

Ciclo 4

Fundamentos de programación

Aplicaciones móviles Android con Java

Sesión 3: Creación de Interfaz Grafica de Usuario(GUI)

Programa Ciencias de la Computación e Inteligencia Artificial
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda
Bogotá

Agenda

1. Concepto General
2. Descripción General de un proyecto
3. Estructura del proyecto <<MiPrimerPrograma>>
4. Componentes de una aplicación
5. Actividad (Activity)
6. Vista (View)
7. Layout
8. Actividad principal (MainActivity)
9. Carga del recurso XLM
10. Eventos de Entrada
11. Ejercicio: suma
12. Menús: Carga del recurso XLM

1. Concepto General

La **interfaz gráfica de usuario**, conocida también como **GUI** (del inglés *Graphical User Interface*), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Habitualmente **las acciones se realizan mediante manipulación directa, para facilitar la interacción del usuario con la computadora**. Surge como evolución de las interfaces de línea de comandos que se usaban para operar los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X, Aqua.

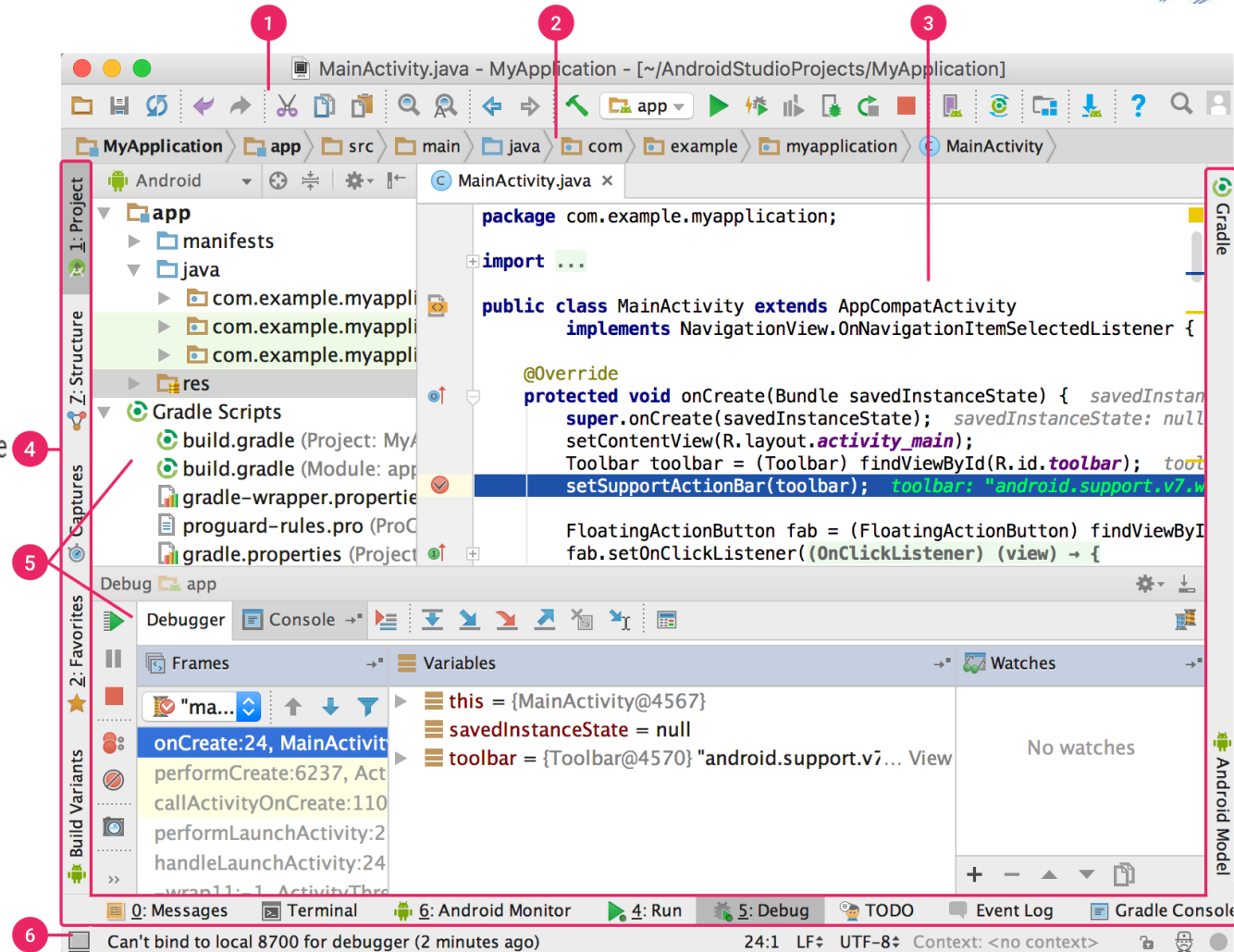
En el contexto del proceso de interacción persona-computadora, **la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático**.

Tomado de: https://es.wikipedia.org/wiki/Interfaz_gr%C3%A1fica_de_usuario

2. Descripción general de proyectos

Estructura del proyecto

- 1 La **barra de herramientas** te permite realizar una gran variedad de acciones, como ejecutar tu app e iniciar las herramientas de Android.
- 2 La **barra de navegación** te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana **Project**.
- 3 La **ventana del editor** es el área en la que puedes crear y modificar código. Según el tipo de actividad actual, el editor puede cambiar. Por ejemplo, cuando ves un archivo de diseño, el editor muestra el Editor de diseño.
- 4 La **barra de la ventana de herramientas** se encuentra afuera de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.
- 5 Las **ventanas de herramientas** te brindan acceso a tareas específicas, como la administración de proyectos, la búsqueda, el control de versiones, entre otras. Puedes expandirlas y contraerlas.
- 6 En la **barra de estado**, se muestra el estado de tu proyecto y el IDE, además de advertencias o mensajes.



Tomado de: https://developer.android.com/studio/intro#project_structure

#Misión TIC 2022  <https://developer.android.com/studio/projects?hl=es-419>



3. Estructura del proyecto: MiPrimerPrograma

Retomamos lo visto en la **Sesión 1**, para desarrollar la **GUI** de esta sesión:

En la ventana **Project** (**View > Tool Windows > Project**), desplegamos para trabajar sobre la **actividad principal**.

app > java > usa.sesion1.miprimerprograma > MainActivity

Esta es la actividad principal. Es el punto de entrada de tu app. Cuando compilas y ejecutas la app, el sistema inicia una instancia del elemento Activity y carga su diseño.

app > res > layout > activity_main.xml

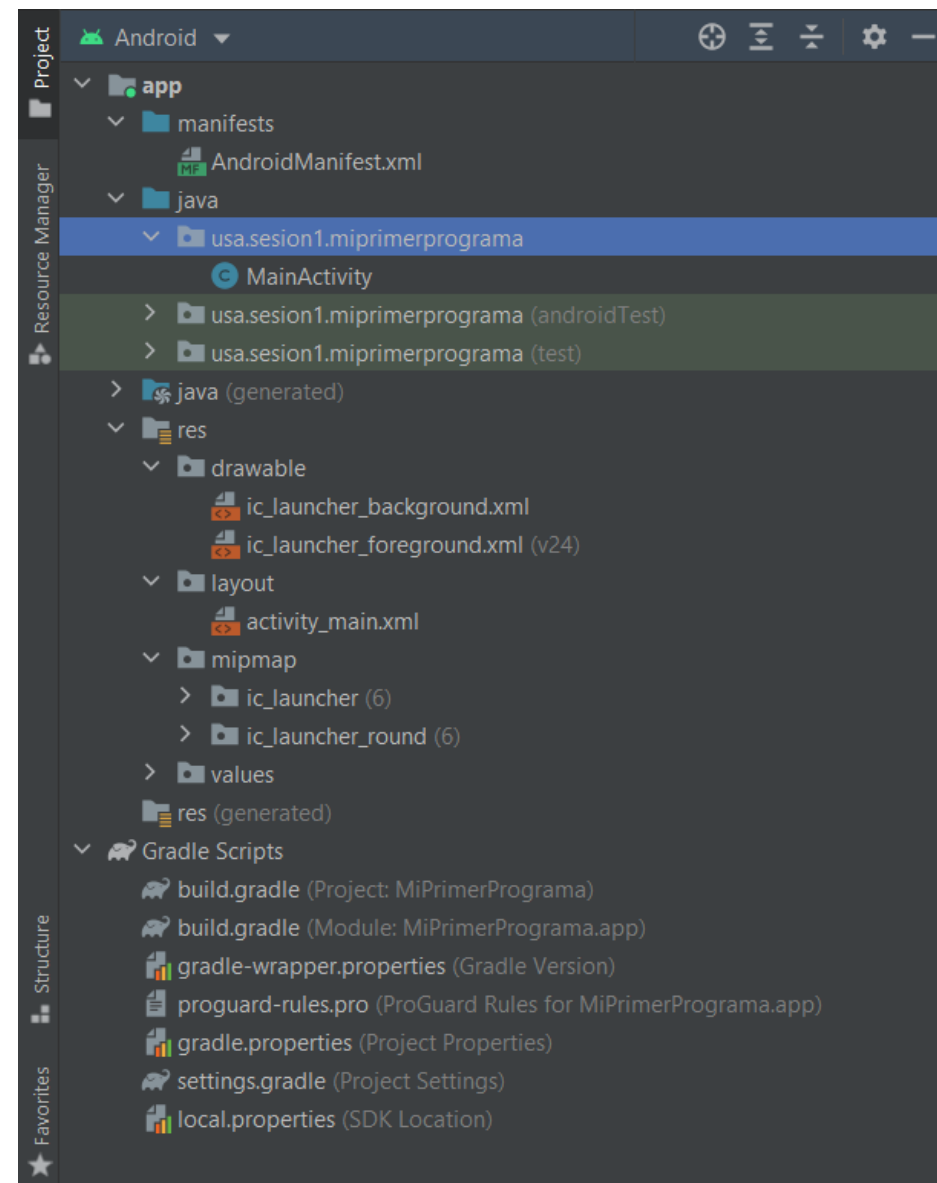
Este archivo en formato XML define el diseño de la interfaz de usuario (IU) de la actividad. Contiene un elemento **TextView** con el texto "**Hello, World!**".

app > manifests > AndroidManifest.xml

En el archivo de manifiesto, se describen las características fundamentales de la app, y se define cada uno de sus componentes.

