etiquetas: Gato y León, luego cuando se presione reproducir el archivo de audio respectivo. Los archivos de sonidos almacenarlos en la misma aplicación.

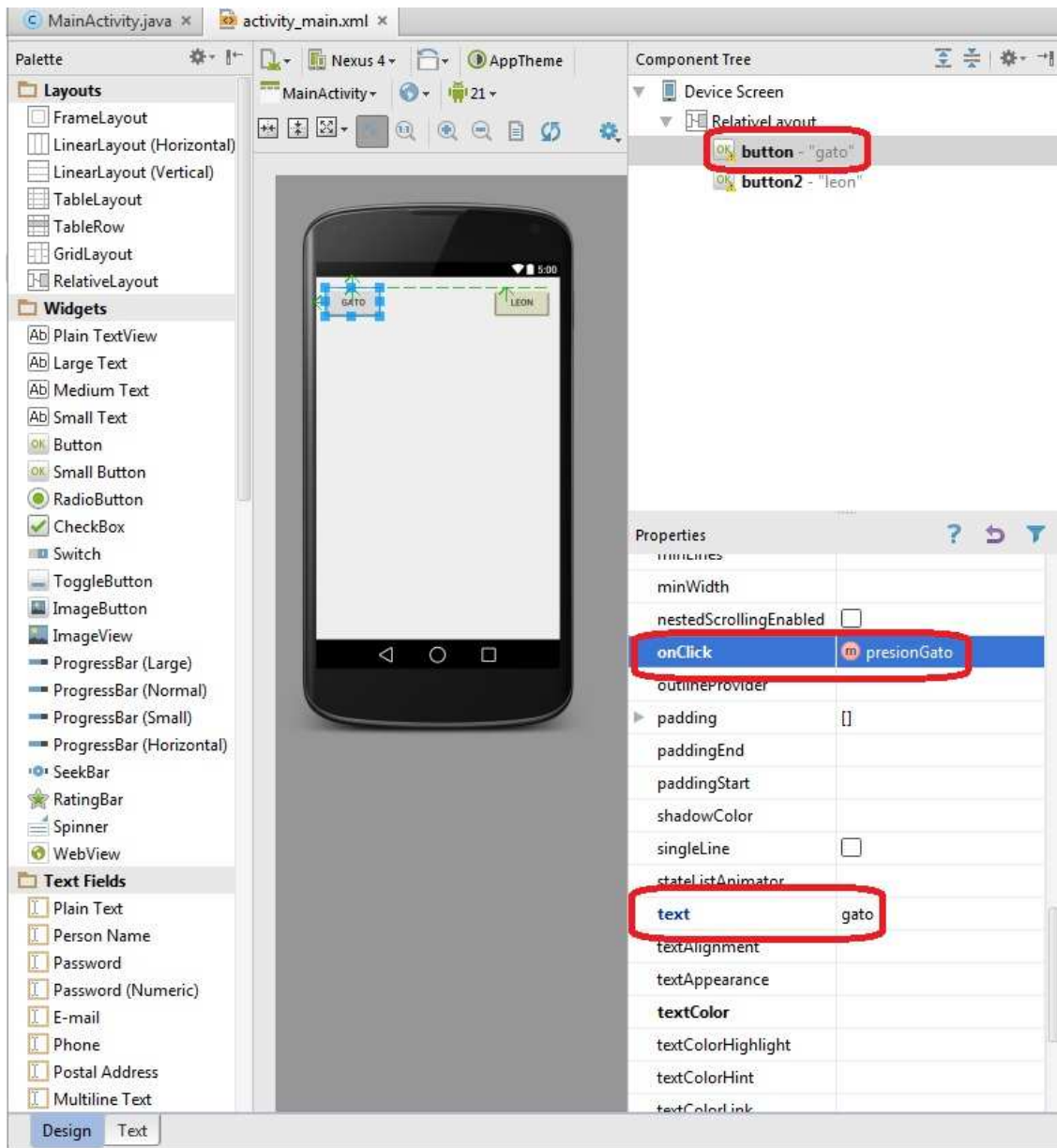
Luego de crear el proyecto procedemos a crear una carpeta llamada raw que dependa de la carpeta res, almacenamos los dos archivos mp3 en dicha carpeta (para crear la carpeta presionamos el botón derecho del mouse sobre la carpeta res y seleccionamos New -> Directory):



Luego de copiar los archivos a la carpeta (en Android Studio funciona el Copy/Paste desde el administrador de archivos del sistema operativo Windows):



Creamos una interfaz con dos botones e inicializamos las propiedades text y onClick de cada botón:



El código fuente es:

```
package ar.com.tutorialesya.proyecto025;

import android.media.MediaPlayer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
```

```

        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void presionGato(View v) {
        MediaPlayer mp = MediaPlayer.create(this, R.raw.gato);
        mp.start();
    }

    public void presionLeon(View v) {
        MediaPlayer mp = MediaPlayer.create(this, R.raw.leon);
        mp.start();
    }
}

```

Cuando copiamos los archivos mp3 se genera luego en la clase R la referencia a los dos archivos y posteriormente los podemos rescatar cuando creamos un objeto de la clase MediaPlayer:

```
MediaPlayer mp=MediaPlayer.create(this,R.raw.gato);
```

Seguidamente llamamos al método start:

```
mp.start();
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto025.zip](#)

[Retornar](#)

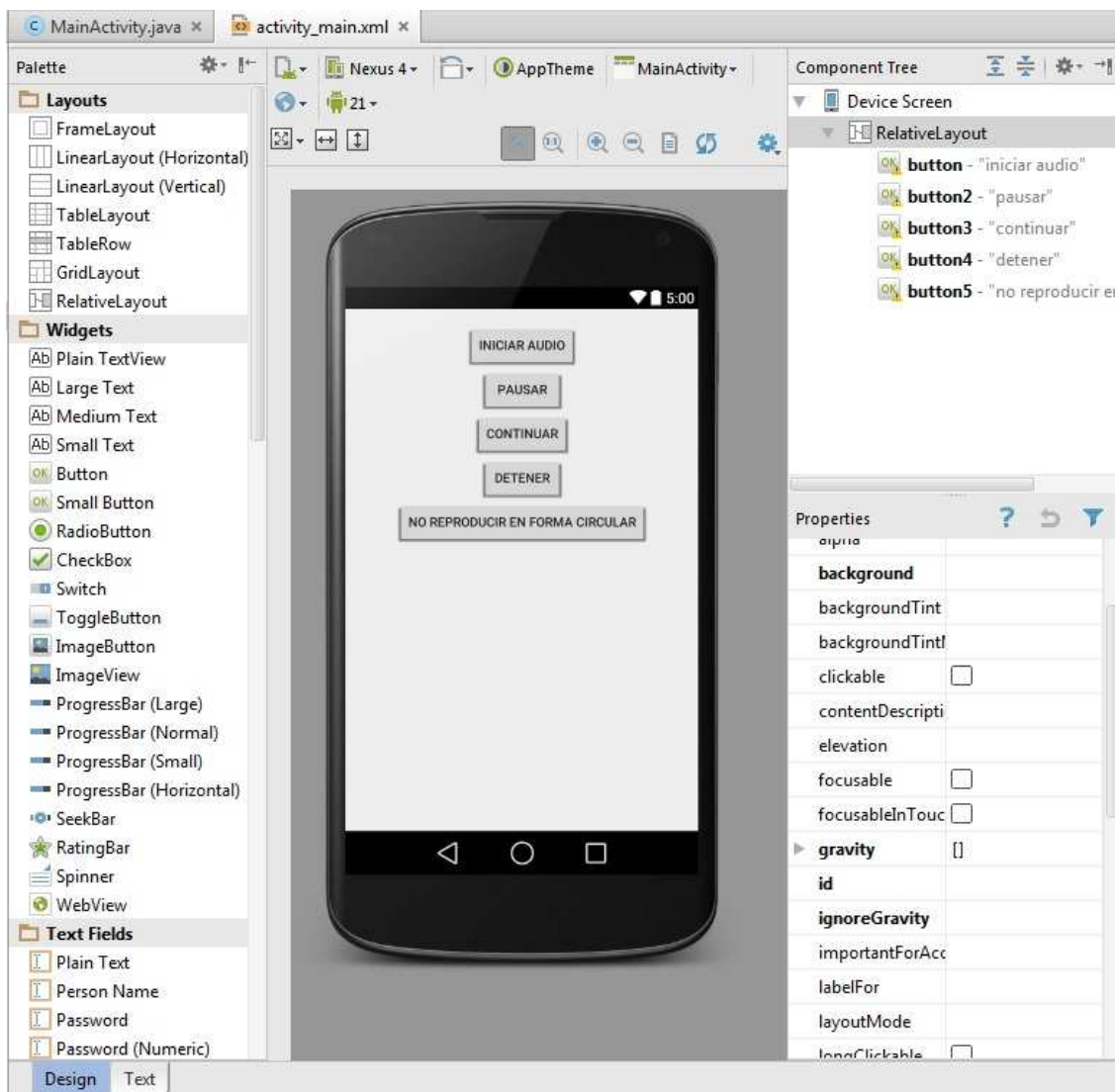
24 - Reproducción, pausa, continuación y detención de un archivo de audio.

Problema:

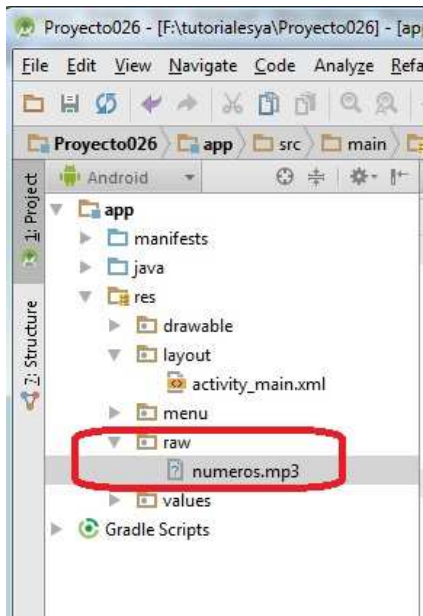
Confeccionar una aplicación que permita Iniciar un archivo mp3, detener, continuar, detener en forma definitiva y activación o no de la reproducción en forma circular.

Crear un archivo mp3 con el programa [Audacity](#) contando del 1 al 30.

Primero creamos un proyecto (Proyecto026) y definimos los 5 botones y métodos a ejecutar cuando se presionen los botones respectivos:



Creamos la carpeta raw y almacenamos en la misma el archivo mp3 creado previamente:



El código fuente es:

```
package ar.com.tutorialesya.proyecto026;

import android.media.MediaPlayer;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;

public class MainActivity extends ActionBarActivity {
    MediaPlayer mp;
    Button b5;
    int posicion = 0;

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

        b5=(Button)findViewById(R.id.button5);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
```

```

        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void destruir() {
    if (mp != null)
        mp.release();
}

public void iniciar(View v) {
    destruir();
    mp = MediaPlayer.create(this, R.raw.numeros);
    mp.start();
    String op = b5.getText().toString();
    if (op.equals("no reproducir en forma circular"))
        mp.setLooping(false);
    else
        mp.setLooping(true);
}

public void pausar(View v) {
    if (mp != null && mp.isPlaying()) {
        posicion = mp.getCurrentPosition();
        mp.pause();
    }
}

public void continuar(View v) {
    if (mp != null && mp.isPlaying() == false) {
        mp.seekTo(posicion);
        mp.start();
    }
}

public void detener(View v) {
    if (mp != null) {
        mp.stop();
        posicion = 0;
    }
}

public void circular(View v) {
    detener(null);
    String op = b5.getText().toString();
    if (op.equals("no reproducir en forma circular"))
        b5.setText("reproducir en forma circular");
    else
        b5.setText("no reproducir en forma circular");
}
}

```

Primero definimos tres atributos uno de la clase MediaPlayer para administrar el archivo mp3, un entero donde se almacena la posición actual de reproducción en milisegundos (para poder continuarla en el futuro) y la referencia de un objeto de la clase Button:

```

MediaPlayer mp;
Button b5;
int posicion = 0;

```

El método destruir verifica con un if si el objeto de la clase MediaPlayer está creado procede a liberar recursos del mismo llamando al método release:

```

public void destruir() {
    if(mp!=null)
        mp.release();
}

```

El método iniciar que se ejecuta al presionar el botón "iniciar" primero llama al método destruir (para el caso que el mp3 este en ejecución actualmente) seguidamente creamos un objeto de la clase MediaPlayer llamando al método create (en este hacemos referencia al archivo que copiamos a la carpeta raw) Llamamos al método start. Por último extraemos el texto del quinto botón y verificamos si la reproducción debe ejecutarse en forma circular (en forma indefinida una y otra vez):

```

public void iniciar(View v) {
    destruir();
    mp = MediaPlayer.create(this,R.raw.numeros);
    mp.start();
    String op=b5.getText().toString();
    if (op.equals("no reproducir en forma circular"))
        mp.setLooping(false);
    else
        mp.setLooping(true);
}

```

El método pausar verifica que el objeto de la clase MediaPlayer este creado y en ejecución, en caso afirmativo recuperamos la posición actual de reproducción y llamamos seguidamente al método pause:

```

public void pausar(View v) {
    if(mp != null && mp.isPlaying()) {
        posicion = mp.getCurrentPosition();
        mp.pause();
    }
}

```

El método continuar verifica que el objeto de la clase MediaPlayer este creado y la propiedad isPlaying retorne false para proceder a posicionar en que milisegundo continuar la reproducción:

```

public void continuar(View v) {
    if(mp != null && mp.isPlaying()==false) {
        mp.seekTo(posicion);
        mp.start();
    }
}

```

El método detener interrumpe la ejecución del mp3 e inicializa el atributo posicion con cero:

```

public void detener(View v) {
    if(mp != null) {
        mp.stop();
        posicion = 0;
    }
}

```

Cuando se presiona el botón que cambia si la reproducción se efectúa en forma circular o no procedemos a extraer su texto y según dicho valor almacenamos el valor opuesto:

```

public void circular(View v) {
    detener(null);
    String op=b5.getText().toString();
    if (op.equals("no reproducir en forma circular"))
        b5.setText("reproducir en forma circular");
    else
        b5.setText("no reproducir en forma circular");
}

```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto026.zip](#)

[Retornar](#)

25 - Reproducción de audio (archivo contenido en una tarjeta SD)

El objetivo de este concepto es acceder a un archivo mp3 almacenado en la tarjeta SD de nuestro equipo (tener en cuenta que esto solo funciona con un equipo que dispone la capacidad de tarjeta SD) Debemos utilizar un emulador que tenga configurado tarjeta SD.

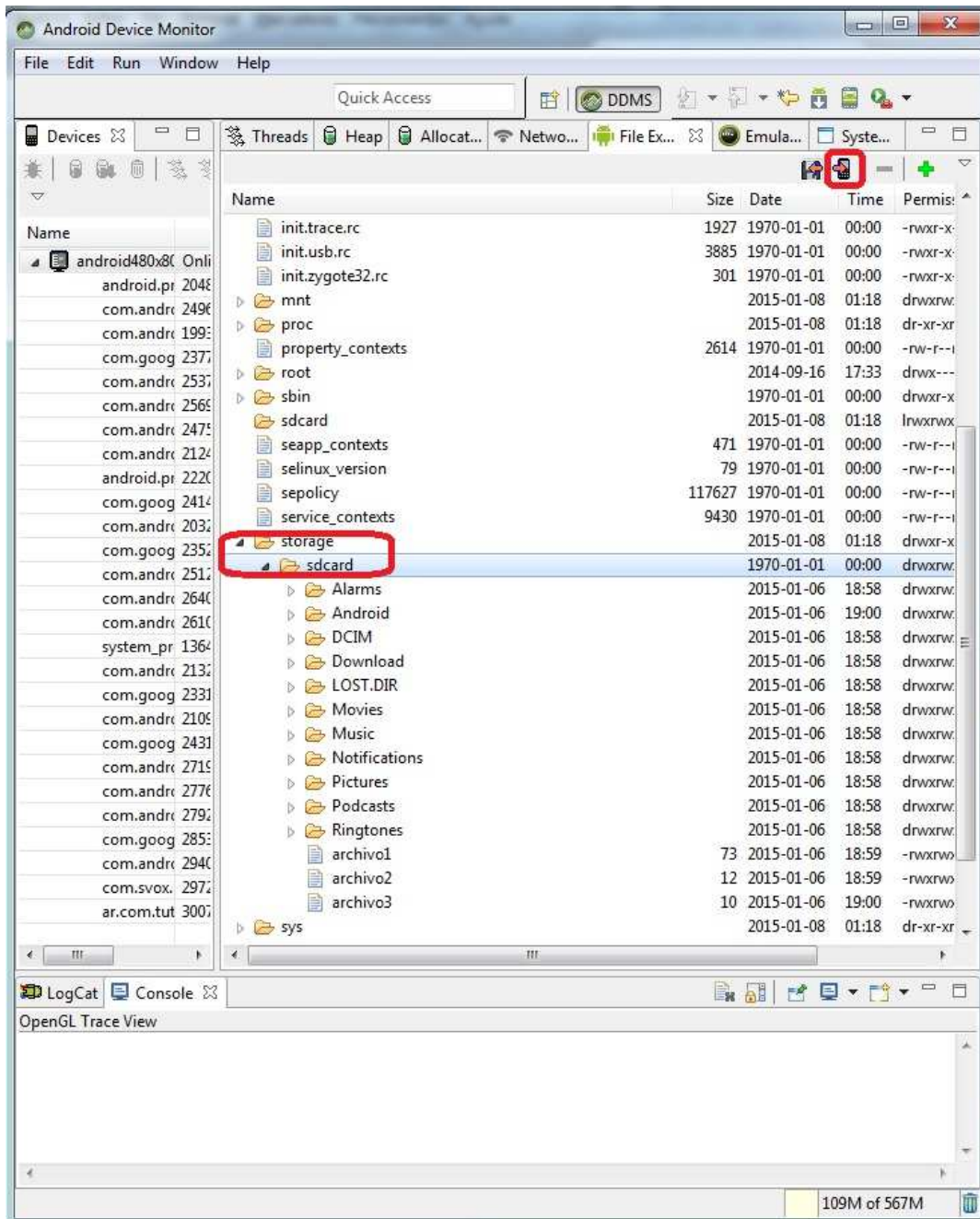
Problema:

Disponer un botón con la etiqueta: "gato", luego cuando se presione reproducir el archivo de audio respectivo. El archivo de sonido almacenarlo en la tarjeta SD.

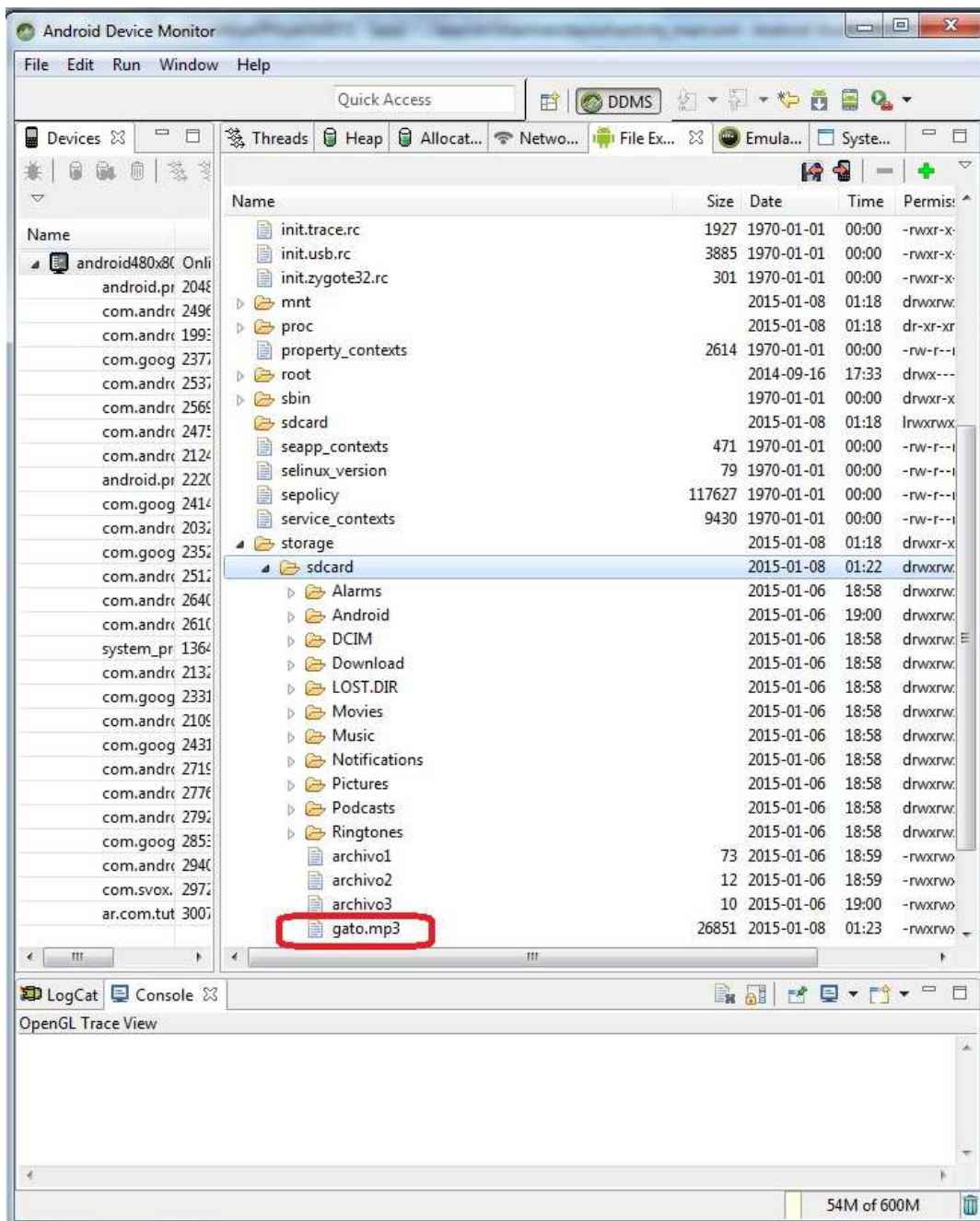
Luego de crear el proyecto (Proyecto027) debemos ejecutar la aplicación para poder acceder a la tarjeta SD que crea el emulador Android.

Una vez que se este ejecutando la aplicación (no importa que todavía no hemos implementado su funcionalidad) procedemos a abrir la ventana "Android Device Monitor". Seleccionamos del menú de opciones de Tools->Android->Android Device Monitor.

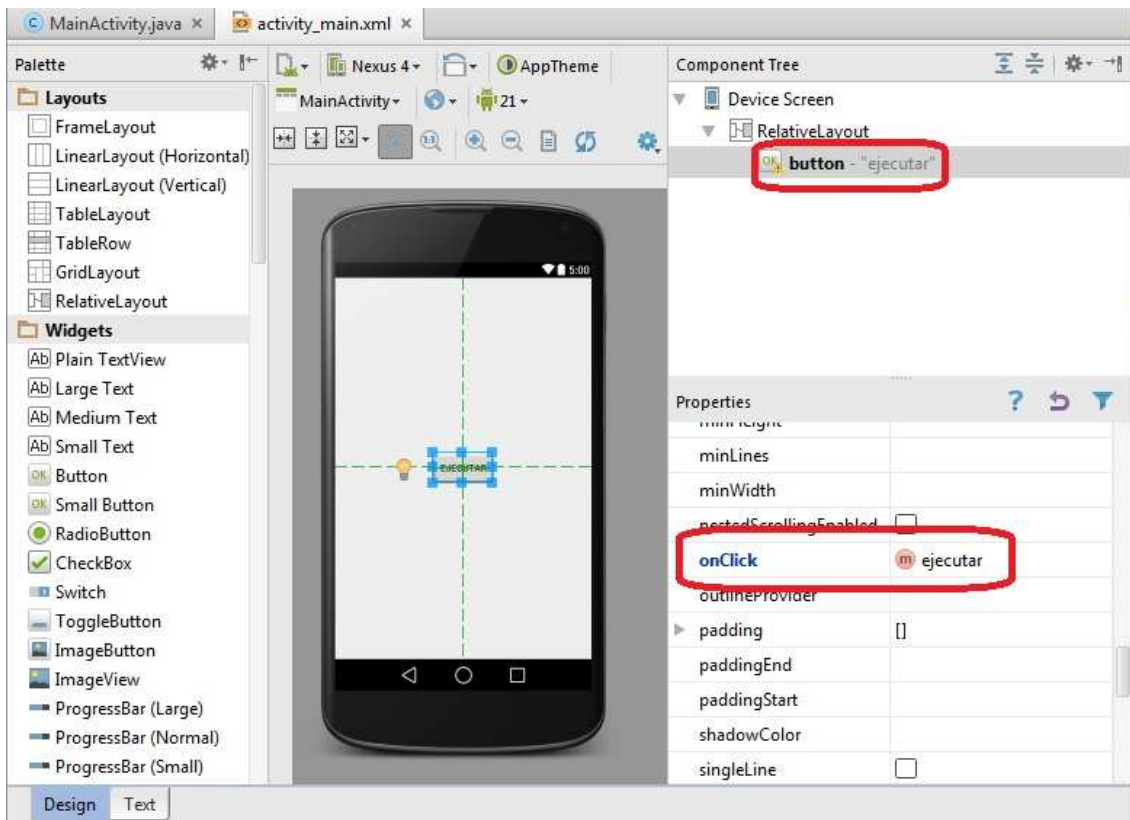
En la carpeta storage/sdcard debemos subir el archivo mp3 (debemos seleccionar esta carpeta con el mouse) Esto lo hacemos mediante un botón que aparece en la parte derecha de esta ventana "Push a file onto device", luego de esto tenemos el archivo montado en la tarjeta SD:



Luego de subirlo debe aparecer dentro de la lista de archivos que tiene la tarjeta SD:



Ahora implementemos la interfaz de nuestra aplicación (un solo botón) que cuando se presione llame al método ejecutar:



El código fuente es:

```
package ar.com.tutorialesya.proyecto027;

import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Environment;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
    }
```

```

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void ejecutar(View v) {
        Uri datos = Uri.parse(Environment.getExternalStorageDirectory()
            .getPath() + "/gato.mp3");
        MediaPlayer mp = MediaPlayer.create(this, datos);
        mp.start();
    }
}

```

Creamos un objeto de la clase Uri llamando al método parse donde indicamos el path y nombre del archivo a recuperar:

```
Uri datos = Uri.parse(Environment.getExternalStorageDirectory().getPath() + "/gato.mp3");
```

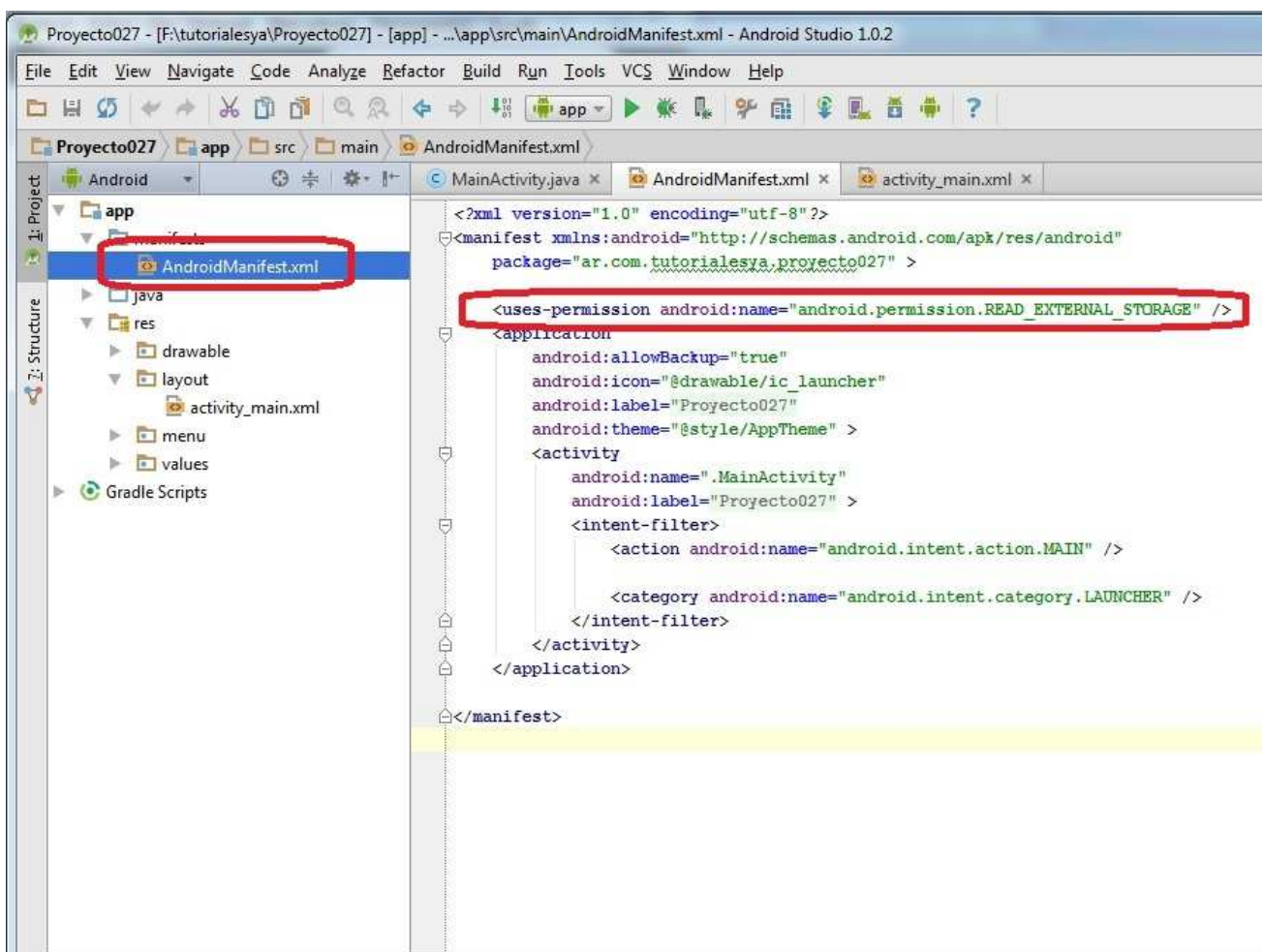
Creamos el objeto de la clase MediaPlayer pasando ahora la referencia del objeto de la clase Uri:

```
MediaPlayer mp=MediaPlayer.create(this, datos);
```

Iniciamos la reproducción del mp3:

```
mp.start();
```

Otro paso importante antes de ejecutar la aplicación es agregar el permiso de leer la tarjeta SD, para ello modificamos el archivo AndroidManifest.xml agregando este permiso:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto027.zip](#)

Recordar que para ejecutar este proyecto se debe subir un archivo mp3 llamado "gato.mp3" a la tarjeta SD.

[Retornar](#)

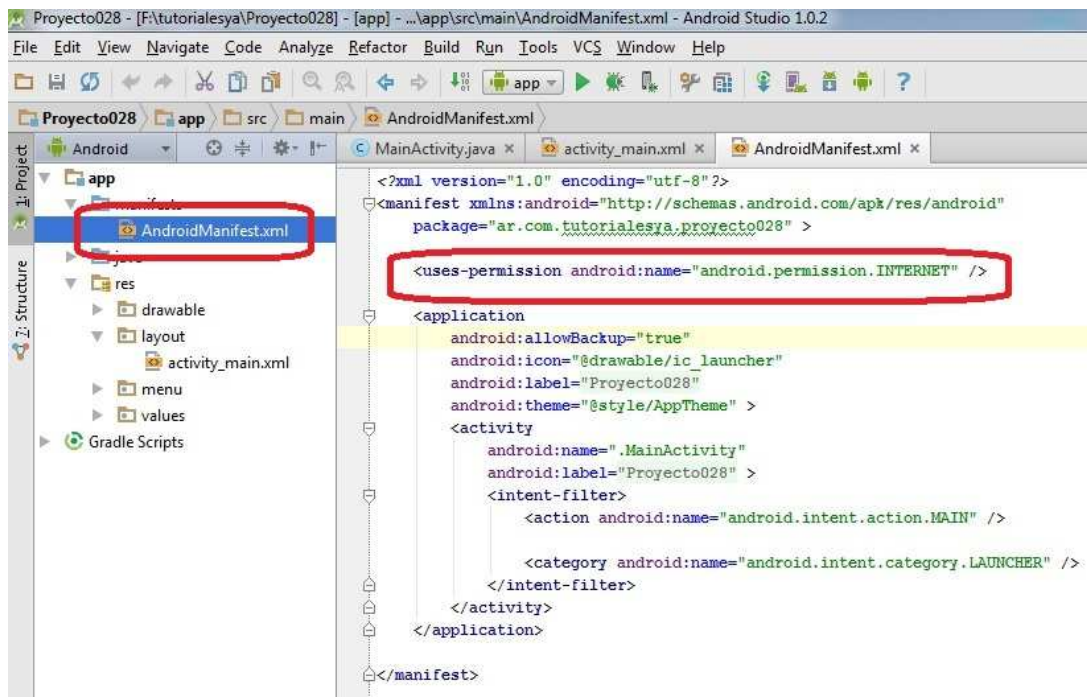
26 - Reproducción de audio (archivo localizado en internet)

Ahora vamos a ver los pasos para reproducir un archivo almacenado en un servidor de internet.

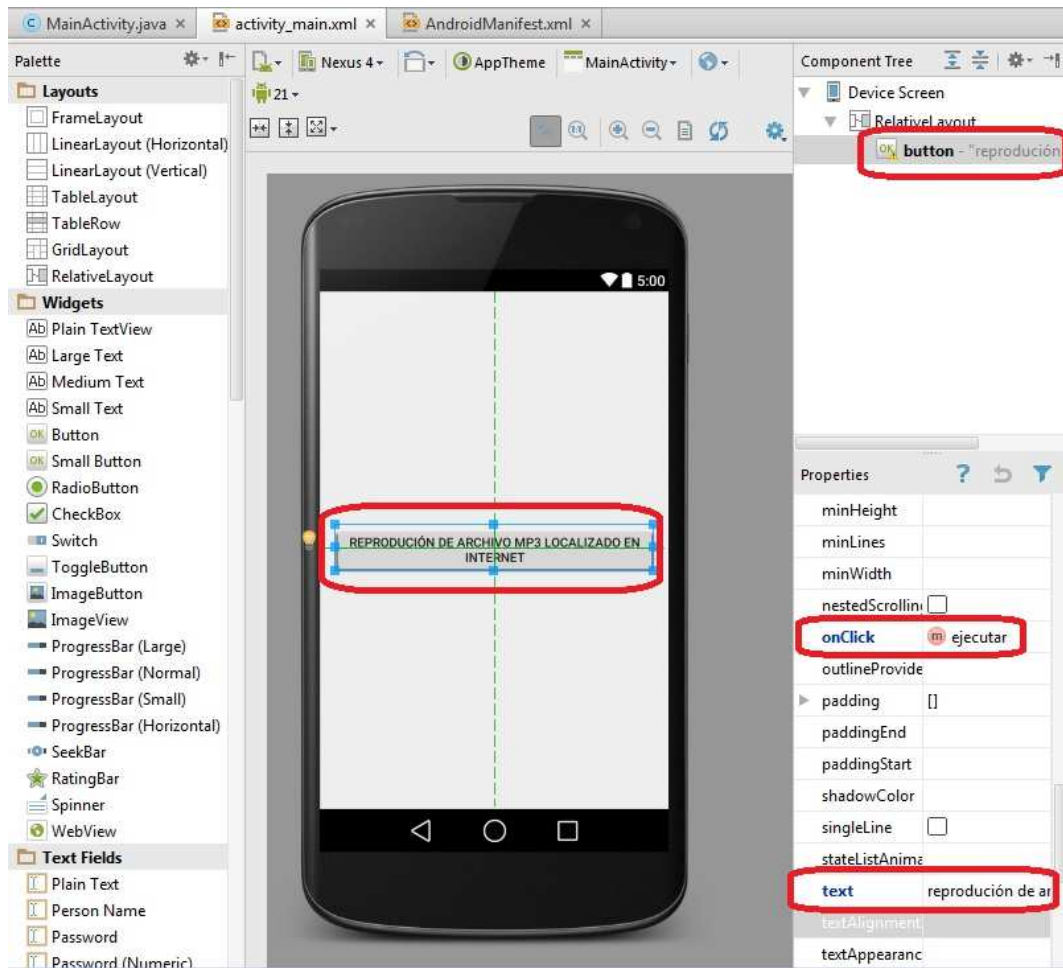
Problema:

Disponer un botón con la etiqueta: "gato", luego cuando se presione reproducir el archivo de audio respectivo. El archivo de sonido se encuentra almacenado en <http://www.javaya.com.ar/recursos/gato.mp3>

Creamos un proyecto (Proyecto028) y el primer paso es modificar el archivo AndroidManifest.xml donde autorizamos a la aplicación a acceder a recursos localizados en internet (debemos abrir este archivo y proceder a agregar el permiso de acceso a internet agregando la marca uses-permission respectivo):



Creamos la interfaz de la aplicación e inicializamos el evento onClick del Button con el método que implementaremos llamado "ejecutar":



El código fuente es:

```
package ar.com.tutorialesya.proyecto028;

import android.media.MediaPlayer;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import java.io.IOException;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
    }
}
```



```

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void ejecutar(View v) {
        MediaPlayer mp = new MediaPlayer();
        try {
            mp.setDataSource("http://www.javaya.com.ar/recursos/gato.mp3");
            mp.prepare();
            mp.start();
        } catch (IOException e) {
        }
    }
}

```

Para recuperar un archivo mp3 de internet procedemos de la siguiente manera, primero creamos un objeto de la clase MediaPlayer:

```
MediaPlayer mp=new MediaPlayer();
```

Luego llamamos al método setDataSource indicando la dirección de internet donde se almacena el archivo mp3:

```
mp.setDataSource("http://www.javaya.com.ar/recursos/gato.mp3");
```

Llamamos al método prepare y seguidamente llamamos a start:

```
mp.prepare();
mp.start();
```

Todo esto lo hacemos en un bloque try/catch para capturar excepciones de tipo IOException.

Esta primera aproximación para ejecutar un mp3 localizado en internet bloquea la aplicación hasta que se carga por completo el archivo, es decir queda ejecutándose el método mp.prepare() hasta que finaliza la recuperación en forma completa.

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto028.zip](#)

Problema:

Confeccionar otra aplicación similar a la anterior pero que no se congele la interfaz de la aplicación mientras se carga el mp3. Mostrar un mensaje que el archivo se está cargando.

Desarrollamos un nuevo proyecto (Proyecto029), asignamos el permiso de acceder a internet en el archivo AndroidManifest.xml y creamos una interfaz similar al problema anterior.

El código fuente es:

```

package ar.com.tutorialesya.proyecto029;

import android.media.MediaPlayer;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;

import java.io.IOException;

public class MainActivity extends ActionBarActivity implements MediaPlayer.OnPreparedListener {

    private MediaPlayer mp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void ejecutar(View v) {
        mp = new MediaPlayer();
        mp.setOnPreparedListener(this);
        try {
            mp.setDataSource("http://www.javaya.com.ar/recursos/gato.mp3");
            mp.prepareAsync();
        } catch (IOException e) {
        }
        Toast t = Toast.makeText(this,
            "Espere un momento mientras se carga el mp3",
            Toast.LENGTH_SHORT);
        t.show();
    }

    public void onPrepared(MediaPlayer mp) {
        mp.start();
    }
}

```

Para poder capturar el evento que el archivo se terminó de recuperar debemos implementar la interface `MediaPlayer.OnPreparedListener`:

```
public class MainActivity extends ActionBarActivity implements MediaPlayer.OnPreparedListener {
```

Con esto decimos que nuestra clase implementará el método `onPrepared` donde iniciamos la ejecución del mp3:

```

    public void onPrepared(MediaPlayer mp) {
        mp.start();
    }

```

En el evento click del botón creamos el objeto de la clase `MediaPlayer`, le pasamos al método `setOnPreparedListener` la dirección del objeto que capturará el evento de que el recurso está completo. Luego llamamos a los métodos `setDataSource` y `prepareAsync` para inicializar la carga del mp3. Finalmente mostramos un mensaje para informar al usuario que el archivo se está descargando:

```

    public void ejecutar(View v) {
        mp=new MediaPlayer();
        mp.setOnPreparedListener(this);
        try {
            mp.setDataSource("http://www.javaya.com.ar/recursos/gato.mp3");
            mp.prepareAsync();
        }catch(IOException e) {
        }
        Toast t=Toast.makeText(this,"Espere un momento mientras se carga el mp3", Toast.LENGTH_SHORT);
    }

```

```
        t.show();  
    }
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto029.zip](#)

[Retornar](#)

27 - Reproducción de audio utilizando el reproductor propio de Android (vía Intent)

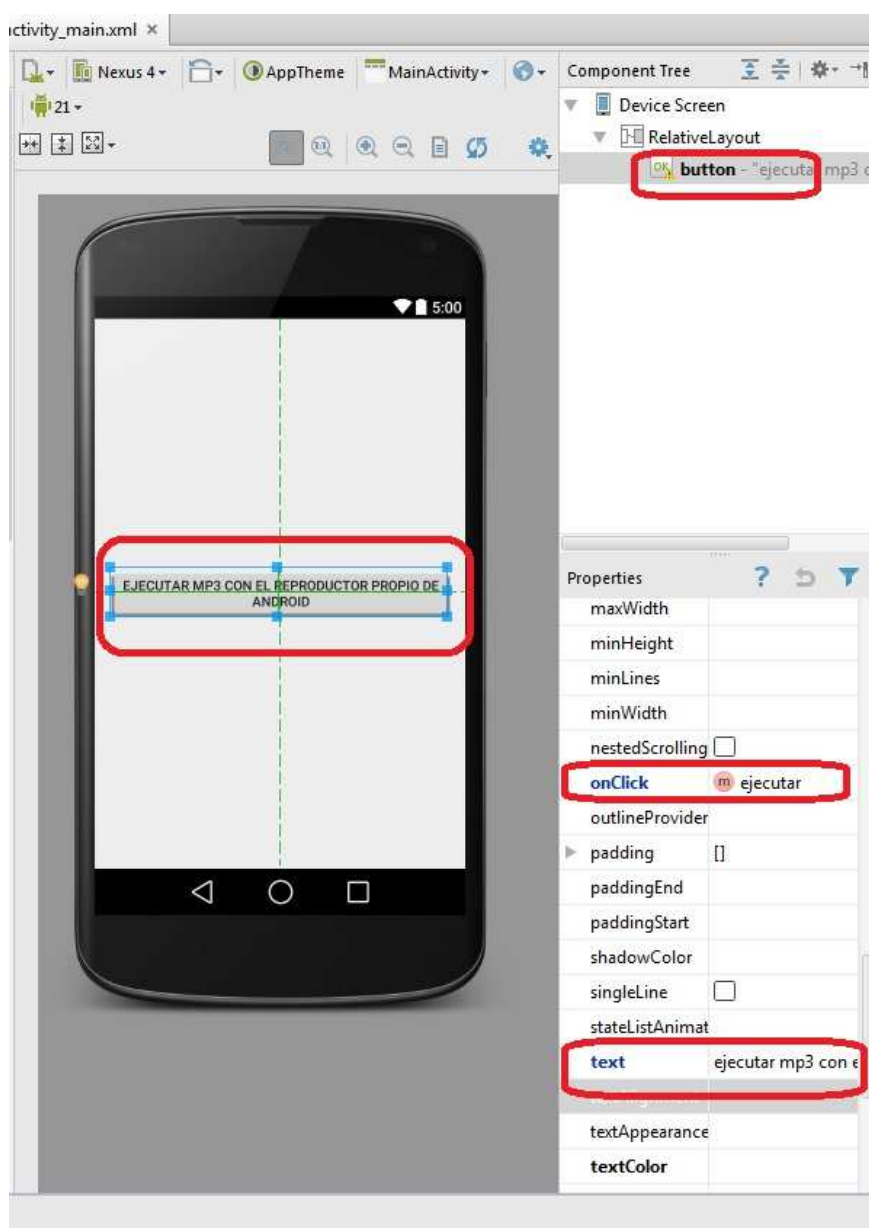
Otra forma de ejecutar un archivo mp3 es mediante el reproductor interno de Android. Esta aplicación reproduce todos los formatos soportados por Android y tiene una interfaz que le será familiar al usuario de nuestra aplicación.

Problema:

Disponer un botón con la etiqueta: "ejecutar mp3 con el reproductor propio de android", luego cuando se presione reproducir el archivo de audio respectivo con el reproductor de Android via Intent. El archivo de sonido almacenarlo en la tarjeta SD, utilizaremos el archivo "gato.mp3" que lo subimos a la tarjeta SD en conceptos anteriores.

Crearemos el Proyecto030.

Creemos la interfaz con el Button y especificamos el evento onClick con el método "ejecutar":



El código fuente es:

```
package ar.com.tutorialesya.proyecto030;

import android.content.Intent;
```

```

import android.net.Uri;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void ejecutar(View v) {
        Intent intent = new Intent(android.content.Intent.ACTION_VIEW);
        Uri data = Uri.parse("file:///sdcard" + "/gato.mp3");
        intent.setDataAndType(data, "audio/mp3");
        startActivity(intent);
    }
}

```

Creamos un objeto de la clase Intent y un objeto de la clase Uri referenciando al archivo mp3 almacenado en la tarjeta SD. Indicamos mediante el método setDataAndType el Uri y el tipo de archivo a reproducir. Activamos la aplicación mediante startActivity.

Cuando presionamos el botón vemos como se activa el reproductor propio de android y es el que realmente reproduce el archivo de audio:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto030.zip](#)

[Retornar](#)

28 - Grabación de audio mediante el grabador provisto por Android (via Intent)

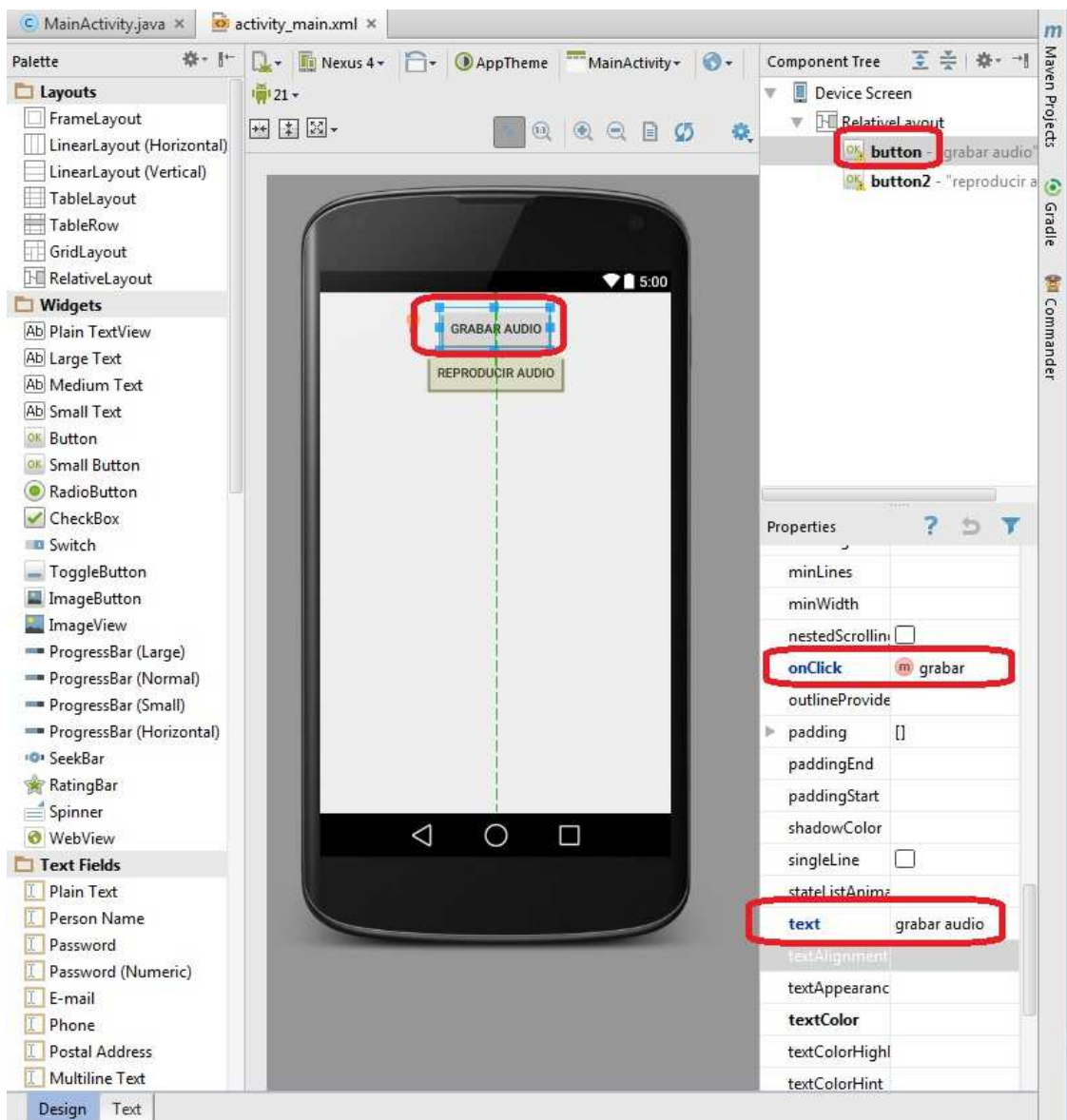
La forma más sencilla de capturar audio en Android es mediante el grabador que provee el sistema operativo Android. Invocamos la aplicación de grabación y luego recuperamos el audio grabado.

Tiene como ventaja que la interfaz le es familiar al usuario, ya que muchas aplicaciones utilizan esta característica.

Problema:

Disponer dos objetos de la clase Button con las etiquetas "grabar" y "reproducir". Cuando se presione el primer botón proceder a activar la grabadora provista por Android. Cuando se presione el segundo botón reproducir el audio grabado.

Crear un proyecto (Proyecto031) e implementar la interfaz, inicializar los eventos onClick de cada botón:



El código fuente es:

```
package ar.com.tutorialesya.proyecto031;

import android.content.Intent;
import android.media.MediaPlayer;
import android.net.Uri;
import android.provider.MediaStore;
```



```

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends ActionBarActivity {

    int petition = 1;
    Uri url1;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void grabar(View v) {
        Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
        startActivityForResult(intent, petition);
    }

    public void reproducir(View v) {
        MediaPlayer mediaPlayer = MediaPlayer.create(this, url1);
        mediaPlayer.start();
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK && requestCode == petition) {
            url1 = data.getData();
        }
    }
}

```

Cuando se presiona el botón de grabar el audio mediante un Intent activamos la aplicación de grabación propia de Android.

Seguidamente llamamos al método `startActivityForResult` para poder recuperar la grabación luego de finalizada a través del método `onActivityResult`:

```
public void grabar(View v) {
```

```

        Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
        startActivityForResult(intent, petition);
    }

```

Debemos pasar al método `startActivityForResult` además de la referencia del `Intent` una variable con un valor 0 o positivo (luego este valor retornará al método `onActivityResult`)

Cuando finalizamos la grabación se ejecuta el método `onActivityResult`, donde almacenamos en la variable `url1` la referencia al archivo de audio creado:

```

protected void onActivityResult (int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == petition) {
        url1 = data.getData();
    }
}

```

Por último para ejecutar el contenido de la grabación utilizamos la clase ya vista `MediaPlayer`:

```

public void reproducir(View v) {
    MediaPlayer mediaPlayer = MediaPlayer.create(this, url1);
    mediaPlayer.start();
}

```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto031.zip](#)

[Retornar](#)

29 - Captura de audio mediante la clase MediaRecorder

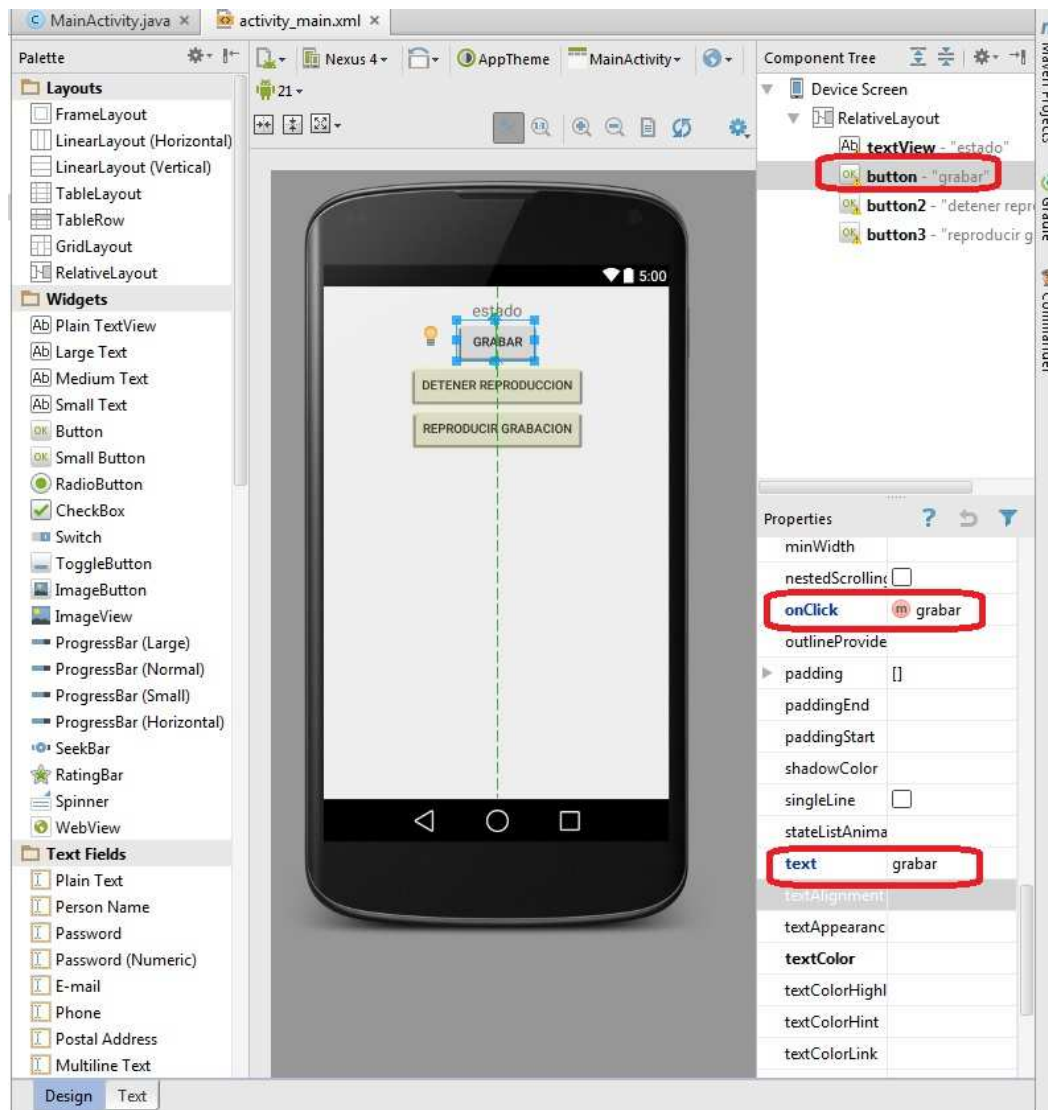
Otra forma de grabar audio en Android es el empleo de la clase MediaRecorder. Esta clase nos da más libertades a la hora de construir una aplicación que requiere grabar audio.

Problema:

Crear un proyecto (Proyecto032) Disponer tres objetos de la clase Button con las etiquetas "Grabar", "Detener Grabación" y "Reproducir Grabación". Disponer además un TextView para informar del estado actual.

Cuando se presione el botón "Grabar" permitir registrar todos los sonidos hasta que se presione el botón "Detener Grabación". Cuando se presione el botón "Reproducir Grabación" emitir el archivo de audio previamente generado.

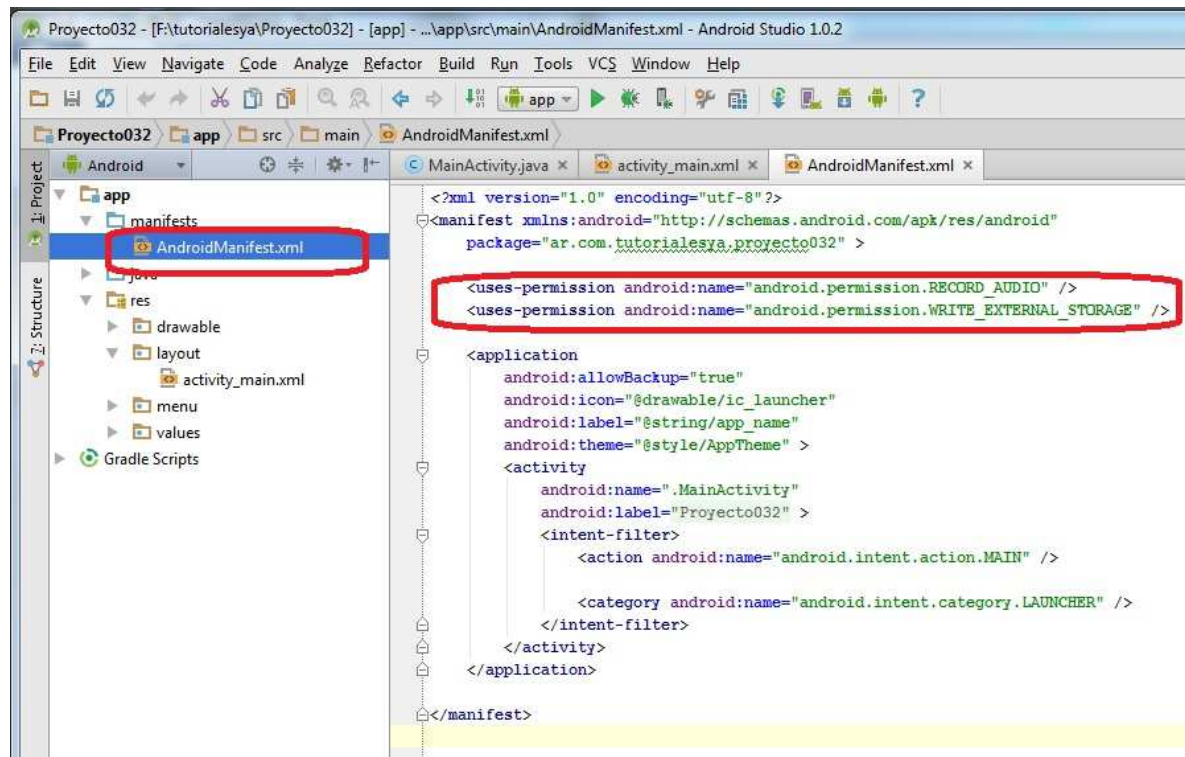
La interfaz visual a implementar es la siguiente:



Tener en cuenta de no olvidar definir los tres métodos para los tres botones: grabar, detener y reproducir en las propiedades onClick de cada botón.

También debemos modificar el archivo AndroidManifest.xml donde debemos indicar que nuestra aplicación accederá a la grabadora de sonido y a la tarjeta SD donde se almacenará el archivo de sonido.

Esto lo hacemos seleccionando el archivo AndroidManifest.xml y escribiendo los permisos respectivos:



El código fuente es:

```
package ar.com.tutorialesya.proyecto032;

import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Environment;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.io.File;
import java.io.IOException;

public class MainActivity extends ActionBarActivity implements MediaPlayer.OnCompletionListener {

    TextView tv1;
    MediaRecorder recorder;
    MediaPlayer player;
    File archivo;
    Button b1, b2, b3;

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

        tv1 = (TextView) this.findViewById(R.id.textView);
        b1 = (Button) findViewById(R.id.button);
        b2 = (Button) findViewById(R.id.button2);
        b3 = (Button) findViewById(R.id.button3);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

```

        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void grabar(View v) {
        recorder = new MediaRecorder();
        recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        File path = new File(Environment.getExternalStorageDirectory()
            .getPath());
        try {
            archivo = File.createTempFile("temporal", ".3gp", path);
        } catch (IOException e) {
        }
        recorder.setOutputFile(archivo.getAbsolutePath());
        try {
            recorder.prepare();
        } catch (IOException e) {
        }
        recorder.start();
        tv1.setText("Grabando");
        b1.setEnabled(false);
        b2.setEnabled(true);
    }

    public void detener(View v) {
        recorder.stop();
        recorder.release();
        player = new MediaPlayer();
        player.setOnCompletionListener(this);
        try {
            player.setDataSource(archivo.getAbsolutePath());
        } catch (IOException e) {
        }
        try {
            player.prepare();
        } catch (IOException e) {
        }
        b1.setEnabled(true);
        b2.setEnabled(false);
        b3.setEnabled(true);
        tv1.setText("Listo para reproducir");
    }

    public void reproducir(View v) {
        player.start();
        b1.setEnabled(false);
        b2.setEnabled(false);
        b3.setEnabled(false);
        tv1.setText("Reproduciendo");
    }

    public void onCompletion(MediaPlayer mp) {
        b1.setEnabled(true);
    }

```

```

        b2.setEnabled(true);
        b3.setEnabled(true);
        tv1.setText("Listo");
    }
}

```

Declaramos un objeto de la clase `MediaRecorder` para grabar audio:

```
MediaRecorder recorder;
```

Declaramos un objeto de la clase `MediaPlayer` para reproducir el archivo de sonido generado:

```
MediaPlayer player;
```

Declaramos un objeto de la clase `File` que hace referencia al archivo que se creará:

```
File archivo;
```

Declaramos las variables que harán referencia a los tres botones y al `TextView`:

```
TextView tv1;
Button b1,b2,b3;
```

En el método `onCreate` obtenemos la referencia de los cuatro objetos creados en el archivo XML:

```

tv1 = (TextView) this.findViewById(R.id.textView);
b1 = (Button) findViewById(R.id.button);
b2 = (Button) findViewById(R.id.button2);
b3 = (Button) findViewById(R.id.button3);

```

El método más importante de este concepto es el grabar:

```

public void grabar(View v) {
    recorder = new MediaRecorder();
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
    File path = new File(Environment.getExternalStorageDirectory().getPath());
    try {
        archivo = File.createTempFile("temporal", ".3gp", path);
    } catch (IOException e) {
    }
    recorder.setOutputFile(archivo.getAbsolutePath());
    try {
        recorder.prepare();
    } catch (IOException e) {
    }
    recorder.start();
    tv1.setText("Grabando");
    b1.setEnabled(false);
    b2.setEnabled(true);
}

```

Creamos un objeto de la clase `MediaRecorder`:

```
recorder = new MediaRecorder();
```

Seguidamente definimos el micrófono como fuente de audio:

```
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

Luego llamamos al método `setOutputFormat` especificando que el archivo será almacenado con la especificación 3GPP y con extensión `.3gp`

```
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
```

Especificamos el codec a emplear llamando al método `setAudioEncoder`:

```
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
```

Obtenemos el path de la tarjeta SD y creamos un archivo temporal con extensión `3gp`:

```

File path = new File(Environment.getExternalStorageDirectory().getPath());
try {
    archivo = File.createTempFile("temporal", ".3gp", path);
} catch (IOException e) {
}

```

Con el método `setOutputFile` de la clase `MediaRecorder` le indicamos el archivo donde debe almacenarse la grabación:

```

recorder.setOutputFile(archivo.getAbsolutePath());

```

Llamamos al método `prepare` y finalmente al método `start` para comenzar la grabación:

```

try {
    recorder.prepare();
} catch (IOException e) {
}
recorder.start();

```

El método `detener`:

```

public void detener(View v) {
    recorder.stop();
    recorder.release();
    player = new MediaPlayer();
    player.setOnCompletionListener(this);
    try {
        player.setDataSource(archivo.getAbsolutePath());
    } catch (IOException e) {
    }
    try {
        player.prepare();
    } catch (IOException e) {
    }
    b1.setEnabled(true);
    b2.setEnabled(false);
    b3.setEnabled(true);
    tv1.setText("Listo para reproducir");
}

```

Llamamos primero al método `stop` de la clase `MediaRecorder` y liberamos los recursos consumidos llamando a `release`:

```

recorder.stop();
recorder.release();

```

Creamos un objeto de la clase `MediaPlayer` para poder reproducir el archivo de audio que acabamos de grabar. Indicamos mediante el método `setOnCompletionListener` la referencia de la clase que será informada cuando el audio finalice:

```

player = new MediaPlayer();
player.setOnCompletionListener(this);

```

Referenciamos el archivo a que debe reproducir:

```

try {
    player.setDataSource(archivo.getAbsolutePath());
} catch (IOException e) {
}

```

Finalmente llamamos al método `prepare` de la clase `MediaPlayer`:

```

try {
    player.prepare();
} catch (IOException e) {
}

```

El método `reproducir` simplemente llama al método `start` de la clase `MediaPlayer` para iniciar la reproducción del archivo previamente grabado:

```

public void reproducir(View v) {
    player.start();
    b1.setEnabled(false);
    b2.setEnabled(false);
    b3.setEnabled(false);
    tv1.setText("Reproduciendo");
}

```


El método onCompletion se ejecuta cuando termina de reproducirse el archivo de audio:

```
public void onCompletion(MediaPlayer mp) {  
    b1.setEnabled(true);  
    b2.setEnabled(true);  
    b3.setEnabled(true);  
    tv1.setText("Listo");  
}
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto032.zip](#)

[Retornar](#)

30 - Dibujar: graficar un píxel

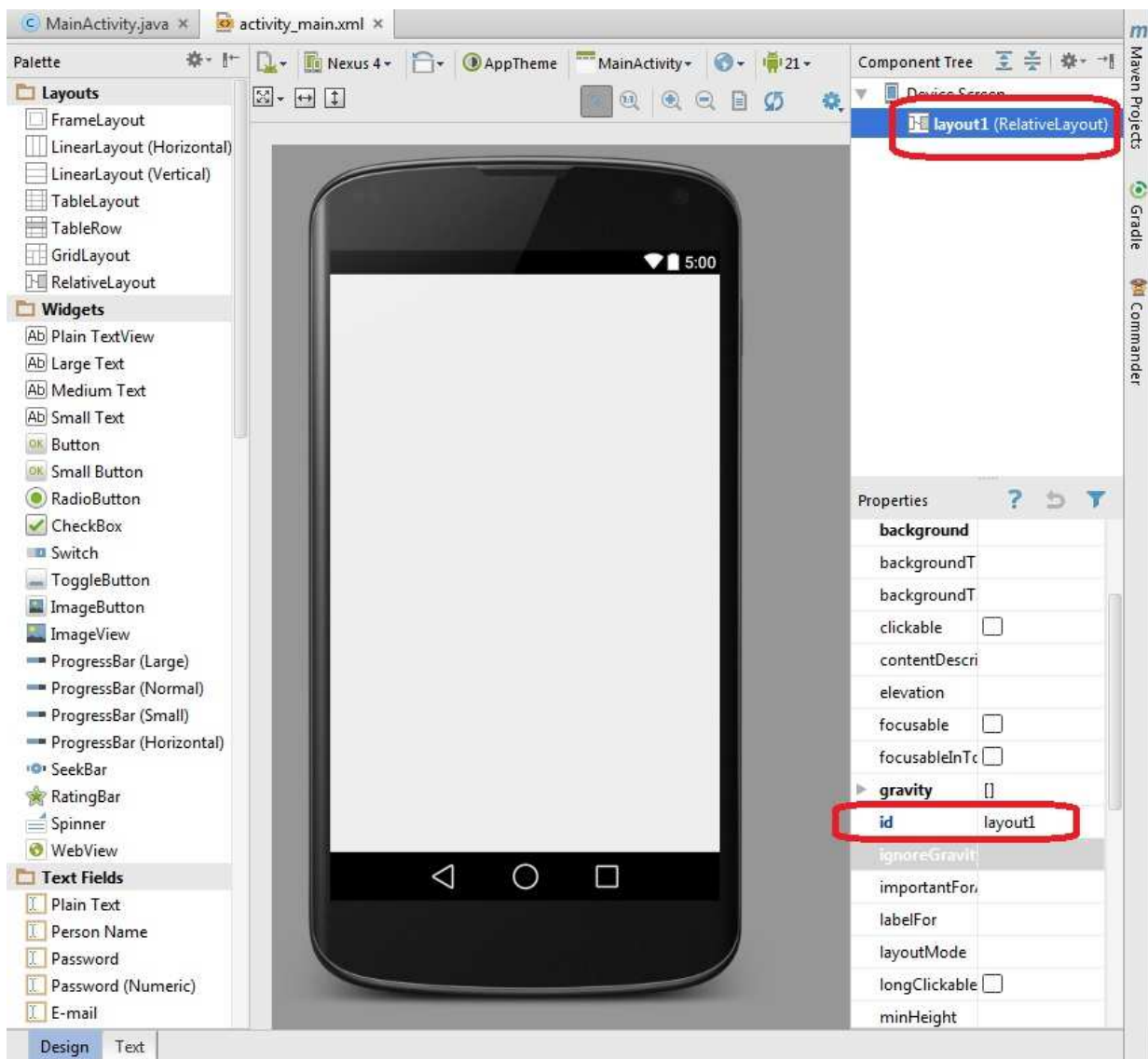
Problema:

Graficar un píxel de color negro en medio de la pantalla del dispositivo.

Para poder hacer esta simple tarea debemos seguir una serie de pasos:

1 - Creamos un proyecto llamado: Proyecto033

Borramos el TextView que agrega automáticamente el plug-in del Android Studio y procedemos a definir un id para el RelativeLayout (le asignamos como id el valor @layout1):



Ahora codificamos la clase donde se encuentra toda la lógica para encender el píxel:

```
package ar.com.tutorialesya.proyecto033;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```

import android.view.View;
import android.widget.RelativeLayout;

public class MainActivity extends ActionBarActivity {

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

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            int ancho = canvas.getWidth();
            int alto = canvas.getHeight();
            Paint pincel1 = new Paint();
            pincel1.setARGB(255, 0, 0, 0);
            canvas.drawPoint(ancho / 2, alto / 2, pincel1);
        }
    }
}

```

Debemos crear una clase que herede de la clase View (todos los controles visuales en Android heredan de esta clase) y sobrescribir el método onDraw. El método onDraw se ejecuta cuando tiene que graficarse. Para acceder a las primitivas gráficas hay una clase llamada Canvas que encapsula todo lo relacionado a pintar píxeles, líneas, rectángulos etc.:

```

protected void onDraw(Canvas canvas) {
    int ancho=canvas.getWidth();

```

```

        int alto=canvas.getHeight();
        Paint pincel1=new Paint();
        pincel1.setARGB(255,0,0,0);
        canvas.drawPoint(ancho/2, alto/2, pincel1);
    }

```

Lo primero que hacemos en el método onDraw es obtener el ancho y alto en píxeles del dispositivo mediante los métodos getWidth() y getHeight() que nos provee la clase Canvas. Seguidamente creamos un objeto de la clase Paint. Llamamos al método setARGB para definir el color del pincel (el primer parámetro indica el valor de transparencia con 255 indicamos que no hay transparencia, luego indicamos la cantidad de rojo, verde y azul.

Por último debemos llamar al método drawPoint que dibuja el píxel en la columna, fila y pincel que le pasamos como parámetros.

Como podemos observar la clase Lienzo hereda de View e implementa el constructor donde llegará la referencia del Activity donde se inserta:

```

class Lienzo extends View {

    public Lienzo(Context context) {
        super(context);
    }

    protected void onDraw(Canvas canvas) {
        int ancho=canvas.getWidth();
        int alto=canvas.getHeight();
        Paint pincel1=new Paint();
        pincel1.setARGB(255,0,0,0);
        canvas.drawPoint(ancho/2, alto/2, pincel1);
    }
}

```

Por último en el método onCreate del Activity obtenemos la referencia del RelativeLayout que tiene el Activity. Creamos un objeto de la clase Lienzo (llamado fondo) y agregamos el objeto fondo al RelativeLayout llamando al método addView:

```

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

    RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
    Lienzo fondo = new Lienzo(this);
    layout1.addView(fondo);
}

```

Cuando ejecutemos el programa veremos un píxel en medio de la pantalla de color negro.

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto033.zip](#)

[Retornar](#)

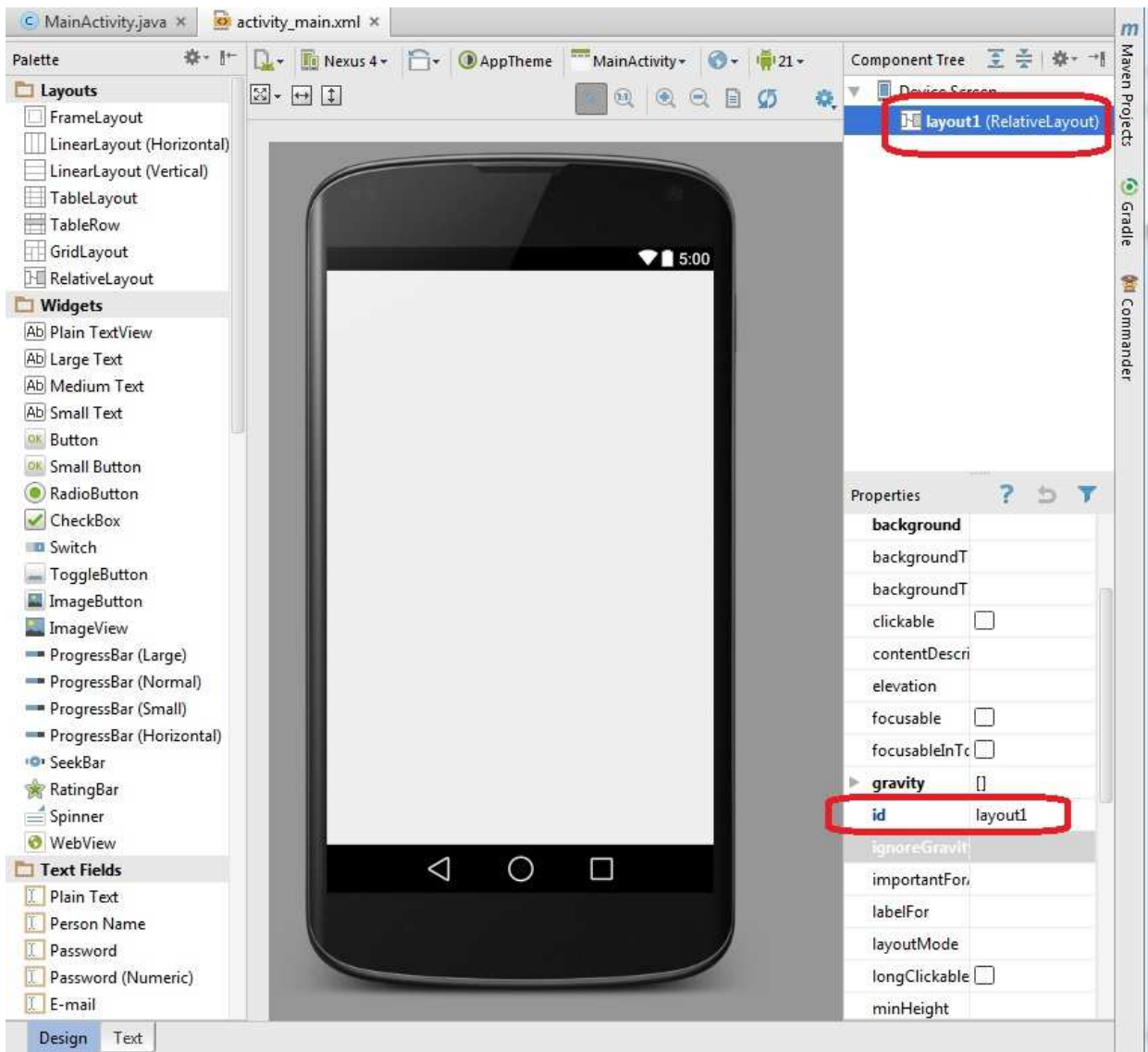
31 - Dibujar: pintar fondo y dibujar líneas

Problema:

Pintar el fondo de color amarillo y dibujar una serie de líneas con distintos estilos.

1 - Creamos un proyecto llamado: proyecto034

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica para pintar el fondo y dibujar las líneas:

```
package ar.com.tutorialesya.proyecto034;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;
```

```

public class MainActivity extends ActionBarActivity {

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

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255, 255, 0);
            int ancho = canvas.getWidth();
            Paint pincel1 = new Paint();
            pincel1.setARGB(255, 255, 0, 0);
            canvas.drawLine(0, 30, ancho, 30, pincel1);
            pincel1.setStrokeWidth(4);
            canvas.drawLine(0, 60, ancho, 60, pincel1);
        }

    }

}

```

Veamos el método onDraw donde pintamos el fondo de la componente llamando al método drawRGB donde indicamos la cantidad de rojo, verde a azul:

```

protected void onDraw(Canvas canvas) {
    canvas.drawRGB(255,255,0);

```

Creamos un objeto de la clase paint y definimos el color rojo (recordemos que el primer parámetro indica el valor de la

transparencia, si vale 255 es totalmente opaco, con un valor menor el trazo de la línea tendrá transparencia.

El método para graficar una línea se llama `drawLine` y tiene como parámetros la columna y fila del punto inicial y el tercer y cuarto parámetro indica la columna y fila del punto final de la línea (en este ejemplo se dibujará una línea horizontal en la fila 30 y tendrá un ancho que coincide con el ancho del dispositivo), el último parámetro es el objeto de la clase `Paint` que indica el color de la línea:

```
Paint pincel1=new Paint();
pincel1.setARGB(255,255,0,0);
canvas.drawLine(0, 30, ancho, 30, pincel1);
```

La siguiente línea la dibujamos en la fila 60 pero previamente cambiamos el grosor del pincel llamando al método `setStrokeWidth` indicando que serán 4 píxeles el grosor:

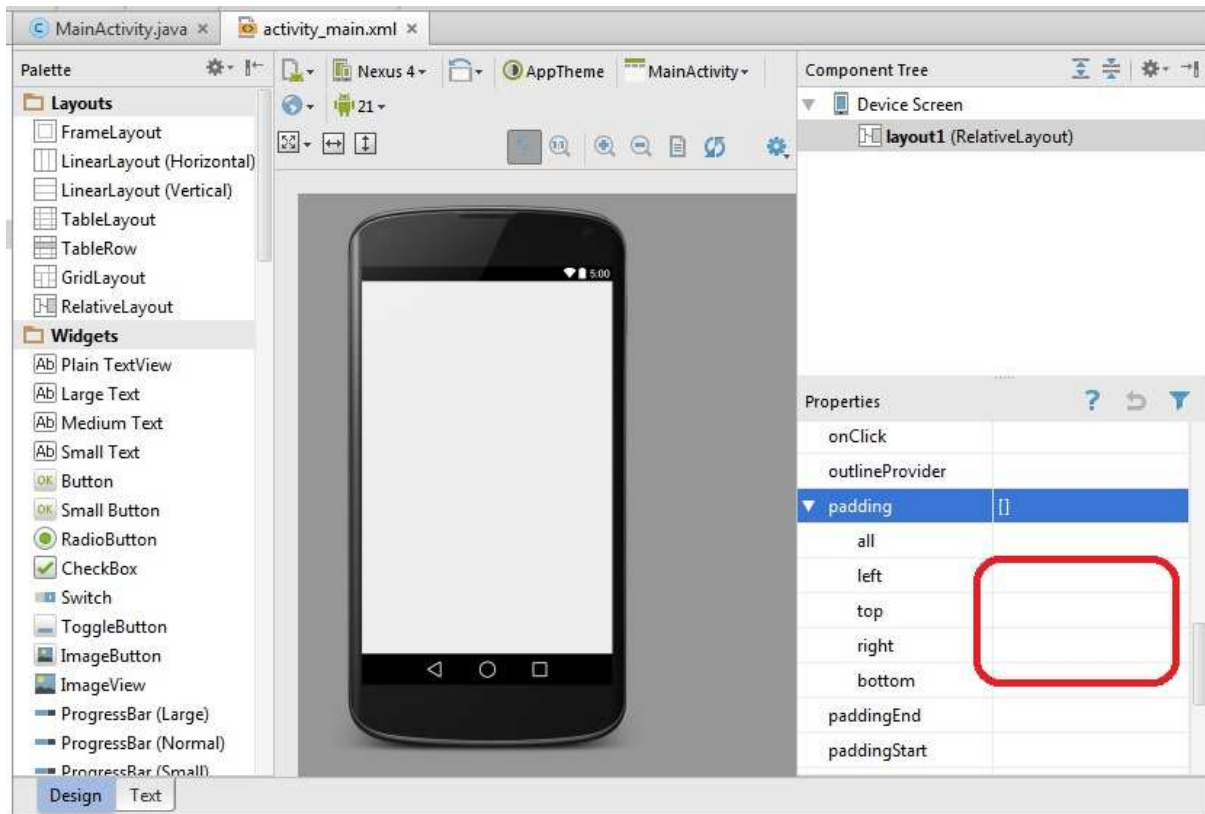
```
pincel1.setStrokeWidth(4);
canvas.drawLine(0, 60, ancho, 60, pincel1);
```

La vista previa en el dispositivo será:



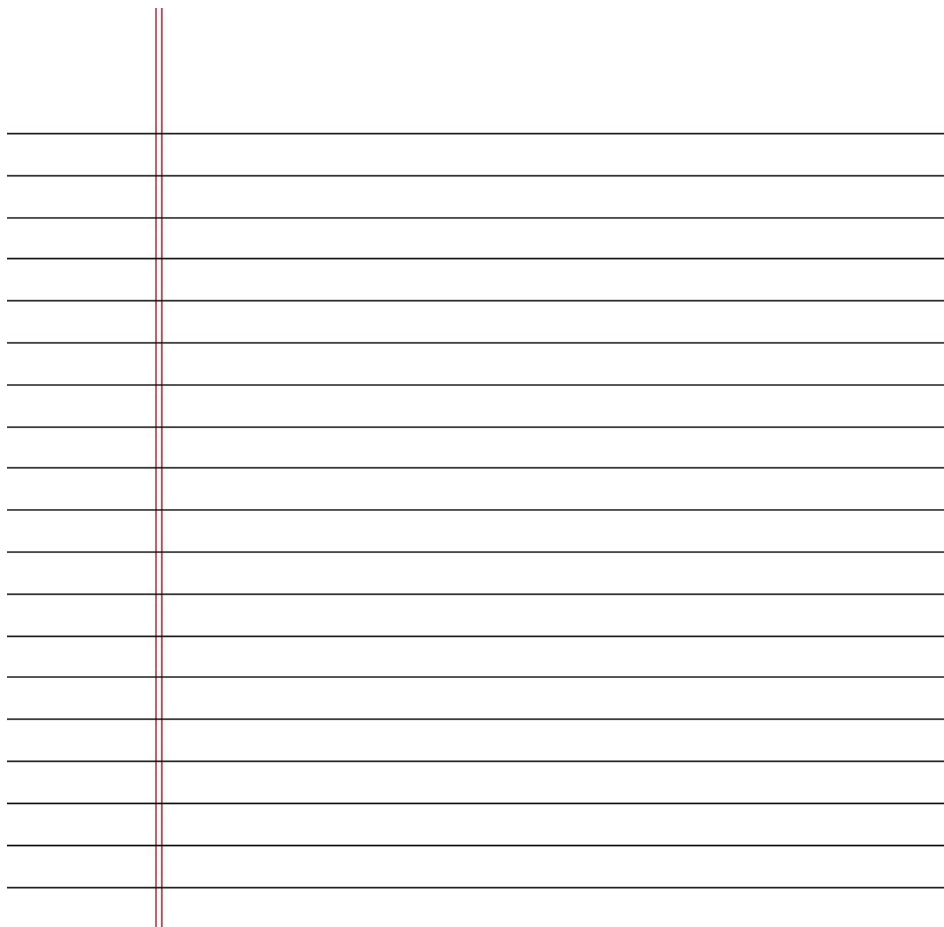
Este proyecto lo puede descargar en un zip desde este enlace: [proyecto034.zip](#)

Como vemos el objeto fondo que creamos en el método `onCreate` del `ActionBarActivity` ocupa todo el espacio del `RelativeLayout`, si queremos que ocupe todo el espacio de la pantalla debemos modificar la propiedad `padding` del `RelativeLayout` borrando el `Padding left, top, right y bottom`:



Problema propuesto

1. Confeccionar una aplicación que muestre una hoja en la pantalla similar a esta:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto035.zip](#)

[Retornar](#)

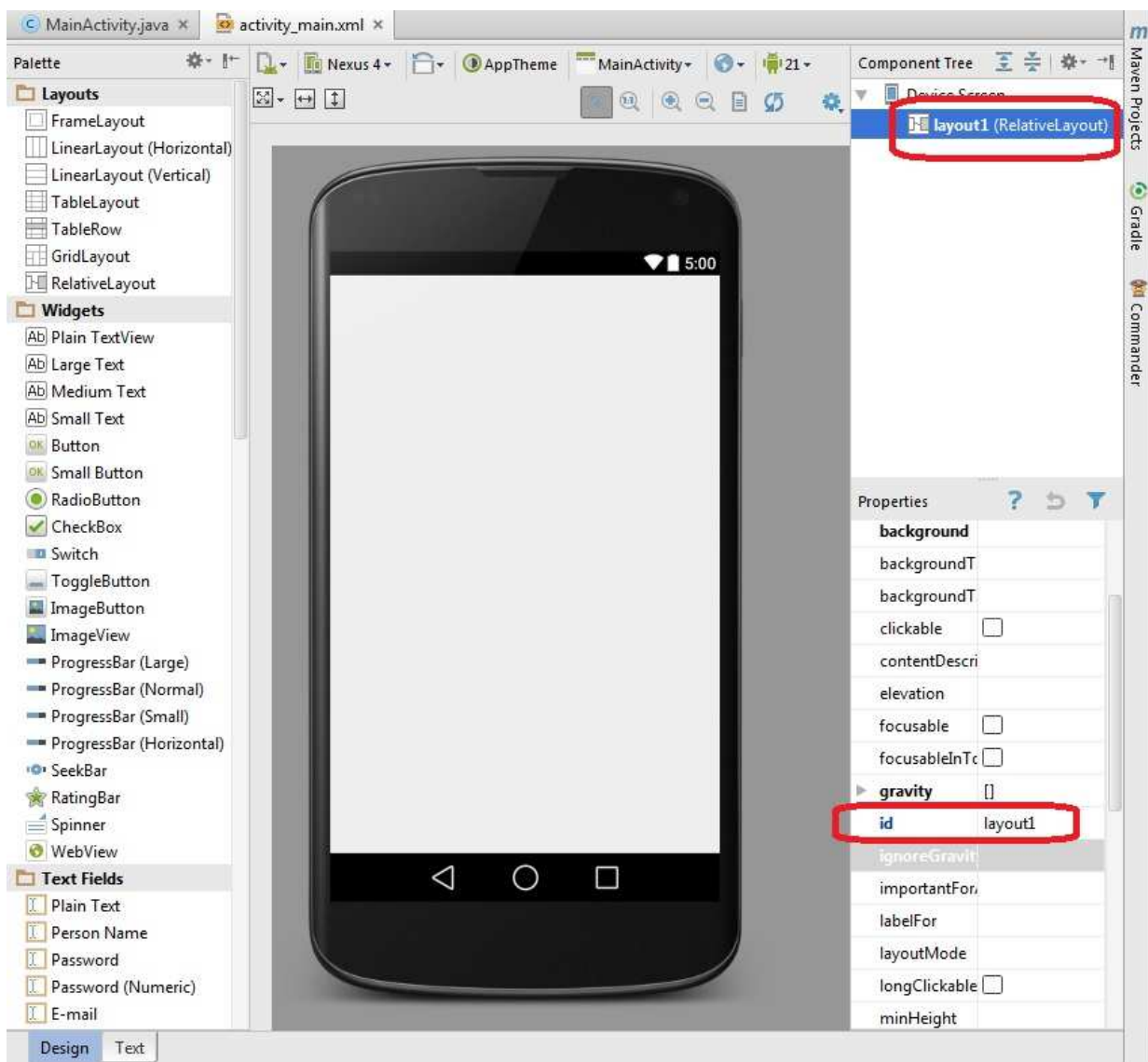
32 - Dibujar: rectángulos

Problema:

Pintar el fondo de color blanco y dibujar una serie de rectángulos con distintos estilos.

1 - Creamos un proyecto llamado: Proyecto036

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica:

```
package ar.com.tutorialesya.proyecto036;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;
```

```

public class MainActivity extends ActionBarActivity {

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

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255,255,255);
            int ancho=canvas.getWidth();
            Paint pincell=new Paint();

            pincell.setARGB(255,255,0,0);
            canvas.drawRect(10,10,ancho-10,40,pincell);

            pincell.setStyle(Paint.Style.STROKE);
            canvas.drawRect(10,60,ancho-10,90,pincell);

            pincell.setStrokeWidth(3);
            canvas.drawRect(10,110,ancho-10,140,pincell);

        }

    }

}

```

En el método onDraw de la clase Lienzo procedemos a pintar el fondo de color blanco:

```
canvas.drawRGB(255,255,255);
```

Obtenemos el ancho del dispositivo:

```
int ancho=canvas.getWidth();
```

Creamos un objeto de la clase Paint:

```
Paint pincel1=new Paint();
```

Activamos el color rojo:

```
pincel1.setARGB(255,255,0,0);
```

Dibujamos un rectángulo desde la coordenada columna:10 y fila 10 hasta la columna que coincide con el ancho de la pantalla menos 10 píxeles y la fila 40. Además le pasamos el pincel a utilizar:

```
canvas.drawRect(10,10,ancho-10,40,pincel1);
```

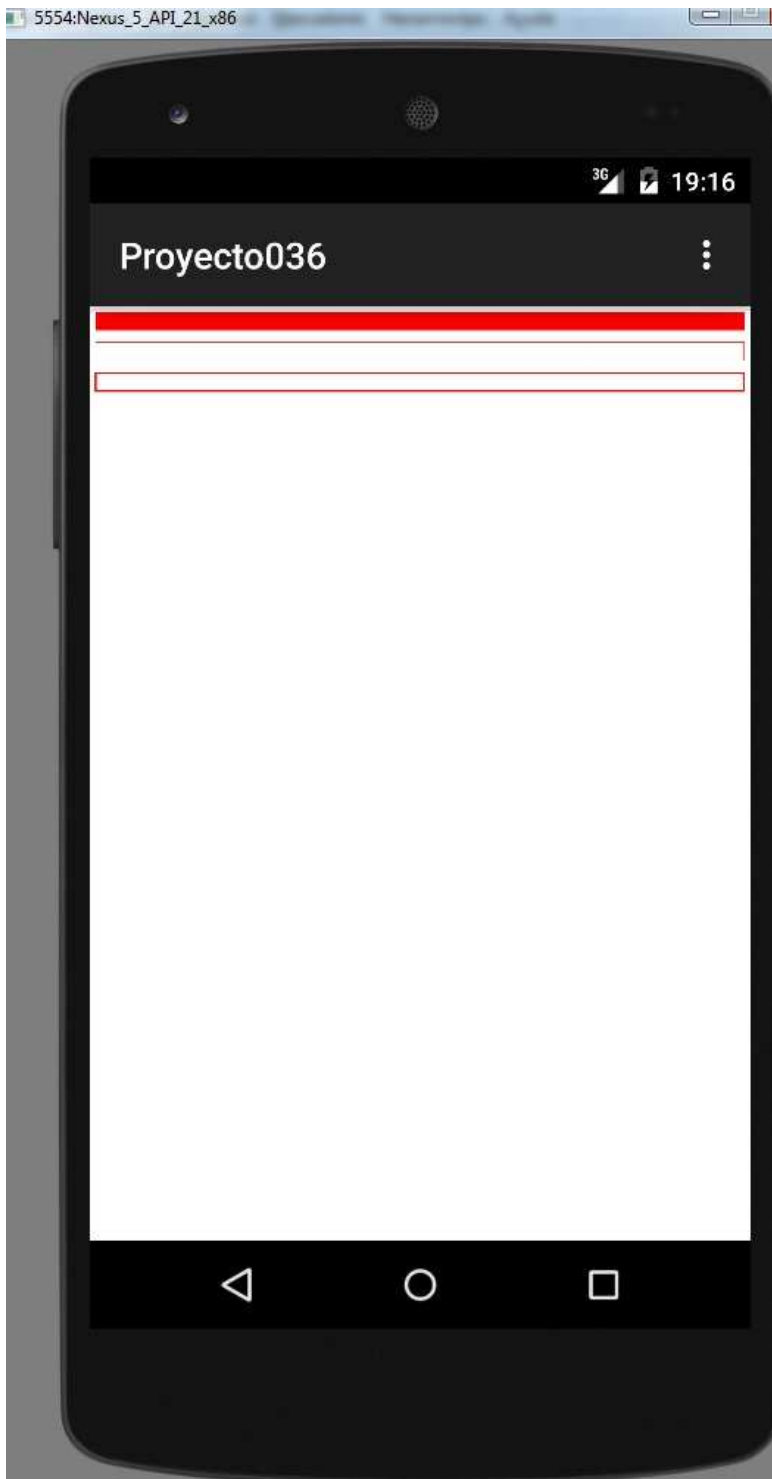
Para el siguiente rectángulo configuramos el pincel para que solo pinte el perímetro llamando al método `setStyle` y pasando la constante `STROKE`:

```
pincel1.setStyle(Style.STROKE);  
canvas.drawRect(10,60,ancho-10,90,pincel1);
```

Por último dibujamos otro rectángulo que solo se pinta el perímetro pero cambiamos el grosor del lápiz llamando al método `setStrokeWidth`:

```
pincel1.setStrokeWidth(3);  
canvas.drawRect(10,110,ancho-10,140,pincel1);
```

La vista previa de la aplicación es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto036.zip](#)

[Retornar](#)

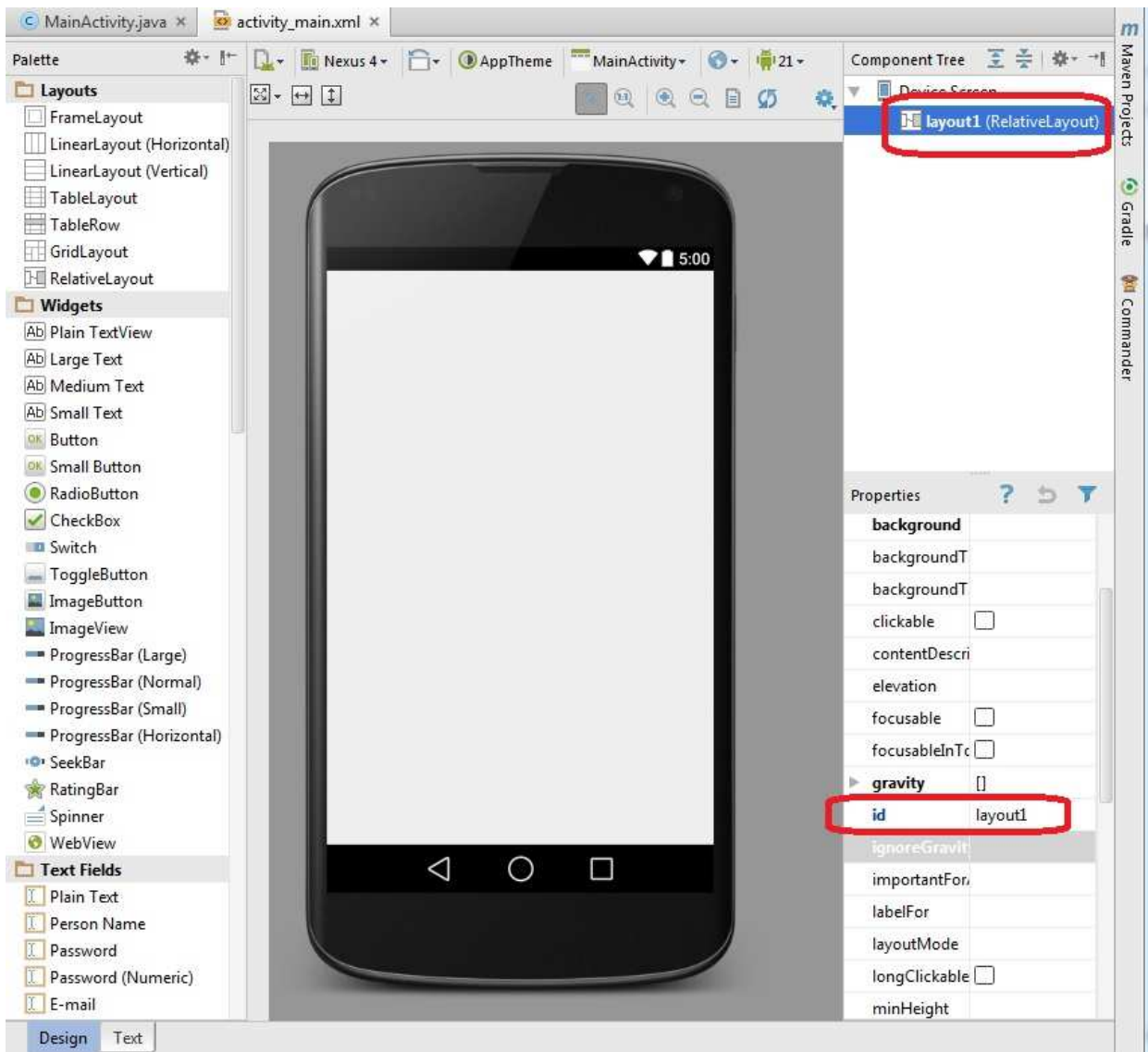
33 - Dibujar: círculos

Problema:

Pintar el fondo de color blanco y dibujar 10 círculos crecientes desde el medio de la pantalla.

1 - Creamos un proyecto llamado: Proyecto037

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica:

```
package ar.com.tutorialesya.proyecto037;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;
```



```

public class MainActivity extends ActionBarActivity {

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

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255, 255, 255);
            int ancho = canvas.getWidth();
            int alto = canvas.getHeight();
            Paint pincell = new Paint();
            pincell.setARGB(255, 255, 0, 0);
            pincell.setStyle(Paint.Style.STROKE);
            for (int f = 0; f < 10; f++) {
                canvas.drawCircle(ancho / 2, alto / 2, f * 15, pincell);
            }
        }
    }
}

```

Pintamos el fondo de blanco y obtenemos el ancho y alto del control:

```

protected void onDraw(Canvas canvas) {
    canvas.drawRGB(255,255,255);
    int ancho=canvas.getWidth();
    int alto=canvas.getHeight();
}

```

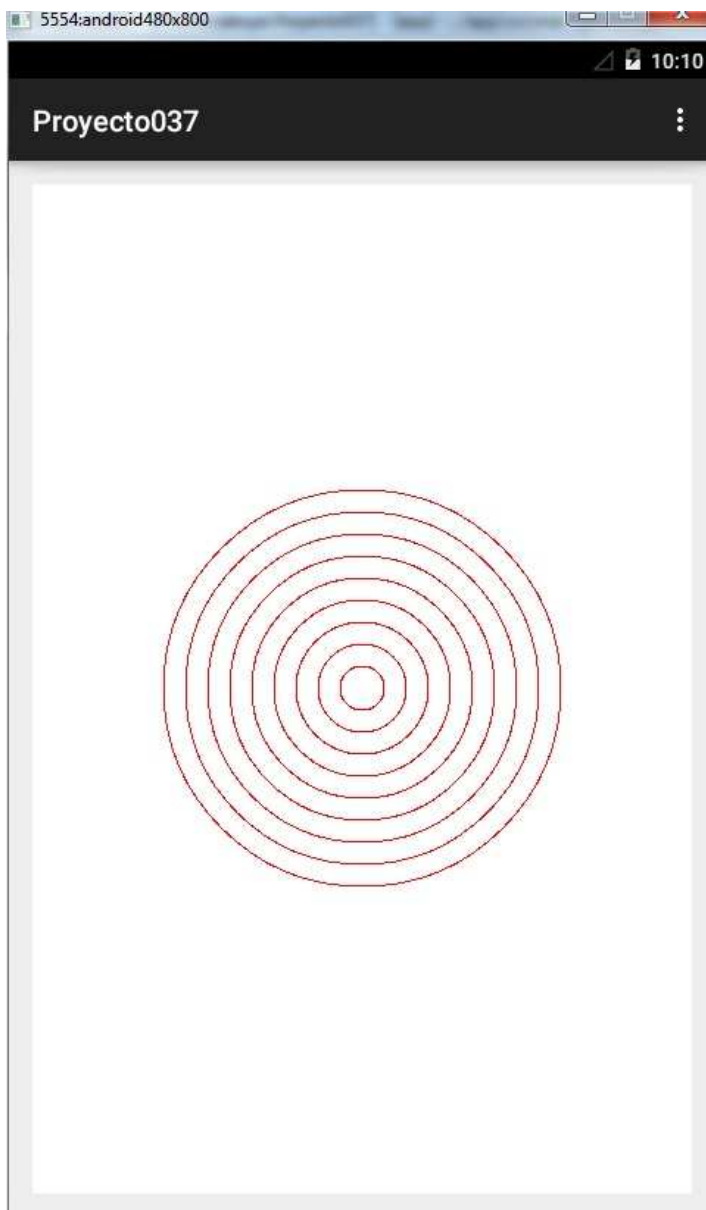
Creamos un objeto de la clase Paint, fijamos el color rojo y mediante setStyle indicamos que solo se debe pintar el perímetro:

```
Paint pincel1=new Paint();
pincel1.setARGB(255, 255, 0, 0);
pincel1.setStyle(Style.STROKE);
```

Disponemos un for para dibujar los 10 círculos concéntricos (indicamos en los dos primeros parámetros el punto central del círculo y en el tercer parámetro el radio del círculo:

```
for(int f=0;f<10;f++) {
    canvas.drawCircle(ancho/2, alto/2, f*15, pincel1);
}
```

La vista previa de la aplicación es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto037.zip](#)

[Retornar](#)

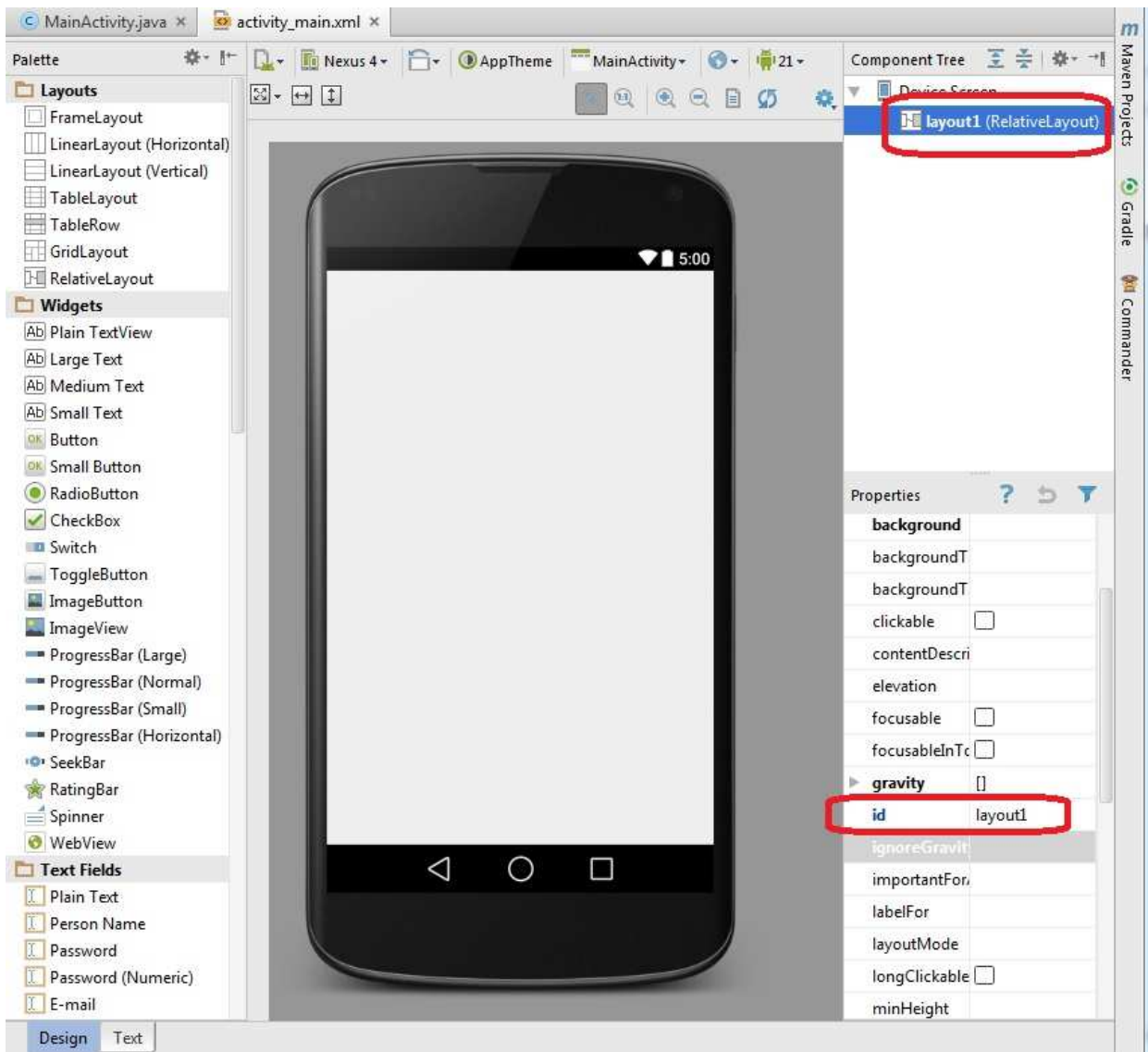
34 - Dibujar: óvalos

Problema:

Dibujar un óvalo que ocupe toda la pantalla y un círculo en su interior.

1 - Creamos un proyecto llamado: proyecto038

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica:

```
package ar.com.tutorialesya.proyecto038;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
```

```

import android.view.Window;
import android.view.WindowManager;
import android.widget.RelativeLayout;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);
        getSupportActionBar().hide();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255, 255, 255);
            int ancho = canvas.getWidth();
            int alto = canvas.getHeight();

            Paint pincel1 = new Paint();
            pincel1.setARGB(255, 0, 0, 0);
            pincel1.setStrokeWidth(5);
            pincel1.setStyle(Paint.Style.STROKE);
            RectF rectangulo1 = new RectF(0, 0, ancho, alto);
            canvas.drawOval(rectangulo1, pincel1);

            int menor;
            if (ancho < alto)
                menor = ancho;

```

```

        else
            menor = alto;

        pincel1.setStyle(Paint.Style.FILL);
        pincel1.setARGB(255, 255, 255, 0);
        canvas.drawCircle(ancho / 2, alto / 2, menor / 2, pincel1);
    }
}

```

Para ocultar la barra del título y la barra superior debemos hacer lo siguiente en el método onCreate:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
    Lienzo fondo = new Lienzo(this);
    layout1.addView(fondo);
    getSupportActionBar().hide();
}

```

En el método onDraw pintamos el fondo de blanco y obtenemos el ancho y alto de la pantalla:

```

protected void onDraw(Canvas canvas) {
    canvas.drawRGB(255,255,255);
    int ancho=canvas.getWidth();
    int alto=canvas.getHeight();
}

```

Seguidamente creamos el objeto de la clase Paint y configuramos el color, grosor de línea y estilo:

```

Paint pincel1=new Paint();
pincel1.setARGB(255,0,0,0);
pincel1.setStrokeWidth(5);
pincel1.setStyle(Style.STROKE);

```

Creamos un objeto de la clase RectF pasando como dato la coordenada superior izquierda del óvalo y la coordenada inferior derecha del mismo (teniendo en cuenta que el óvalo se dibujará en ese rectángulo imaginario que indicamos con dichas coordenadas):

```

RectF rectangulo1=new RectF(0,0,ancho,alto);
canvas.drawOval(rectangulo1, pincel1);

```

Obtenemos el valor menor del ancho y alto del dispositivo:

```

int menor;
if (ancho<alto)
    menor=ancho;
else
    menor=alto;

```

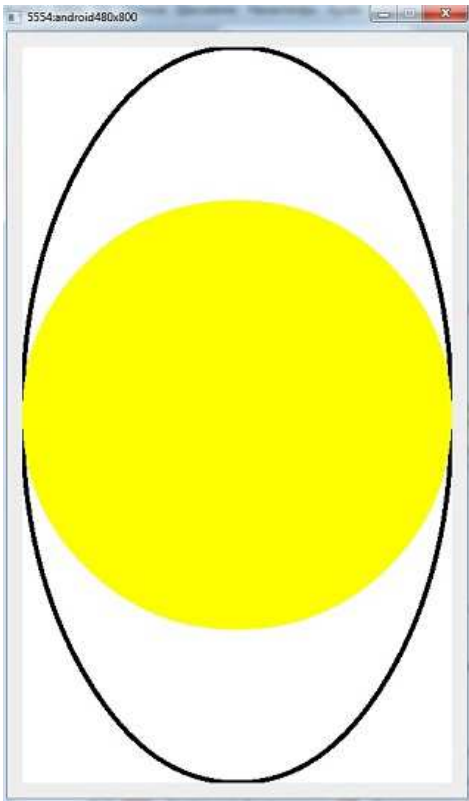
Configuramos ahora el pincel para dibujar el círculo y lo dibujamos:

```

pincel1.setStyle(Style.FILL);
pincel1.setARGB(255, 255, 255, 0);
canvas.drawCircle(ancho/2, alto/2, menor/2, pincel1);
}

```

La vista previa de la aplicación es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto038.zip](#)

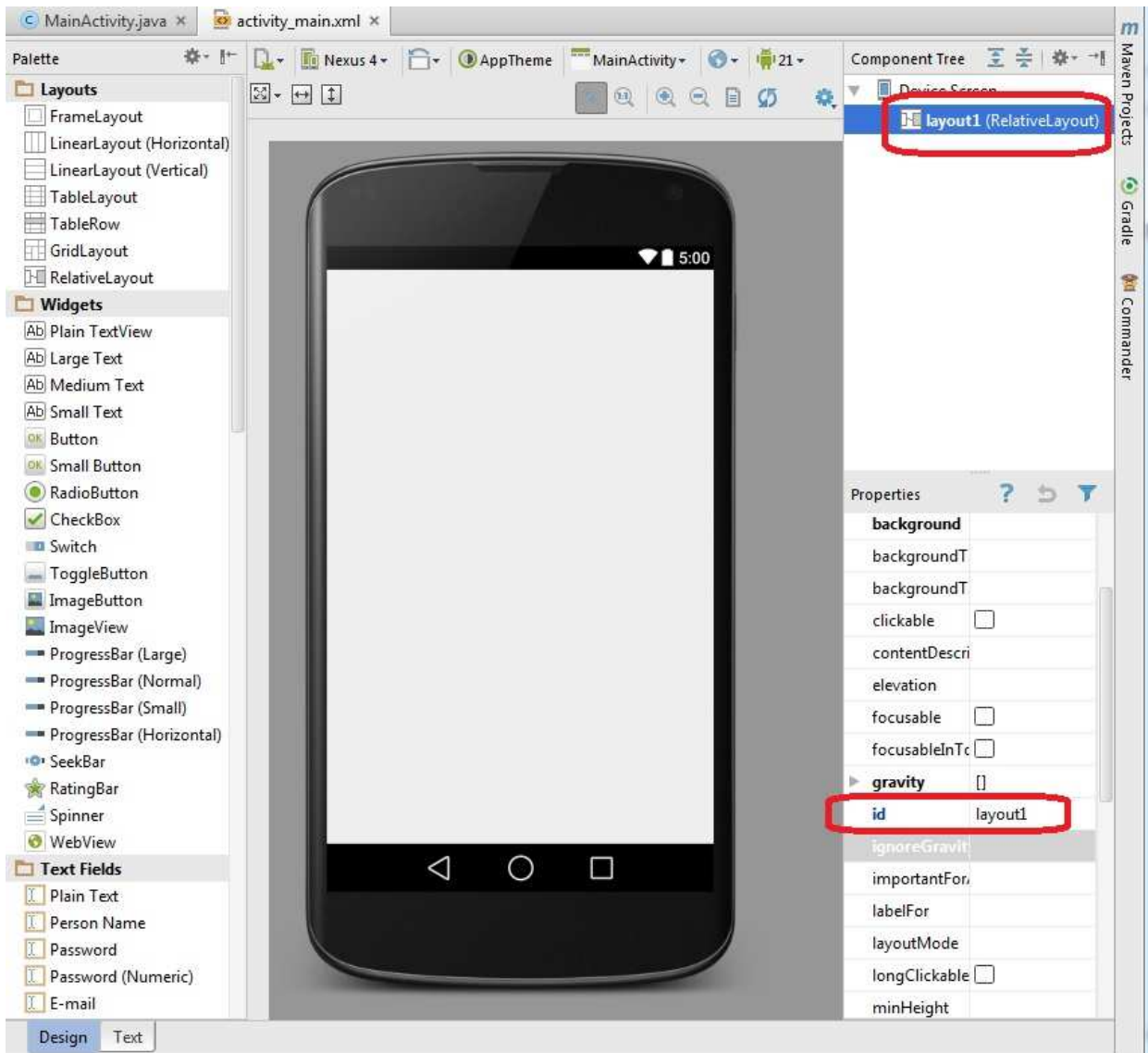
[Retornar](#)

35 - Dibujar: texto

Otra recurso que nos permite la clase Canvas es el de graficar texto.

1 - Creamos un proyecto llamado: Proyecto040

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica:

```
package ar.com.tutorialesya.proyecto039;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Typeface;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.RelativeLayout;
```



```

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);
        getSupportActionBar().hide();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(0, 0, 255);
            Paint pincell = new Paint();
            pincell.setARGB(255, 255, 0, 0);
            pincell.setTextSize(30);
            pincell.setTypeface(Typeface.SERIF);
            canvas.drawText("Hola Mundo (SERIF)", 0, 70, pincell);
            pincell.setTypeface(Typeface.SANS_SERIF);
            canvas.drawText("Hola Mundo (SANS_SERIF)", 0, 100, pincell);
            pincell.setTypeface(Typeface.MONOSPACE);
            canvas.drawText("Hola Mundo (MONOSPACE)", 0, 140, pincell);
            Typeface tf = Typeface.create(Typeface.SERIF, Typeface.ITALIC);
            pincell.setTypeface(tf);
            canvas.drawText("Hola Mundo (SERIF ITALIC)", 0, 180, pincell);
            tf = Typeface.create(Typeface.SERIF, Typeface.ITALIC
                | Typeface.BOLD);
            pincell.setTypeface(tf);
        }
    }
}

```

```

        canvas.drawText("Hola Mundo (SERIF ITALIC BOLD)", 0, 220, pincel1);
    }
}

```

Para graficar texto disponemos del método `drawText` que nos permite imprimir un `String` en una determinada columna, fila con un determinado pincel que podemos definir su color:

```

Paint pincel1=new Paint();
pincel1.setARGB(255,255,0,0);

```

El tamaño de la letra:

```

pincel1.setTextSize(30);

```

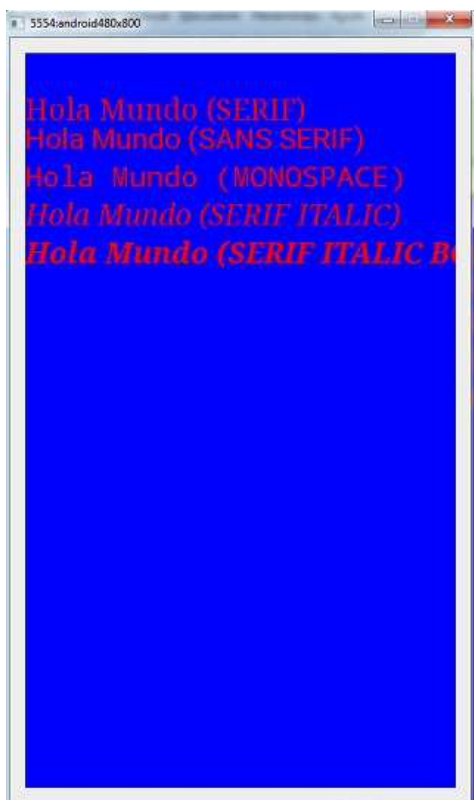
El estilo de la letra:

```

pincel1.setTypeface(Typeface.SERIF);

```

La interfaz del programa es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto039.zip](#)

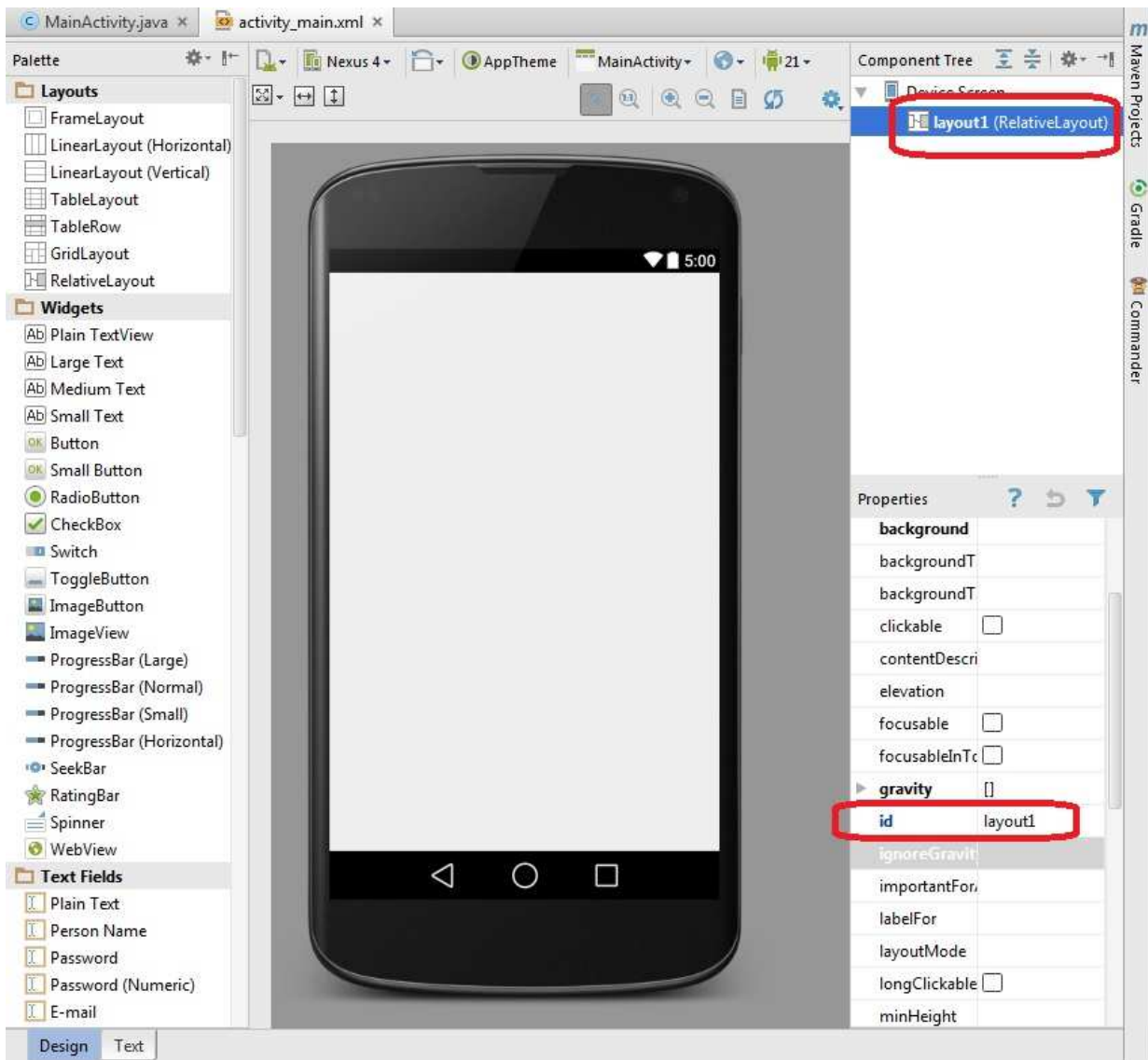
[Retornar](#)

36 - Dibujar: texto con fuentes externas

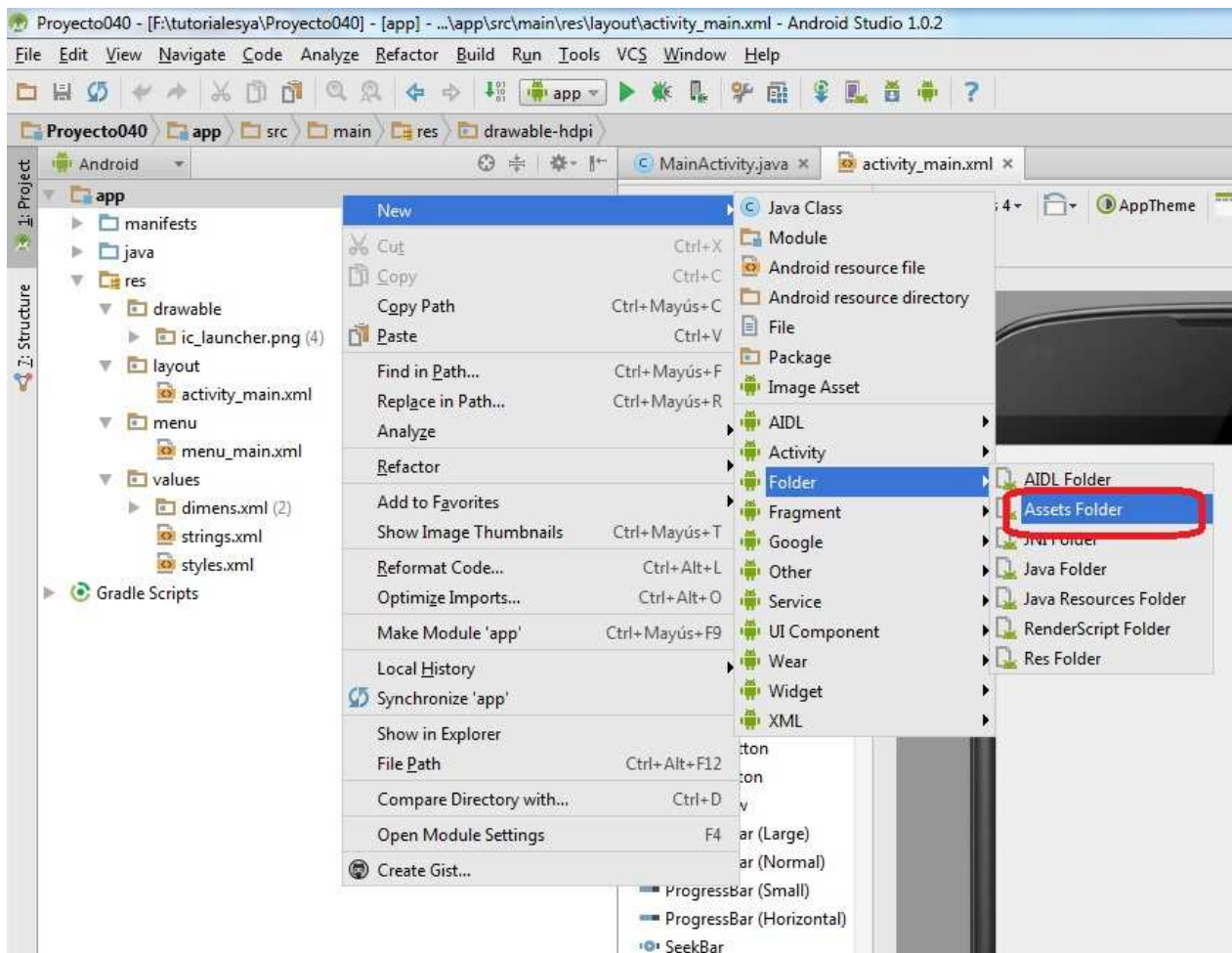
Podemos confeccionar aplicaciones e incorporar fuentes True Type externas. Para ello implementaremos una aplicación que muestre un texto con una fuente externa.

1 - Creamos un proyecto llamado: proyecto041

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:

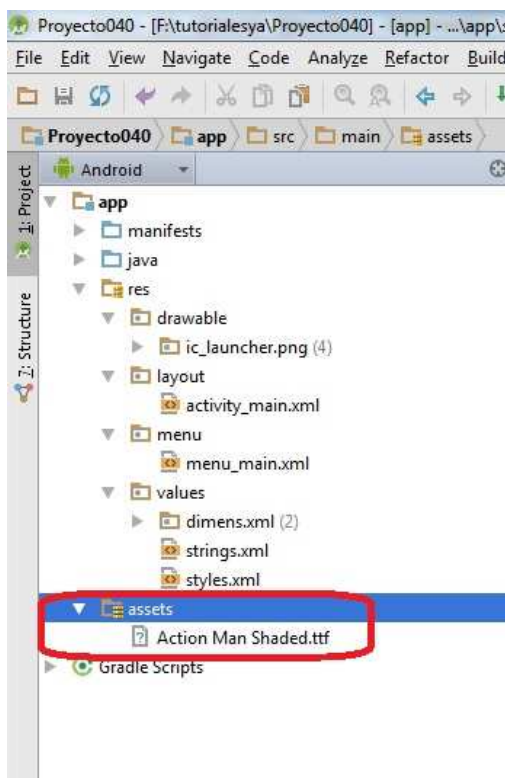


Primero tenemos que crear una carpeta donde almacenaremos nuestras fuentes externas. Para esto presionamos el botón derecho del mouse sobre app y seleccionamos New->Folder->Assets Folder:



Aparece un diálogo y presionamos Finish. Ya tenemos creada la carpeta Assets donde podemos almacenar las fuentes.

Procedemos ahora a descargar una fuente del sitio [Creamundo](#) y copiamos el archivo de la fuente a la carpeta assets como se muestra:



Ahora codificamos la clase donde se encuentra toda la lógica:

```

package ar.com.tutorialesya.proyecto040;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Typeface;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.RelativeLayout;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);
        getSupportActionBar().hide();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255, 255, 255);
            Paint pincell = new Paint();

```

```

        pincell.setARGB(255, 0, 0, 0);
        pincell.setTextSize(30);
        Typeface face = Typeface.createFromAsset(getAssets(),
            "Action Man Shaded.ttf");
        pincell.setTypeface(face);
        canvas.drawText("Hola Mundo", 0, 120, pincell);
    }
}
}

```

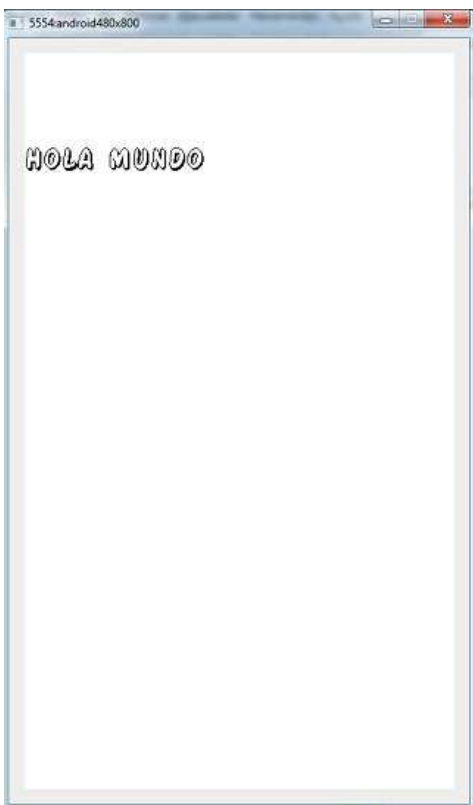
En el método onDraw procedemos a crear una fuente llamando al método createFromAsset e indicando el nombre del archivo que descargamos de internet (disponga en el segundo parámetro el nombre del archivo que eligió y descargó de internet):

```

Typeface face = Typeface.createFromAsset(getAssets(), "Action Man Shaded.ttf");
pincell.setTypeface(face);

```

El resultado en pantalla es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto040.zip](#)

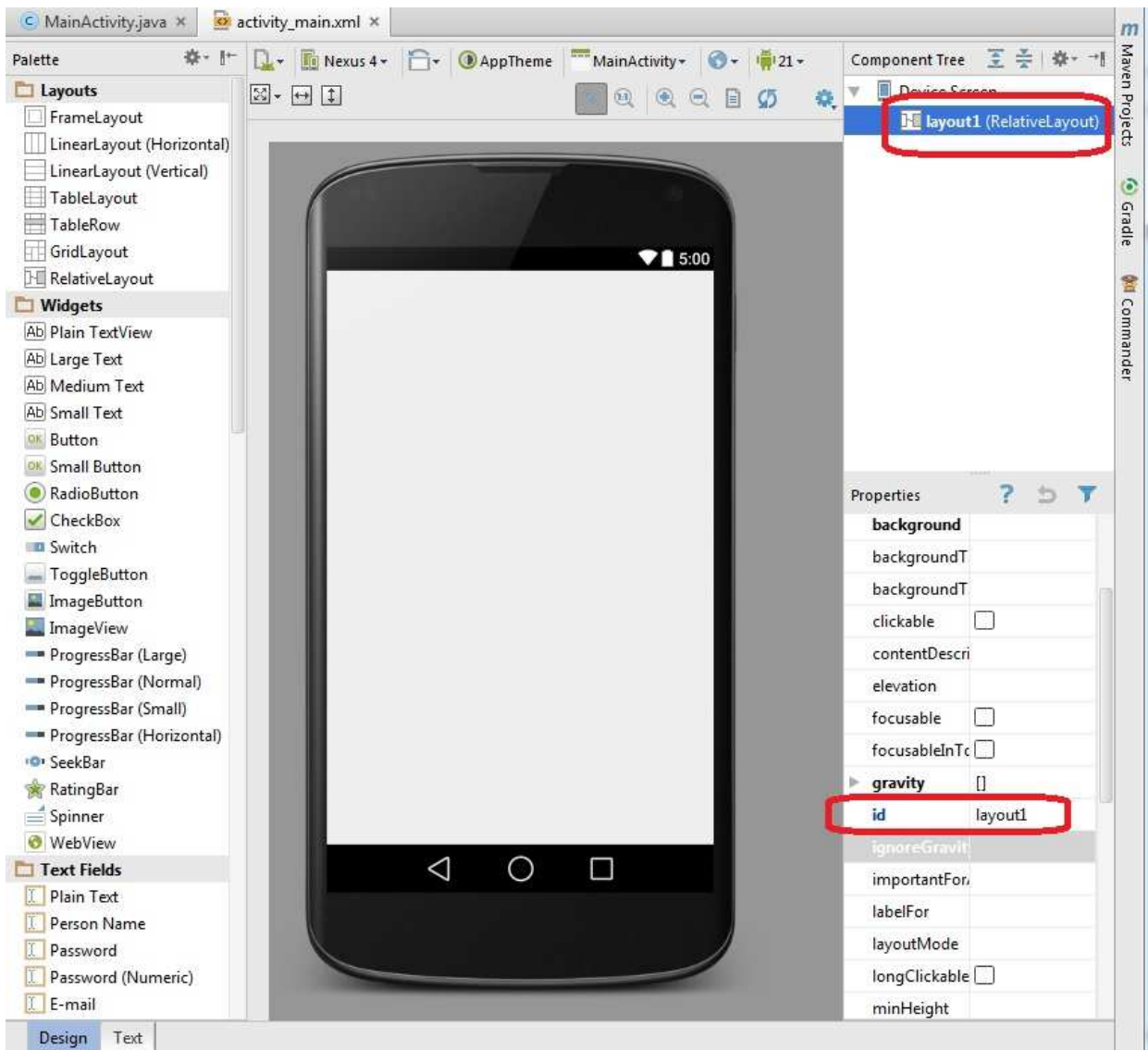
[Retornar](#)

37 - Dibujar: texto sobre un camino

Si queremos disponer texto que no esté horizontal debemos crear un camino indicando los puntos donde pasará el texto.

1 - Creamos un proyecto llamado: Proyecto041

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica:

```
package ar.com.tutorialesya.proyecto041;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Path;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
```



```

import android.widget.RelativeLayout;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);
        getSupportActionBar().hide();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(0, 0, 255);
            int alto = canvas.getHeight();

            Path camino = new Path();
            camino.moveTo(0, alto / 2);
            camino.lineTo(40, alto / 2 - 30);
            camino.lineTo(80, alto / 2 - 60);
            camino.lineTo(120, alto / 2 - 90);
            camino.lineTo(160, alto / 2 - 120);
            camino.lineTo(220, alto / 2 - 150);
            camino.lineTo(280, alto / 2 - 180);
            camino.lineTo(340, alto / 2 - 210);
            Paint pincell = new Paint();
            pincell.setARGB(255, 255, 0, 0);
            pincell.setTextSize(30);
        }
    }
}

```

```

        canvas.drawTextOnPath("Hola Mundo Hola Mundo", camino, 0, 0,
                               pincell1);
    }
}

```

Creamos un objeto de la clase Path e indicamos el primer punto del camino llamando al método moveTo:

```

Path camino = new Path();
camino.moveTo(0, alto/2);

```

Luego indicamos todos los otros puntos en forma consecutiva llamando al método lineTo:

```

camino.lineTo(40, alto/2-30);
camino.lineTo(80, alto/2-60);
camino.lineTo(120, alto/2-90);
camino.lineTo(160, alto/2-120);
camino.lineTo(220, alto/2-150);
camino.lineTo(280, alto/2-180);
camino.lineTo(340, alto/2-210);

```

Luego para graficar el texto llamamos al método drawTextOnPath y le pasamos en el segundo parámetro la referencia del objeto de tipo Path:

```

canvas.drawTextOnPath("Hola Mundo Hola Mundo",camino, 0, 0, pincell1);

```

La salida del programa es:



Este proyecto lo puede descargar en un zip desde este enlace:

[proyecto041.zip](#)

[Retornar](#)

38 - Dibujar: una imagen

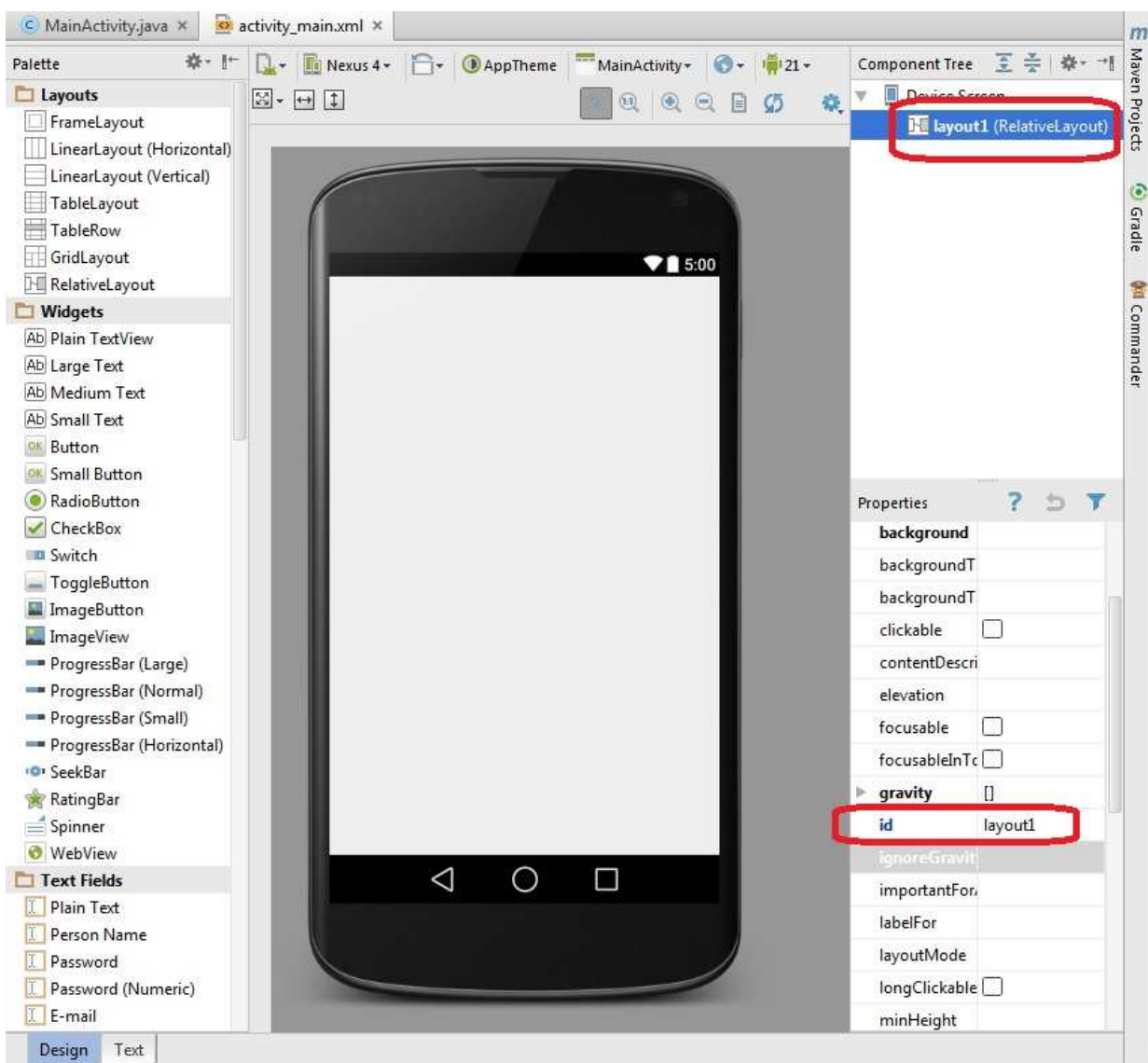
Para mostrar un archivo jpg, png etc. disponemos en la clase Canvas de un método llamado drawBitmap.

