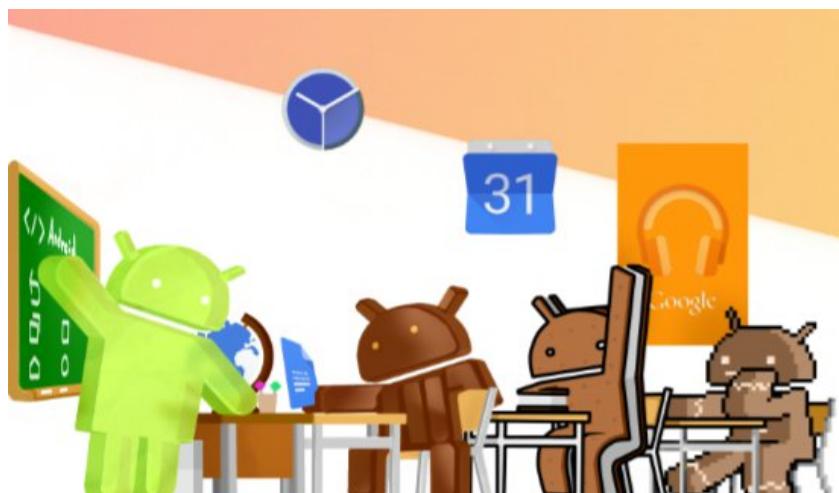


UT2. DESARROLLO DE APLICACIONES PARA ANDROID



www.iesriberadeltajo.com
facebook.com/ies.riberadeltajo
[@IESRiberaTajo](https://twitter.com/IESRiberaTajo)

Android – Tipos de Componentes



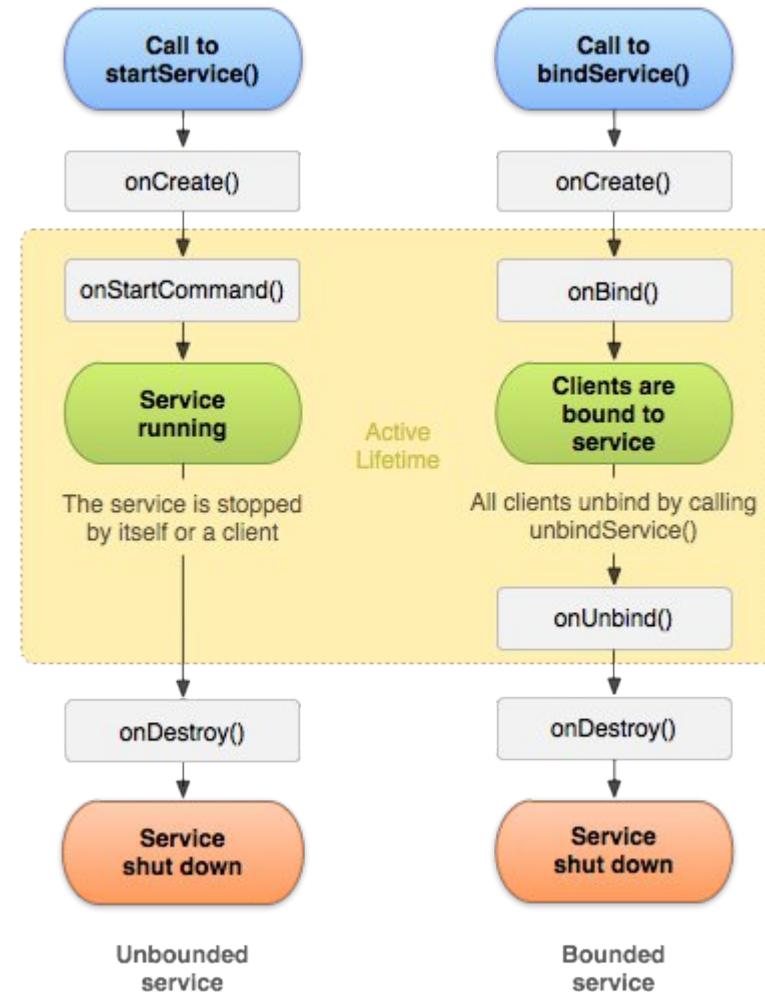
Actividades



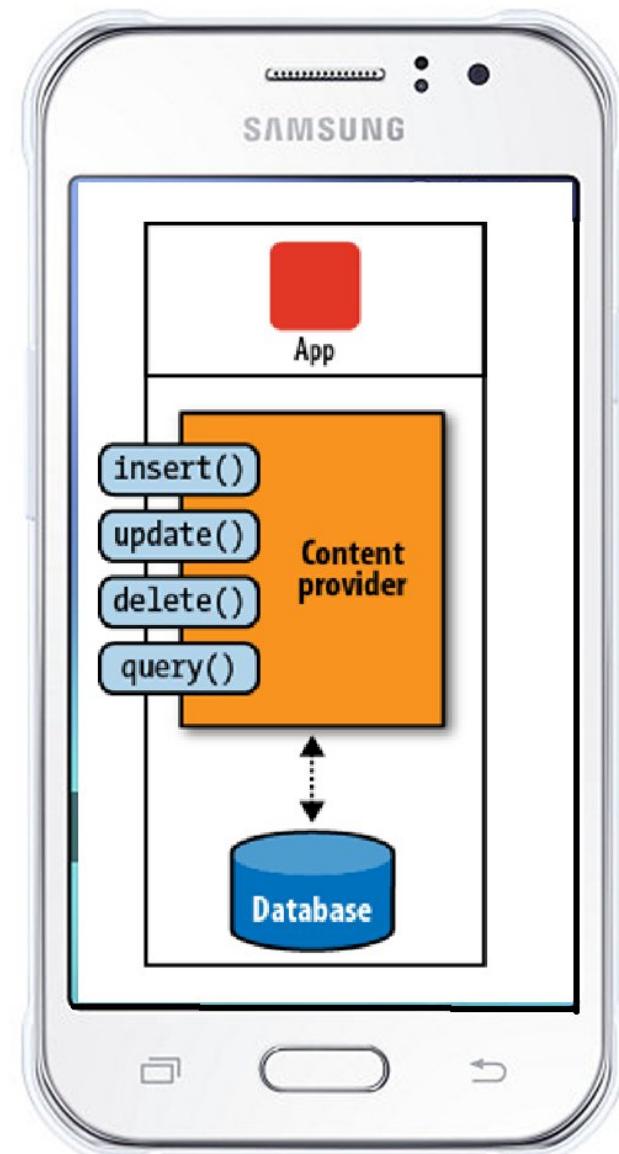
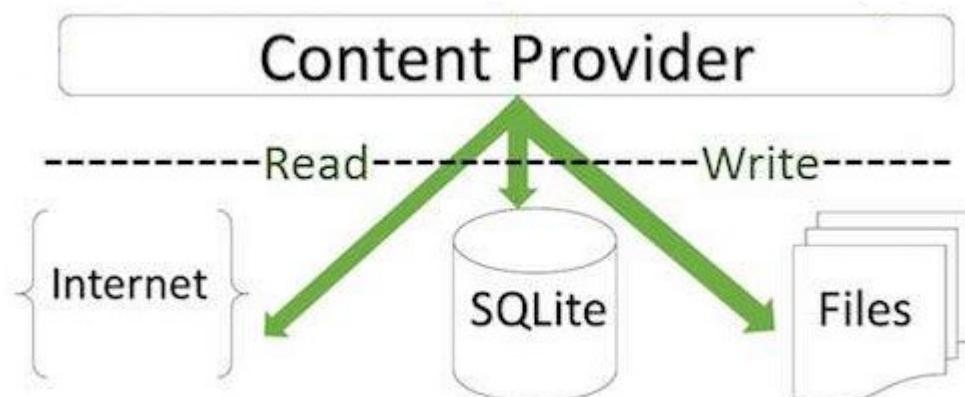
Servicios



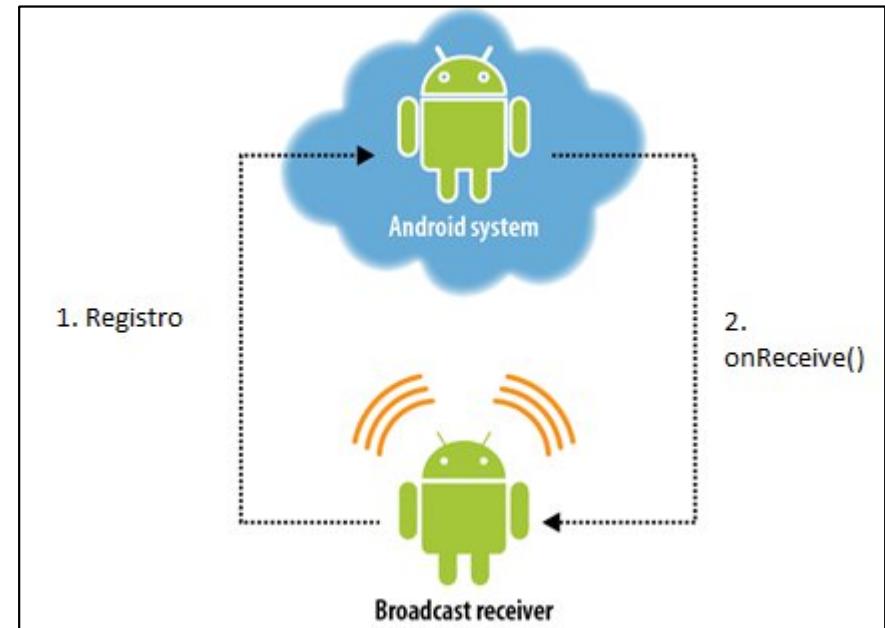
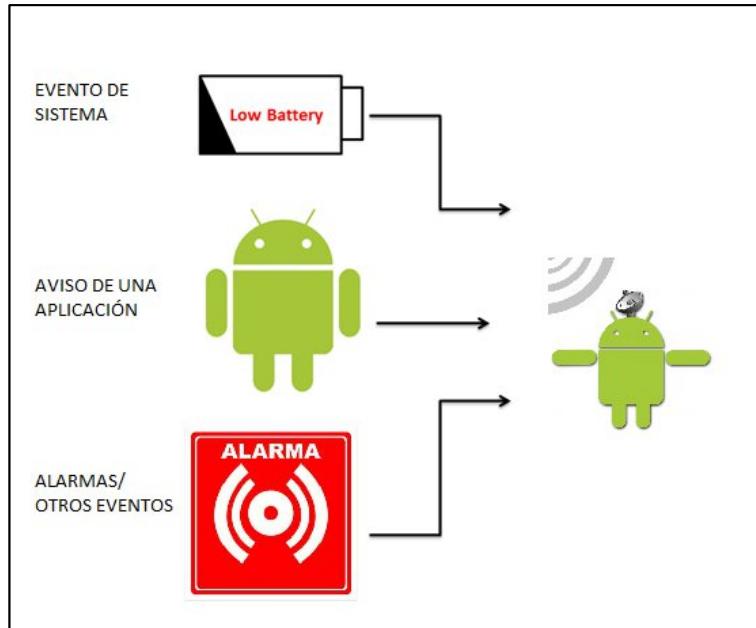
Sin interfaz gráfica