Gradle Scripts > build.gradle

Hay dos archivos con este nombre: uno para el proyecto ("Project: MiPrimerPrograma") y otro para el módulo de la app ("Module: MiPrimerPrograma"). Cada módulo tiene su propio archivo build.gradle, pero este proyecto, por el momento, tiene un solo módulo. Usa el archivo build.gradle de cada módulo para controlar cómo el complemento de Gradle compila tu app.



4. Componentes de una aplicación



Estos componentes son esenciales para desarrollar una App móvil, principalmente la GUI de la aplicación, las primeras cuatro se verán en este **Modulo 1**, el resto en el siguiente modulo:

- Actividad (Activity)
- Vista (View)
- Layout
- Actividad principal (MainActivity)
- Menús
- Fragmentos (Fragment) se estudiarán en el **Modulo 2**
- Servicio (Service) se estudiarán en el **Modulo 2**
- Intención (Intent) se estudiarán en el **Modulo 2**
- Receptor de anuncios (Broadcast Receiver) se estudiarán en el **Modulo 2**
- Proveedores de Contenido (Content Provider) se estudiarán en el **Modulo 2**

Tomado de: <http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion>

5. Actividad (Activity)

Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización, mas conocidos como **pantallas de la aplicación**. En Android cada uno de estos elementos, o pantallas, se conoce como **actividad (Activity)**. Su función principal es la creación de la interfaz de usuario **UI**.

Una aplicación suele necesitar **varias actividades para crear la interfaz de usuario**. Las diferentes actividades creadas serán independientes entre sí, aunque todas trabajarán en un objetivo común.

Una actividad se define en una clase descendiente de **Activity** y utiliza un **Layout** que define su apariencia.

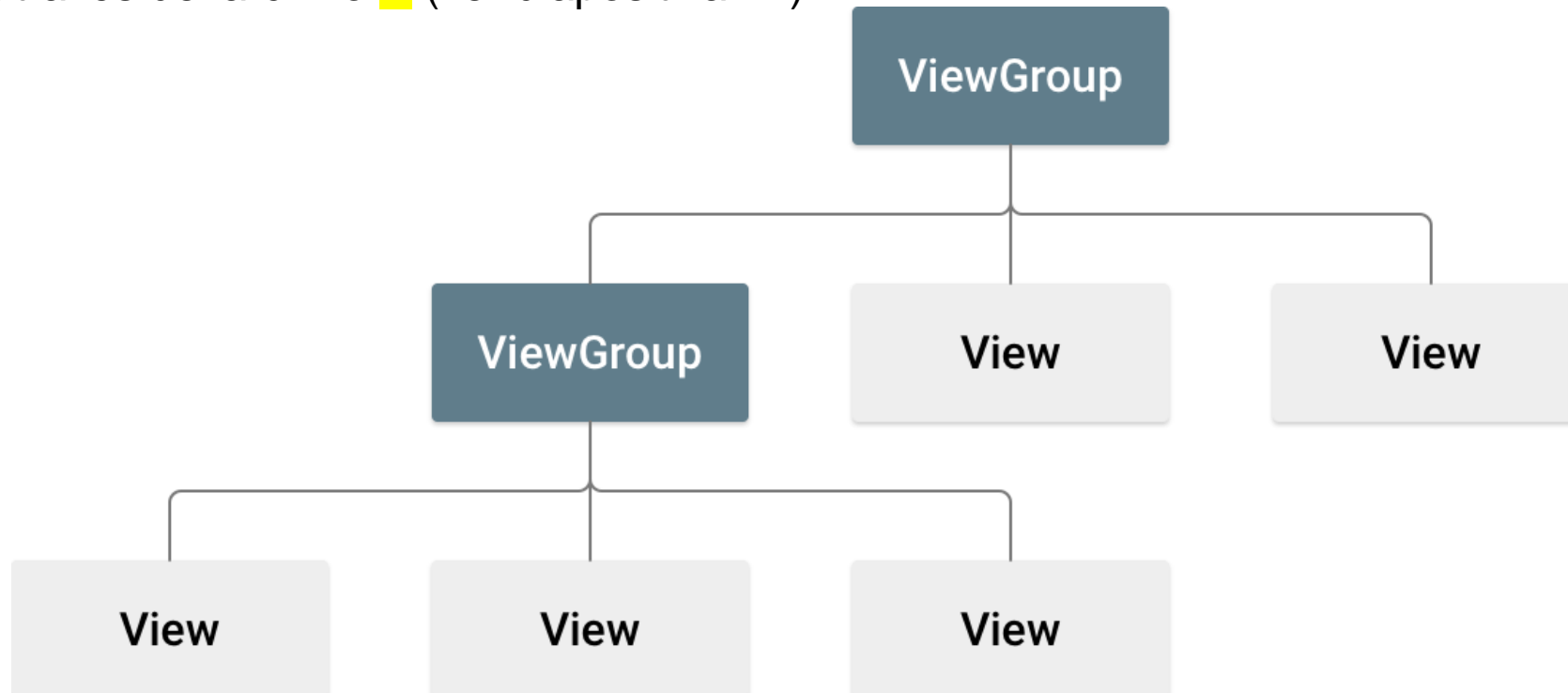
```
<manifest ... >
  <application ... >
    <activity android:name=".ExampleActivity" />
    ...
  </application ... >
  ...
</manifest >
```

Tomado de: <http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion>

Tomado de: <https://developer.android.com/guide/components/activities/intro-activities?hl=es>

6. Vista (View)

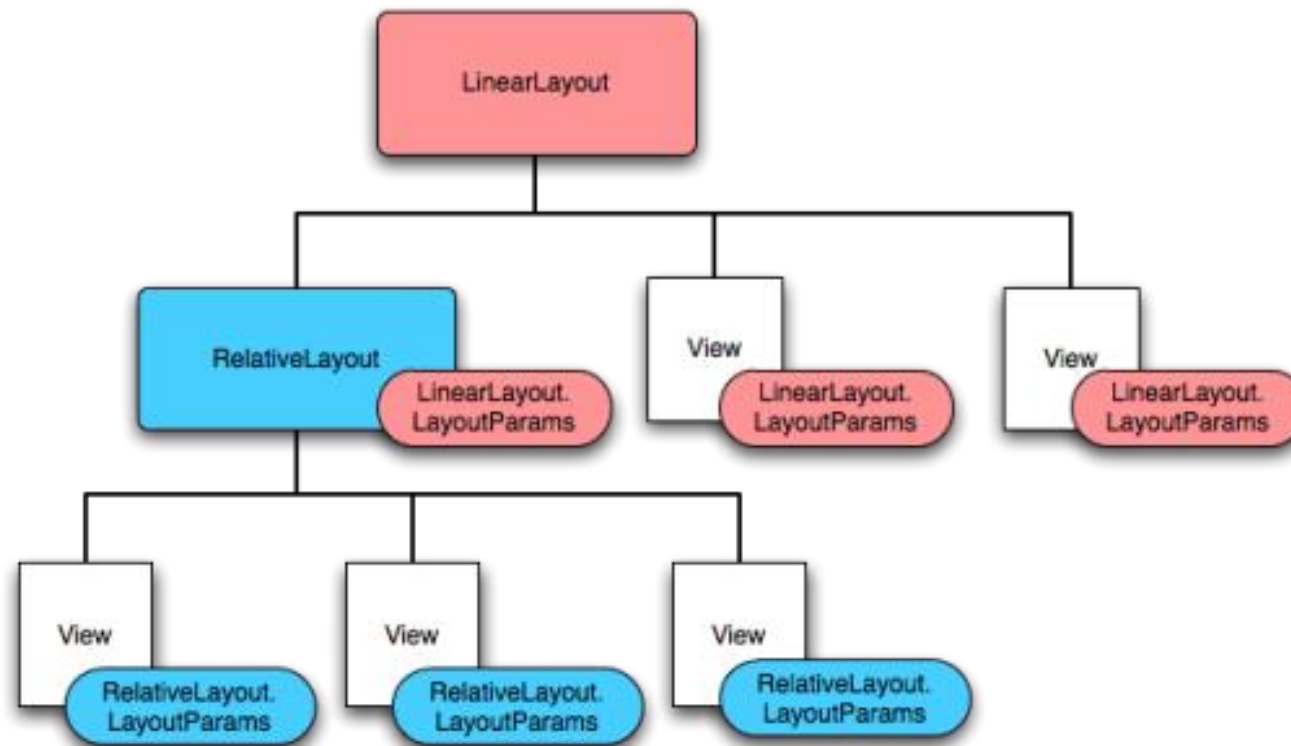
Las **vistas** son los elementos que componen la interfaz de usuario **UI** de una aplicación: por ejemplo, un **botón** o una **entrada de texto**. Todas las vistas van a ser objetos descendientes de la clase View, y por tanto, pueden ser definidas utilizando código Java. Sin embargo, lo habitual será definir las vistas utilizando **XML** y dejar que el sistema cree los objetos visuales por nosotros. Luego se conectan a un objeto java a través del archivo **R** (ver diapositiva 12).