Problema:

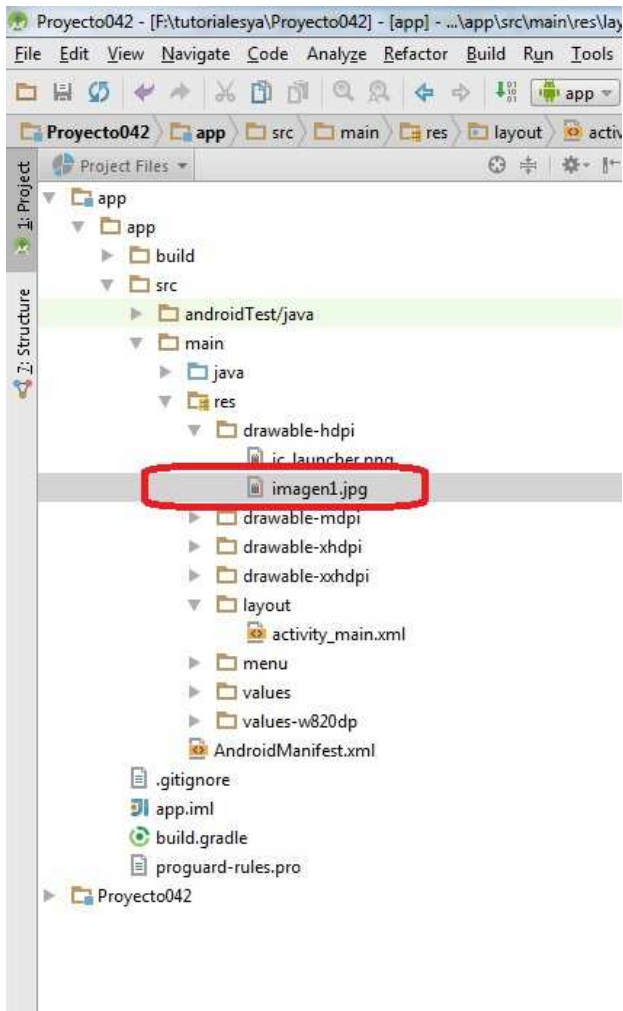
Mostrar el contenido de un archivo jpg centrado en la pantalla sabiendo que tiene un tamaño de 250 píxeles de ancho por 200 de alto.

1 - Creamos un proyecto llamado: proyecto042

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Descargar de internet una imagen y redimensionarla a 250*200 píxeles. Copiar dicho archivo a la carpeta res/drawable-hdpi de nuestro proyecto:



Ahora codificamos la clase donde se encuentra toda la lógica:

```
package ar.com.tutorialesya.proyecto042;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.RelativeLayout;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        Lienzo fondo = new Lienzo(this);
        layout1.addView(fondo);
    }
}
```

```

        getSupportActionBar().hide();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(0, 0, 255);
            int ancho = canvas.getWidth();
            int alto = canvas.getHeight();
            Bitmap bmp = BitmapFactory.decodeResource(getResources(),
                R.drawable.imagen1);
            canvas.drawBitmap(bmp, (ancho - 250) / 2, (alto - 200) / 2, null);
        }
    }
}

```

Para recuperar la imagen del archivo de la carpeta res/drawable-hdpi debemos utilizar el método decodeResource:

```
Bitmap bmp=BitmapFactory.decodeResource(getResources(), R.drawable.imagen1);
```

Una vez que tenemos creado el objeto de la clase Bitmap procedemos a posicionar la imagen en forma centrada en la pantalla del dispositivo:

```
canvas.drawBitmap(bmp, (ancho-250)/2,(alto-200)/2, null);
```

La vista previa de la aplicación es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto042.zip](#)

[Retornar](#)

39 - Evento touch: dibujar un círculo

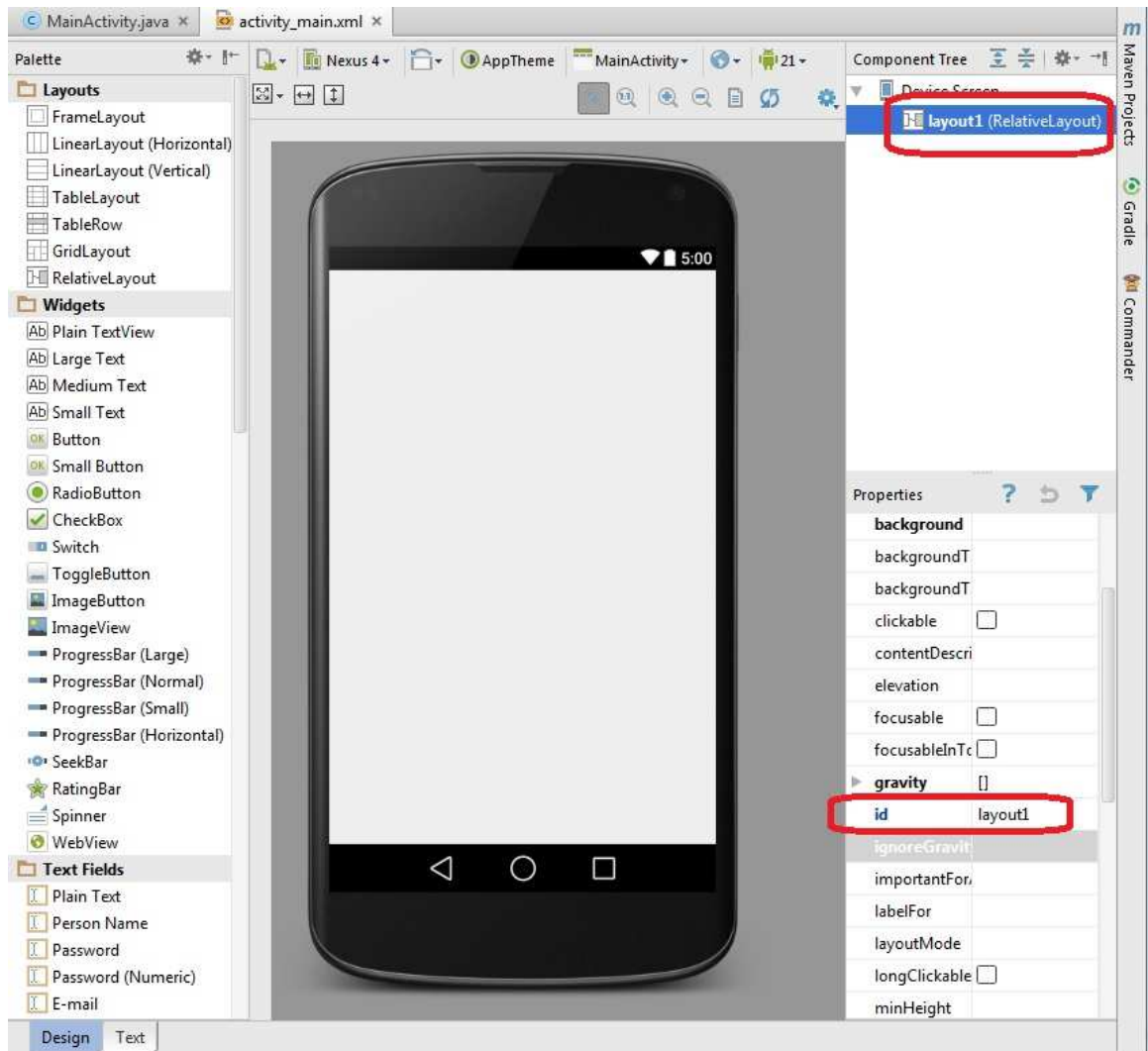
Una actividad fundamental es poder detectar cuando el usuario selecciona o toca la pantalla táctil.

Problema:

Desarrollar un programa que dibuje un círculo en la coordenada 100,100. Cuando se presione otra parte de la pantalla táctil proceder a trasladar el círculo a dicha coordenada.

1 - Creamos un proyecto llamado: Proyecto044

Borramos el TextView que agrega automáticamente el Android Studio y definimos el id del RelativeLayout con el valor: layout1:



Ahora codificamos la clase donde se encuentra toda la lógica para capturar el evento onTouch:

```
package ar.com.tutorialesya.proyecto043;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.RelativeLayout;
```



```

public class MainActivity extends ActionBarActivity implements View.OnTouchListener{

    private int corx, cory;
    private Lienzo fondo;

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

        corx = 100;
        cory = 100;
        RelativeLayout layout1 = (RelativeLayout) findViewById(R.id.layout1);
        fondo = new Lienzo(this);
        fondo.setOnTouchListener(this);
        layout1.addView(fondo);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public boolean onTouch(View v, MotionEvent event) {
        corx = (int) event.getX();
        cory = (int) event.getY();
        fondo.invalidate();
        return true;
    }

    class Lienzo extends View {

        public Lienzo(Context context) {
            super(context);
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255, 255, 0);
            Paint pincell = new Paint();
            pincell.setARGB(255, 255, 0, 0);
            pincell.setStrokeWidth(4);
            pincell.setStyle(Paint.Style.STROKE);
            canvas.drawCircle(corx, cory, 20, pincell);
        }

    }
}

```

clase debe implementar el método onTouch:

```
public class MainActivity extends ActionBarActivity implements View.OnTouchListener {
```

Definimos como atributos la coordenada donde se debe dibujar el círculo y la referencia al objeto de la clase Lienzo:

```
private int corx,cory;  
private Lienzo fondo;
```

En el método onCreate del ActionBarActivity inicializamos los tres atributos de la clase y mediante el método setOnTouchListener indicamos que la propia clase capturará el evento onTouch del objeto fondo:

```
corx=100;  
cory=100;  
fondo=new Lienzo(this);  
fondo.setOnTouchListener(this);  
linearLayout.addView(fondo);
```

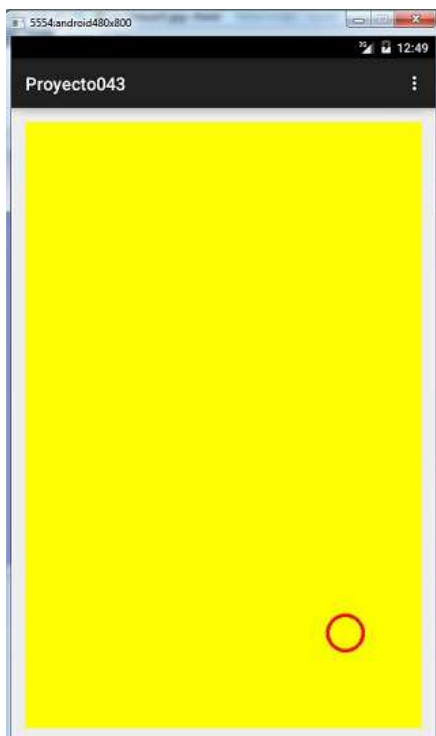
El método onTouch es el que implementamos de la clase OnTouchListener. En este método mediante el objeto event obtenemos la coordenada x e y donde el usuario seleccionó con el dedo y procedemos a llamar al método invalidate para que vuelva a pintarse el control fondo (el método invalidate borra el contenido del objeto de la clase Lienzo y vuelve a ejecutarse el método onDraw):

```
public boolean onTouch(View v, MotionEvent event) {  
    corx=(int)event.getX();  
    cory=(int)event.getY();  
    fondo.invalidate();  
    return true;  
}
```

El método onDraw pinta el fondo de amarillo, crea un objeto de la clase Paint y procede a dibujar un círculo en las coordenadas indicadas por los atributos corx y cory:

```
protected void onDraw(Canvas canvas) {  
    canvas.drawRGB(255,255,0);  
    Paint pincel1=new Paint();  
    pincel1.setARGB(255,255,0,0);  
    pincel1.setStrokeWidth(4);  
    pincel1.setStyle(Paint.Style.STROKE);  
    canvas.drawCircle(corx, cory, 20, pincel1);  
}
```

La vista previa de la aplicación es:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto043.zip](#)

[Retornar](#)

40 - Evento touch: juego del buscaminas

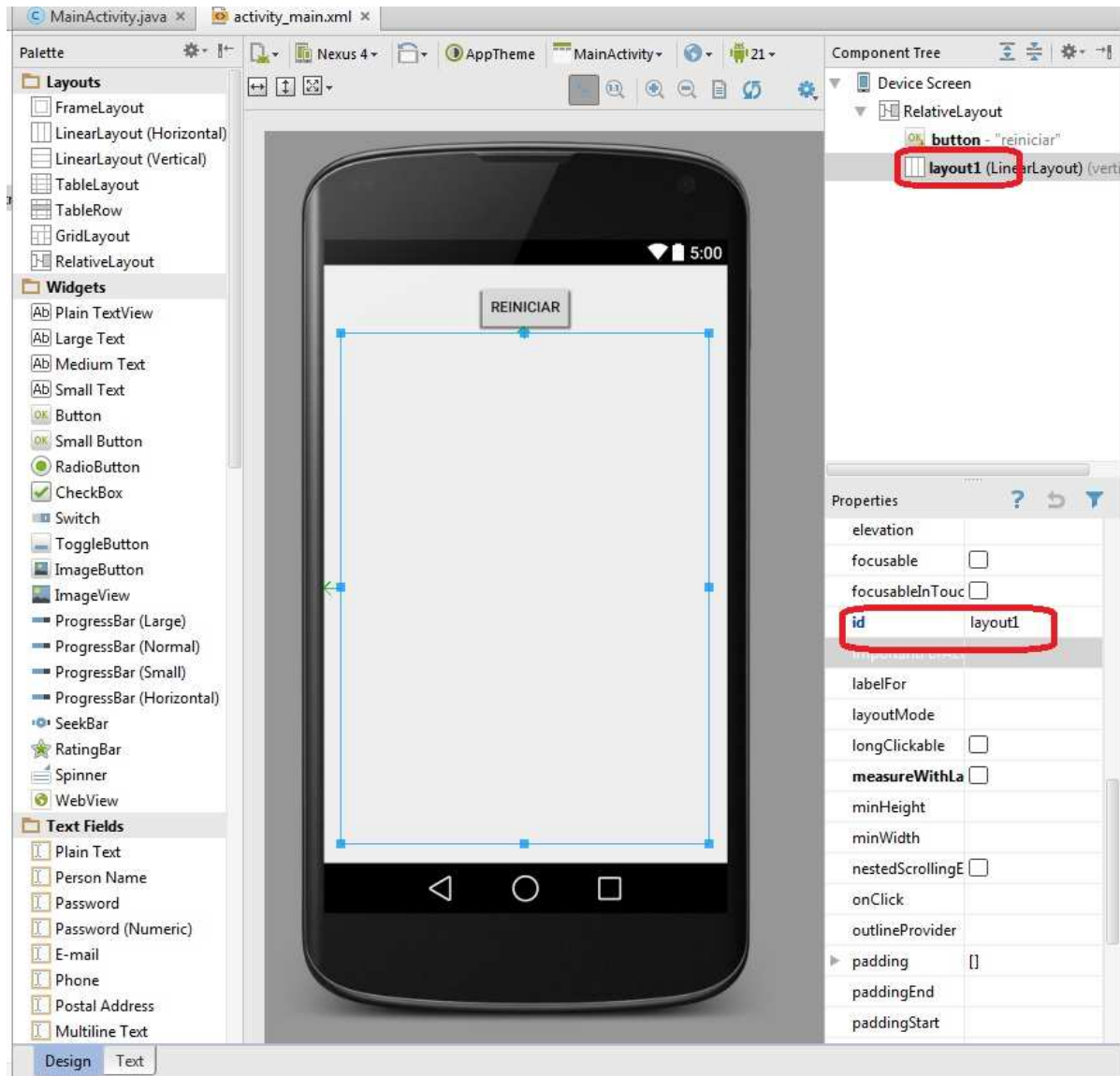
Problema:

Implementar el juego del Buscaminas. Crear una grilla de 8*8 celdas.

1 - Creamos un proyecto llamado: BuscaMinas

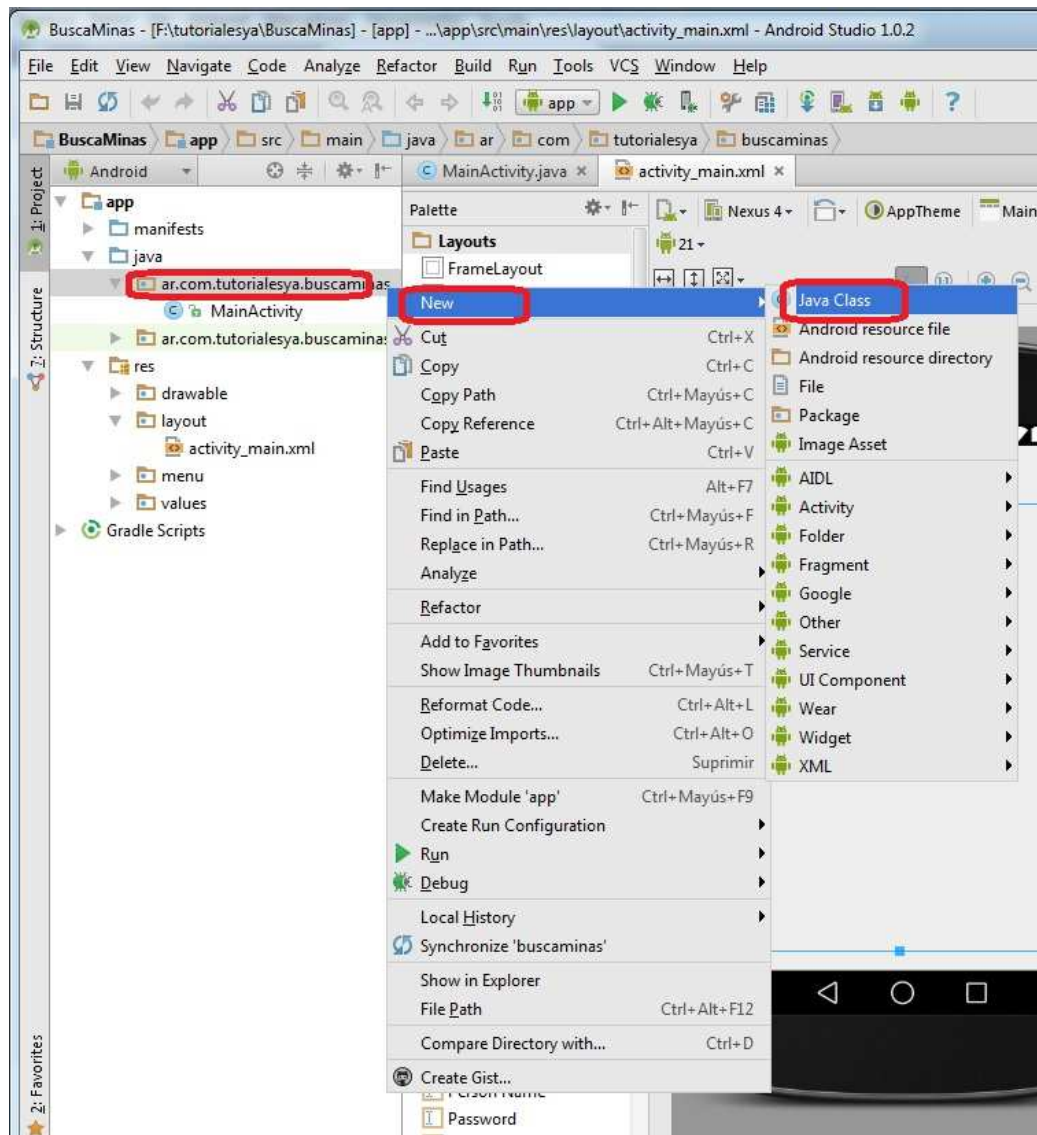
Borramos el TextView que agrega automáticamente el Android Studio y disponemos un Button y un LinearLayout:

Al botón inicializamos la propiedad onClick con el valor "reiniciar" y al LinearLayout le asignamos el id como "layout1".



Luego codificamos las clases BuscaMinasActivity y Casilla:

Creamos una clase llamada Casilla desde el Android Studio:



```
package ar.com.tutorialesya.buscaminas;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Typeface;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.WindowManager;
import android.widget.LinearLayout;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity implements View.OnTouchListener{
    private Tablero fondo;
    int x, y;
    private Casilla[][] casillas;
    private boolean activo = true;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
    }
}
```

```

        setContentView(R.layout.activity_main);

        LinearLayout layout = (LinearLayout) findViewById(R.id.layout1);
        fondo = new Tablero(this);
        fondo.setOnTouchListener(this);
        layout.addView(fondo);
        casillas = new Casilla[8][8];
        for (int f = 0; f < 8; f++) {
            for (int c = 0; c < 8; c++) {
                casillas[f][c] = new Casilla();
            }
        }
        this.disponerBombas();
        this.contarBombasPerimetro();
        getSupportActionBar().hide();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void reiniciar(View v) {
        casillas = new Casilla[8][8];
        for (int f = 0; f < 8; f++) {
            for (int c = 0; c < 8; c++) {
                casillas[f][c] = new Casilla();
            }
        }
        this.disponerBombas();
        this.contarBombasPerimetro();
        activo = true;

        fondo.invalidate();
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (activo)
            for (int f = 0; f < 8; f++) {
                for (int c = 0; c < 8; c++) {
                    if (casillas[f][c].dentro((int) event.getX(),
                        (int) event.getY())) {
                        casillas[f][c].destapado = true;
                        if (casillas[f][c].contenido == 80) {
                            Toast.makeText(this, "Booooooooooooooooooooooooooooo",
                                Toast.LENGTH_LONG).show();
                            activo = false;
                        } else if (casillas[f][c].contenido == 0)

```

```

        recorrer(f, c);
        fondo.invalidate();
    }
}
}
if (gano() && activo) {
    Toast.makeText(this, "Ganaste", Toast.LENGTH_LONG).show();
    activo = false;
}

return true;
}

class Tablero extends View {

    public Tablero(Context context) {
        super(context);
    }

    protected void onDraw(Canvas canvas) {
        canvas.drawRGB(0, 0, 0);
        int ancho = 0;
        if (canvas.getWidth() < canvas.getHeight())
            ancho = fondo.getWidth();
        else
            ancho = fondo.getHeight();
        int anchocua = ancho / 8;
        Paint paint = new Paint();
        paint.setTextSize(20);
        Paint paint2 = new Paint();
        paint2.setTextSize(20);
        paint2.setTypeface(Typeface.DEFAULT_BOLD);
        paint2.setARGB(255, 0, 0, 255);
        Paint paintlineal = new Paint();
        paintlineal.setARGB(255, 255, 255, 255);
        int filaact = 0;
        for (int f = 0; f < 8; f++) {
            for (int c = 0; c < 8; c++) {
                casillas[f][c].fijarxy(c * anchocua, filaact, anchocua);
                if (casillas[f][c].destapado == false)
                    paint.setARGB(153, 204, 204, 204);
                else
                    paint.setARGB(255, 153, 153, 153);
                canvas.drawRect(c * anchocua, filaact, c * anchocua
                    + anchocua - 2, filaact + anchocua - 2, paint);
                // linea blanca
                canvas.drawLine(c * anchocua, filaact, c * anchocua
                    + anchocua, filaact, paintlineal);
                canvas.drawLine(c * anchocua + anchocua - 1, filaact, c
                    * anchocua + anchocua - 1, filaact + anchocua,
                    paintlineal);

                if (casillas[f][c].contenido >= 1
                    && casillas[f][c].contenido <= 8
                    && casillas[f][c].destapado)
                    canvas.drawText(
                        String.valueOf(casillas[f][c].contenido), c
                            * anchocua + (anchocua / 2) - 8,
                        filaact + anchocua / 2, paint2);

                if (casillas[f][c].contenido == 80
                    && casillas[f][c].destapado) {
                    Paint bomba = new Paint();
                    bomba.setARGB(255, 255, 0, 0);
                    canvas.drawCircle(c * anchocua + (anchocua / 2),
                        filaact + (anchocua / 2), 8, bomba);
                }
            }
        }
    }
}

```



```

        }
        filaact = filaact + anchocua;
    }
}

private void disponerBombas() {
    int cantidad = 8;
    do {
        int fila = (int) (Math.random() * 8);
        int columna = (int) (Math.random() * 8);
        if (casillas[fila][columna].contenido == 0) {
            casillas[fila][columna].contenido = 80;
            cantidad--;
        }
    } while (cantidad != 0);
}

private boolean gano() {
    int cant = 0;
    for (int f = 0; f < 8; f++)
        for (int c = 0; c < 8; c++)
            if (casillas[f][c].destapado)
                cant++;
    if (cant == 56)
        return true;
    else
        return false;
}

private void contarBombasPerimetro() {
    for (int f = 0; f < 8; f++) {
        for (int c = 0; c < 8; c++) {
            if (casillas[f][c].contenido == 0) {
                int cant = contarCoordenada(f, c);
                casillas[f][c].contenido = cant;
            }
        }
    }
}

int contarCoordenada(int fila, int columna) {
    int total = 0;
    if (fila - 1 >= 0 && columna - 1 >= 0) {
        if (casillas[fila - 1][columna - 1].contenido == 80)
            total++;
    }
    if (fila - 1 >= 0) {
        if (casillas[fila - 1][columna].contenido == 80)
            total++;
    }
    if (fila - 1 >= 0 && columna + 1 < 8) {
        if (casillas[fila - 1][columna + 1].contenido == 80)
            total++;
    }

    if (columna + 1 < 8) {
        if (casillas[fila][columna + 1].contenido == 80)
            total++;
    }
    if (fila + 1 < 8 && columna + 1 < 8) {
        if (casillas[fila + 1][columna + 1].contenido == 80)
            total++;
    }

    if (fila + 1 < 8) {
        if (casillas[fila + 1][columna].contenido == 80)
            total++;
    }
}

```


41 - Archivo strings.xml

El archivo strings.xml se utiliza para almacenar todas las constantes de cadenas de caracteres que se necesitan en un programa (por ejemplo las etiquetas de los objetos Button, los textos de los controles TextView y todos los controles que muestran un texto en el dispositivo)

La idea fundamental es tener todos los mensajes que muestra nuestra aplicación en un archivo (strings.xml) no lo hemos hecho hasta este momento ya que nos torno un grado más complejo el desarrollo de ejercicios que tienen por objetivo el aprendizaje de programación java en Android. Cuando hagamos una aplicación más importante es necesario aplicar este estilo de organizar los mensajes de nuestra aplicación.

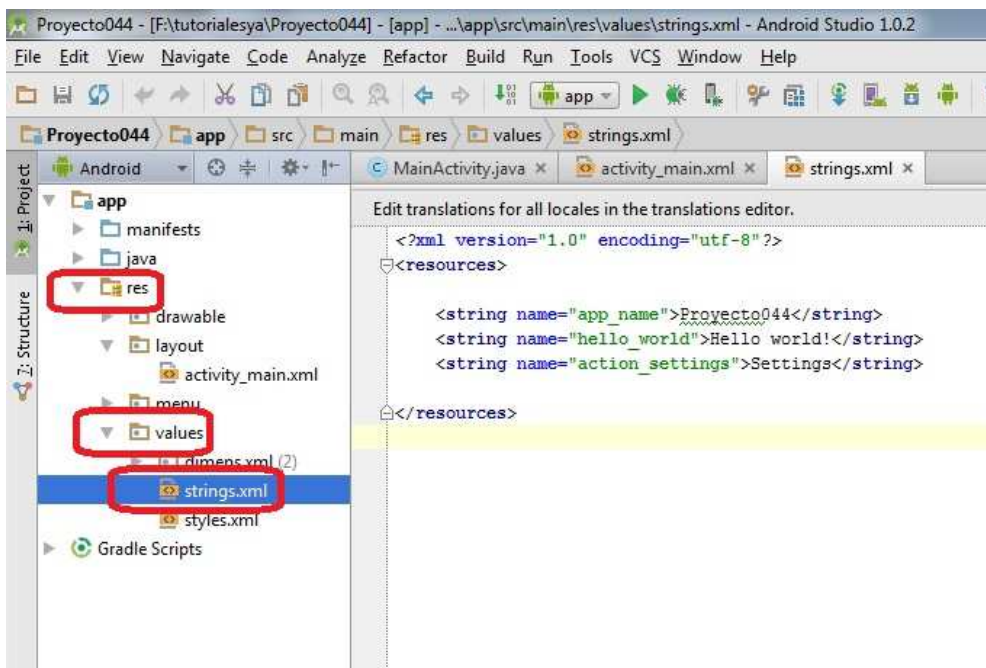
Ya veremos en el próximo concepto que uno de las ventajas que presenta esta agrupación de string es la posibilidad de facilitar la implementación de aplicaciones en múltiples idiomas.

Problema:

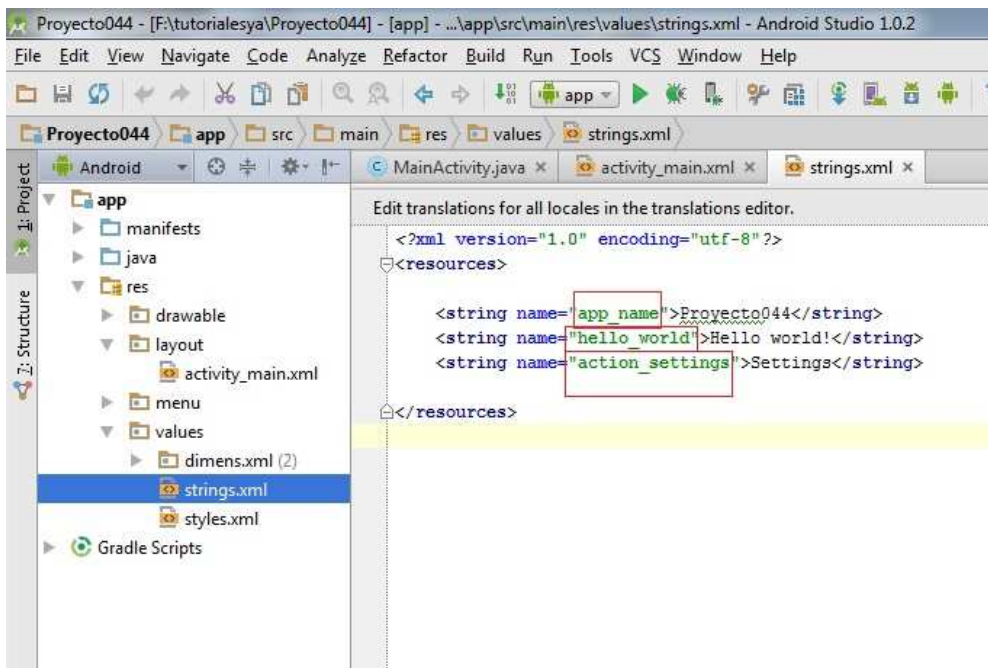
Crear un proyecto que solicite la carga de dos valores. Mediante dos RadioButton permitir seleccionar si queremos sumar o restar. Cuando se presione un botón mostrar en un TextView el resultado de la operación.