PROVEEDORES DE CONTENIDO

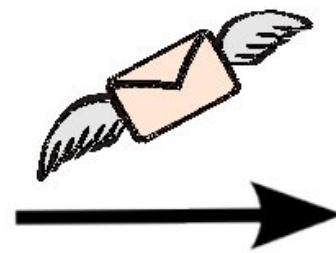
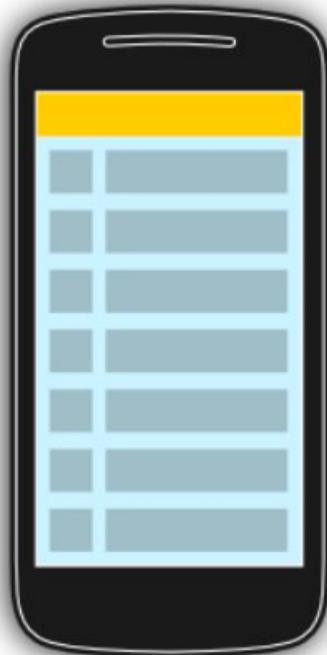


RECEPTORES DE BROADCAST



INTENTS

ACTIVIDAD



SERVICIO



El fichero de manifiesto

The screenshot shows the Android Studio interface with the project 'MiprimeraApp' open. The left sidebar displays the project structure, including the app module with its build, libs, src (containing androidTest and main), and res directories. The main area shows the code editor with the 'AndroidManifest.xml' file selected. The XML code defines the application manifest with an application tag containing an activity tag for 'MiPrimeraActivity'. The activity is set to handle the main intent filter and has a launcher category.

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="Mi primera App"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MiPrimeraActivity"
            android:label="Mi primera App" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

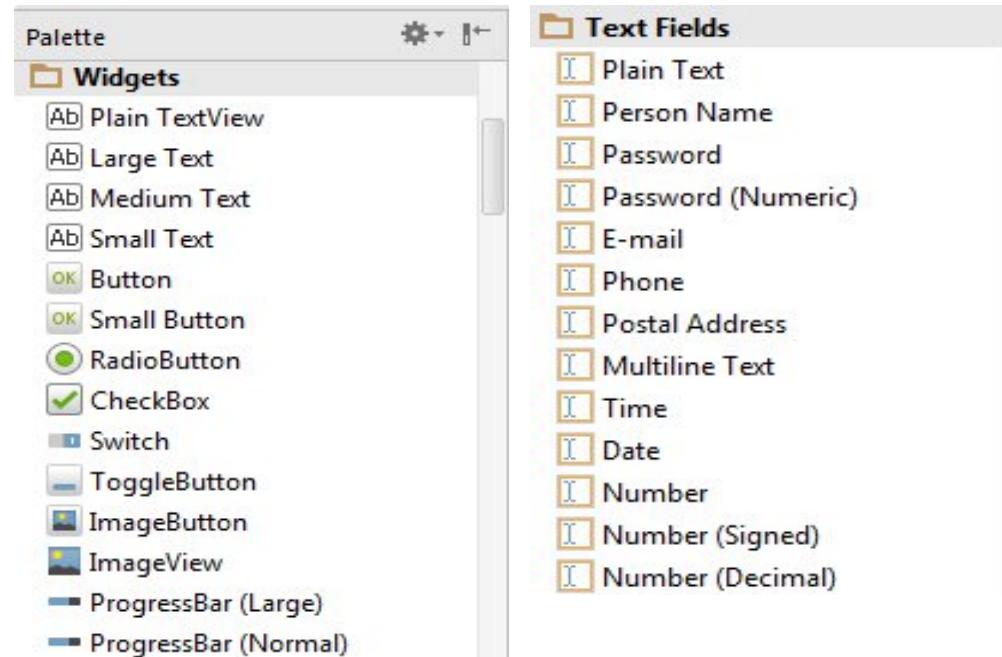
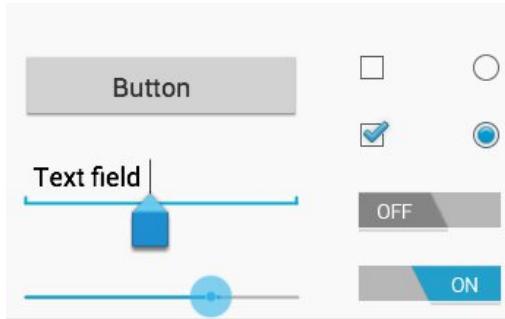
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

<activity> componentes de tipo actividad
<service> componentes de tipo servicio
<receiver> componentes de tipo receptores de broadcast
<provider> componentes de tipo proveedor.

WIDGETS SENCILLOS PARA TUS ACTIVIDADES

- CLASES

- TextView
- Button
- RadioGroup y RadioButton
- Checkbox
- ToggleButton y Switch
- ImageView e ImageButton
- EditText



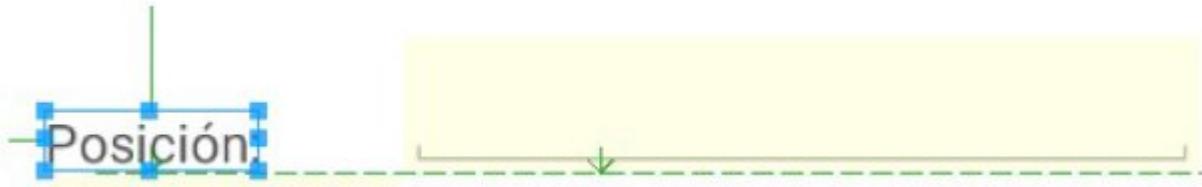
Documentación para los widgets

- <https://developer.android.com/reference/android/widget/package-summary.html>

The screenshot shows a browser window displaying the Android Developers website at <https://developer.android.com/reference/android/widget/package-summary.html>. The page is titled "android.widget" and includes the following content:

- Overview**: A brief description stating that the widget package contains mostly visual UI elements and allows for custom design.
- Implementation**: A list of three required files:
 - Java implementation file**: Implements the behavior of the widget.
 - XML definition file**: Defines the XML element used to instantiate the widget and its attributes.
 - Layout XML [optional]**: An optional XML file describing the layout of the widget.
- Implementation Examples**: A note about the ApiDemos sample application providing an example of creating a custom layout XML tag, `LabelView`.

TextView



- ***android:typeface*** permite cambiar el tipo de fuente (normal, sans, serif, monospace) para la etiqueta.
- ***android:textStyle*** permite seleccionar el estilo de la letra (cursiva, negrita, o cursiva y negrita)
- ***android:textColor*** permite seleccionar el color de la etiqueta en formato RGB hexadecimal.
- ***Android:textSize*** permite especificar el tamaño de la fuente, por ejemplo 20dp (20 density-independent pixels)

Ejemplo

```
<TextView  
    android:id="@+id/txtHelloWorld"  
    android:text="@string/hello_world"  
    android:typeface="monospace"  
    android:textStyle="italic"  
    android:textColor="#FF0000"  
    android:textSize="20dp"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```



TextView Otras fuentes....

- ¿No puedo cambiar en un TextView al tipo de fuente que yo quiera?
 - Si, por supuesto, pero tienes que tener en cuenta que las fuentes incluidas por defecto en Android son Droid Sans (sans), Droid Sans Mono (monospace) y Droid Serif (serif). Cualquier otra opción debe ser cargada.
- Por ejemplo, para poder usar una fuente Arial, debes cargar el fichero arial.ttf en el directorio de Assets (herramientas) y cargarlo desde código:

```
TextView  
txtHelloWorld=(TextView)findViewById(R.id.txtHelloWorld);  
  