<https://developer.android.com/training/basics/firstapp/building-ui?hl=es>

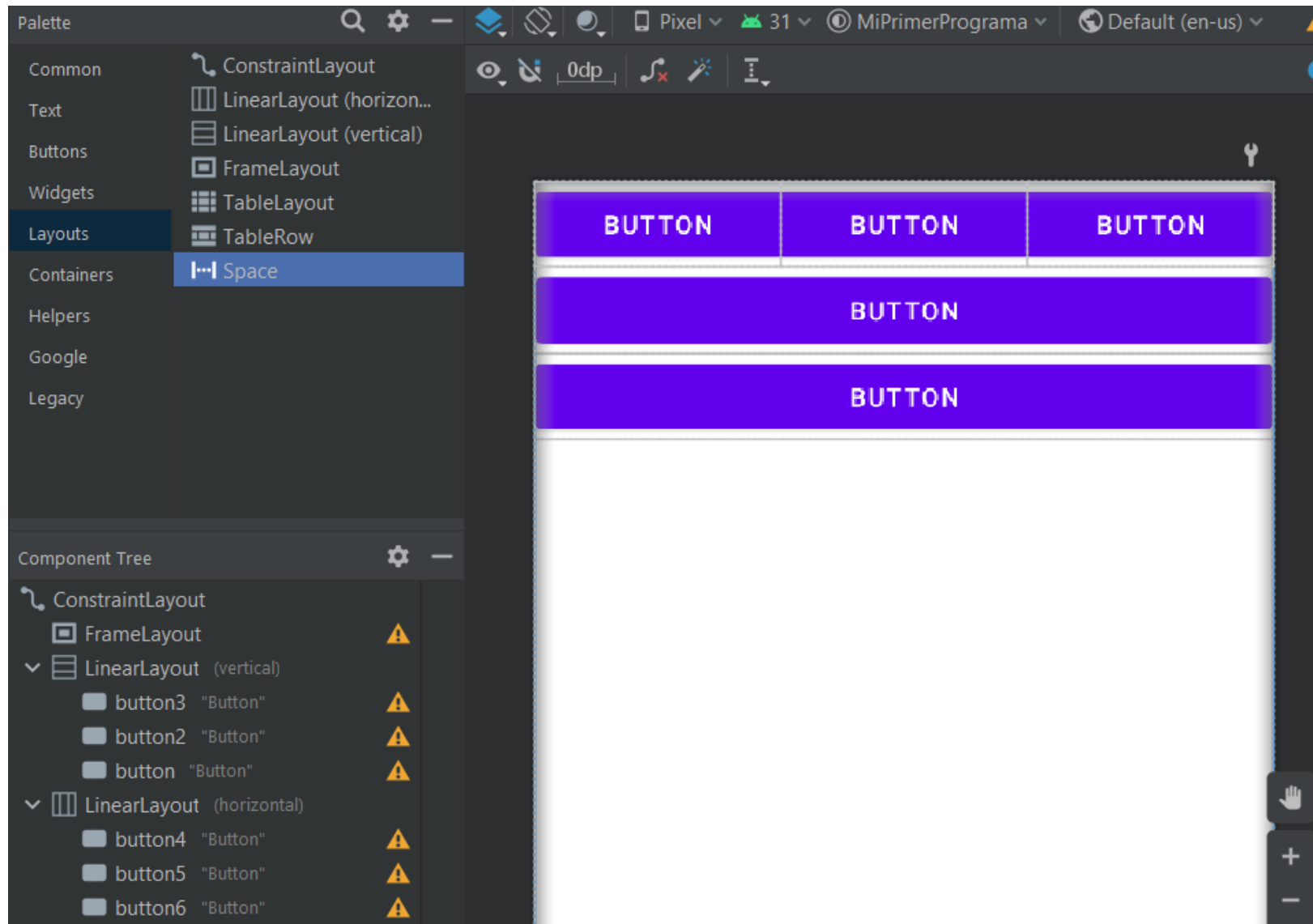
7. Layout

Un **layout** es un conjunto de vistas agrupadas de una determinada forma. Vamos a disponer de diferentes tipos de layouts para organizar las vistas de **forma lineal**, **en cuadrícula** o **indicando la posición absoluta de cada vista**. Los layouts también son objetos descendientes de la **clase View**. Igual que las vistas, los layouts pueden ser definidos en código, aunque la forma habitual de definirlos es utilizando código XML.



<https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419>

7. Layout



8. Actividad Principal (MainActivity)



La clase **Activity** es un componente clave de una app para Android, y la forma en que se inician y se crean las actividades es una parte fundamental del modelo de aplicación de la plataforma.

A diferencia de los paradigmas de programación en los que las apps se inician con un método **main()**, el sistema Android inicia el código en una instancia de **Activity** invocando métodos de devolución de llamada específicos que corresponden a etapas específicas de su ciclo de vida (se verá en el Modulo 2), el primero que veremos es el método **onCreate()**.

```
package usa.sesion1.miprimerprograma;

import ...

public class MainActivity extends AppCompatActivity {

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

Carga el recurso XML: **layout**

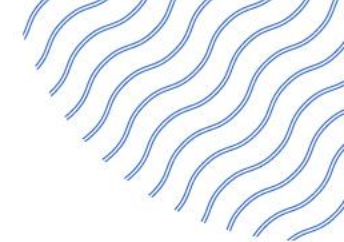
Cuando compilas tu aplicación, cada archivo XML de diseño se compila en un recurso **View**. Debes cargar el recurso de layout desde el código de tu aplicación, en la implementación de devolución de llamada

Activity.onCreate(). Para eso, se usa a **setContentView()** pasando la referencia a tu recurso de diseño en forma de **<<R.layout.layout_file_name>>**. Por ejemplo, si tu diseño XML se guarda como **activity_main.xml**, los cargarás para tu actividad de la siguiente manera: **R.layout.activity_main**

<https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419>



7. Actividad Principal (MainActivity)



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="usa.sesion1.miprimerprograma">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MiPrimerPrograma"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MiPrimerPrograma">

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

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

</manifest>
```

Al revisar el archivo **Manifest** después de la creación del proyecto, **app > manifests > AndroidManifest.xml**.

se puede observar como automáticamente quedo incluida la Actividad principal, y además su característica de inicio de inicio de la aplicación a través de **LAUNCHER**.



9. Carga el recurso XML

Cargando las vistas **TextView**, **Button** dentro de un **LinearLayout** en el **MainActivity** de nuestro proyecto **MiPrimerPrograma**.

Luego, crear una instancia del objeto View (**TextView**, **Button**) y capturarla desde el diseño (**LinearLayout**) generalmente a través del método **onCreate()** del **MainActivity** se Carga el recurso XML a través de archivo **R** con la siguiente sintaxis:

```
Clase_objeto_visual nombreJava= (Clase_objeto_visual) findViewById(R.id.nombreXML)
```

Suponiendo que el objeto visual **Button** se ha nombrado en XML como **boton1** la carga queda así:

```
Button myBoton1 = (Button) findViewById(R.id.boton1)
```