1 - Creamos un proyecto llamado: Proyecto044

El Android Studio nos crea automáticamente el archivo strings.xml en la carpeta values que se encuentra en la carpeta res:



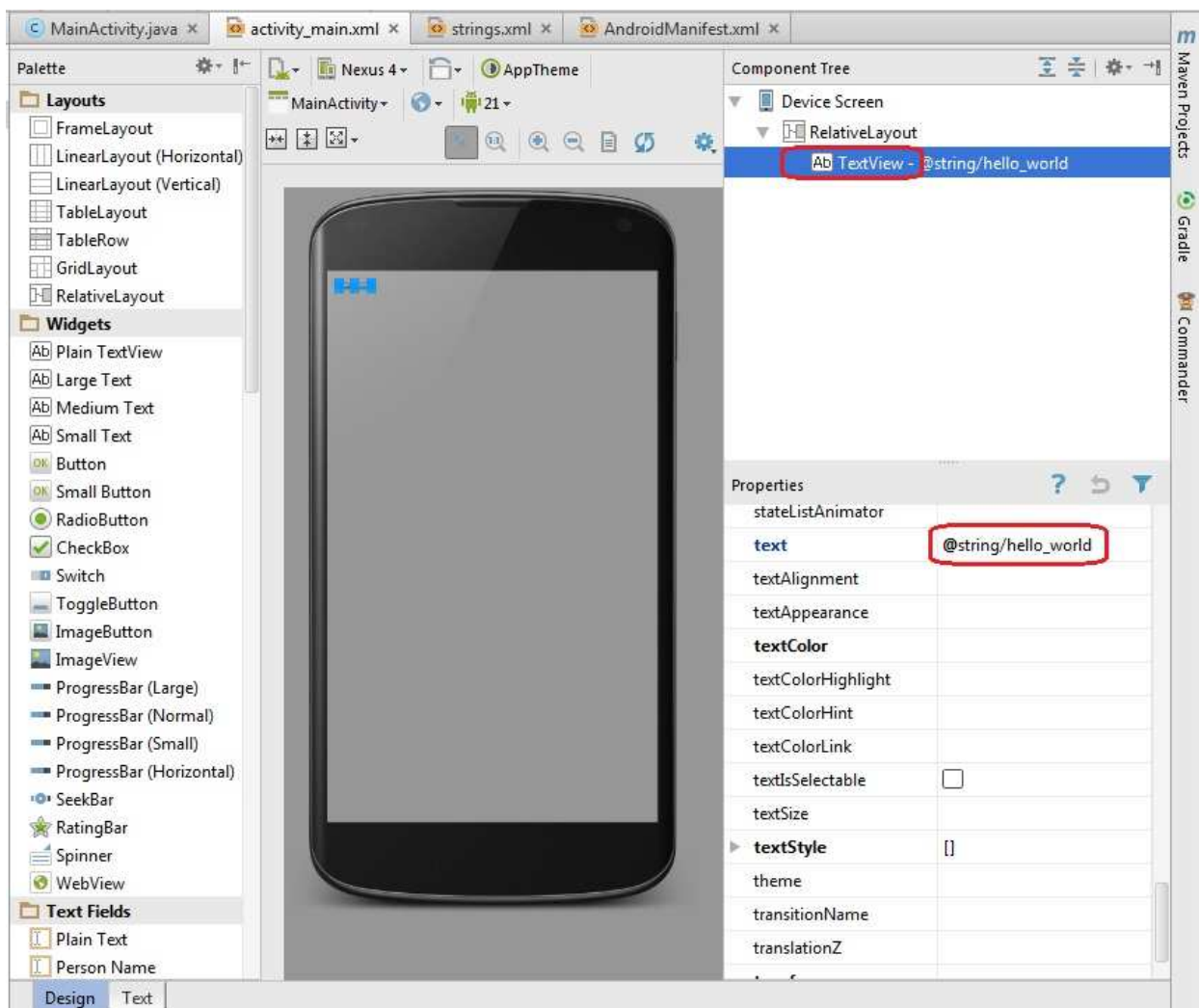
Si vemos en su interior podemos ver que ya define tres string llamados app_name, hello_world y action_settings:



Lo que se encuentra entre las marcas string es lo que se visualizará, por ejemplo en la constante `hello_world` se almacena el valor "Hello world!". En este archivo crearemos todas las constantes a incluir en nuestro programa.

Estas tres constantes que crea por defecto el Android Studio se están utilizando en distintas partes del programa.

La constante `hello_world` por ejemplo la utiliza en el `TextView` que dispone dentro del `ActionBarActivity`:



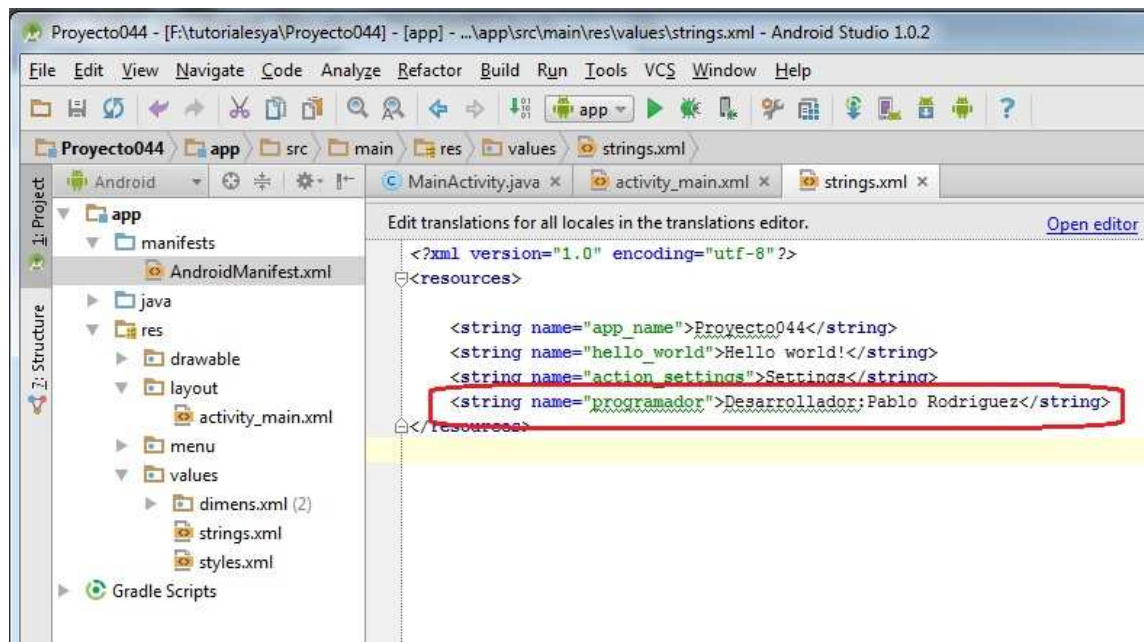
Como podemos ver para indicar una constante definida en el archivo string.xml debemos utilizar la sintaxis:

`@string/hello_world`

Es decir disponemos @string/ y seguidamente el nombre de la constante definida en el archivo string.xml.

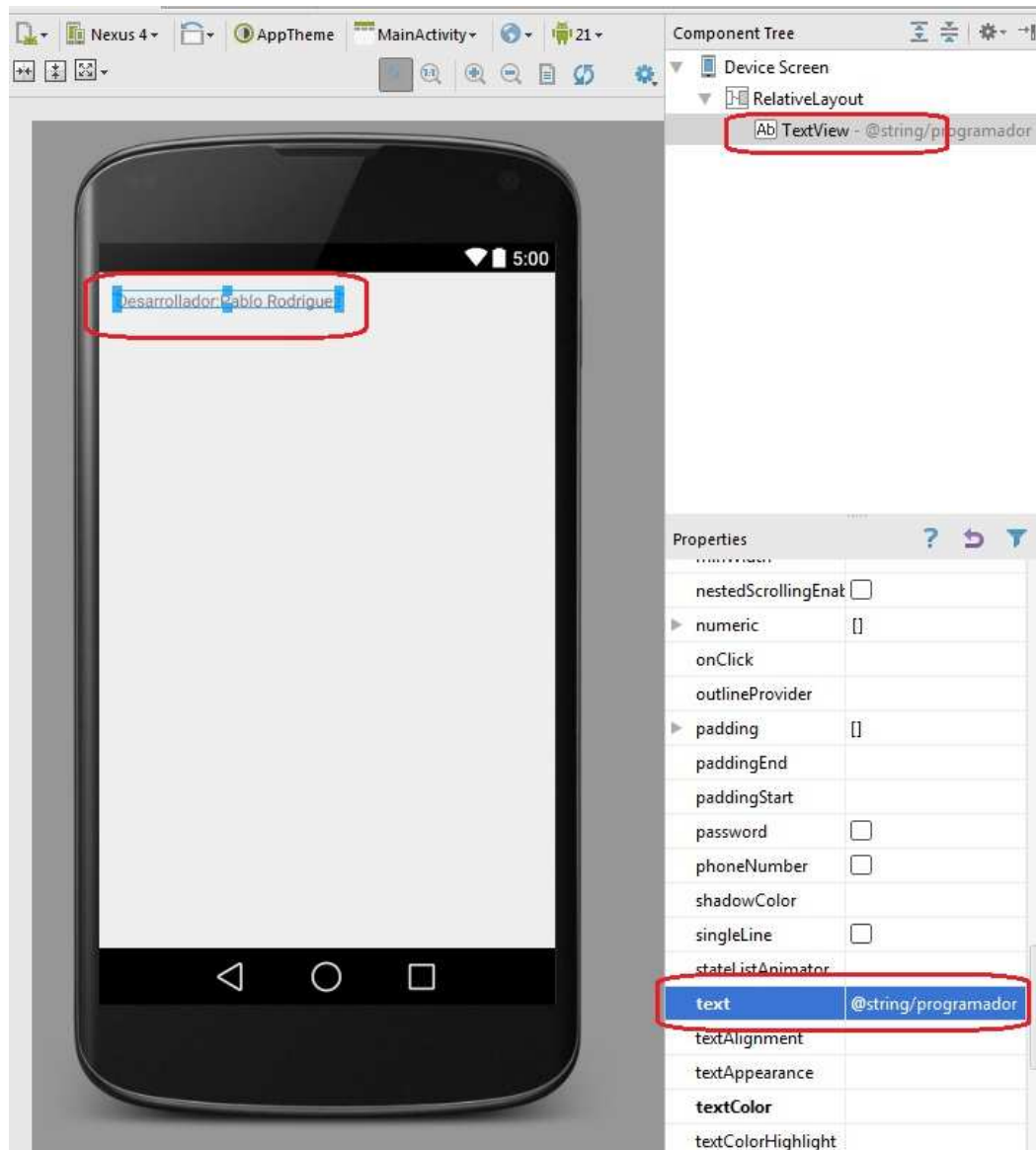
Hay varias formas de crear estas constantes:

1. La escribimos directamente en el editor de texto del archivo string.xml:

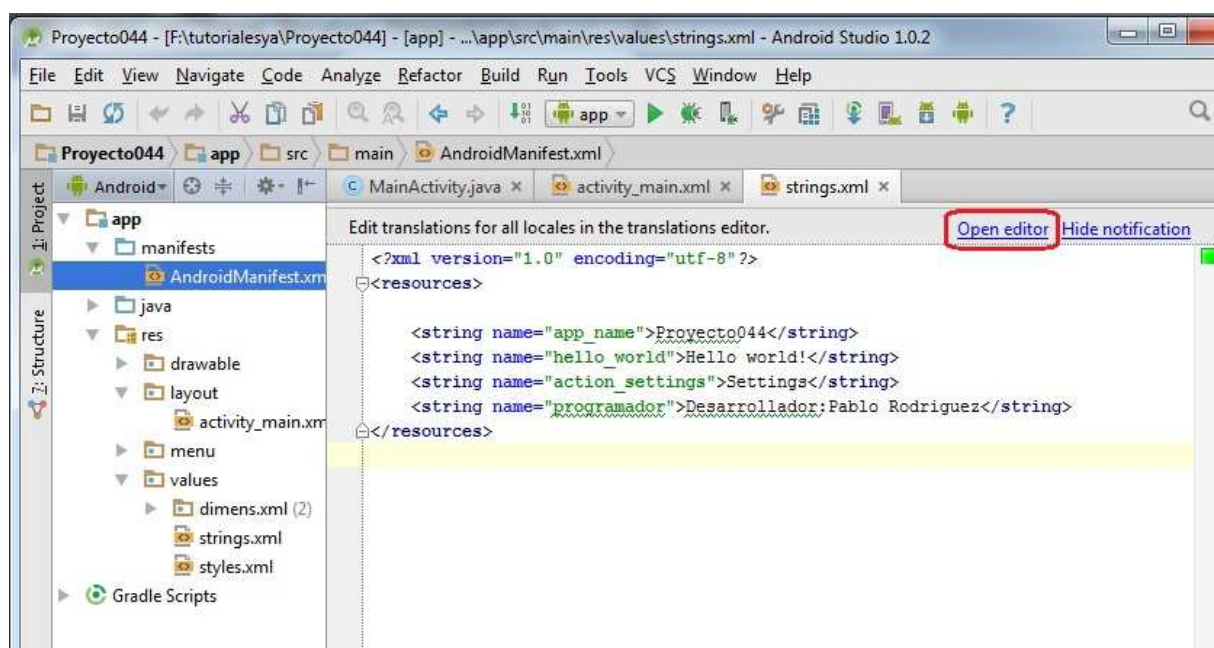


Es decir estamos creando una constante llamada "programador". Para hacer referencia en nuestro programa indicaremos @string/programador y el dato que se mostrará será: "Desarrollador:Pablo Rodriguez".

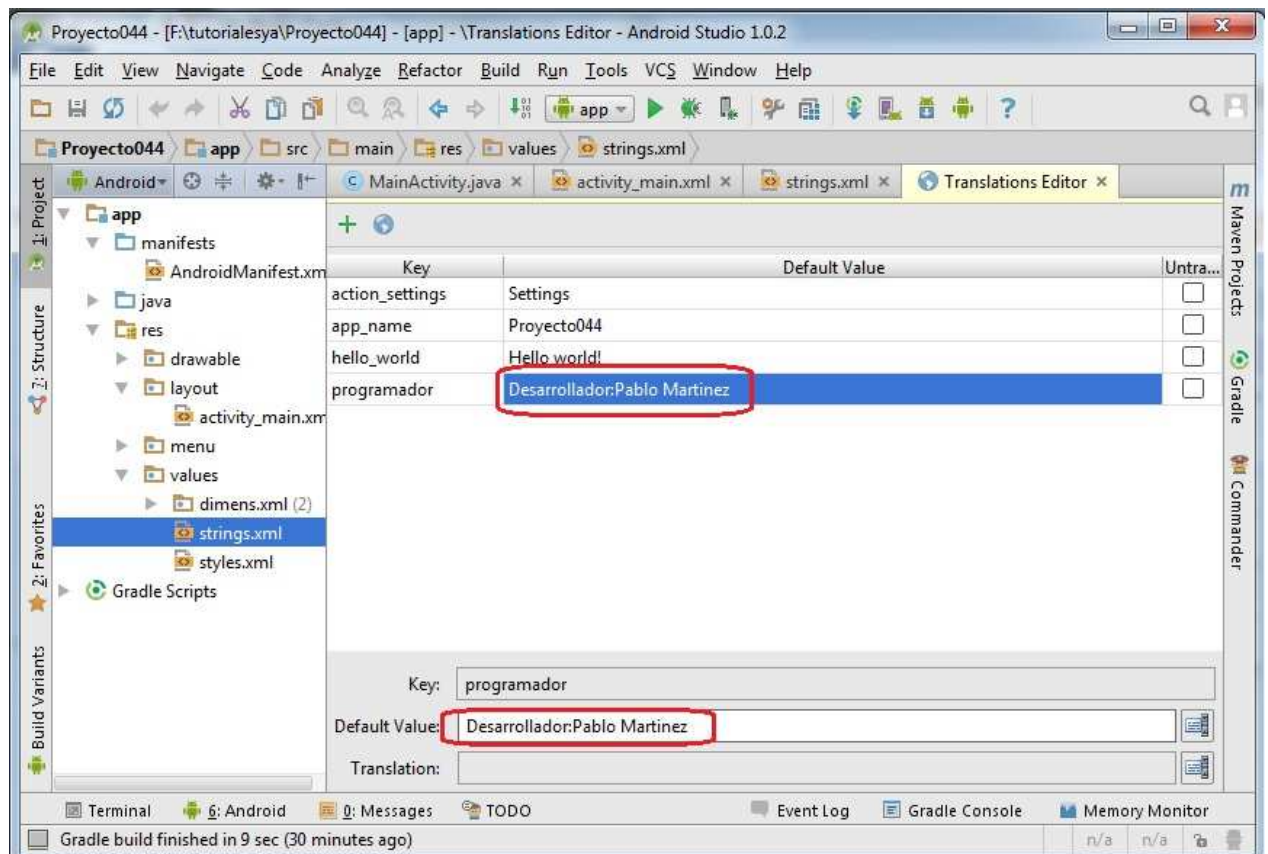
Probar de cambiar la propiedad text del EditText por defecto que agregó el Android Studio por la constante "programador", el resultado debe ser el siguiente:



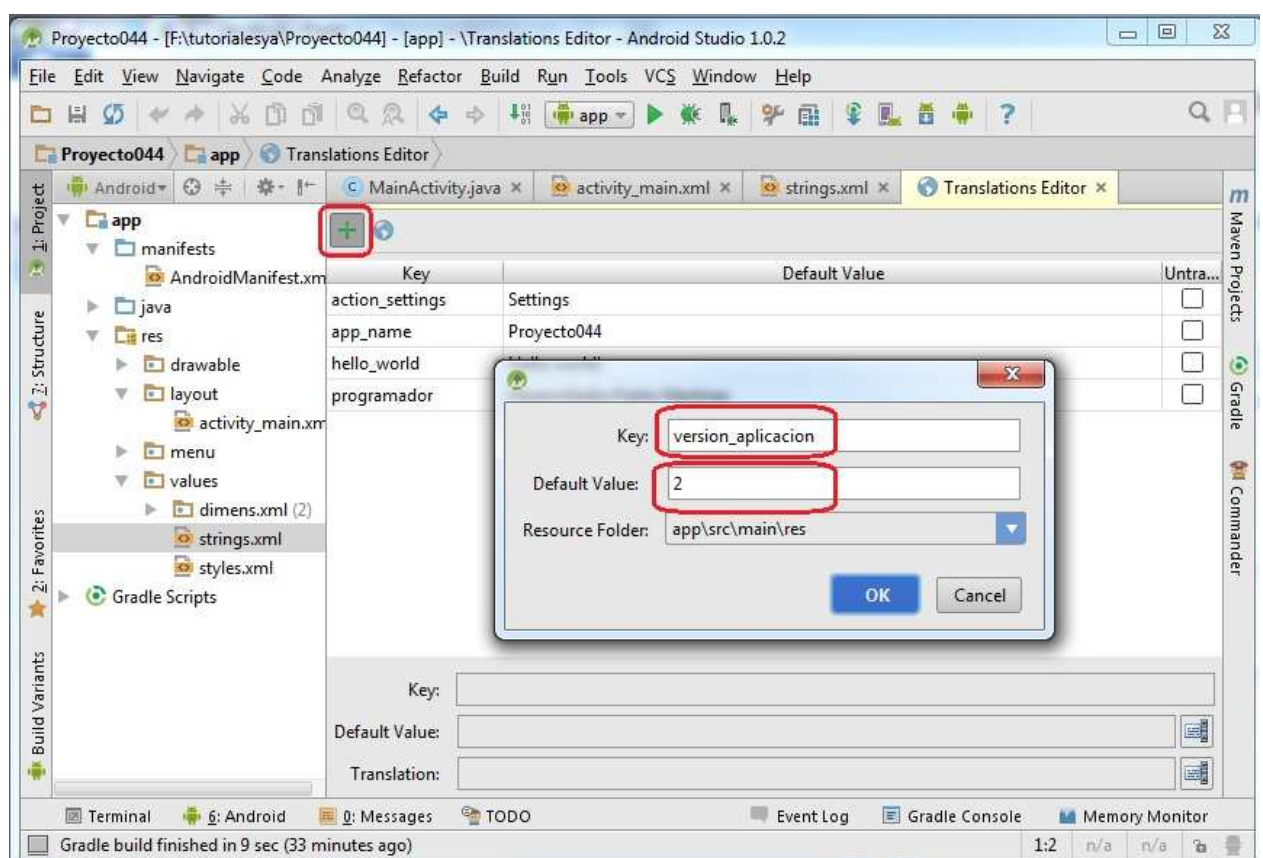
- La segunda forma es utilizar un editor visual que nos escriba el xml, esto se accede presionando la opción "Open editor":



En esta pantalla tenemos dos columnas con todas las constantes en la primera columna y los valores de las mismas en la segunda. Podemos con el mouse seleccionar cualquiera de ellas y editarlas en la parte inferior:



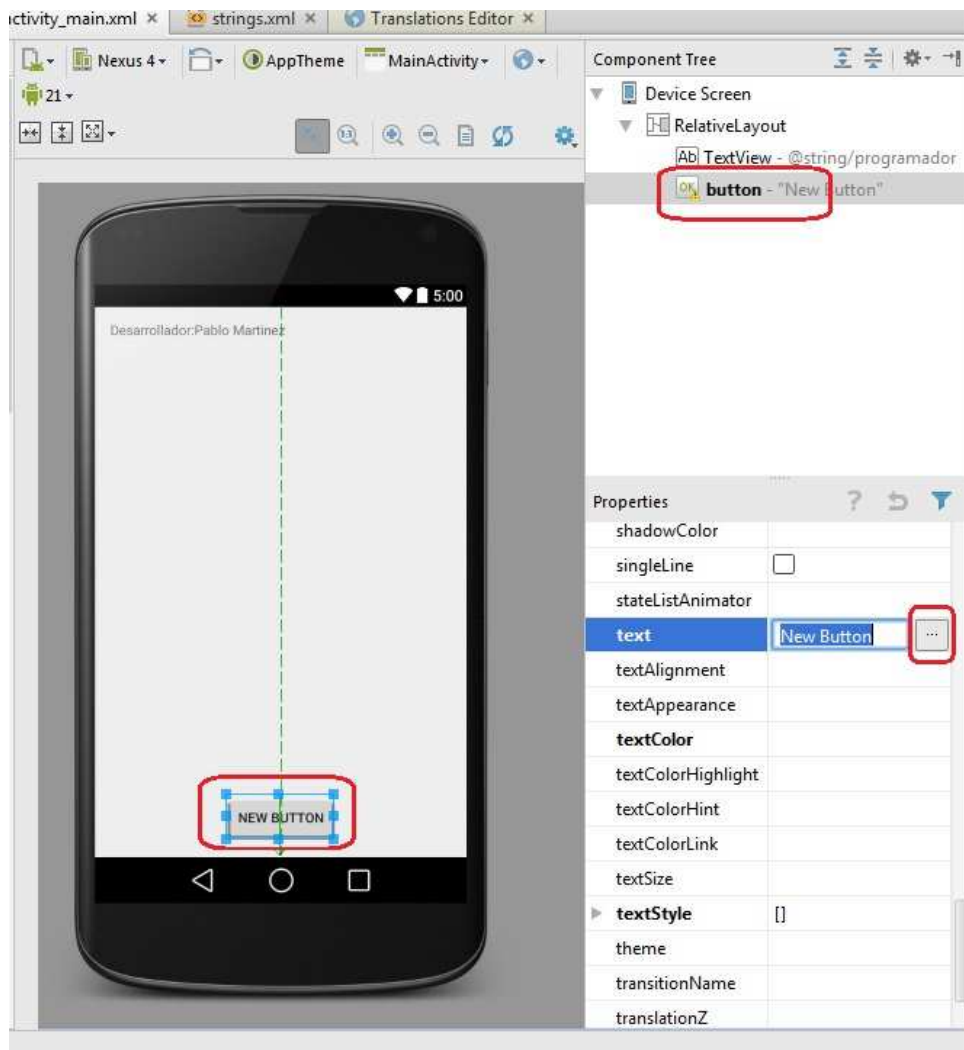
También podemos añadir nuevas constantes presionando el botón con el ícono "más" de la parte superior izquierda, creemos nuestra segunda constante llamada "version_aplicacion" con el valor 2:



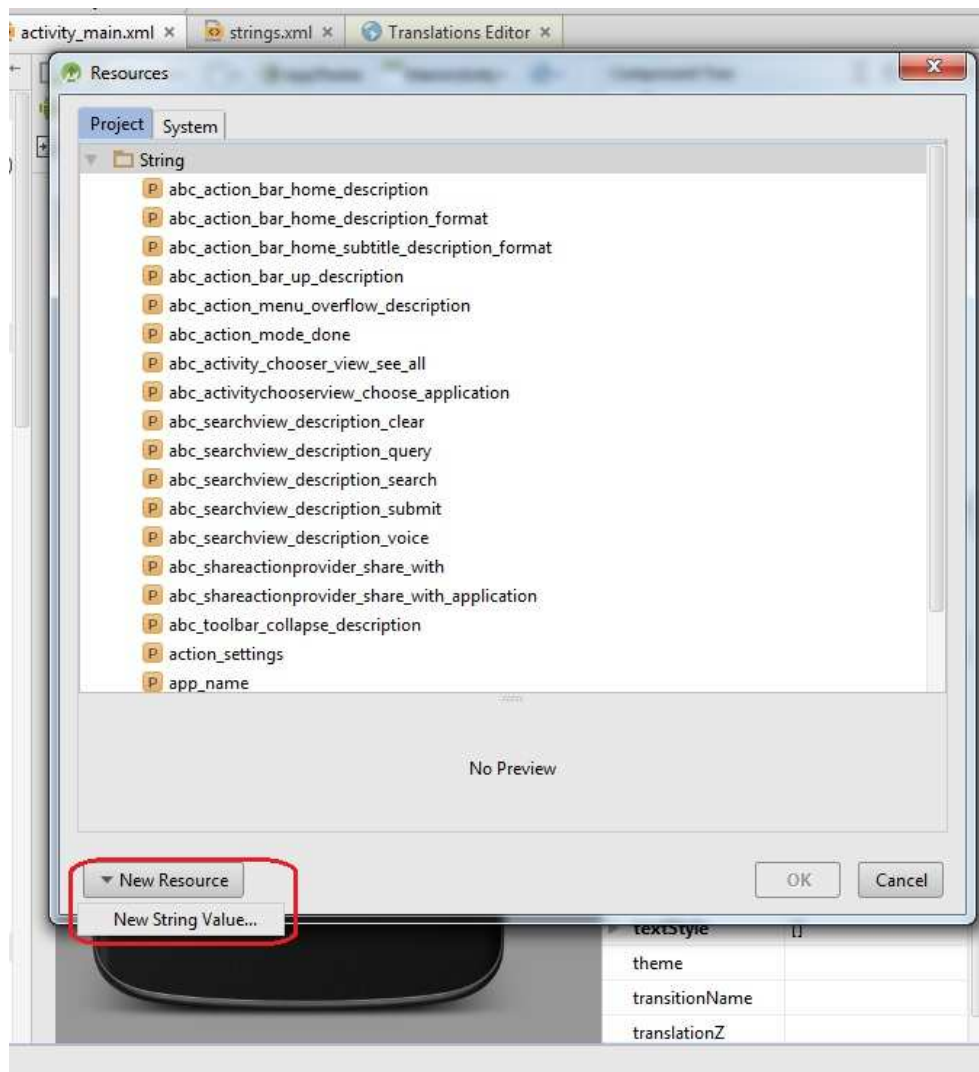
En principio con este editor de constantes nos evitamos de escribir y cometer algún error sintáctico en nuestro archivo strings.xml

- La tercer forma es crear la constante en el momento que disponemos un control en la interfaz visual, por ejemplo dispongamos un botón en la parte inferior que muestre el mensaje "Finalizar Programa" localizando

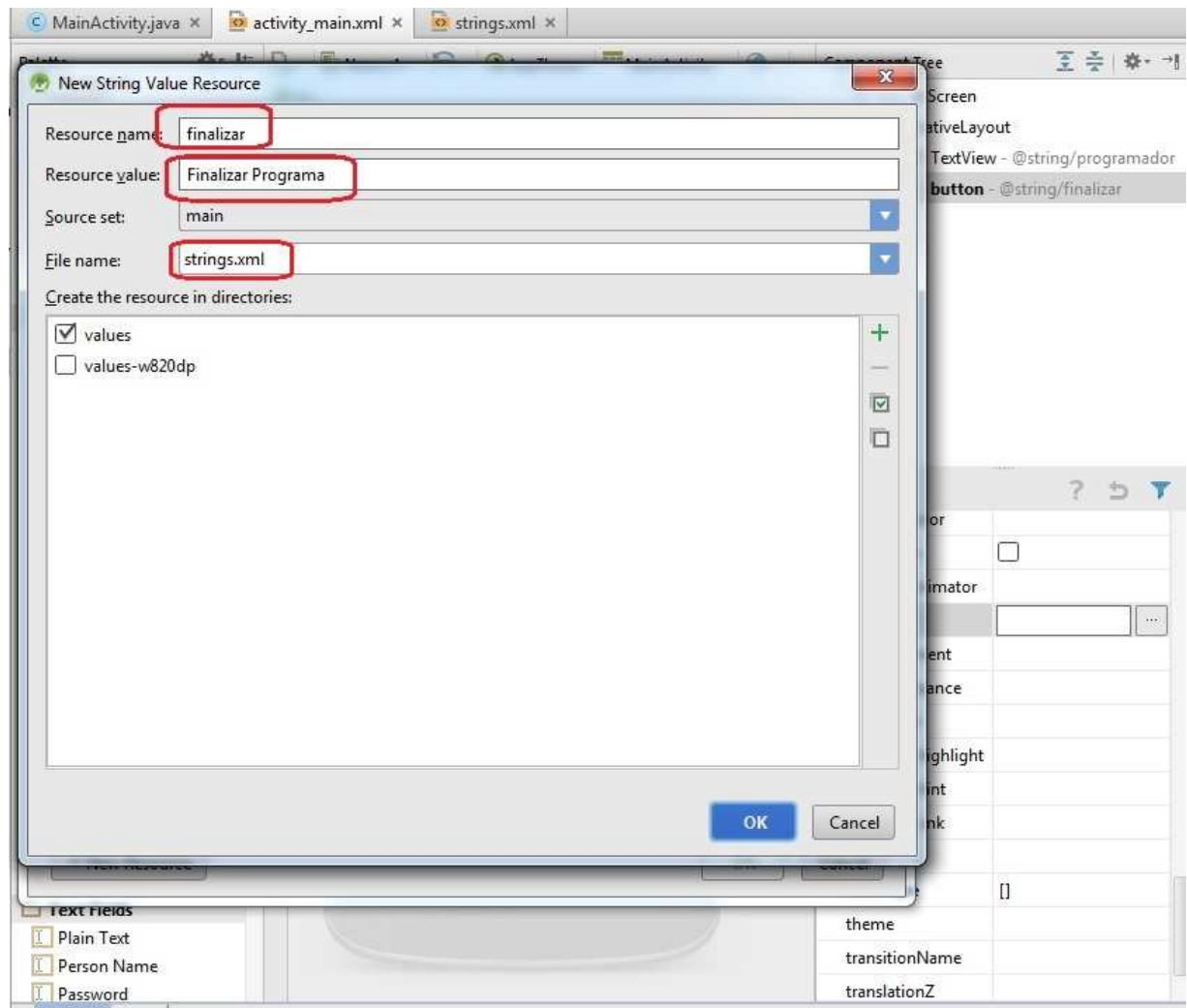
este texto en el archivo strings.xml:



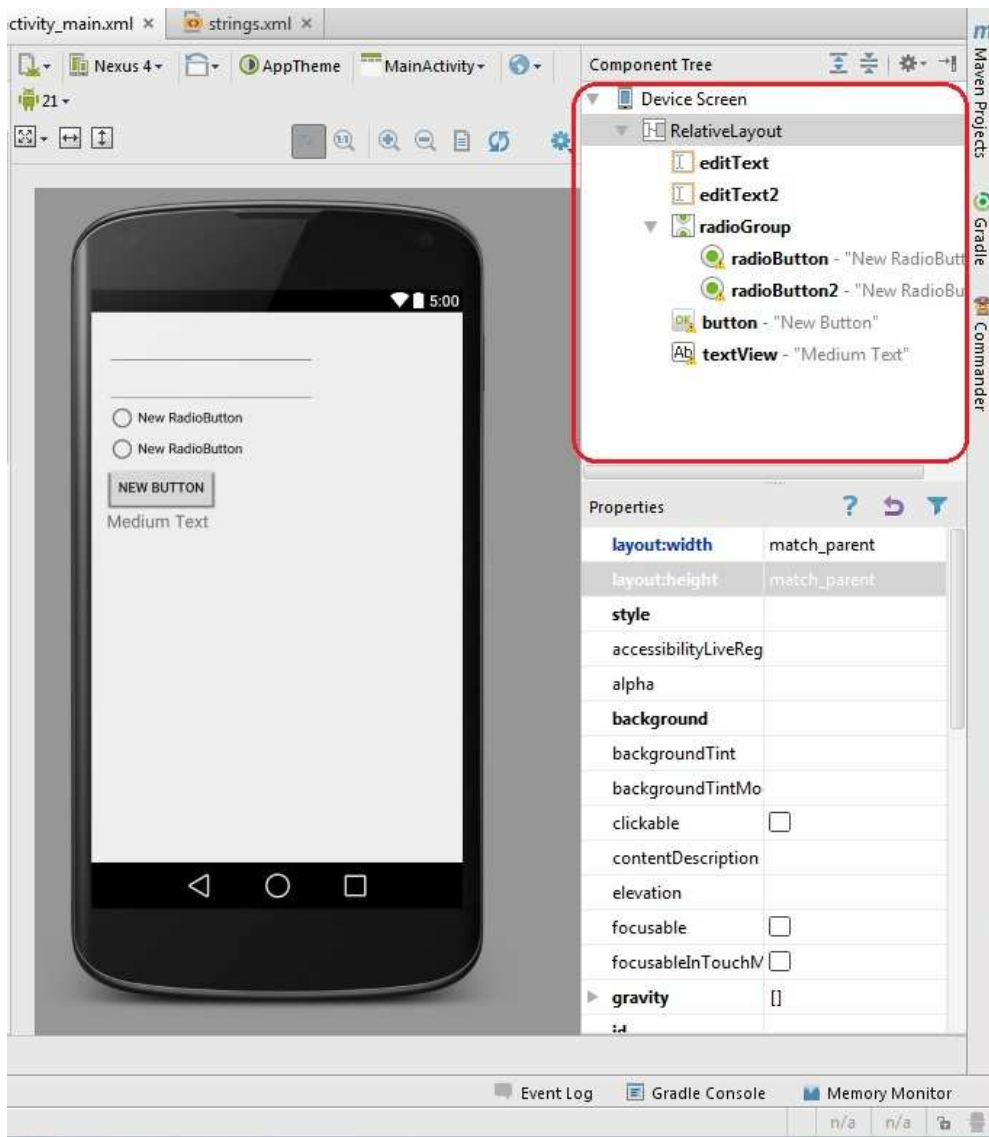
Presionamos el botón que aparece al lado de donde ingresamos el texto y en este diálogo procedemos a elegir la opción de "New String Value":