Typeface type =  
Typeface.createFromAsset(getAssets(),"arial.ttf");  
  
txtHelloWorld.setTypeface(type);
```

- Cuidado: Este tipo de ficheros son muy “pesados” y recuerda que

Button

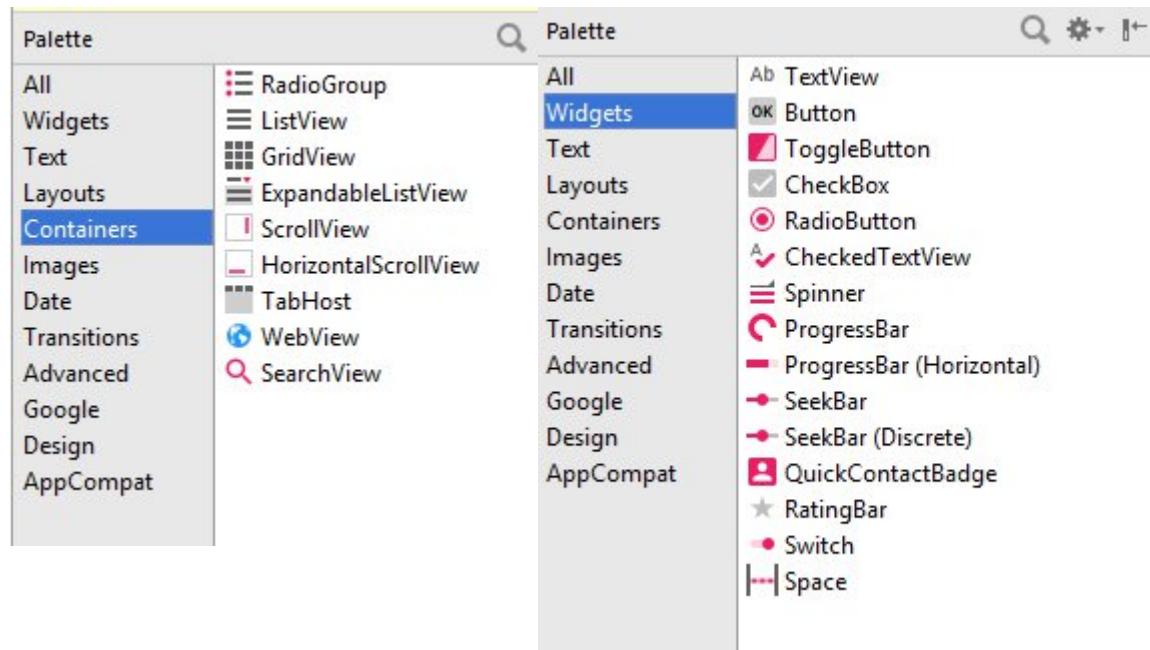
```
java.lang.Object  
  ↳ android.view.View  
    ↳ android.widget.TextView  
      ↳ android.widget.Button
```

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Botón Gigante"  
    android:id="@+id/button"  
    android:typeface="serif"  
    android:textStyle="italic"  
    android:textColor="#FF0000"  
    android:textSize="80dp"
```



LAS CLASES RadioGroup y RadioButton

- Los RadioGroup son contenedores, por tanto los encontrarás en la carpeta de contenedores
- los RadioButton los tienes justo debajo de los widget de tipo Button.



XML

```
<RadioGroup  
...  
    <!--Propiedades del radio group-->  
    ...  
    <RadioButton  
        ...  
        android:text="Talavera C.F."  
            android:id="@+id radioButton2"  
            android:checked="true" />  
    <RadioButton  
        ...  
        android:text="Gimástico de Alcazar"  
        android:id="@+id radioButton3"  
        android:checked="false" />  
    ...  
</RadioGroup>
```

métodos

Clase	Método	Acción
RadioGroup	void check(int id)	Activa el RadioButton con identificador id
RadioGroup	void clearCheck()	"Limpia", es decir, quita la selección dejando todos los RadioButton desactivados
RadioGroup	int getCheckedRadioButtonId()	Retorno el Id del RadioButton seleccionado
RadioButton	void toggle()	Cambia el estado de activación del RadioButton

Ejercicio: programa esta app

Paso 1. Implementar la interfaz RadioGroup.OnCheckedChangeListener

```
public class MainActivity extends AppCompatActivity implements  
    RadioGroup.OnCheckedChangeListener{  
    ...  
}
```

Paso 2. Registrar el listener

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    RadioGroup r=(RadioGroup) findViewById(R.id.radioGroup);  
    r.setOnCheckedChangeListener(this);  
}
```

Paso 3. Codificarlo

```
public void onCheckedChanged(RadioGroup group, int checkedId) {  
    TextView t=(TextView) findViewById(R.id.txtEstado);  
    if (checkedId == R.id.rbTalavera) { //Talavera  
        t.setText("Buena elección!: El Talavera promete!!");  
    } else if (checkedId == R.id.rbAlcazar) { //Alcazar  
        t.setText("Gran equipo la gimnástica!!");  
    } else if (checkedId == R.id.rbAlbacete) { //Albacete  
        t.setText("El Albacete no es el mismo desde que se fué Iniesta");  
    } else if (checkedId == R.id.rbOtros) { //Otros  
        t.setText("El dinero no lo es todo....");  
    }  
}
```

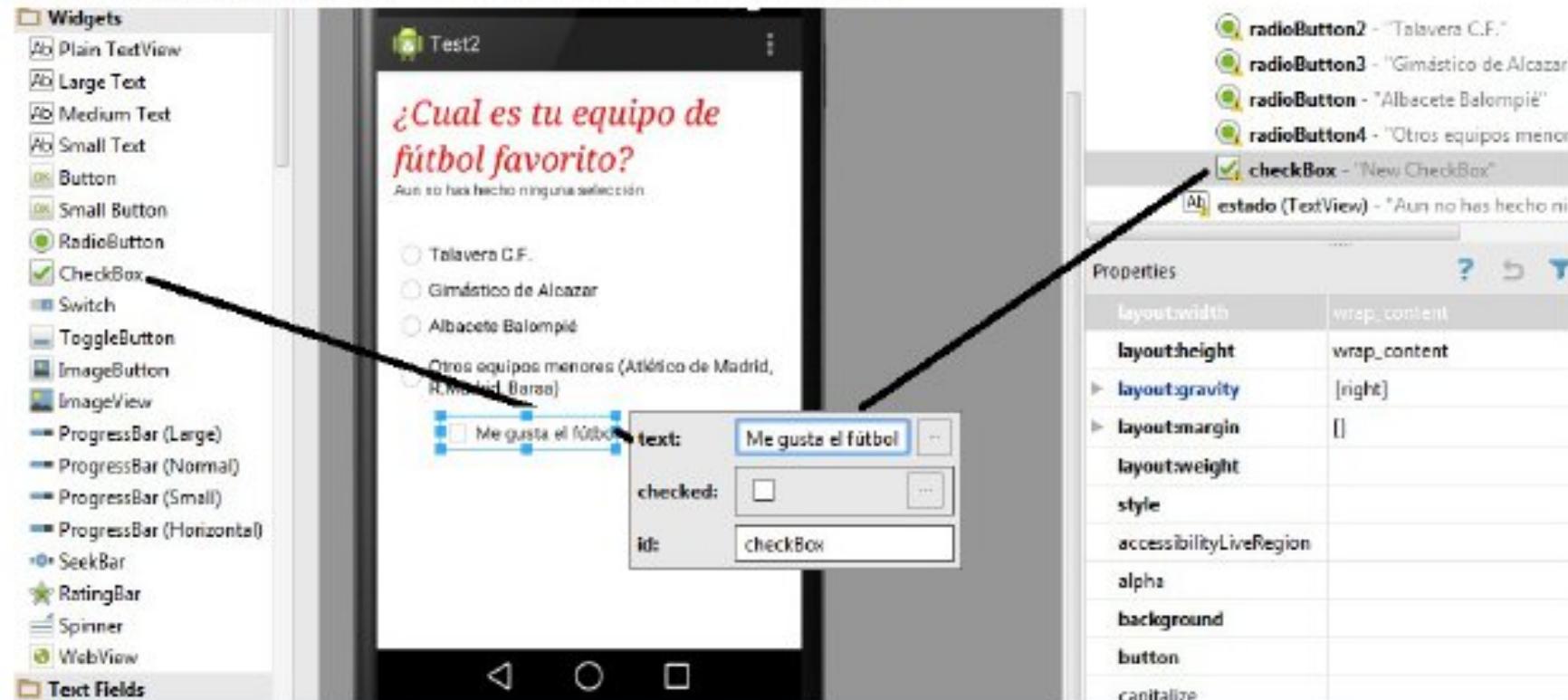


Public methods

abstract void	onCheckedChanged(RadioGroup group, int checkedId)
	Called when the checked radio button has changed.

Los checkboxes

`setChecked(true)` o `setChecked(false)`.

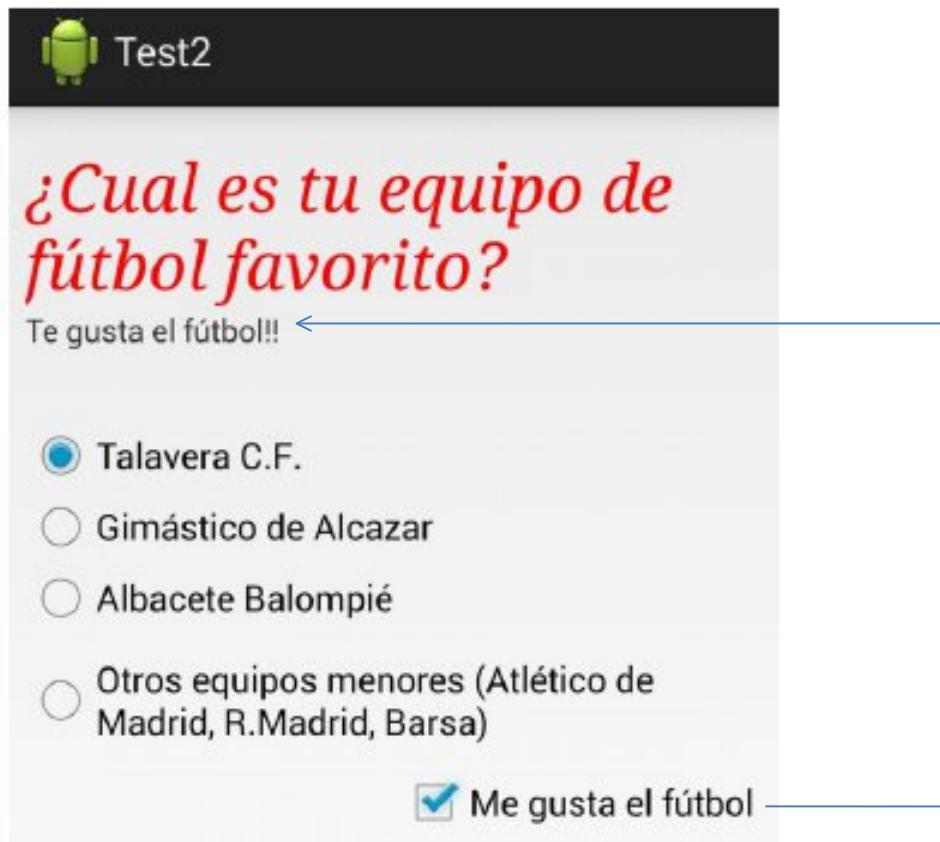


Public methods

`abstract void onCheckedChanged(CompoundButton buttonView, boolean isChecked)`

Called when the checked state of a compound button has changed.

Ejercicio: A la app anterior, añade un checkbox

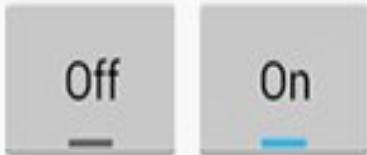


ficheros tema 2/test3.zip

¿ y qué pasa si..., como es el caso de esta App, tengo que implementar dos interfaces, una para RadioGroup y otra para Checkbox? ¿No habrá conflictos entre los dos métodos que hay que implementar? ¿Cómo sabe Android a qué método invocar cuando se produzca un evento de cambio de estado si los dos métodos se llaman OnCheckedChangeListener?

La respuesta es que **no**. Gracias al polimorfismo estático, Java detecta qué método hay que ejecutar dependiendo de los parámetros de entrada.

Toggle buttons y switches



Toggle buttons



Switches (in Android 4.0+)

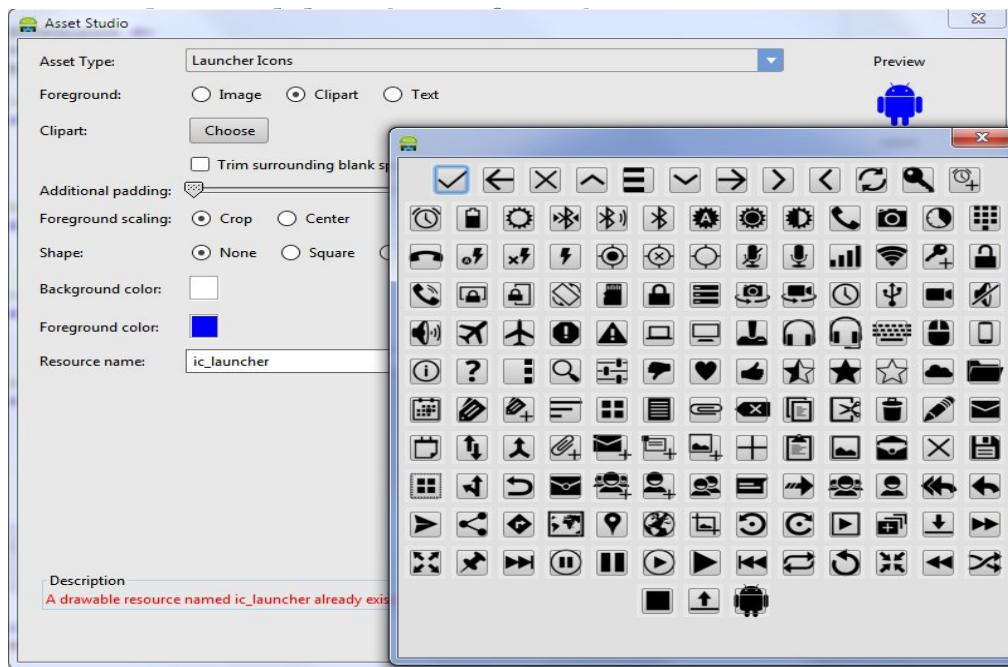
```
<ToggleButton  
    android:text="Sombrero"  
    android:id="@+id/toggleButton"  
    android:textOn="Puesto"  
    android:textOff="Quitado"  
    android:checked="false"  
/>
```