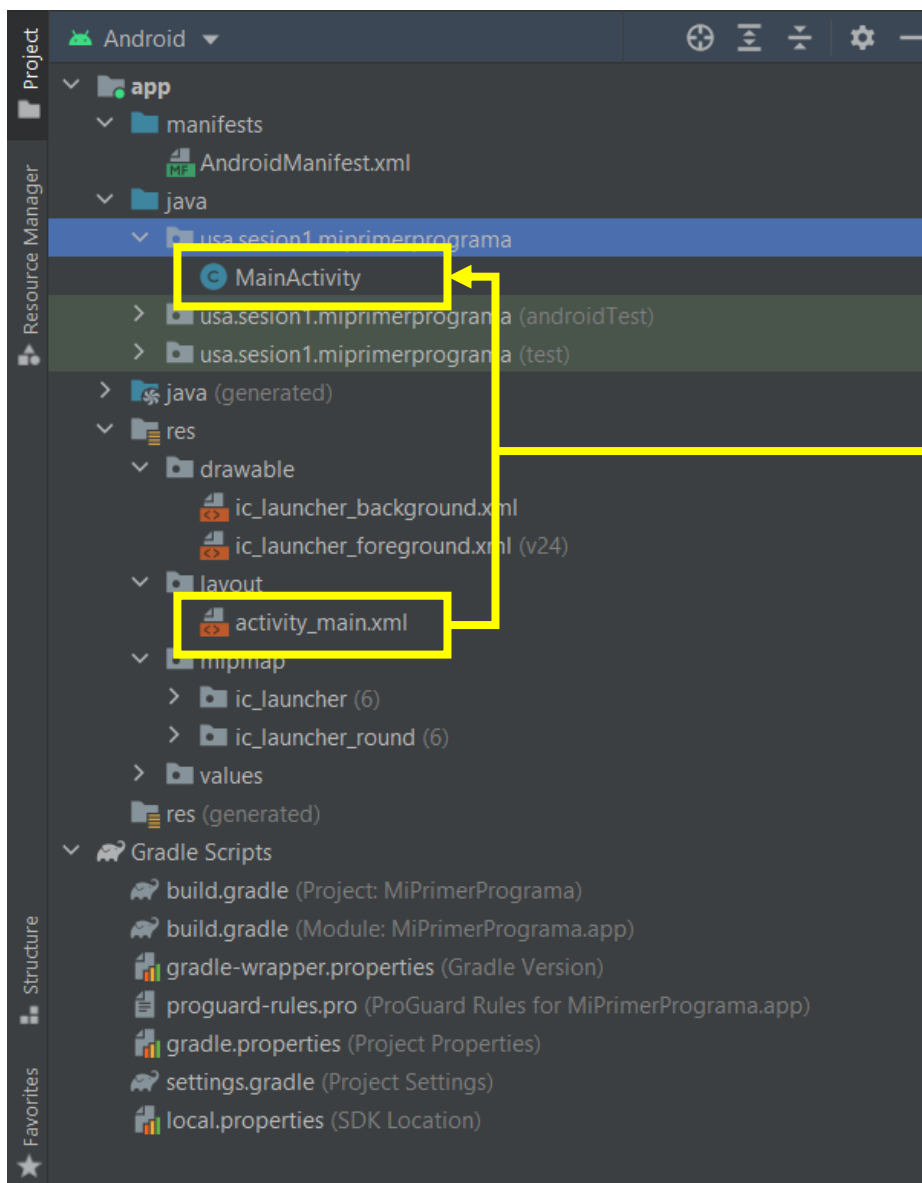
Suponiendo que el objeto visual **TextView** se ha nombrado en XML como **texto1** la carga queda así:

```
TextView myBoton1 = (TextView) findViewById(R.id.texto1)
```



9. Carga el recurso XML

La **MainActivity** (Java) queda relacionada con **activity_main.xml** de la siguiente manera: **R.layout.activity_main**



```
package usa.sesion1.miprimerprograma;

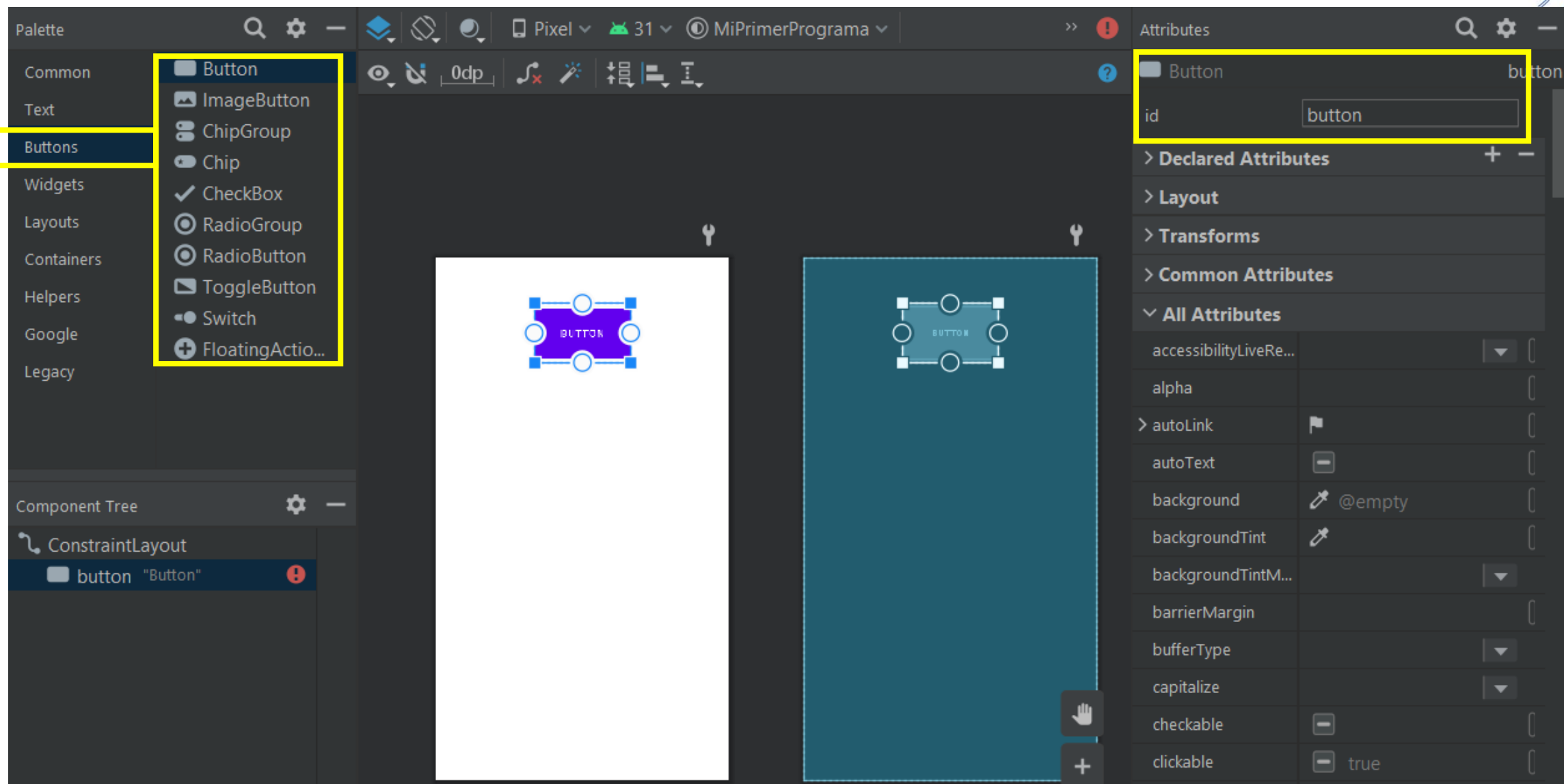
import ...

public class MainActivity extends AppCompatActivity {

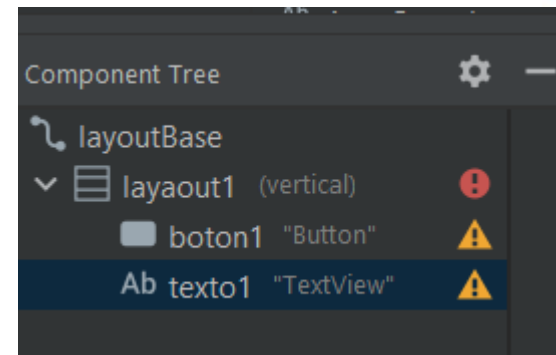
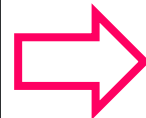
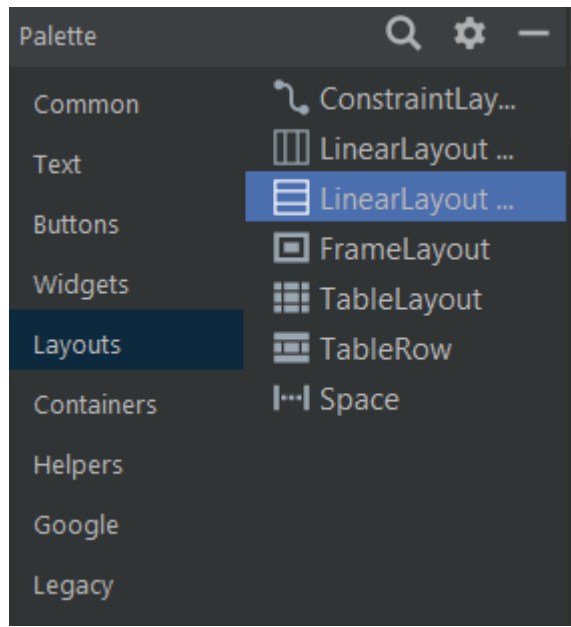
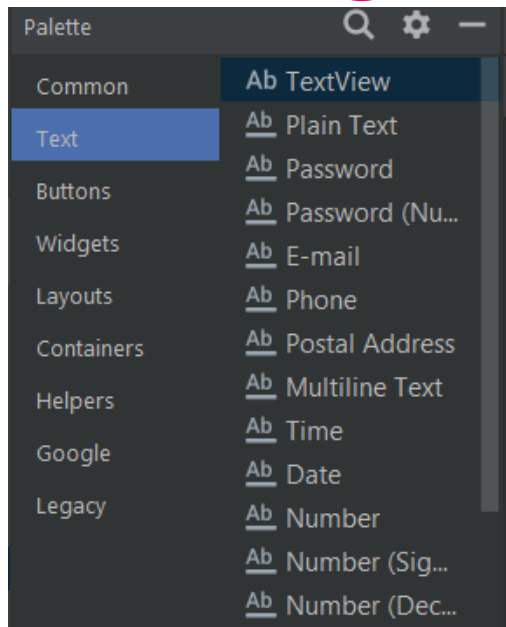
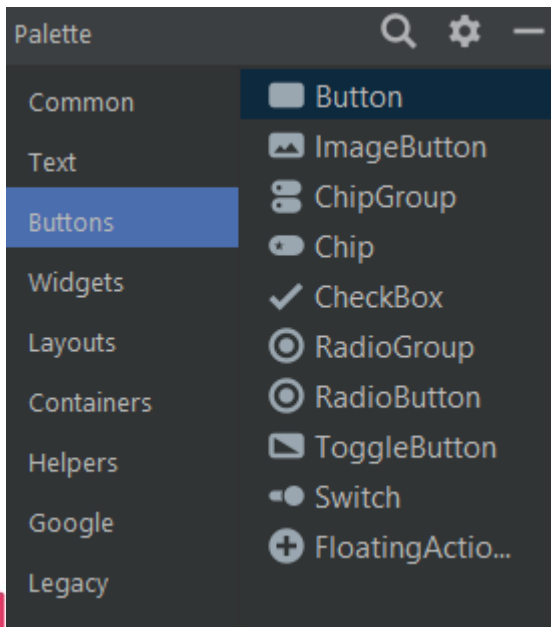
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



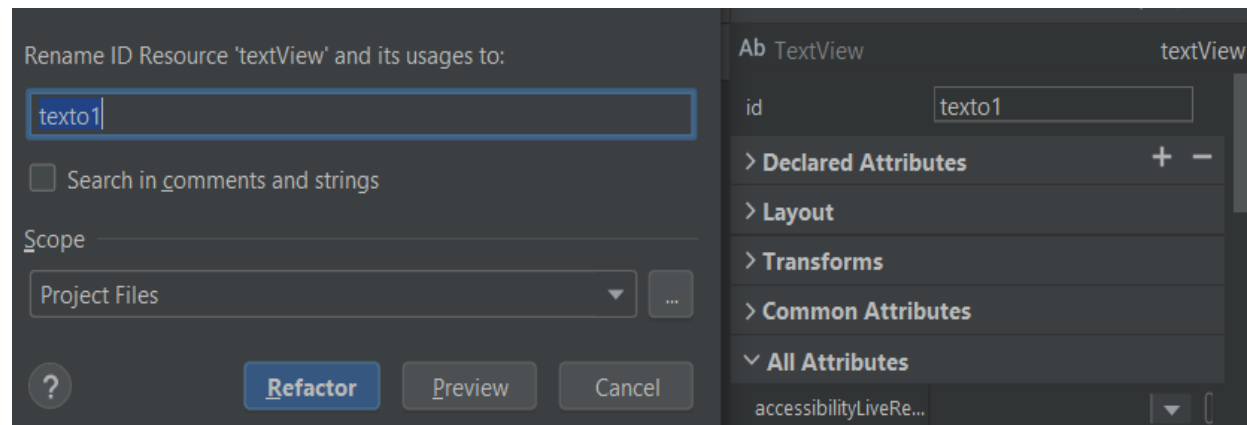
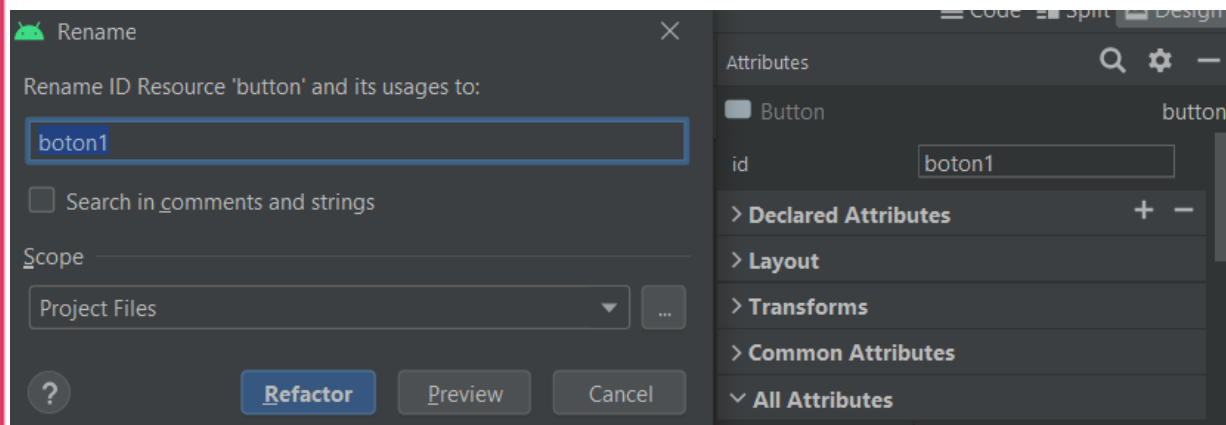
9. Carga el recurso XML



9. Carga el recurso XML



Cambiar los nombres con **Refactor Rename**:



9. Carga el recurso XML

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button myBoton1 =(Button) findViewById(R.id.bo);  
    }  
}
```