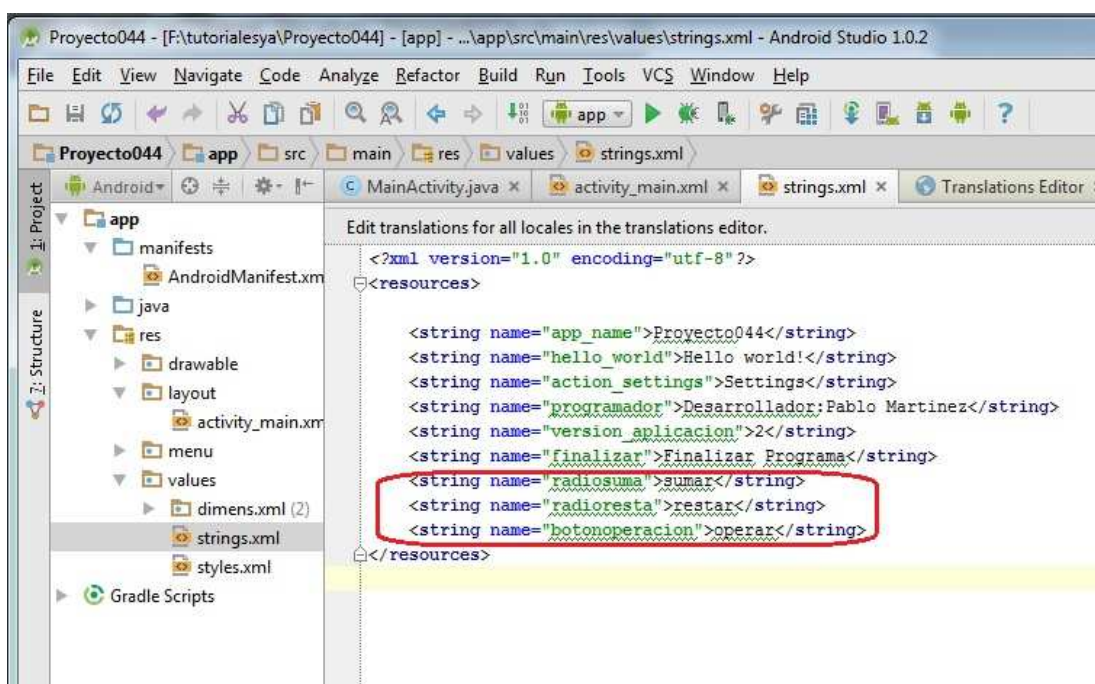
Finalmente es este diálogo procedemos a especificar el nombre de la constante, el valor y en "File Name" elegimos el archivo strings.xml:



Crearemos ahora la interfaz visual propuesta por el problema planteado al principio de este concepto. Disponiendo dos controles de tipo `EditText`, un `RadioGroup` con dos `RadioButton`, un `Button` y un `TextView` donde se muestra el resultado de la operación:

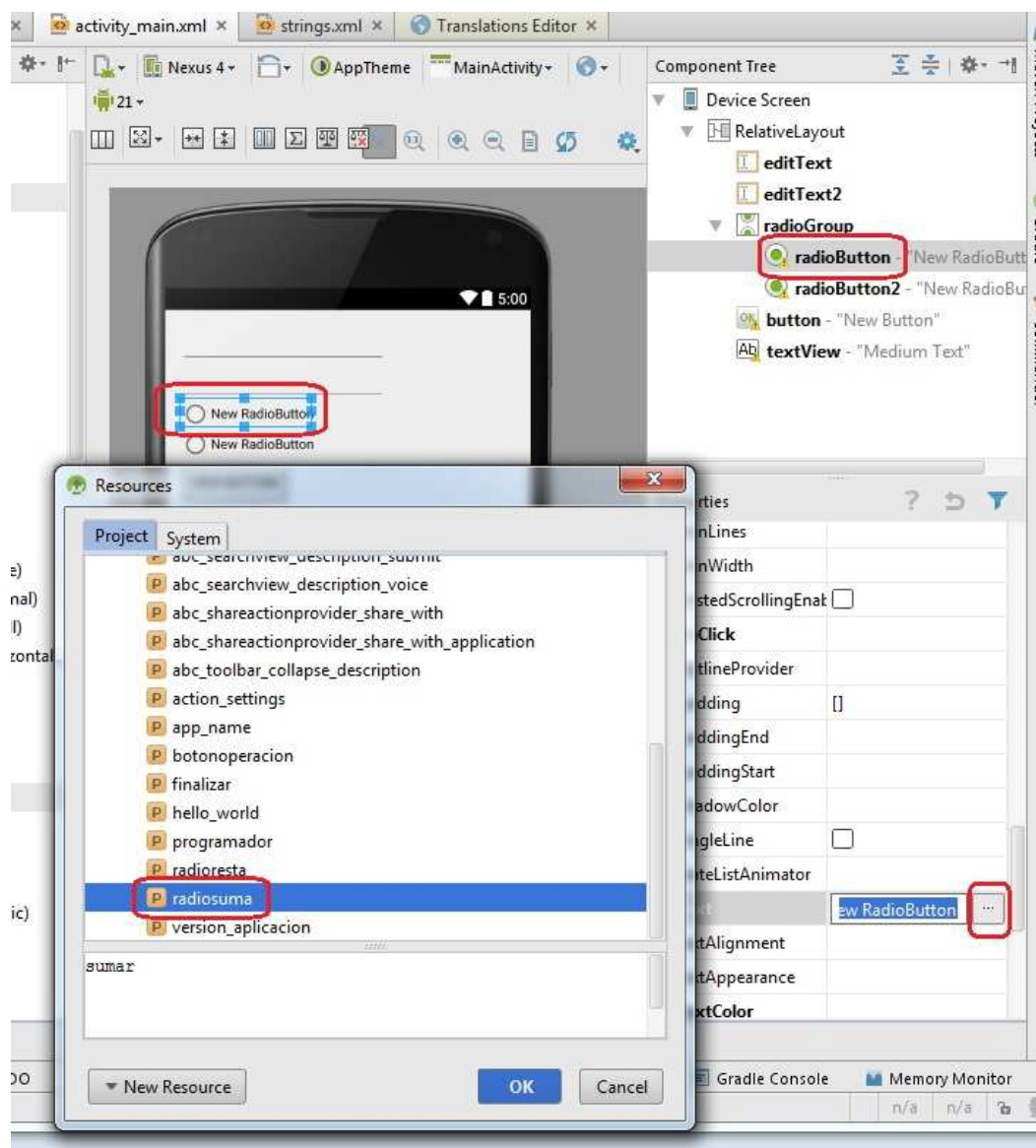


Ahora si crearemos los mensajes que mostrarán los dos RadioButton y el Button. Para esto seleccionamos el archivo strings y procedemos a crear las tres constantes (de la primera o segunda forma creamos las tres constantes):



Ahora debemos asociar estas constantes a los controles visuales. Seleccionamos el archivo activity_main.xml y seleccionamos primero el primer RadioButton y buscamos la propiedad text, presionamos el botón que aparece a la

derecha:



En este diálogo seleccionamos la constante de string que acabamos de crear ("radiosuma")

Lo mismo hacemos para asociar las constantes para el segundo RadioButton y el Button de operar. Es decir las propiedades text del radio1 debe quedar con el valor "@string/radioresta" y el valor de la propiedad text del objeto button debe ser "@string/botonoperacion"

Como vemos cuando asociamos las propiedades text de los controles con las constantes automáticamente vemos en pantalla que los textos de los controles se actualizan con los valores almacenados en dichas constantes.

Para que funcione nuestro programa inicializamos la propiedad onClick del Button con el método que sumará o restará (llamado operar):

El código fuente del programa será:

```
package ar.com.tutorialesya.proyecto044;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;
```

```

public class MainActivity extends ActionBarActivity {
    EditText et1, et2;
    RadioButton rb1, rb2;
    TextView tv1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        et1 = (EditText) findViewById(R.id.editText);
        et2 = (EditText) findViewById(R.id.editText2);
        rb1 = (RadioButton) findViewById(R.id.radioButton);
        rb2 = (RadioButton) findViewById(R.id.radioButton2);
        tv1 = (TextView) findViewById(R.id.textView);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

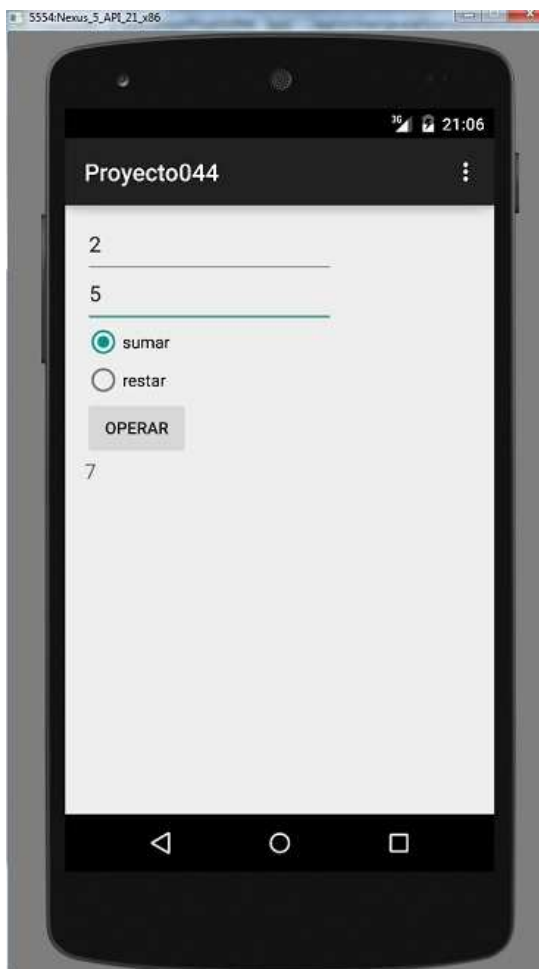
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void operar(View v) {
        int v1 = Integer.parseInt(et1.getText().toString());
        int v2 = Integer.parseInt(et2.getText().toString());
        if (rb1.isChecked()) {
            int suma = v1 + v2;
            tv1.setText(String.valueOf(suma));
        } else if (rb2.isChecked()) {
            int resta = v1 - v2;
            tv1.setText(String.valueOf(resta));
        }
    }
}

```

El resultado final del programa en el dispositivo es:



Este proyecto lo puede descargar en un zip desde este enlace: proyecto044.zip

[Retornar](#)

42 - Internacionalización y archivo strings.xml

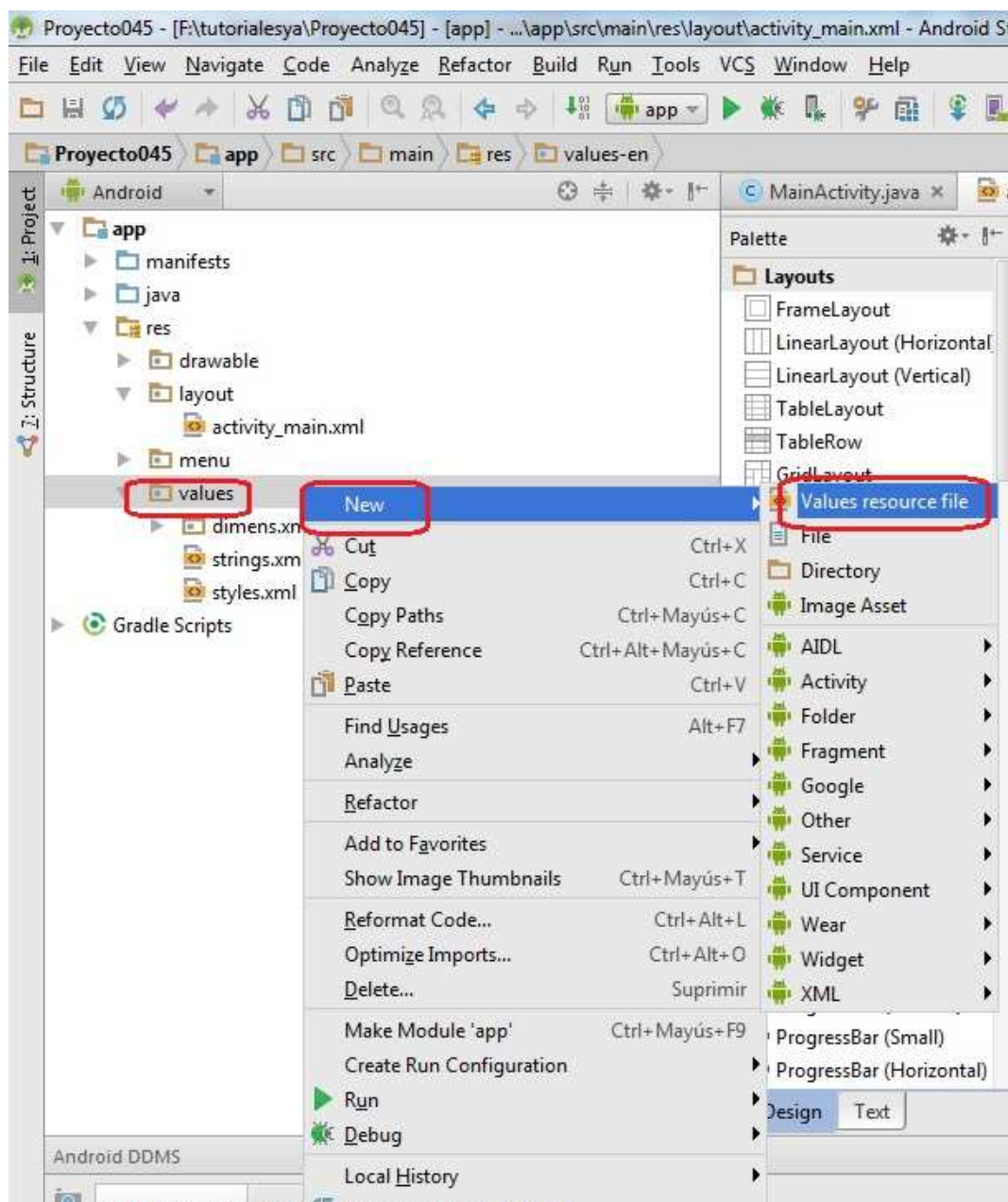
La internacionalización de un programa en Android se resuelve implementando varios archivos strings.xml (uno por cada idioma que necesitemos implementar)

Problema:

Crear un nuevo proyecto llamado Proyecto045 y confeccionar el mismo problema del concepto anterior (Proyecto044) donde estudiamos el archivo strings.xml.

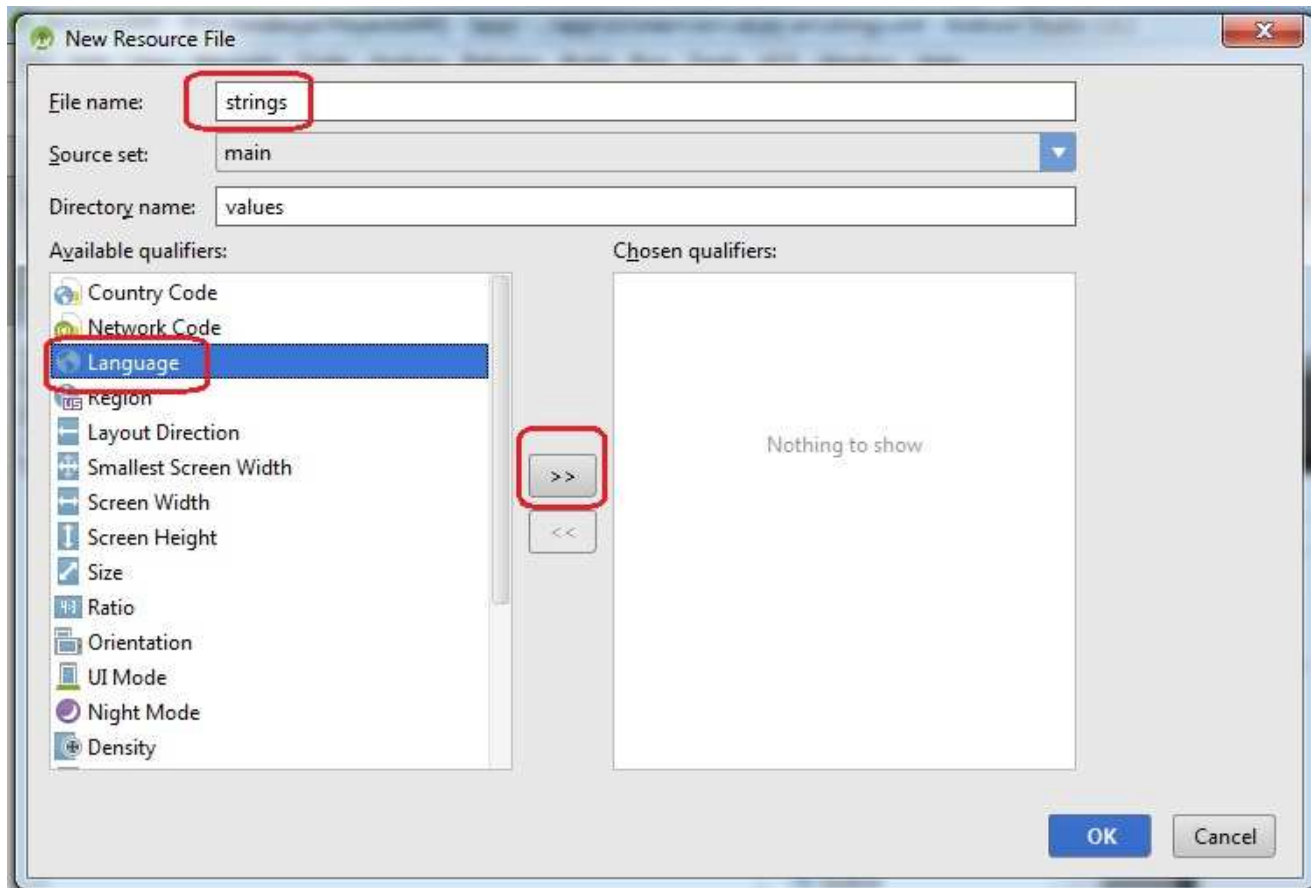
Ahora haremos que muestre su interfaz en castellano o ingles según la configuración del idioma seleccionado en el dispositivo.

Presionamos el botón derecho del mouse sobre la carpeta "values" y seleccionamos "Values Resource File":

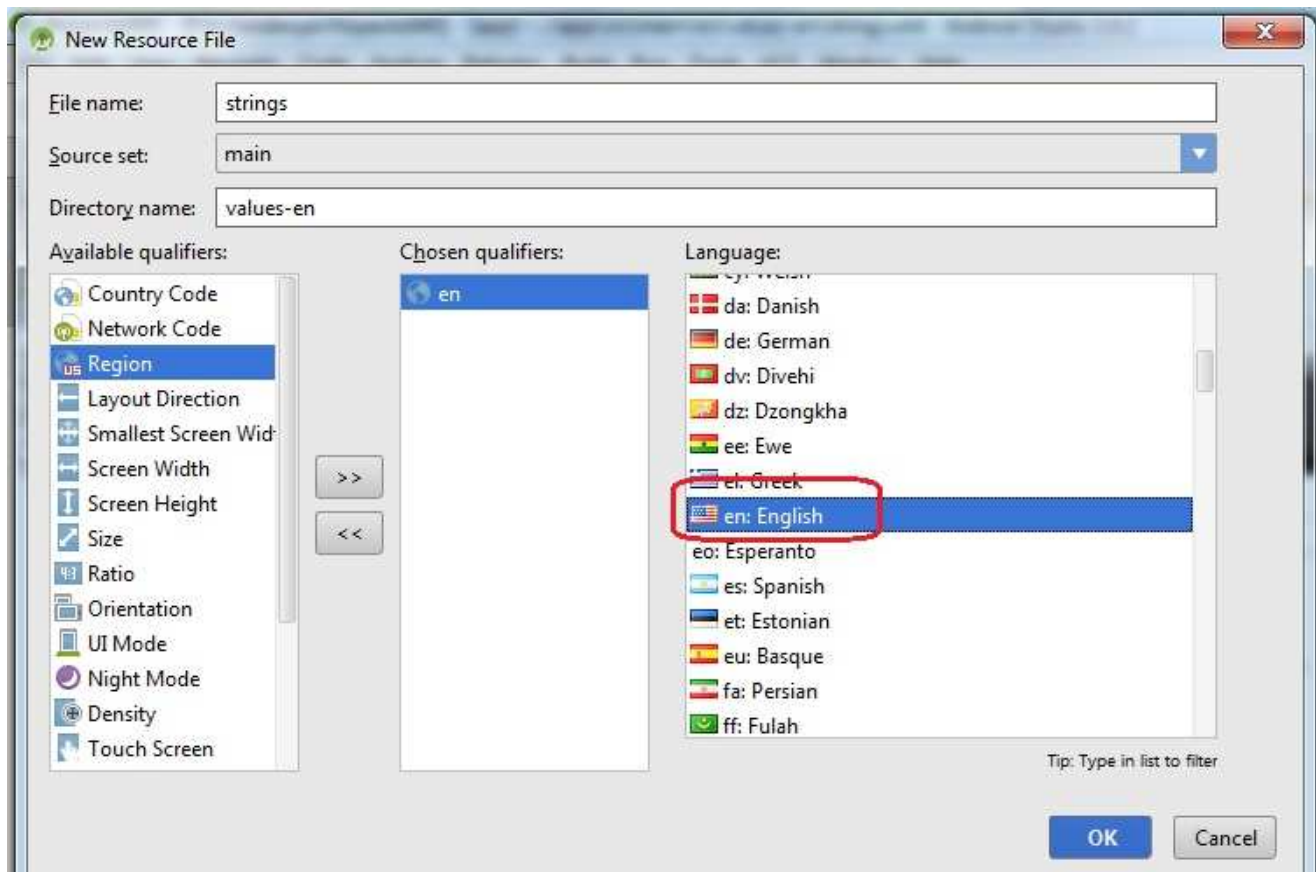


Aparece un diálogo donde debemos indicar que se creará un archivo llamado "strings"

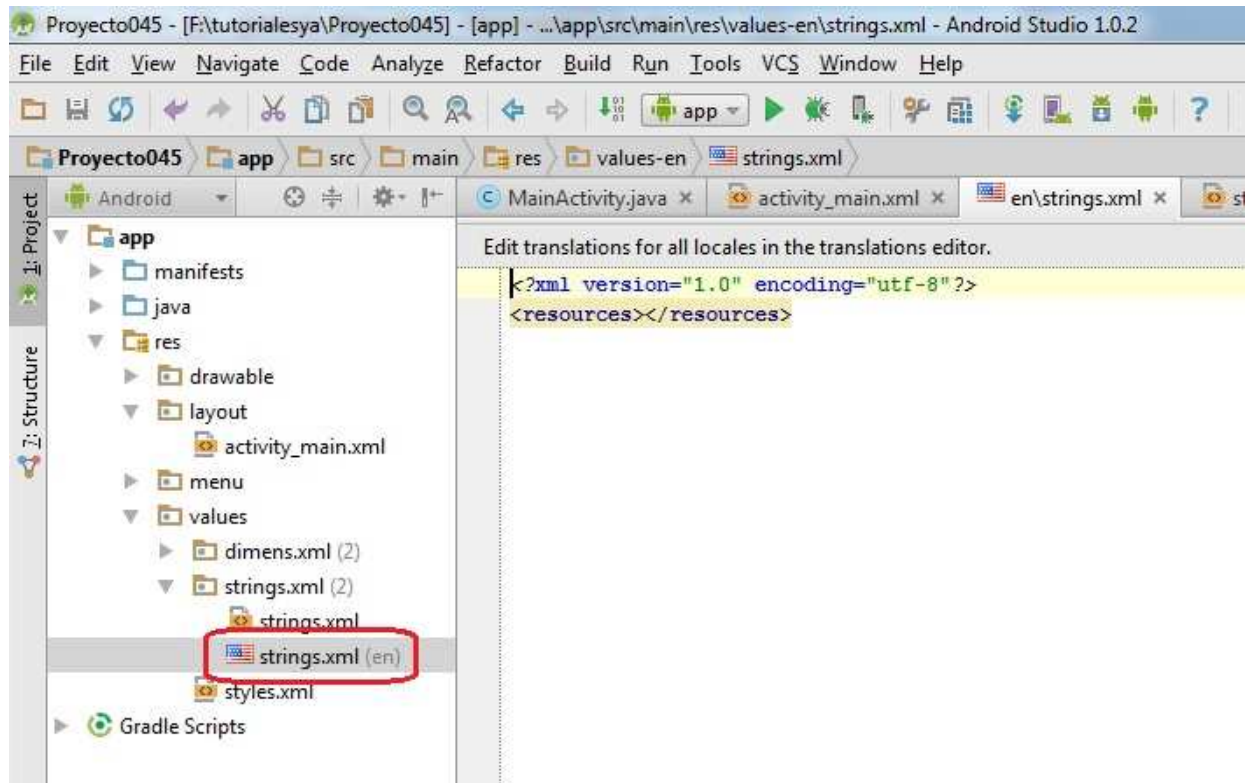
seleccionamos "Lenguaje" presionamos el botón ">>":



Luego seleccionamos en:English:



Ya tenemos el archivo strings.xml donde debemos disponer las traducciones del archivo strings.xml original:



Procedemos a insertar el siguiente texto:

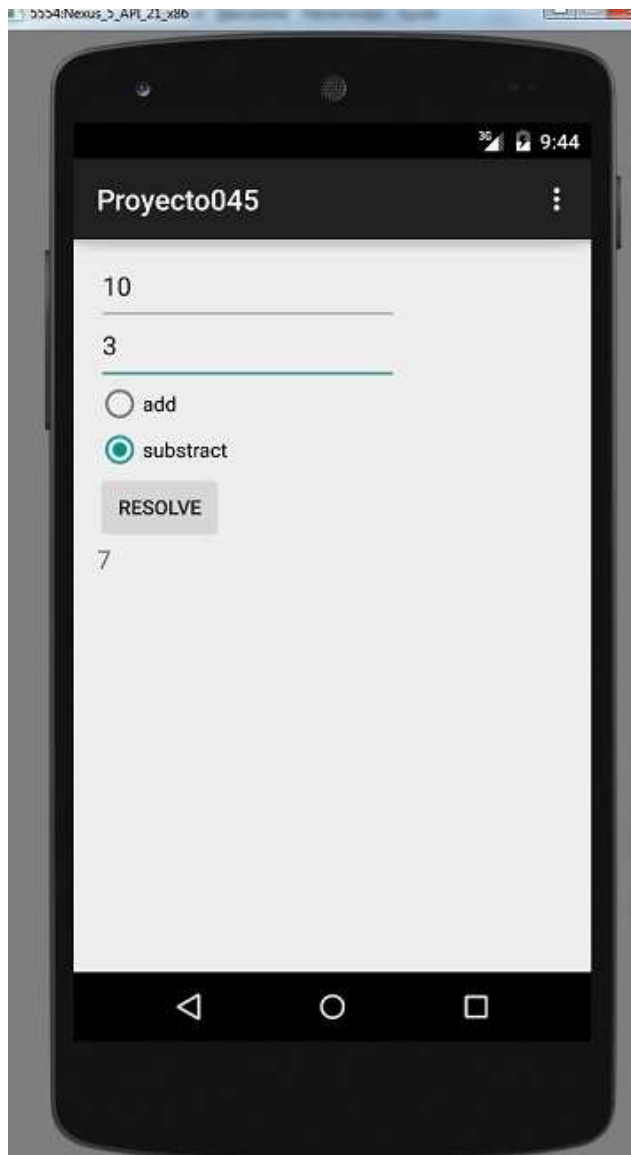
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Proyecto045</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="radiosuma">add</string>
    <string name="radioresta">subtract</string>
    <string name="botonoperacion">resolve</string>