```
<Switch  
    android:text="Bigote:"  
    android:checked="true"  
    android:id="@+id/switch1"  
    android:textOn="Con bigote"  
    android:textOff="Sin bigote"  
/>
```



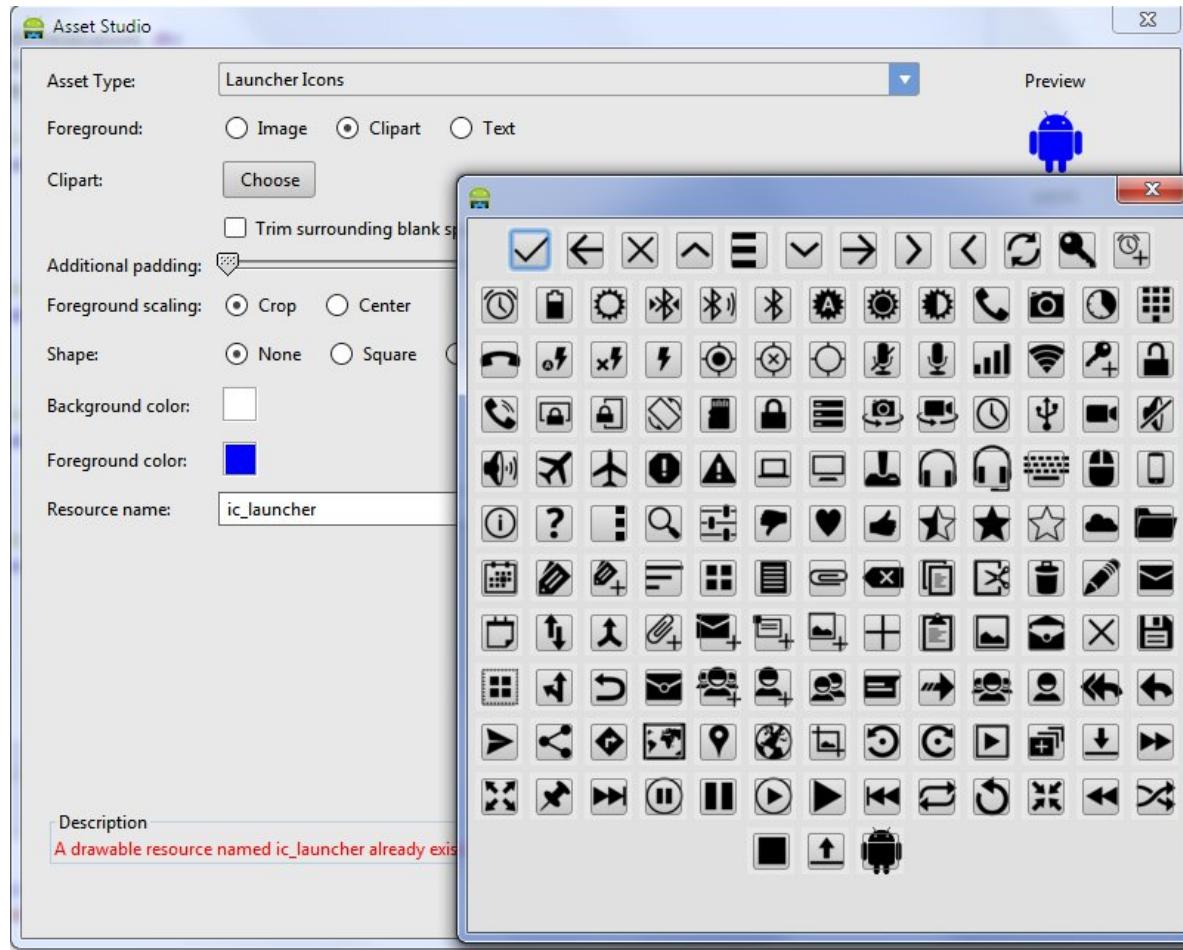
ImageView y ImageButton

- Hereda de Button
- Cada widget toma un atributo android:src en el fichero XML de la actividad para especificar qué imagen usar mediante una referencia a un recurso dibujable (@drawable)
- Debemos añadir a las carpetas “src->main->res->drawable*****” de nuestro proyecto los ficheros de las imágenes que queremos utilizar
- Android recomienda usar ficheros con formato png



```
java.lang.Object
↳ android.view.View
  ↳ android.widget.ImageView
    ↳ android.widget.ImageButton
```

Los drawables

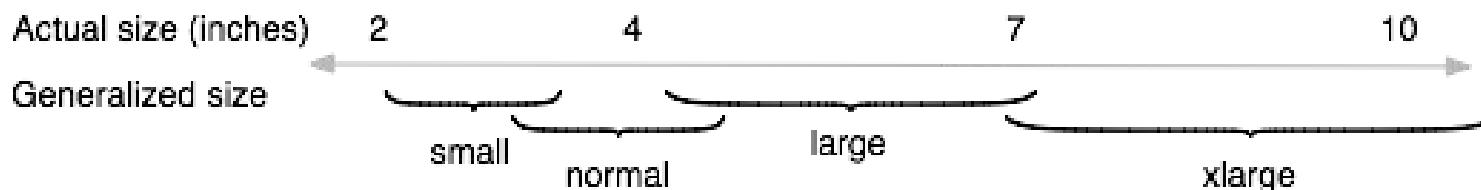


Admitiendo múltiples pantallas

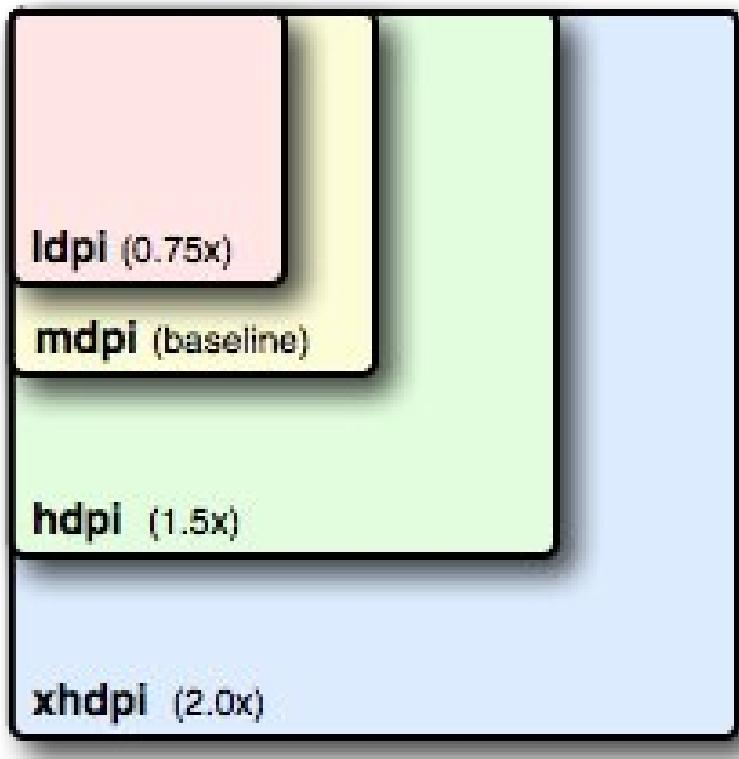
- https://developer.android.com/guide/practices/screens_support.html
- Declara en tu manifiesto de forma explícita los tamaños que admite tu app
- Proporciona diseños diferentes para diferentes tamaños de pantalla