f boton1 (= 1000102)
f bounce (= 1000193)
Ctrl+Abajo and Ctrl+Arriba will move care

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button myBoton1 =(Button) findViewById(R.id.boton1);  
        TextView myText1 =(TextView) findViewById(R.id.tex);  
    }  
}
```

f text01 (= 1000024)
Press Ctrl+. to choose the selected (or first



Toast.makeText(context, text, duration);
Toast.makeText(context, text, duration);

10. Eventos de entrada

En Android, existe más de una forma de interceptar los eventos desde una interacción del usuario con tu aplicación. Al considerar los eventos dentro de tu interfaz de usuario, el enfoque consiste en capturar los eventos desde el objeto de vista específico con el que interactúa el usuario. La clase View proporciona los medios para hacerlo. En este caso solo usaremos como ejemplo la interfaz **OnClickListener** a través de un **escuchador de eventos** y luego el mismo ejemplo pero usando el concepto de **Clase anónima**.

Usaremos el aviso Toast, creando una instancia de un objeto de aviso, con el método `makeText()`, que toma los siguientes parámetros:

```
Toast.makeText(context, text, duration).show();
```

<https://developer.android.com/guide/topics/ui/notifiers/toasts?hl=es-419>



De ambas maneras se obtiene el mismo resultado:

1. Implementando la interfaz
2. Usando Clase anónima

10. Eventos de entrada

Implementando interfaz

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
    private TextView myTexto1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button myBoton1 =(Button) findViewById(R.id.boton1);  
        myBoton1.setOnClickListener(this); // Se pone a escuchar el evento clic  
  
        myTexto1 =(TextView) findViewById(R.id.texto1);  
    }  
    // La interfaz implementa el metodo abstracto onClick  
    @Override  
    public void onClick(View view) {  
        //Toast toast = Toast.makeText(context, text, duration).show();  
        Toast.makeText(context: this, text: "se presiono el boton1 y puso un mensaje en text1",Toast.LENGTH_SHORT).show();  
        myTexto1.setText("hola mundo");  
    }  
}
```

10. Eventos de entrada

```
public class MainActivity extends AppCompatActivity {  
  
    private TextView myTexto1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        myTexto1 = (TextView) findViewById(R.id.texto1);  
  
        Button myBoton1 = (Button) findViewById(R.id.boton1);  
        // Se implementa el escuchador del evento clic usando una clase anonima  
        myBoton1.setOnClickListener(new View.OnClickListener() {  
            public void onClick(View v) {  
                //Toast toast = Toast.makeText(context, text, duration).show();  
                Toast.makeText(getApplicationContext(), text: "se presiono el boton1 y puso un mensaje en text1", Toast.LENGTH_SHORT).show();  
                myTexto1.setText("hola mundo");  
            }  
        });  
    }  
}
```

Usando Clase Anónima

Como estamos dentro de una clase anónima, ya no usamos **this**, ahora usamos **getApplicationContext()** en el contexto.