</resources>
```

Solamente con esto ya tenemos nuestra aplicación funcionando en dos idiomas. Para probar debemos acceder al emulador de Android y configurarlo en inglés (tecla home -> tecla menu -> configuración > Idioma y teclado > Seleccionar idioma -> "English (United States)")

Luego de esto lanzamos la aplicación y deberemos ver:



Este proyecto lo puede descargar en un zip desde este enlace: [proyecto045.zip](#)

[Retornar](#)

43 - Localización y archivo strings.xml

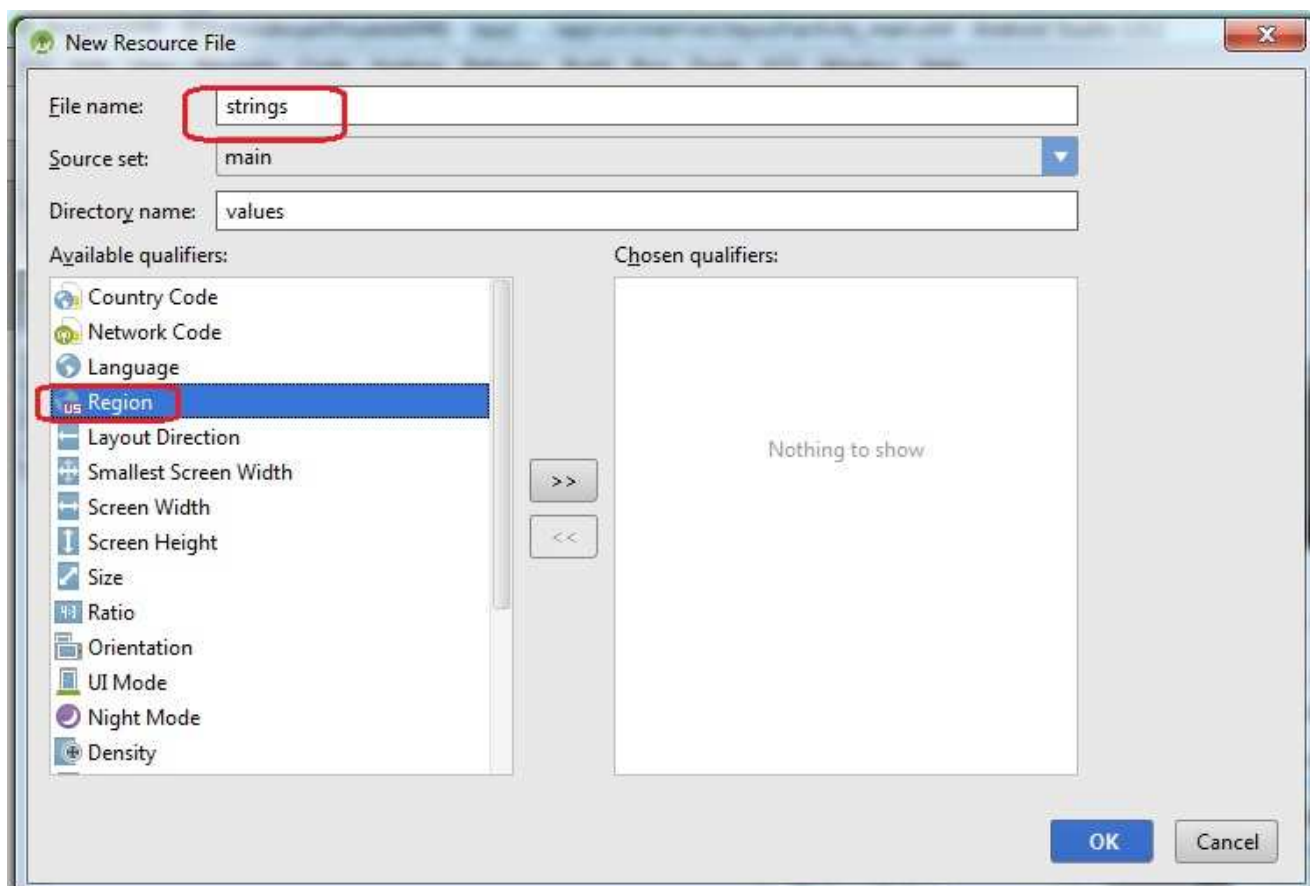
Hemos visto si queremos tener traducida nuestra aplicación a varios idiomas debemos utilizar el concepto de internacionalización. Pero hay muchos casos donde queremos hilar más fino. Por ejemplo el ingles de Estados Unidos no es exactamente igual que el ingles de Inglaterra o de Australia. Para estas situaciones podemos crear varios archivos strings.xml para distintas regiones que tienen el mismo idioma.

Problema:

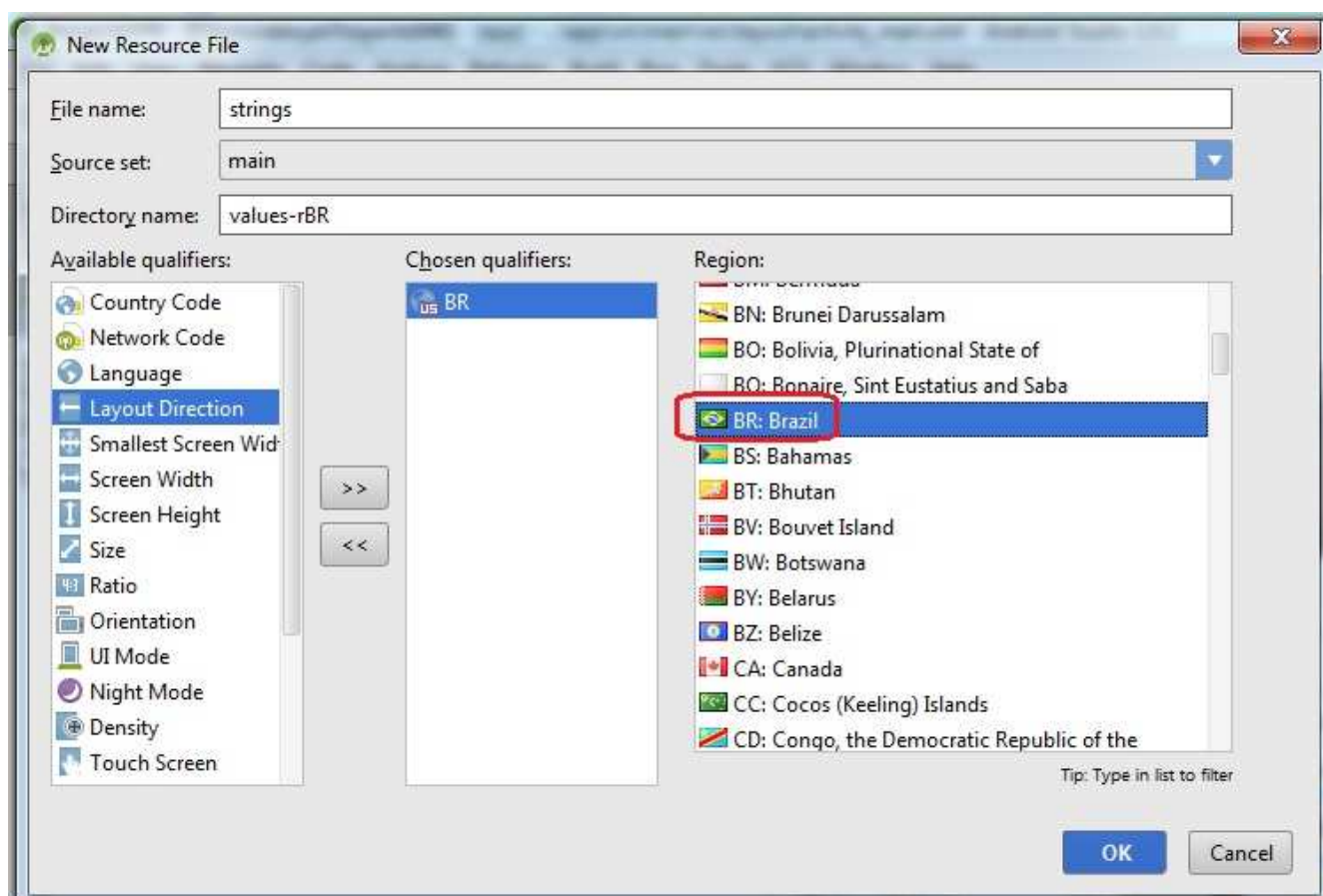
Crear un nuevo proyecto Proyecto046 basado en el proyecto045 de tal manera que muestre su interfaz no solo para el idioma por defecto (español) e ingles, sino para el portugués de Brasil y el portugués de Portugal.

Las abreviaturas para los distintos lenguajes los podemos ver en la página [ISO 639-1 Code](#) (segunda columna) y para obtener las distintas regiones utilizamos la tabla [ISO 3166-1-alpha-2](#)

Para agregar un nuevo idioma de una región procedemos de la misma manera que para cuando seleccionamos un lenguaje: botón derecho sobre la carpeta values :New Values Resource File y en este diálogo especificamos que crearemos un archivo llamado strings.xml y seleccionaremos "Region":



Luego buscamos BR: Brazil y ya tenemos el archivo strings.xml para el portugués de Brasil:



En forma similar creamos el archivo strings.xml para el portugués de Portugal.

Luego tenemos que almacenar las traducciones respectivas en cada archivo xml de cada región.

[proyecto046.zip](#)

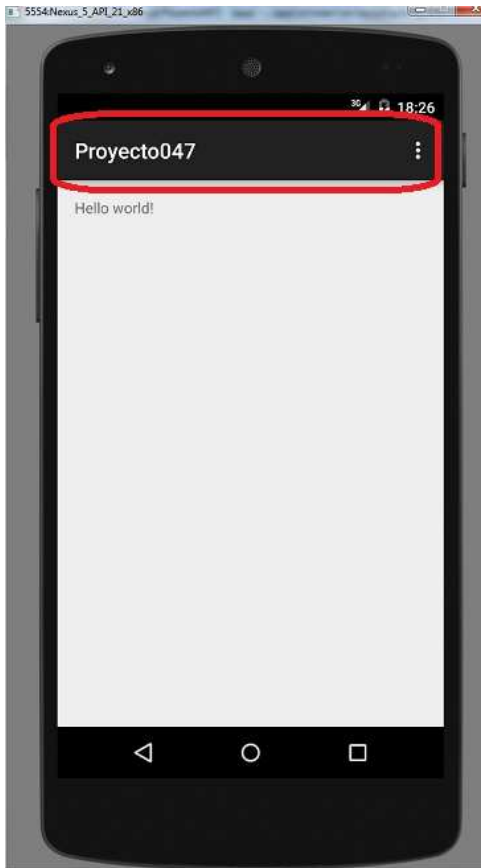
[Retornar](#)

44 - Componente ActionBar (Básica)

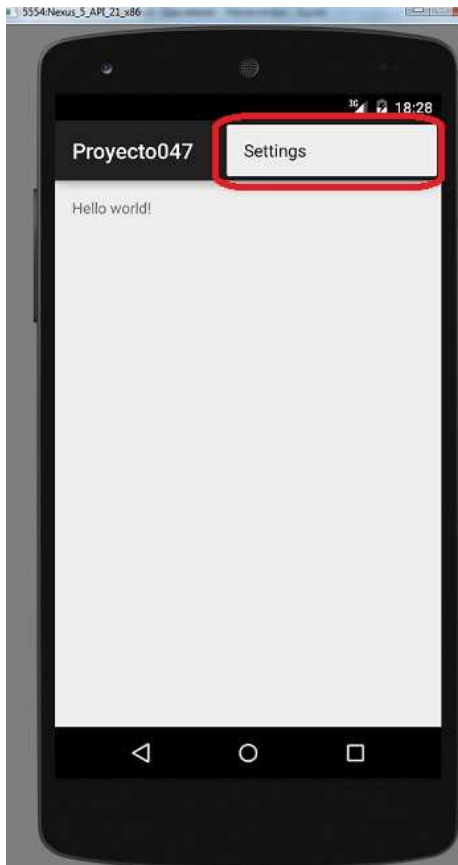
A partir de la versión 4.0 se introduce para el desarrollo de aplicaciones para celulares Android la componente ActionBar. La componente ActionBar es la barra que aparece en la parte superior de gran cantidad de aplicaciones. Esta componente busca estandarizar entre distintas aplicaciones desarrolladas para Android la ubicación de botones de acciones, menú de opciones etc.

Ahora cuando creamos con el Android Studio una aplicación mínima con la opción "Blank Activity" se genera por defecto el ActionBar.

La interfaz mínima del ActionBar es:



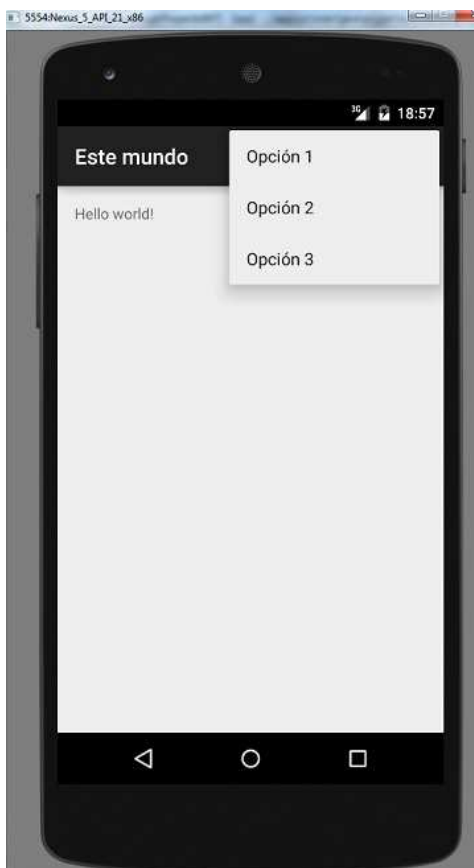
En el lado izquierdo aparece el nombre de la aplicación y del lado derecha aparece un botón que al ser presionado nos muestra un menú desplegable:



Problema:

Confeccionar una aplicación que muestre en el ActionBar el título "Este mundo" y luego tres opciones en el menú desplegable del ActionBar.

La interfaz del ActionBar y el menú desplegable debe ser similar a:



Veamos los pasos que debemos dar para obtener este resultado:

1. El título del ActionBar lo debemos modificar abriendo el archivo strings que se encuentra en la carpeta res/values y su contenido por defecto al crear el proyecto es:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Proyecto047</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

Podemos cambiar el contenido de la constante "app_name":

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Este mundo</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

Recordemos que este archivo de recursos se utiliza para agrupar todos los mensajes que aparecen en pantalla y facilitar la implementación de aplicaciones para varios idiomas independiente de los algoritmos de la misma.

En este archivo hemos creado un string llamado "app_name" con el valor "Este mundo".

Luego este string se asocia con el título de la aplicación con la propiedad label en el archivo AndroidManifest.xml:

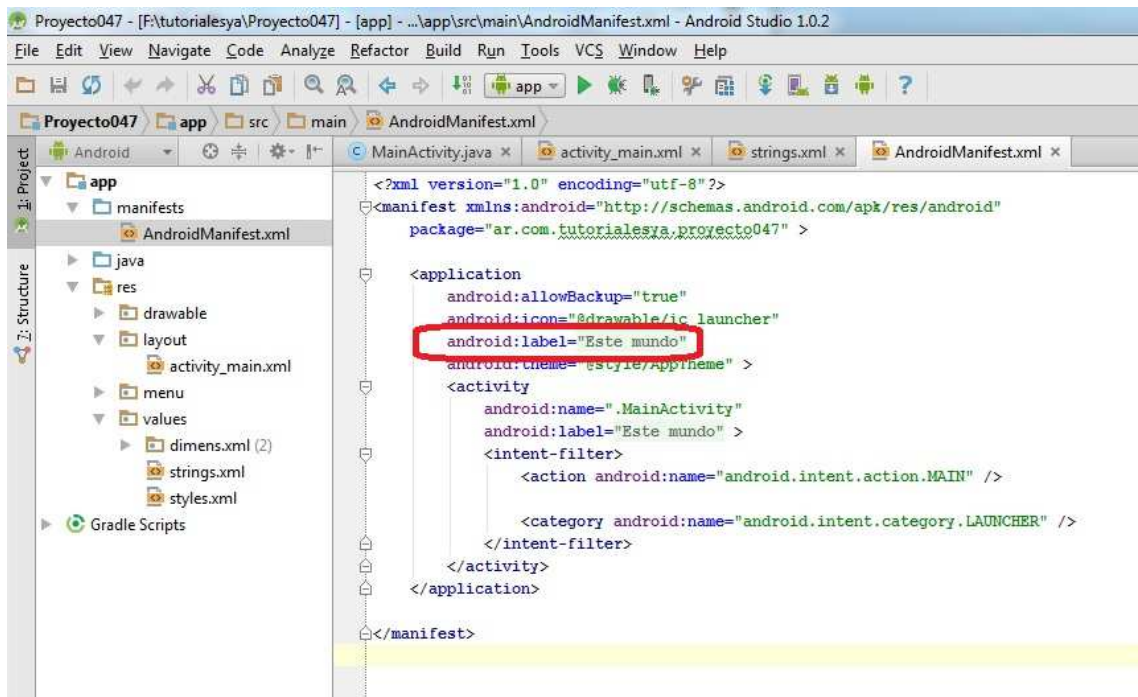
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ar.com.tutorialesya.proyecto047" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

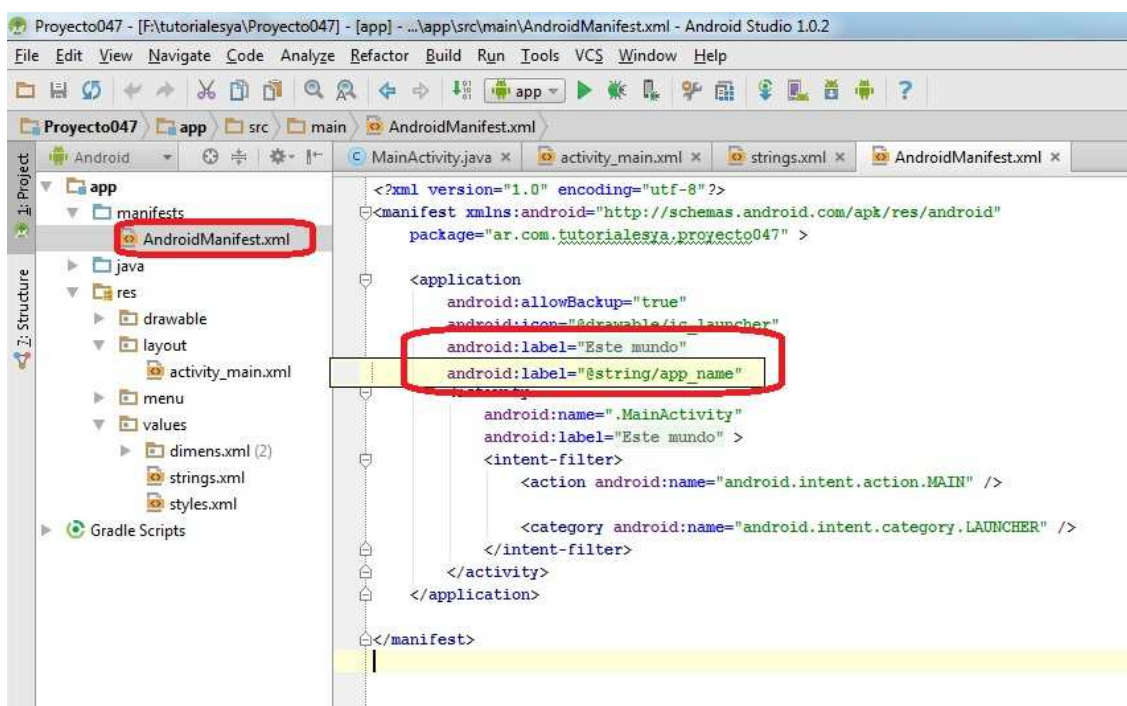
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Desde el Editor del Android Studio cuando vemos el archivo AndroidManifest.xml nos sustituye visualmente las constantes por los valores de las constantes para ver que realmente se almacena:



Pero si posicionamos el mouse sobre el valor veremos que está definida una constante con la sintaxis @string/[nombre de constante]:



Y si hacemos clic veremos el contenido real del valor con el que se inicializa la propiedad.

En este archivo XML ya está inicializada la propiedad label de la marca application con el valor definido en el archivo xml (como vemos acá hacemos referencia al string app_name):

```
android:label="@string/app_name"
```

2. Ahora tenemos que definir las opciones de nuestro menú desplegable, para esto abrimos el archivo menu_main.xml de la carpeta res/menu y definimos las tres opciones que contendrá nuestro menú y eliminamos la que define por defecto cuando creamos un proyecto desde Android Studio:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools" tools:context=".MainActivity">

    <item
        android:id="@+id/opcion1"
        app:showAsAction="never"
        android:title="Opción 1"/>
```

```

<item
    android:id="@+id/opcion2"
    app:showAsAction="never"
    android:title="Opción 2"/>
<item
    android:id="@+id/opcion3"
    app:showAsAction="never"
    android:title="Opción 3"/>
</menu>

```

Como necesitamos tres opciones definimos tres marcas item e inicializamos las propiedades id, showAsAction (indicando con el valor never que deben aparecer en el menú desplegable) y mediante la propiedad title el texto que queremos que muestre (como vemos al asignar directamente el texto a la propiedad title hay que tener en cuenta que la mejor práctica es definir el mensaje en el archivo strings.xml y acá hacer referencia a dicho string mediante un @string/[constante], por el momento para hacerlo más corto y directo al problema lo hacemos con el texto a mostrar directamente)

3. La funcionalidad de nuestro programa será mostrar un Toast cuando se seleccione alguno de las opciones del menú. El código java de la clase debe ser:

```

package ar.com.tutorialesya.proyecto047;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement

        // Debemos borrar este if que creó el Android Studio con la opción por defecto
        // if (id == R.id.action_settings) {
        //     return true;
        // }

        if (id==R.id.opcion1) {
            Toast.makeText(this,"Se seleccionó la primer opción",Toast.LENGTH_LONG).show();
        }

        if (id==R.id.opcion2) {
            Toast.makeText(this,"Se seleccionó la segunda opción",Toast.LENGTH_LONG).show();
        }

        if (id==R.id.opcion3) {
            Toast.makeText(this,"Se seleccionó la tercer opción",Toast.LENGTH_LONG).show();
        }
    }
}

```

```

    }

    return super.onOptionsItemSelected(item);
}
}

```

Como podemos observar hemos modificado el método `onOptionsItemSelected` que se ejecuta cada vez que el usuario selecciona una opción del menú:

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement

    // Debemos borrar este if que creó el Android Studio con la opción por defecto.
    // if (id == R.id.action_settings) {
    //     return true;
    // }

    if (id==R.id.opcion1) {
        Toast.makeText(this,"Se seleccionó la primer opción",Toast.LENGTH_LONG).show();
    }

    if (id==R.id.opcion2) {
        Toast.makeText(this,"Se seleccionó la segunda opción",Toast.LENGTH_LONG).show();
    }

    if (id==R.id.opcion3) {
        Toast.makeText(this,"Se seleccionó la tercer opción",Toast.LENGTH_LONG).show();
    }

    return super.onOptionsItemSelected(item);
}
}

```

Podemos ahora entender el objetivo del método `onCreateOptionsMenu` que se genera al crear el proyecto. En el mismo se procesa el archivo `menu_main.xml` donde definimos el menú de opciones:

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

```

Hemos borrado las líneas:

```

Debemos borrar este if que creó el Android Studio con la opción por defecto.
if (id == R.id.action_settings) {
    return true;
}

```

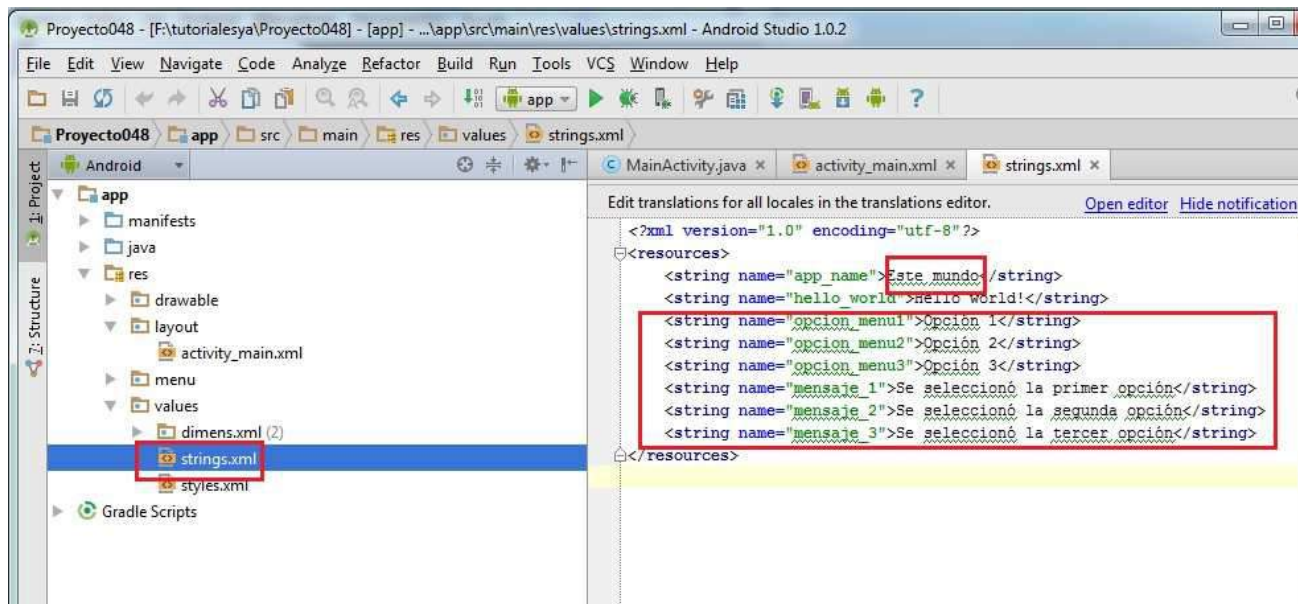
Ya que es la opción que crea por defecto el Android Studio en nuestro esqueleto básico del proyecto.

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto047.zip](#)

Problema:

Confeccionar el mismo problema del ActionBar con el menú de opciones pero utilizando constantes para cada opción del menú y también constantes para los tres mensajes que aparecen cuando se seleccionan cada una de las opciones.

Creemos un proyecto llamado "Proyecto048" y creemos primero el archivo `strings.xml` con todas las constantes necesarias de nuestro programa:



Es decir modificamos el valor de `app_name` y creamos seis constantes más, tres para las etiquetas de las opciones del menú y tres para los mensajes que deben aparecer cuando se presionen las opciones.

El archivo `menu_main` ahora queda codificado así:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" tools:context=".MainActivity">

    <item
        android:id="@+id/opcion1"
        app:showAsAction="never"
        android:title="@string/opcion_menu1"/>
    <item
        android:id="@+id/opcion2"
        app:showAsAction="never"
        android:title="@string/opcion_menu2"/>
    <item
        android:id="@+id/opcion3"
        app:showAsAction="never"
        android:title="@string/opcion_menu3"/>
</menu>
```

Es decir hemos inicializado cada propiedad `title` de los `item` con los nombres de las constantes que creamos en el archivo `strings.xml`.

Finalmente el código fuente de la aplicación sufre los siguientes cambios para utilizar las constantes del archivo `strings.xml` cuando se deben mostrar los mensajes:

```
package ar.com.tutorialesya.proyecto048;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

    @Override
```

```

public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    if (id==R.id.opcion1) {
        Toast.makeText(this,getString(R.string.mensaje_1),Toast.LENGTH_LONG).show();
    }

    if (id==R.id.opcion2) {
        Toast.makeText(this,getString(R.string.mensaje_2),Toast.LENGTH_LONG).show();
    }

    if (id==R.id.opcion3) {
        Toast.makeText(this,getString(R.string.mensaje_3),Toast.LENGTH_LONG).show();
    }

    return super.onOptionsItemSelected(item);
}
}

```

Lo nuevo aquí es ver que antes para mostrar el mensaje directamente en el código Java hacíamos:

```

if (id==R.id.opcion1) {
    Toast.makeText(this,"Se seleccionó la primer opción",Toast.LENGTH_LONG).show();
}

```

Ahora vemos si queremos extraer el dato de la constante que se encuentra en el archivo strings.xml lo debemos hacer llamando al método getString y pasando la referencia al nombre de la constante que se encuentra en la clase R y dentro de esta en una clase interna llamada string:

```

if (id==R.id.opcion1) {
    Toast.makeText(this,getString(R.string.mensaje_1),Toast.LENGTH_LONG).show();
}

```

[Retornar](#)

45 - Componente ActionBar (Botones de acción)

Ahora veremos otra característica que podemos agregarle a la barra de acción (ActionBar). Como habíamos visto la barra de acción muestra a la izquierda un título y del lado derecho un botón que muestra un menú desplegable. Podemos disponer opciones de dicho menú que se muestren directamente en la barra de acción para que tengamos acceso a dichas opciones en forma directa sin tener que abrir el menú desplegable.

Problema:

Confeccionar una aplicación (Proyecto049) que muestre como título "ActionBar" y luego dos botones de acción y tres opciones en el menú desplegable del ActionBar.

La interfaz del ActionBar, el título, los botones de acción y el menú desplegable debe ser similar a:

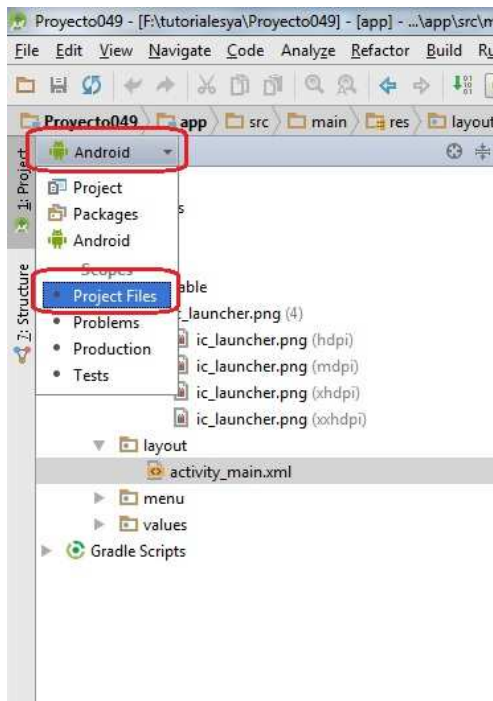


Como podemos observar hay dos botones de acción siempre visibles (podemos mostrar normalmente su ícono, pero si disponemos de dispositivos más grandes podemos mostrar un texto inclusive)

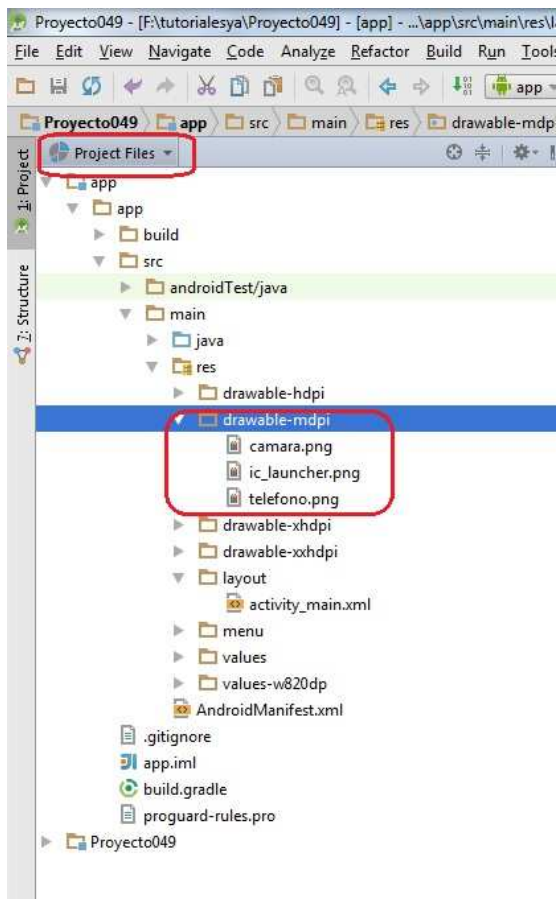
Veamos los pasos que debemos dar para obtener este resultado:

1. Lo primero es disponer dos imágenes de 32x32 píxeles en la carpeta drawable-mdpi, una que represente un teléfono y otra una cámara fotográfica (recordemos que los nombres de archivos deben estar en minúsculas, sin espacios, no empezar con un número y no contener caracteres especiales, salvo el guión bajo), podemos llamar a los archivos telefono.png y camara.png. Dispondremos solo estas dos imágenes en la carpeta drawable-mdpi.

Lo más fácil para insertar estos archivos a dicha carpeta es disponer la ventana de "Project" en vista de "Project File"



Y ahora ubicar la carpeta drawable-mdpi y copiar las dos imágenes:



2. El título del ActionBar lo debemos modificar abriendo el archivo strings que se encuentra en la carpeta res/values y su contenido debe ser:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">ActionBar</string>

</resources>
```

Este archivo de recursos se utiliza para agrupar todos los mensajes que aparecen en pantalla y facilitar la implementación de aplicaciones para varios idiomas independiente de los algoritmos de la misma.

En este archivo hemos creado un string llamado "app_name" con el valor "ActionBar".
Luego este string se asocia con el título de la aplicación mediante el archivo AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ar.com.tutorialesya.proyecto049" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            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>
</manifest>
```

En este archivo XML ya está inicializada la propiedad label de la marca application con el valor definido en el archivo xml (como vemos acá hacemos referencia al string app_name):

```
        android:label="@string/app_name"
```

3. Ahora tenemos que definir las opciones de nuestro menú desplegable y los dos botones de acción, para esto abrimos el archivo menu_main.xml de la carpeta res/menu y definimos las cinco opciones que contendrá nuestro menú y botones de acción:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" tools:context=".MainActivity">

    <item
        android:id="@+id/telefono"
        app:showAsAction="always"
        android:icon="@drawable/telefono"
        android:title="telefono"/>
    <item
        android:id="@+id/camara"
        app:showAsAction="always"
        android:icon="@drawable/camara"
        android:title="camara"/>
    <item
        android:id="@+id/opcion1"
        app:showAsAction="never"
        android:title="Opción 1"/>
    <item
        android:id="@+id/opcion2"
        app:showAsAction="never"
        android:title="Opción 2"/>
    <item
        android:id="@+id/opcion3"
        app:showAsAction="never"
        android:title="Opción 3"/>
</menu>
```

Lo nuevo con respecto al concepto anterior es la definición de dos item que inicializamos la propiedad showAsAction con el valor "always" con lo que indicamos que queremos que siempre se muestre visible esta opción en la barra de acción, y además definimos la propiedad icon con el nombre de la imagen que agregamos a la carpeta drawable-mdpi:

```
    <item
        android:id="@+id/telefono"
        android:showAsAction="always"
        android:icon="@drawable/telefono"
        android:title="telefono"/>
```

4. La funcionalidad de nuestro programa será mostrar un Toast cuando se seleccione alguno de las opciones del menú o un botón de acción. El código java de la clase MainActivity debe ser:

```

package ar.com.tutorialesya.proyecto049;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        if (id == R.id.telefono) {
            Toast.makeText(this, "Se presionó el ícono del teléfono", Toast.LENGTH_LONG).show();
            return true;
        }
        if (id == R.id.camara) {
            Toast.makeText(this, "Se presionó el ícono de la cámara", Toast.LENGTH_LONG).show();
            return true;
        }
        if (id == R.id.opcion1) {
            Toast.makeText(this, "Se presionó la opción 1 del menú", Toast.LENGTH_LONG).show();
            return true;
        }
        if (id == R.id.opcion2) {
            Toast.makeText(this, "Se presionó la opción 2 del menú", Toast.LENGTH_LONG).show();
            return true;
        }
        if (id == R.id.opcion3) {
            Toast.makeText(this, "Se presionó la opción 3 del menú", Toast.LENGTH_LONG).show();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}

```

Como podemos observar la captura del clic de los botones de acción es idéntico a las opciones del menú desplegable:

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    if (id == R.id.telefono) {
        Toast.makeText(this, "Se presionó el ícono del teléfono", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.camara) {

```

```

        Toast.makeText(this, "Se presionó el ícono de la cámara", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.opcion1) {
        Toast.makeText(this, "Se presionó la opción 1 del menú", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.opcion2) {
        Toast.makeText(this, "Se presionó la opción 2 del menú", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.opcion3) {
        Toast.makeText(this, "Se presionó la opción 3 del menú", Toast.LENGTH_LONG).show();
        return true;
    }
    return super.onOptionsItemSelected(item);
}

```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto049.zip](#)

Problema Propuesto

Volver a codificar el problema anterior para mostrar dos botones y tres opciones en el menú del ActionBar localizando todos los mensajes en el archivo strings.xml

La solución a este problema lo puede descargar en un zip desde este enlace: [proyecto050.zip](#)

[Retornar](#)

46 - Componente ActionBar (Ocultarlo y mostrarlo)

La barra de acción se muestra en la parte superior para acceder a las opciones de nuestro programa, pero hay situaciones donde necesitemos ocultarla para tener más espacio. Para ocultar y mostrar el ActionBar disponemos de dos métodos:

```
show()
hide()
```

Problema:

Confeccionar una aplicación (Proyecto051) que mediante dos botones se permita ocultar el ActionBar o mostrarlo. Disponer tres opciones en el menú desplegable del ActionBar.

El código fuente de la aplicación es:

```
package ar.com.tutorialesya.proyecto051;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        if (id==R.id.opcion1) {
            Toast.makeText(this,"Se seleccionó la primer opción",Toast.LENGTH_LONG).show();
        }

        if (id==R.id.opcion2) {
            Toast.makeText(this,"Se seleccionó la segunda opción",Toast.LENGTH_LONG).show();
        }

        if (id==R.id.opcion3) {
            Toast.makeText(this,"Se seleccionó la tercer opción", Toast.LENGTH_LONG).show();
        }
        return super.onOptionsItemSelected(item);
    }

    public void ocultar(View v) {
        getSupportActionBar().hide();
    }
}
```

```
public void mostrar(View v) {  
    getSupportActionBar().show();  
}  
}
```

Luego cuando lo ejecutamos podemos observar según el botón que presionamos el ActionBar se hace visible o se oculta:



O aparece oculta:



Para poder ocultar el ActionBar debemos obtener la referencia del objeto mediante el método `getSupportActionBar()` que se trata de un método heredado de la clase `ActionBarActivity`:

```
public void ocultar(View v) {  
    getSupportActionBar().hide();  
}
```

Para volver a hacer visible el ActionBar llamamos al método `show()`, esto ocurre cuando presionamos el segundo botón:

```
public void mostrar(View v) {  
    getSupportActionBar().show();  
}
```

Este proyecto lo puede descargar en un zip desde este enlace: [proyecto051.zip](#)

[Retornar](#)