```
res/layout/main_activity.xml           # For handsets  
res/layout-land/main_activity.xml    # For handsets in landscape  
res/layout-sw600dp/main_activity.xml  # For 7" tablets  
res/layout-sw600dp-land/main_activity.xml # For 7" tablets in landscape
```

- Proporciona diferentes elementos de diseño del mapa de bits para densidades diferentes
 - ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi



Múltiples densidades



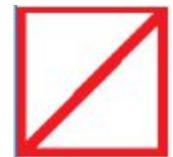
Density	DPI	Example Device	Scale	Pixels
ldpi	120	Galaxy Y	0.75x	1dp = 0.75px
mdpi	160	Galaxy Tab	1.0x	1dp = 1px
hdpi	240	Galaxy S II	1.5x	1dp = 1.5px
xhdpi	320	Nexus 4	2.0x	1dp = 2px
xxhdpi	480	Nexus 5	3.0x	1dp = 3px
xxxhdpi	640	Nexus 6	4.0x	1dp = 4px

Puedes utilizar la siguiente utilidad online (Android Asset Studio) para generar imágenes para todas las densidades de pantalla.

<http://romannurik.github.io/AndroidAssetStudio/nine-patches.html>

Midamos un poco....

- Pon un imageView con una imagen
 - Que la imagen mida 100 * 100 (hazla con msPaint)
 - Ejecuta tu proyecto, mide la imagen y calcula el factor de escalado
 - Ldpi
 - Mdpi
 - Hdpi
 - Xhdpi
 - XXhdpi



Density	DPI	Example Device	Scale	Pixels
ldpi	120	Galaxy Y	0.75x	1dp = 0.75px
mdpi	160	Galaxy Tab	1.0x	1dp = 1px
hdpi	240	Galaxy S II	1.5x	1dp = 1.5px
xhdpi	320	Nexus 4	2.0x	1dp = 2px
xxhdpi	480	Nexus 5	3.0x	1dp = 3px
xxxhdpi	640	Nexus 6	4.0x	1dp = 4px

<https://developer.android.com/training/multiscreen/screendensities.html>

Apps independientes de la densidad

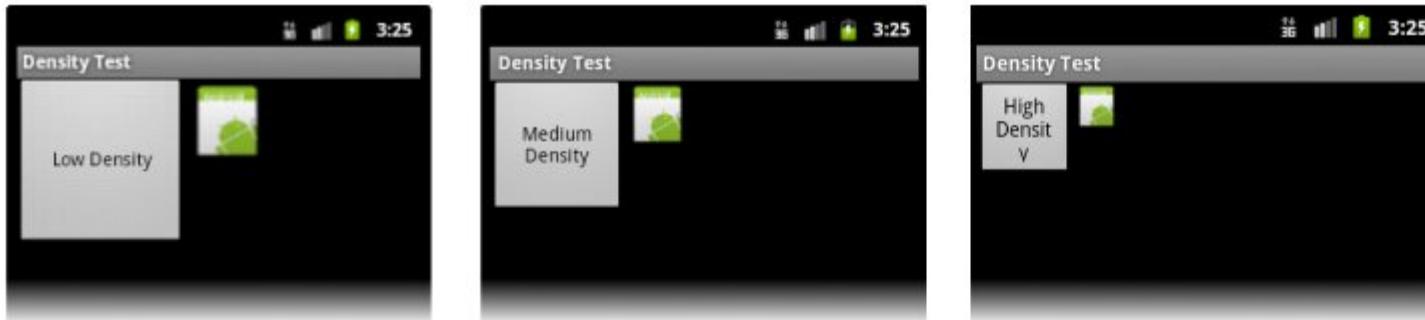


Figura 2: El ejemplo de aplicación es incompatible con diferentes densidades, como se muestra en las pantallas de baja, media y alta densidad.



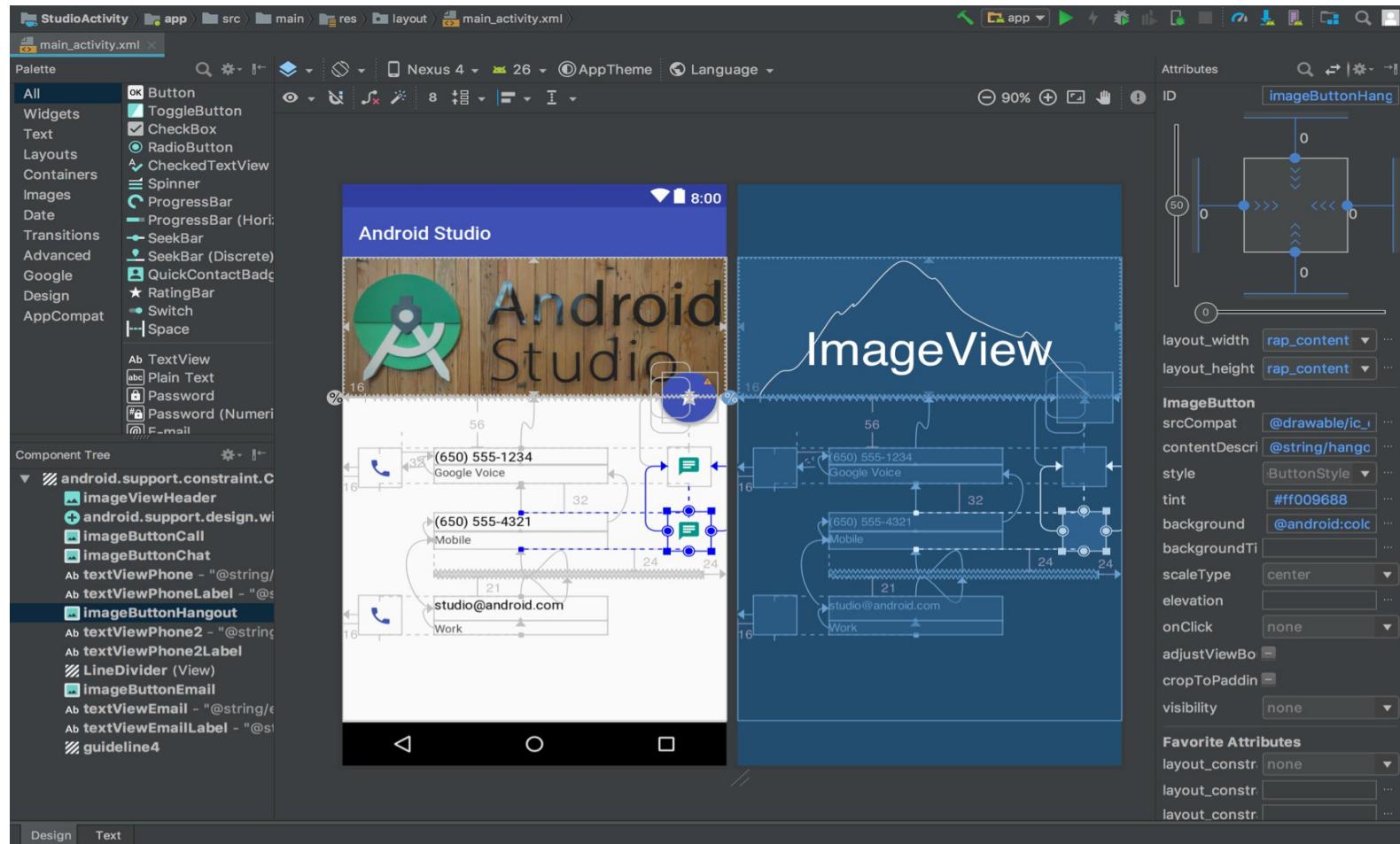
Figura 3: El ejemplo de aplicación tiene buena compatibilidad con las diferentes densidades (es independiente de la densidad), como se muestra en pantallas de densidad baja, media y alta.

El sistema Android permite que tu aplicación alcance la independencia de la densidad de dos maneras:

- * Ajusta unidades dp, según corresponda, a la densidad de la pantalla actual.
- * Si es necesario, ajusta los recursos del elemento de diseño al tamaño apropiado según la densidad de la pantalla actual

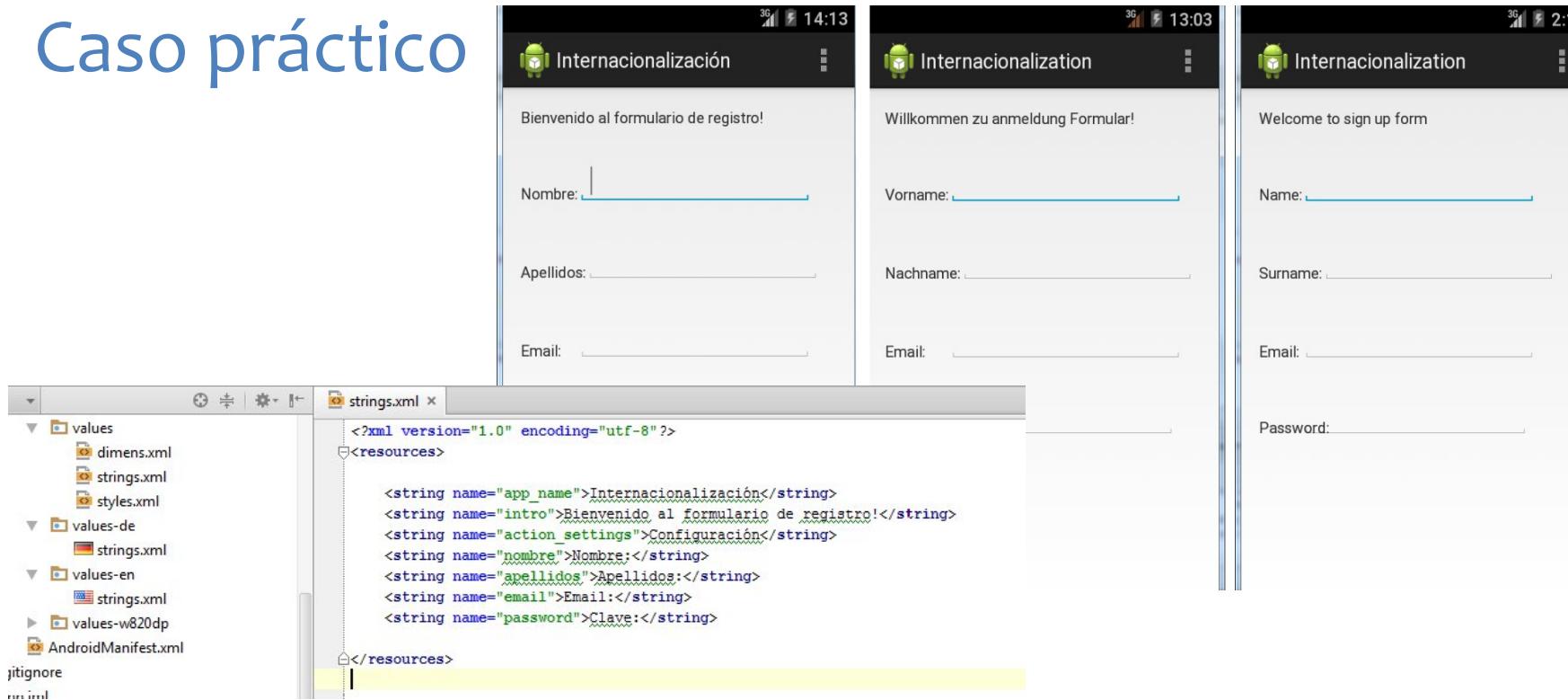
IU reactiva

Independientemente del perfil de hardware con el que quieras brindar compatibilidad primero, debes crear un diseño que sea receptivo incluso para las variaciones más leves de tamaño de pantalla.



INTERNACIONALIZACIÓN

- Buenas prácticas: carpeta res-values
- Caso práctico



ficheros_tema 2/internacionalizacion.zip

Crea esta app



Prueba a ver si te funciona en varias densidades y tamaños de pantalla, si no lo responde de igual forma a los cambios de pantalla, cámbialo para que efectivamente, tu app sea independiente de la densidad y de la configuración regional del dispositivo

La clase EditText

- **android:autotext**, para controlar si el usuario recibirá asistencia automática al escribir el texto
- **android:capitalize**, para controlar si el propio campo de texto debe escribir la primera letra de cada palabra en mayúscula.
- **android:digits**, para configurar si el campo de texto debe admitir solo determinados dígitos.
- **android:singleline**, para especificar si el campo admite solo una línea y al pulsar enter se va al siguiente control, o por el contrario crea una línea nueva al pulsar enter (multilínea)

¡DEPRECIADOS! ->

usar InputType

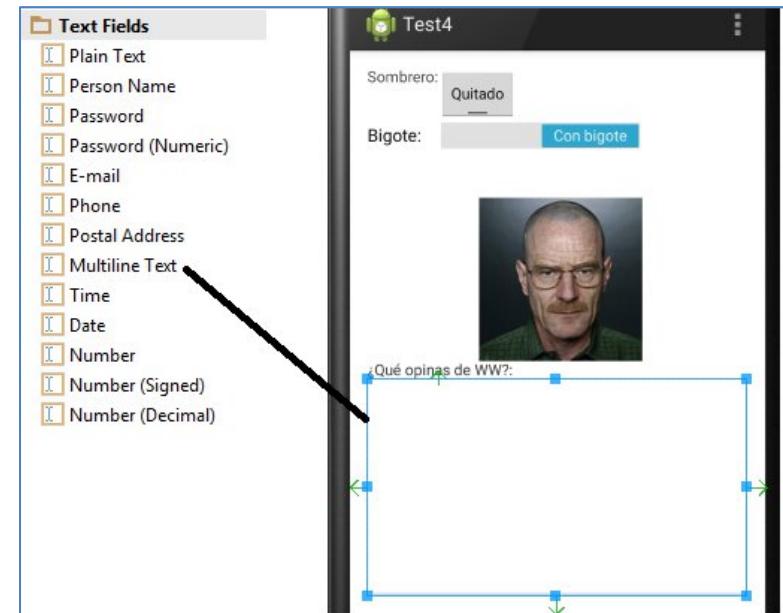
(siguiente transparencia)

`java.lang.Object`

↳ `android.view.View`

↳ `android.widget.TextView`

↳ `android.widget.EditText`



EditText - InputType

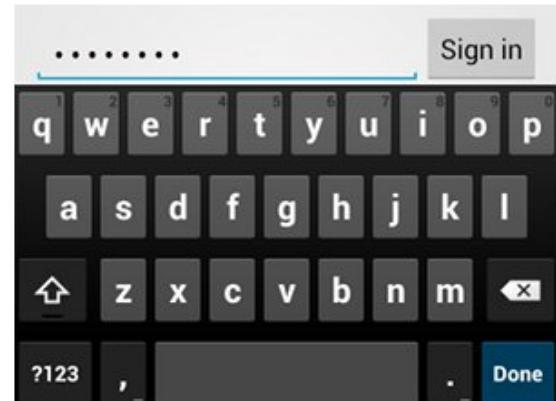
- <https://developer.android.com/training/keyboard-input/style.html>
- https://developer.android.com/reference/android/widget/TextView.html#attr_android:inputType