<https://developer.android.com/guide/topics/ui/controls/button?hl=es-419>

```
private EditText myEditText1;
private EditText myEditText2;
private TextView myTexto1;
private Button myBoton1;
```

```
//*****
```

```
@Override
```

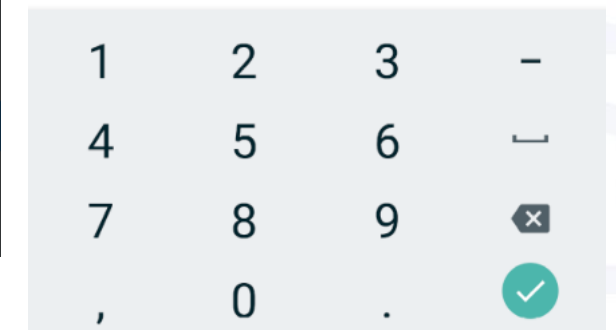
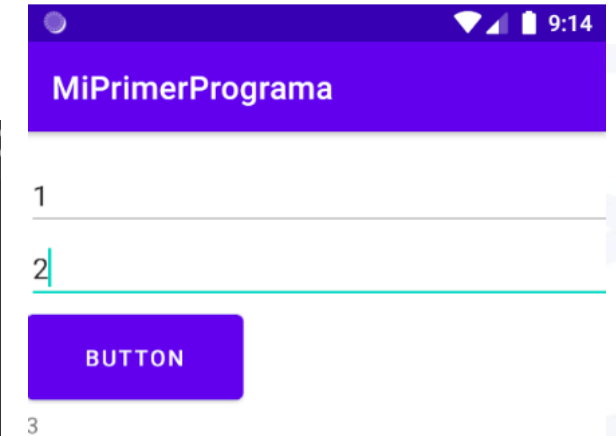
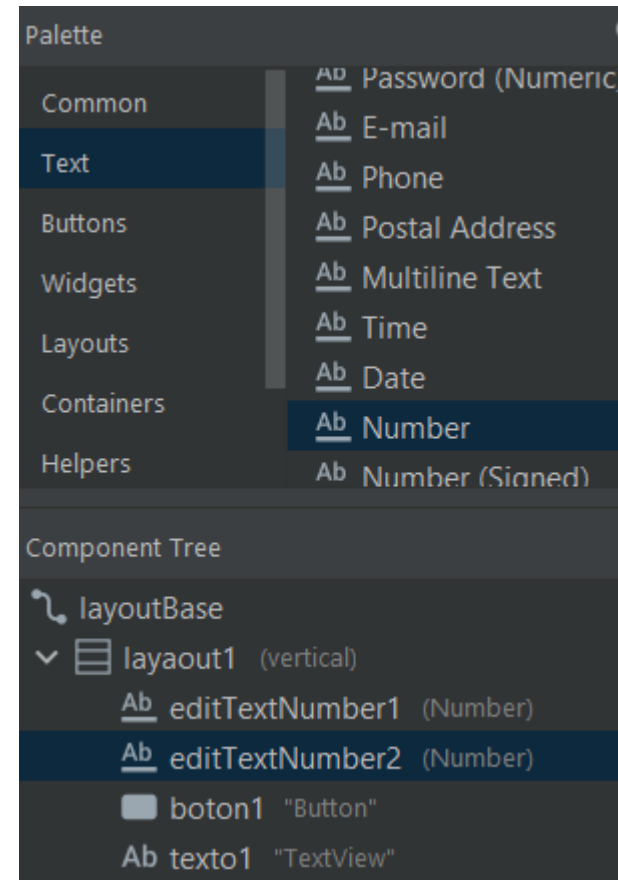
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

```
myEditText1 =(EditText)findViewById(R.id.editTextNumber1);
myEditText2=(EditText) findViewById(R.id.editTextNumber2);
myTexto1 =(TextView) findViewById(R.id.texto1);
```

```
Button myBoton1 =(Button) findViewById(R.id.boton1);
// Se implementa el escuchador del evento clic usando una clase anonima
myBoton1.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String valor1=myEditText1.getText().toString();
        String valor2=myEditText2.getText().toString();
        int nro1=Integer.parseInt(valor1);
        int nro2=Integer.parseInt(valor2);
        int suma=nro1+nro2;
        String resu=String.valueOf(suma);
        myTexto1.setText(resu);
    }
});
```

```
//*****
```

11. Ejercicio: suma



12. Menús



Los **menús** son un componente común de la interfaz de usuario en muchos tipos de aplicaciones. Para proporcionar una experiencia de usuario conocida y uniforme, debes usar las **API de Menu** a fin de presentar al usuario acciones y otras opciones en las actividades.

A partir de Android 3.0 (nivel de API 11), los dispositivos con Android ya no tienen que proporcionar un botón Menú exclusivo. Con este cambio, las apps para Android dejarán de depender de los paneles de menú tradicionales de 6 elementos y, en su lugar, proporcionarán una barra de la app para mostrar las acciones más comunes del usuario.

Aunque haya cambiado el diseño de la experiencia del usuario para algunos elementos de menú, **la semántica para definir un conjunto de acciones y opciones sigue basándose en las API de Menu**. Esta guía muestra cómo crear tres tipos fundamentales de presentaciones de menús o acciones en todas las versiones de Android.

Menú de opciones y barra de la app

Menú contextual y modo de acción contextual

Menú emergente

<https://developer.android.com/guide/topics/ui/menus?hl=es>

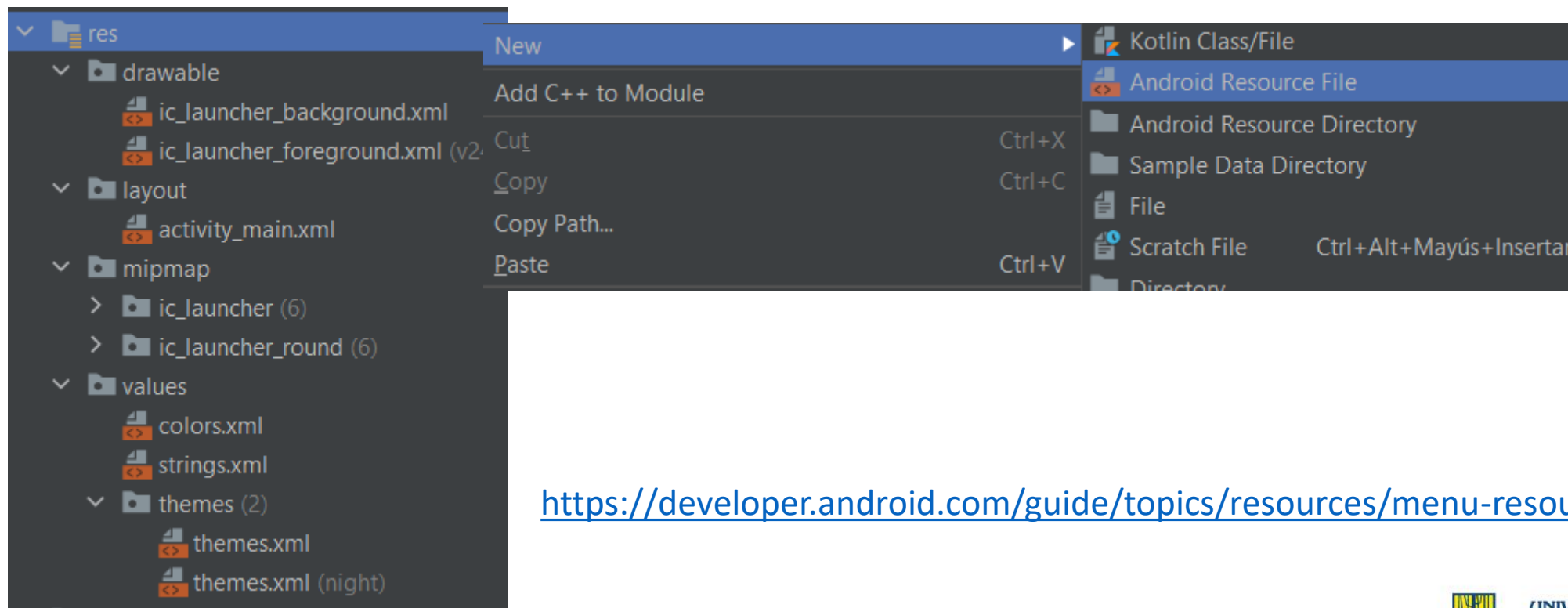
12. Menús

Cómo definir un menú en XML:

En lugar de incorporar un menú en el código de la actividad, debes definir un menú y todos los elementos en un **recurso de menú XML**.

Archivo XML guardado en **res/menu/example_menu.xml**:


1. Clic derecho y crear una carpeta con el nombre de **menu** y luego un archivo XML (Values XML file) con el nombre **example_menu.xml**.