```
<EditText  
    android:id="@+id/phone"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:hint="@string/phone_hint"  
    android:inputType="phone" />
```



Figure 1. The phone input type.

```
<EditText  
    android:id="@+id/password"  
    android:hint="@string/password_hint"  
    android:inputType="textPassword"  
    ... />
```



EditText – Capturar acción de fin de edición

```
<EditText  
    android:id="@+id/search"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:hint="@string/search_hint"  
    android:inputType="text"  
    android:imeOptions="actionSend" />
```



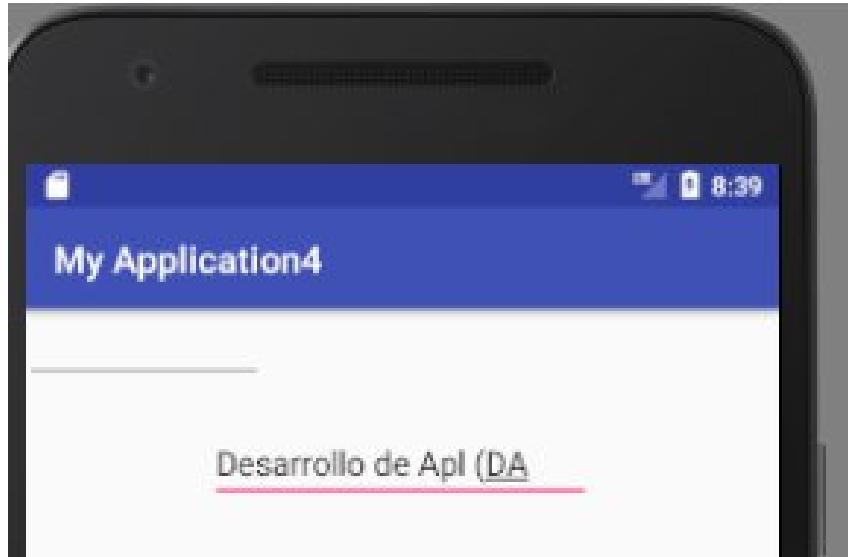
```
EditText editText = (EditText) findViewById(R.id.search);  
editText.setOnEditorActionListener(new OnEditorActionListener() {  
    @Override  
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
        boolean handled = false;  
        if (actionId == EditorInfo.IME_ACTION_SEND) {  
            sendMessage();  
            handled = true;  
        }  
        return handled;  
    }  
});
```

EditText- Capturando cambios en el texto

- Existe una forma de controlar programáticamente cualquier interacción del usuario con un campo de texto.
- `addTextChangedListener(TextWatcher watcher);`
- **Interfaz TextWatcher**
 - `public void beforeTextChanged(CharSequence s, int start, int count, int after)`
 - `public void onTextChanged(CharSequence s, int start, int before, int count)`
 - `public void afterTextChanged(Editable s)`

Ejercicio

- Añadir un TextWatcher a un EditText para que cuando el usuario escribe en cualquier sitio la cadena de caracteres «DAM» se elimine el texto completamente del EditText



Concepto de Adaptador

- ¿Qué es un adapter?
 - Mediante el uso de **Adapters** definimos una forma común de acceder a colecciones de datos
 - Un arrayAdapter es un tipo de adaptador para acceder a Arrays -> StringArray
 - Para crear un arrayAdapter, tenemos que pasar tres parámetros:
 - el contexto
 - un layout que se usará para dibujar cada item (en este caso **simple_list_item_1**, que ya viene definido por android)
 - El array con las cadenas de caracteres

EditText - Autocompletar

- Hay que utilizar un Adaptador (Adapter) que provea las sugerencias de texto

(res/values(strings.xml):

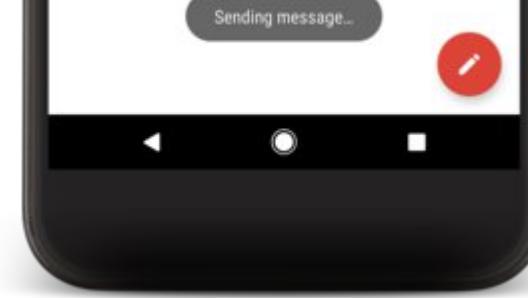
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries_array">
        <item>Afghanistan</item>
        <item>Albania</item>
        <item>Algeria</item>
        <item>American Samoa</item>
        <item>Andorra</item>
        <item>Angola</item>
        <item>Anguilla</item>
        <item>Antarctica</item>
        ...
    </string-array>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<AutoCompleteTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/autocomplete_country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```



```
AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.autocomplete_country);
String[] countries = getResources().getStringArray(R.array.countries_array);
ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, countries);
textView.setAdapter(adapter);
```

Las tostadas (toast)



- Proveen información rápida mediante un

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

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

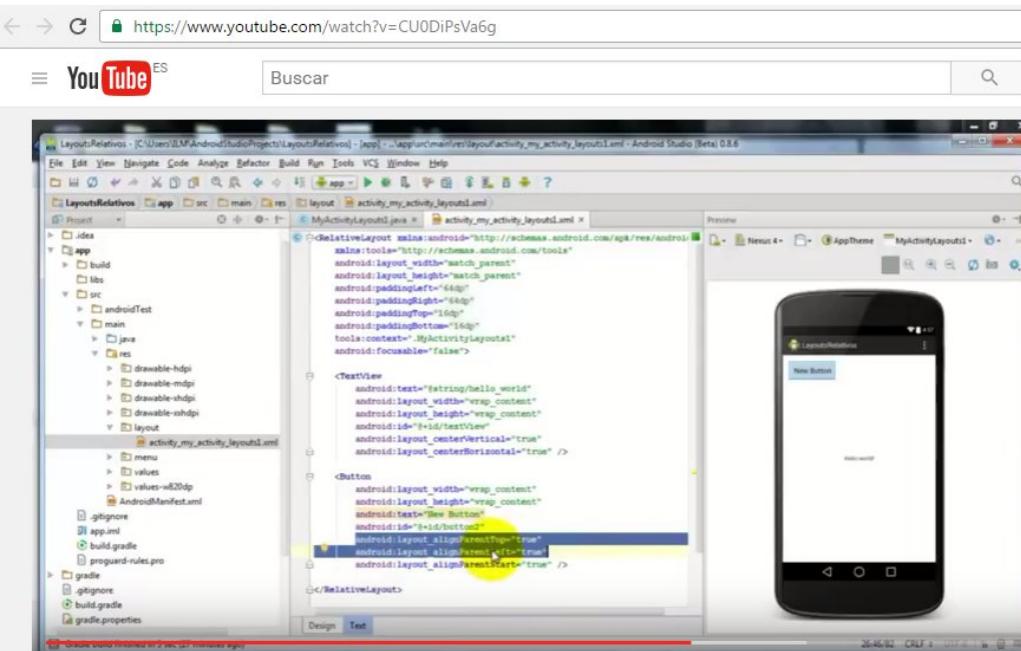
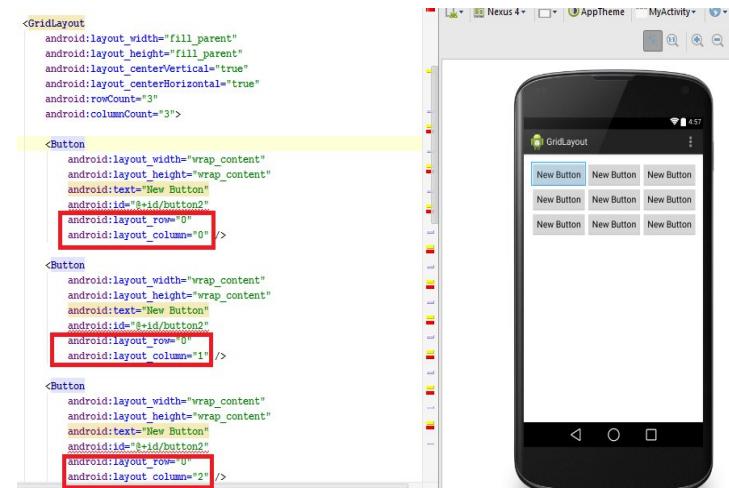


```
Toast.makeText(context, text, duration).show();
```

- Permiten cierto grado de personalización
 - Gravedad y Layout personalizado

LAYOUTS y CONTENEDORES

- ConstraintLayout
 - por defecto
 - Sustituye a RelativeLayout
- LinearLayout
- GridLayout



Unidad 2.- Desarrollando aplicaciones para Android

Recursos de la unidad 2

- [Foro 2](#)
- [Orientaciones para el alumno](#)
- [Código del Tema 2](#)
- [Video sobre los linear layout](#)
- [Video sobre la propiedad peso](#)
- [Video sobre la propiedad gravity](#)
- [Video sobre layouts](#)
- [Video sobre el GridLayout](#)

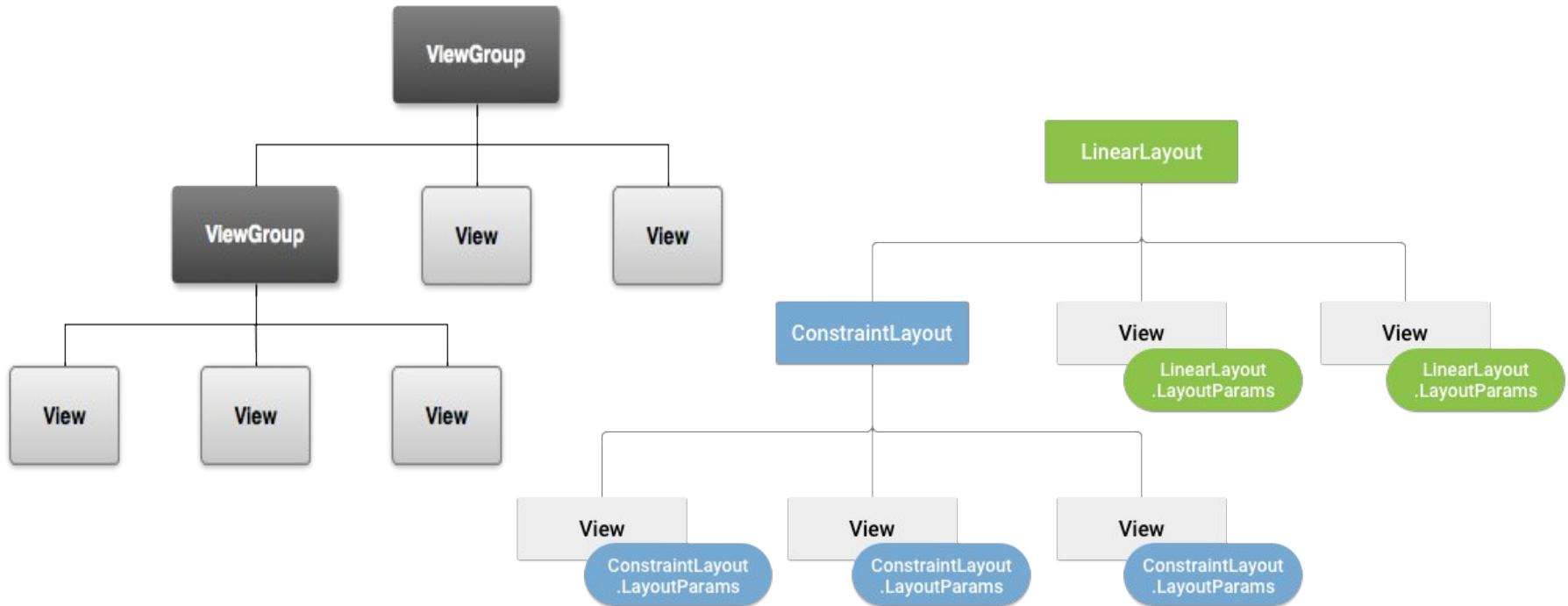


Los layouts

- Permiten organizar un conjunto o colección de widgets en una estructura específica definida como tú quieras
- Se pueden manejar de dos formas:
 - Declarando los ficheros en XML: Como lo has hecho hasta con las vistas de diseño en Android studio.
 - Instanciando objetos programáticamente en tiempo de ejecución.
- Te permite cambiar el diseño de tu interfaz sin tener que tocar código y sin tener que recompilar. Esto te proporciona **FLEXIBILIDAD**
- **Son contenedores!**
 - agrupa controles (también llamados **hijos**) y el layout los ordena/distribuye dentro del contenedor

ViewGroup & LayoutParams

- LayoutParams
 - implementa propiedades importantes que definen la posición y el tamaño de los elementos hijos



LayoutParams

- `layout_width/layout_height`
 - una dimensión determinada, por ejemplo 150dip para indicar que el widget debe tener exactamente ese tamaño.
 - `wrap_content`: Que el widget ocupe su espacio natural
 - `fill_parent/match_parent`: se haría todo lo grande que pudiera hasta llenar el espacio sobrante en el contenedor.

Summary

XML Attributes		
Attribute Name	Related Method	Description
<code>android:layout_height</code>		Specifies the basic height of the view.
<code>android:layout_width</code>		Specifies the basic width of the view.

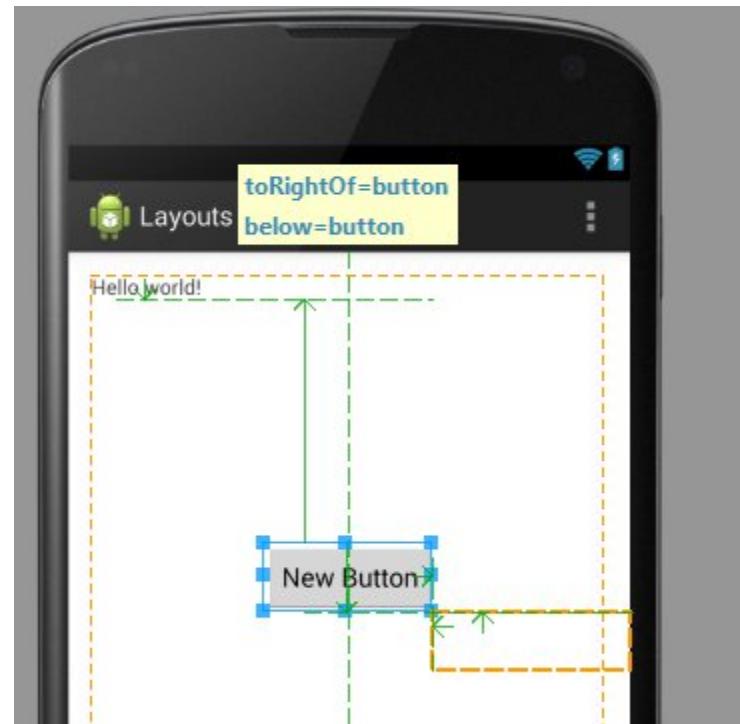
Constants

int	<code>FILL_PARENT</code>	Special value for the height or width requested by a View.
int	<code>MATCH_PARENT</code>	Special value for the height or width requested by a View.
int	<code>WRAP_CONTENT</code>	Special value for the height or width requested by a View.

RelativeLayout

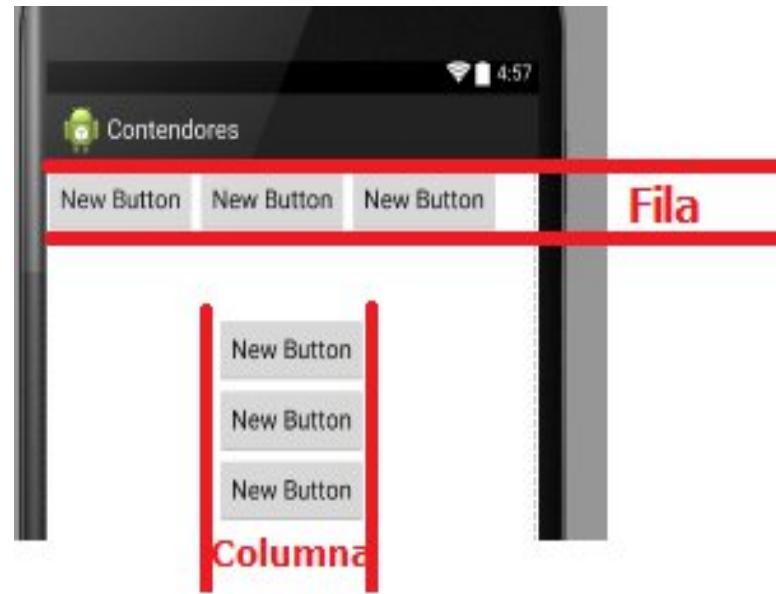
- utiliza un modelo basado en reglas (*rule based model*)
- permite a los hijos (elementos contenidos) especificar su posición con relación al padre (contenedor) o uno de sus hermanos