<https://developer.android.com/guide/topics/resources/menu-resource?hl=es>



12. Menús

 New Resource File ✕

File name: ↑↓

Resource type: ▼

Root element:

Source set: ▼

Directory name:

Available qualifiers:

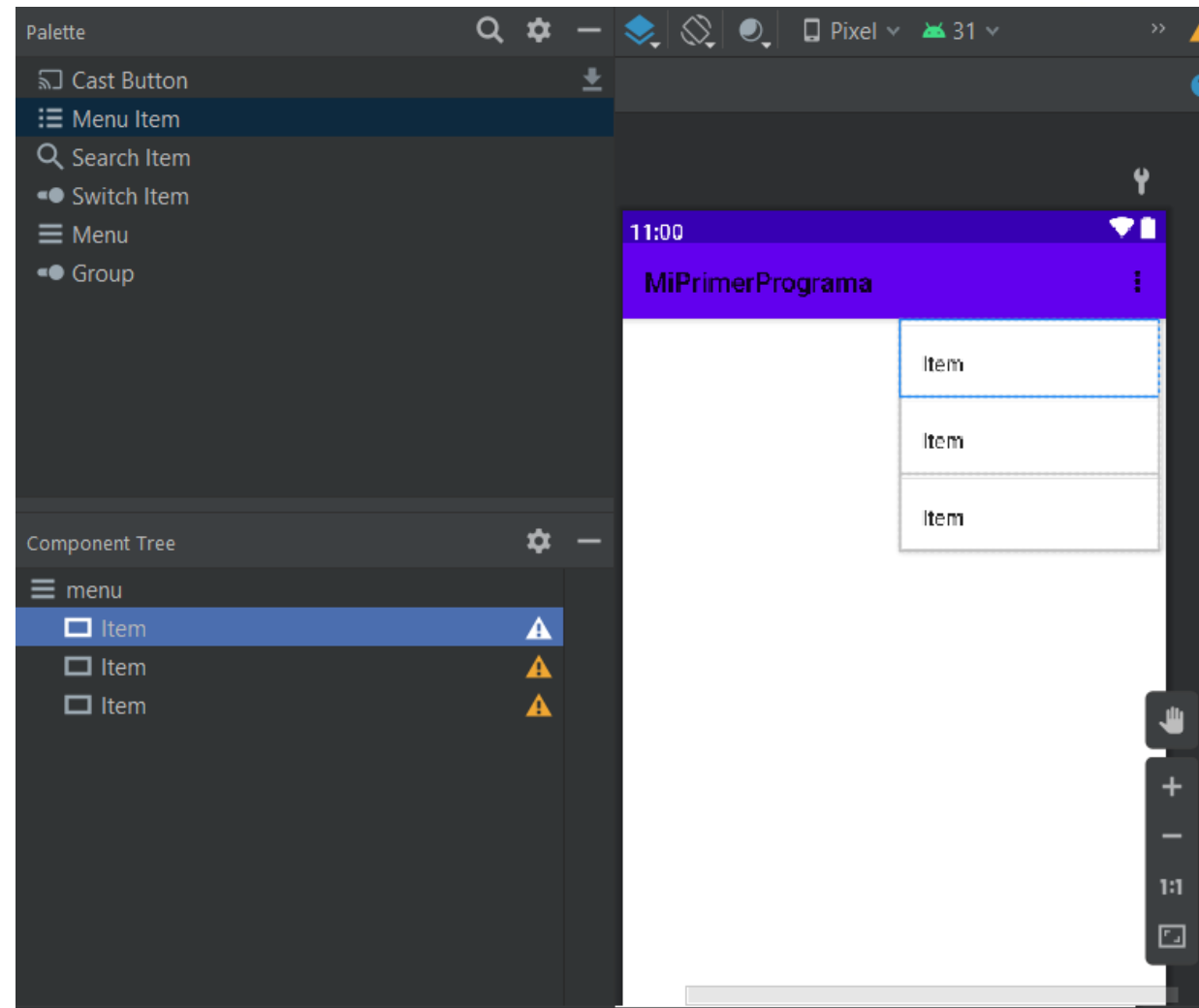
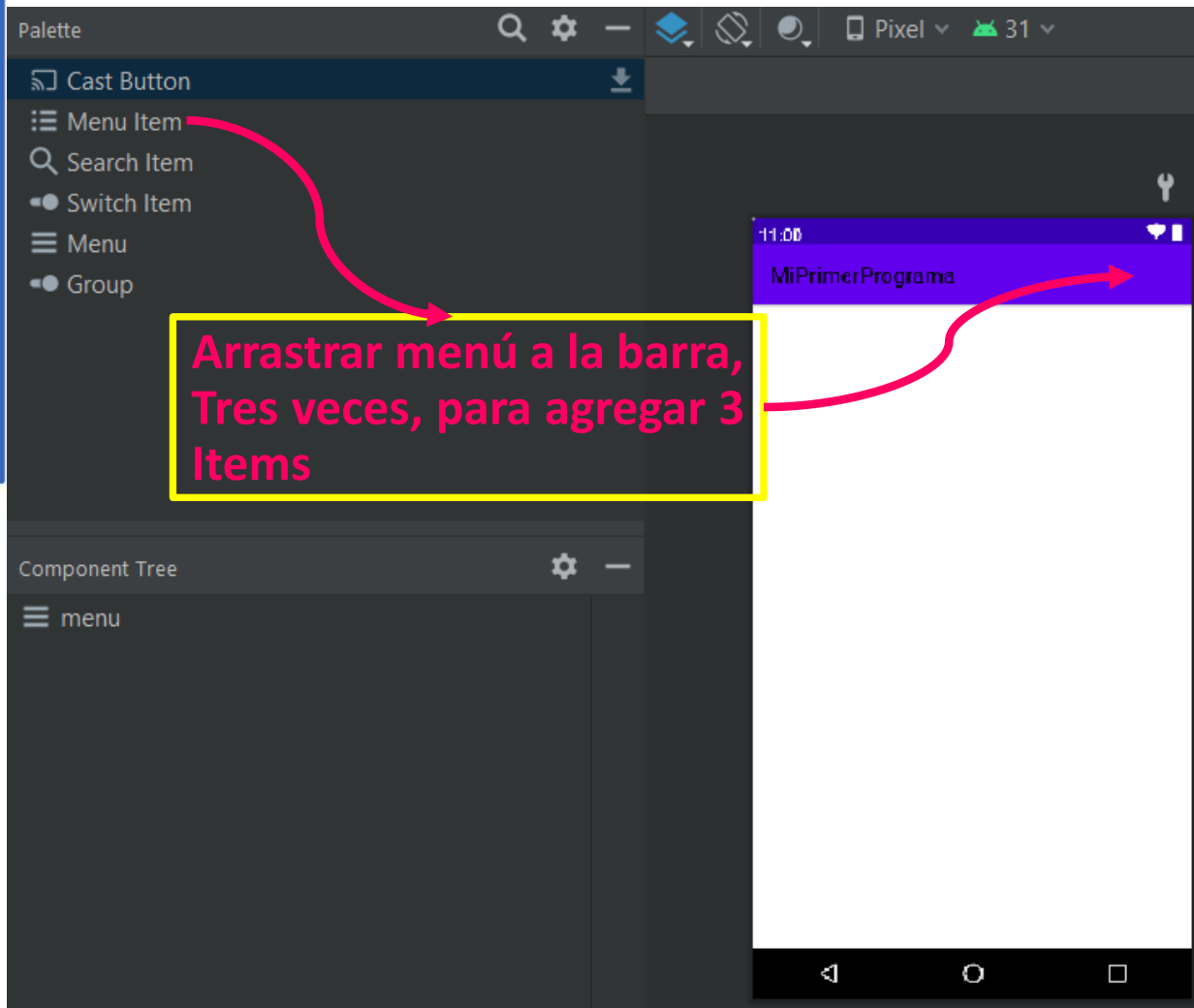
- ☒ Country Code
- ☐ Network Code
- ☐ Locale
- ☐ Layout Direction
- ☐ Smallest Screen Width
- ☐ Screen Width
- ☐ Screen Height
- ☐ Size
- ☐ Ratio
- ☐ Orientation

Chosen qualifiers:

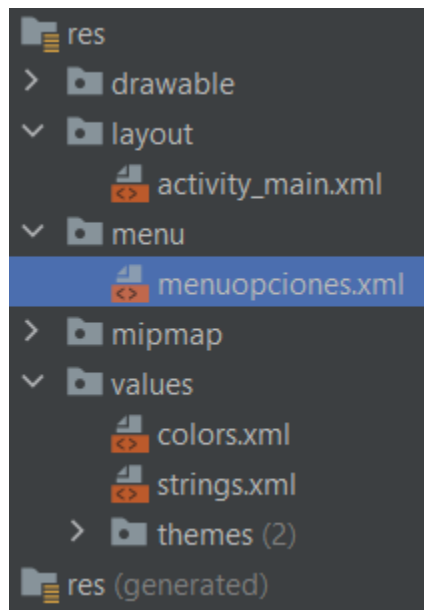
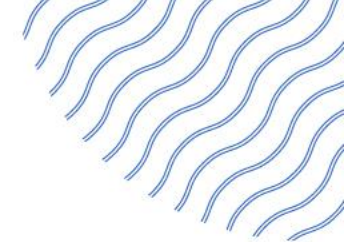
Nothing to show

?? <<

? OK Cancel



12. Menús



```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/opcion1"
        android:title="Opcion 1" />
    <item
        android:id="@+id/opcion2"
        android:title="Opcion 2" />
    <item
        android:id="@+id/opcion3"
        android:title="Opcion 3" />

</menu>
```

El Archivo XML queda guardado en **res/menu/menuopciones.xml**, después de manipularlo en diseño.

Se debe agregar el método con **@Override** sobre el método **onCreateOptionsMenu** y luego el método **onOptionsItemSelected**. La siguiente diapositiva muestra la forma de agregar el método.

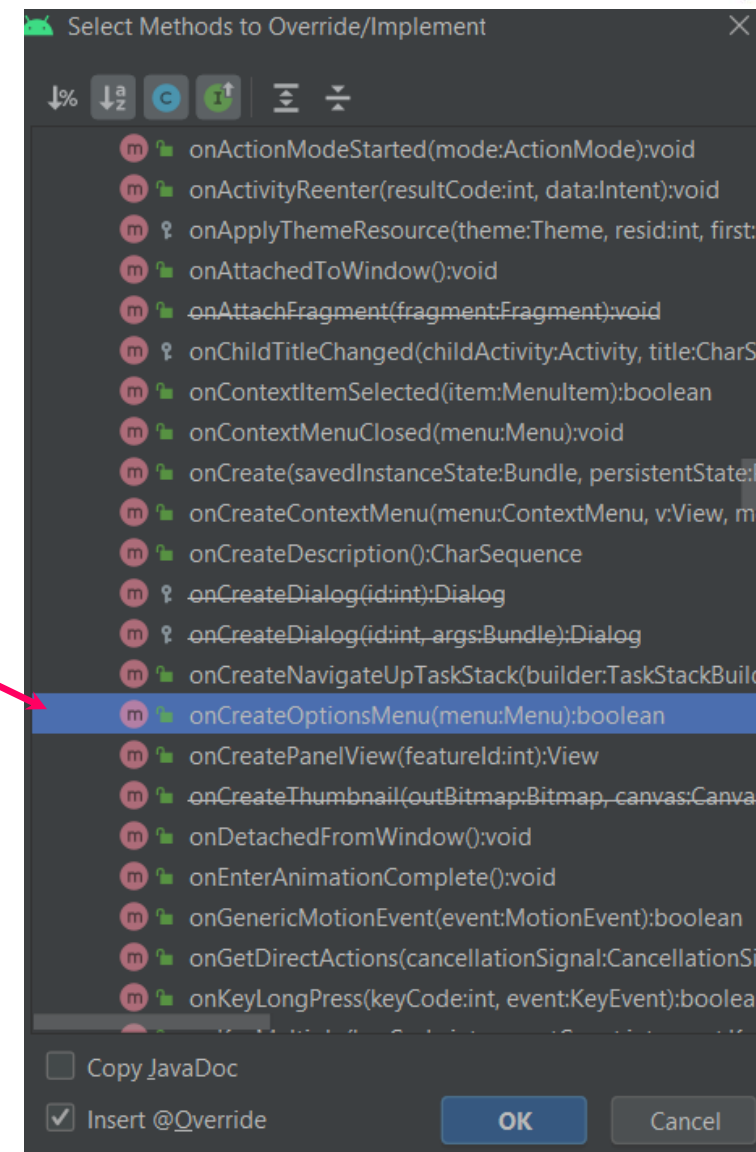
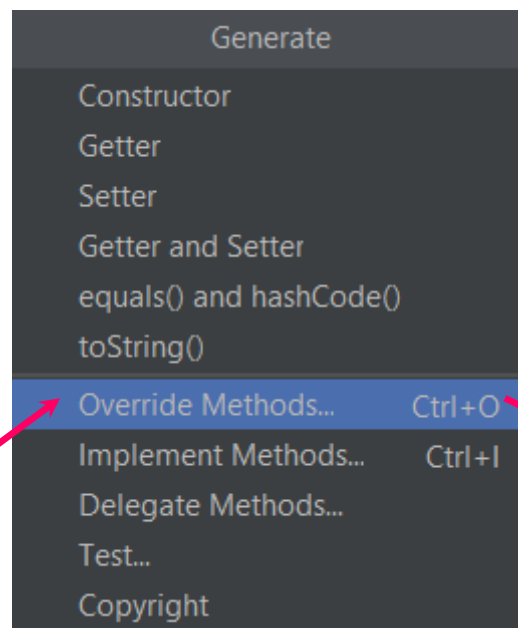
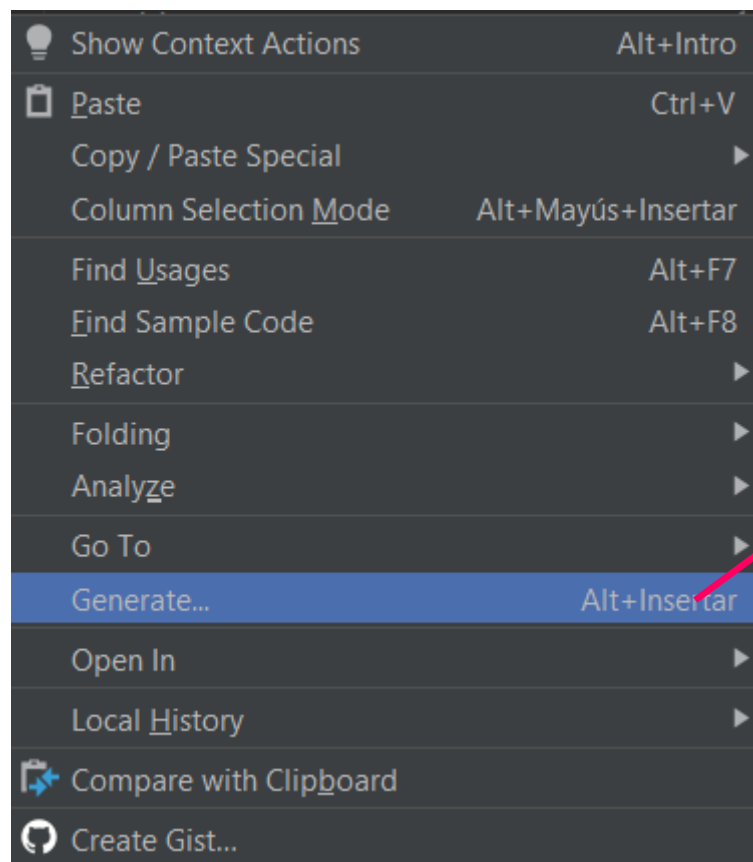
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menuopciones, menu);
    return true;
}
```

```
public void onOptionsItemSelected(MenuItem item) {

}
```

12. Menús

Se agrega el método **onCreateOptionsMenu** de la siguiente manera:



Tomado de: <https://developer.android.com/guide/topics/ui/menus?hl=es>

12. Menús



```
public class MainActivity extends AppCompatActivity {

    private TextView myTexto1;

    //*****

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

        myTexto1 =(TextView) findViewById(R.id.texto1);

        Button myBoton1 =(Button) findViewById(R.id.boton1);
        // Se implementa el escuchador del evento clic usando una clase anonima
        myBoton1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //Toast toast = Toast.makeText(context, text, duration).show();
                Toast.makeText(getApplicationContext(), text: "se presiono el boton1 y puso un mensaje en text1",Toast.LENGTH_SHORT).show();
                myTexto1.setText("hola mundo");
            }
        });
    }

    //*****

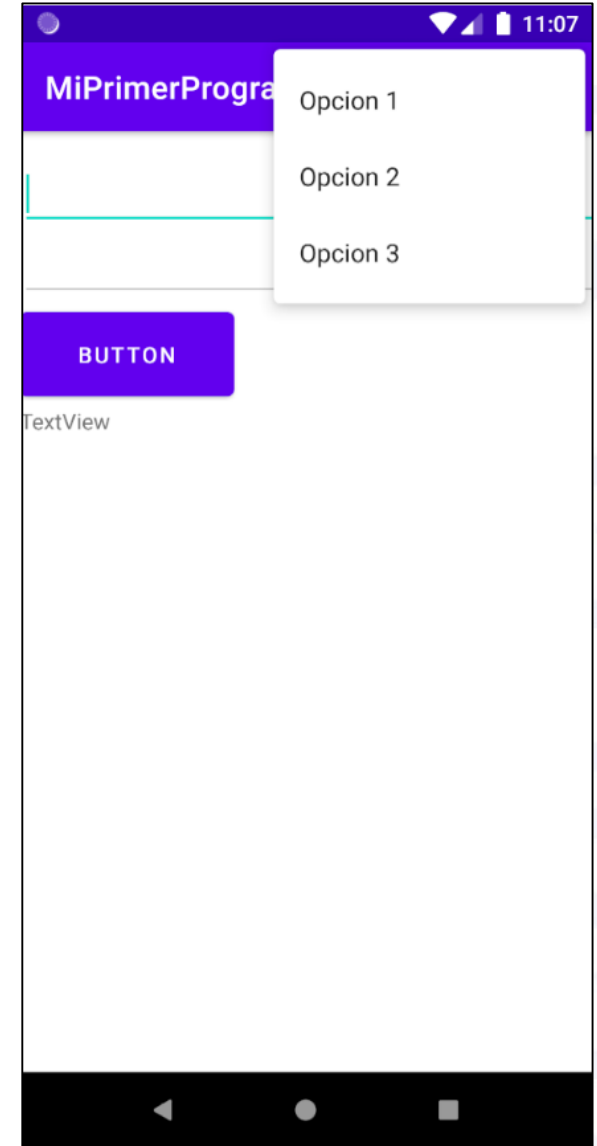
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        return super.onCreateOptionsMenu(menu);
    }

}
```



12. Menús

```
//***** MENU *****/
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menuopciones, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id==R.id.opcion1) {
        Toast.makeText(context: this, text: "Se seleccionó la primer opción", Toast.LENGTH_LONG).show();
    }
    if (id==R.id.opcion2) {
        Toast.makeText(context: this, text: "Se seleccionó la segunda opción", Toast.LENGTH_LONG).show();
    }
    if (id==R.id.opcion3) {
        Toast.makeText(context: this, text: "Se seleccionó la tercer opción", Toast.LENGTH_LONG).show();
    }
    return super.onOptionsItemSelected(item);
}
//*****
} //fin de la clase
```



Preguntas