```
public class  
    RelativeLayout  
extends ViewGroup  
  
java.lang.Object  
└ android.view.View  
    └ android.view.ViewGroup  
        └ android.widget.RelativeLayout
```

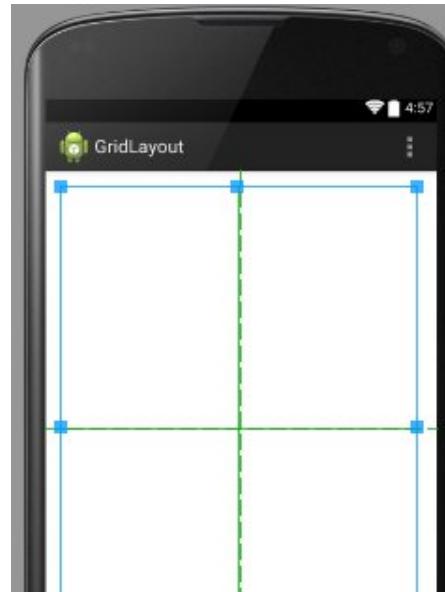


LinearLayouts

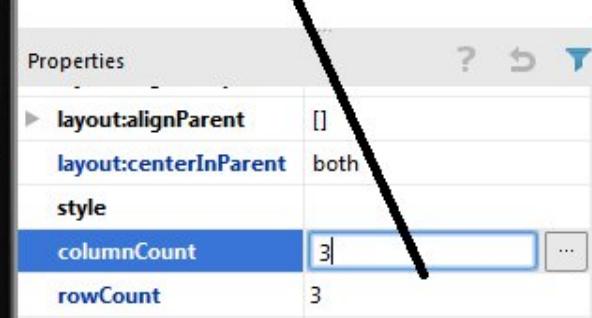
- android:orientation="vertical" si se quiere agrupar los controles en columna
- android:orientation="horizontal" si se quiere agrupar los controles en una fila
- Se puede controlar la «gravedad»
 android:layout_gravity="right"



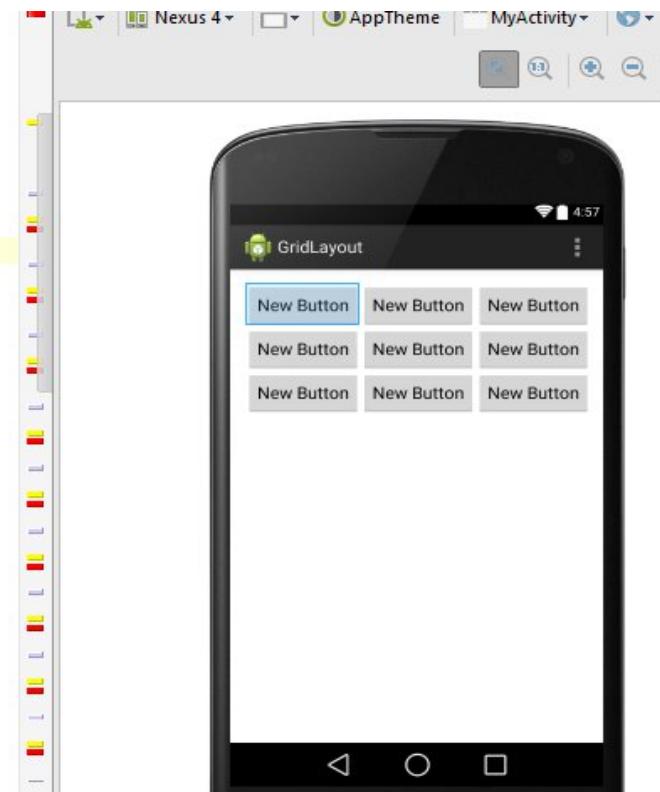
Layout Tabulares (GridLayout)



3 filas, 3 columnas



```
<GridLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_centerVertical="true"  
    android:layout_centerHorizontal="true"  
    android:rowCount="3"  
    android:columnCount="3">  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="New Button"  
    android:id="@+id/button2"  
    android:layout_row="0"  
    android:layout_column="0" />  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="New Button"  
    android:id="@+id/button2"  
    android:layout_row="0"  
    android:layout_column="1" />  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="New Button"  
    android:id="@+id/button2"
```

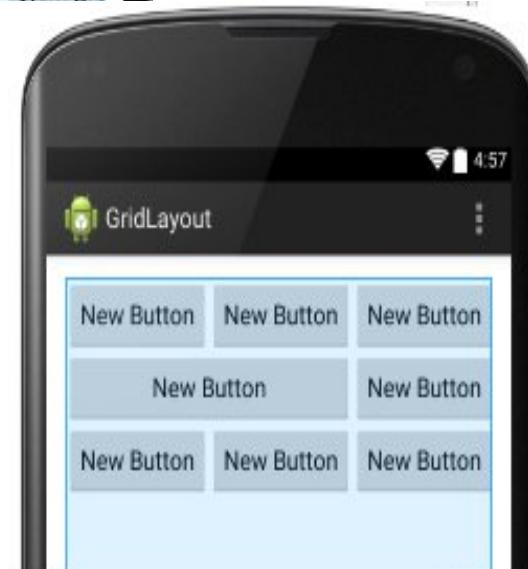


Layout Tabulares – Márgenes y Span

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_margin="20dp"  
    android:text="New Button"  
    android:id="@+id/button2"  
    android:layout_row="0"  
    android:layout_column="0" />
```



```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="New Button"  
    android:id="@+id/button2"  
    android:layout_columnSpan="2"  
    android:layout_gravity="fill"  
    android:layout_row="1"  
    android:layout_column="0" />
```



Recorrido programático de un contenedor

```
public void Recorrer(){  
    View v;  
    GridLayout g = (GridLayout) findViewById(R.id.grid1);  
    for (int i = 0; i < g.getChildCount(); i++) {  
        v=g.getChildAt(i);  
        System.out.println("objeto:"+ v.toString());  
    }  
}
```

GridLayout

```
public class GridLayout  
extends ViewGroup  
  
java.lang.Object  
↳ android.view.View  
→↳ android.view.ViewGroup  
↳ android.widget.GridLayout
```

Diferenciando tipos

```
Button b;  
if(v.getClass().getSimpleName().equals("Button")){  
    b=(Button)v;  
    b.setOnClickListener(...); //o cualquier otro método/prop.de la clase Button  
}
```

GridLayout

Añadiendo elementos

```
public void añadeHijos() {  
    GridLayout g = (GridLayout) findViewById(R.id.grid1);  
    Button b;  
    for(int i=0;i<18;i++) {  
        b = new Button(this);  
        b.setLayoutParams(new ViewGroup.LayoutParams(  
            ViewGroup.LayoutParams.WRAP_CONTENT,  
            ViewGroup.LayoutParams.WRAP_CONTENT));  
        b.setText("btn" + i);  
        b.setId(View.generateViewId());  
        g.addView(b,i);  
    }  
}
```

```
public class GridLayout  
extends ViewGroup  
  
java.lang.Object  
↳ android.view.View  
↳ android.view.ViewGroup  
↳ android.widget.GridLayout
```

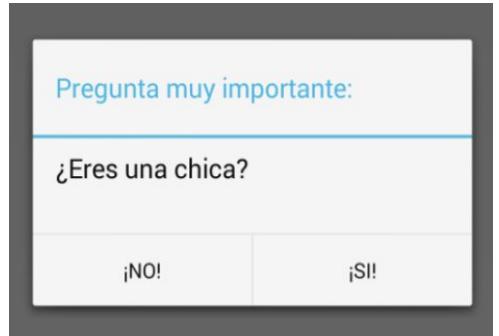


Ejercicio

Realiza una aplicación que programáticamente maneje controles dentro de un GridLayout. El programa deberá crear 18 botones automáticamente con diferentes colores, al pulsar cada uno de ellos, se volverá blanco. El último botón será un botón de RESET, de tal manera que cuando se pulse, se vuelva a la configuración original.

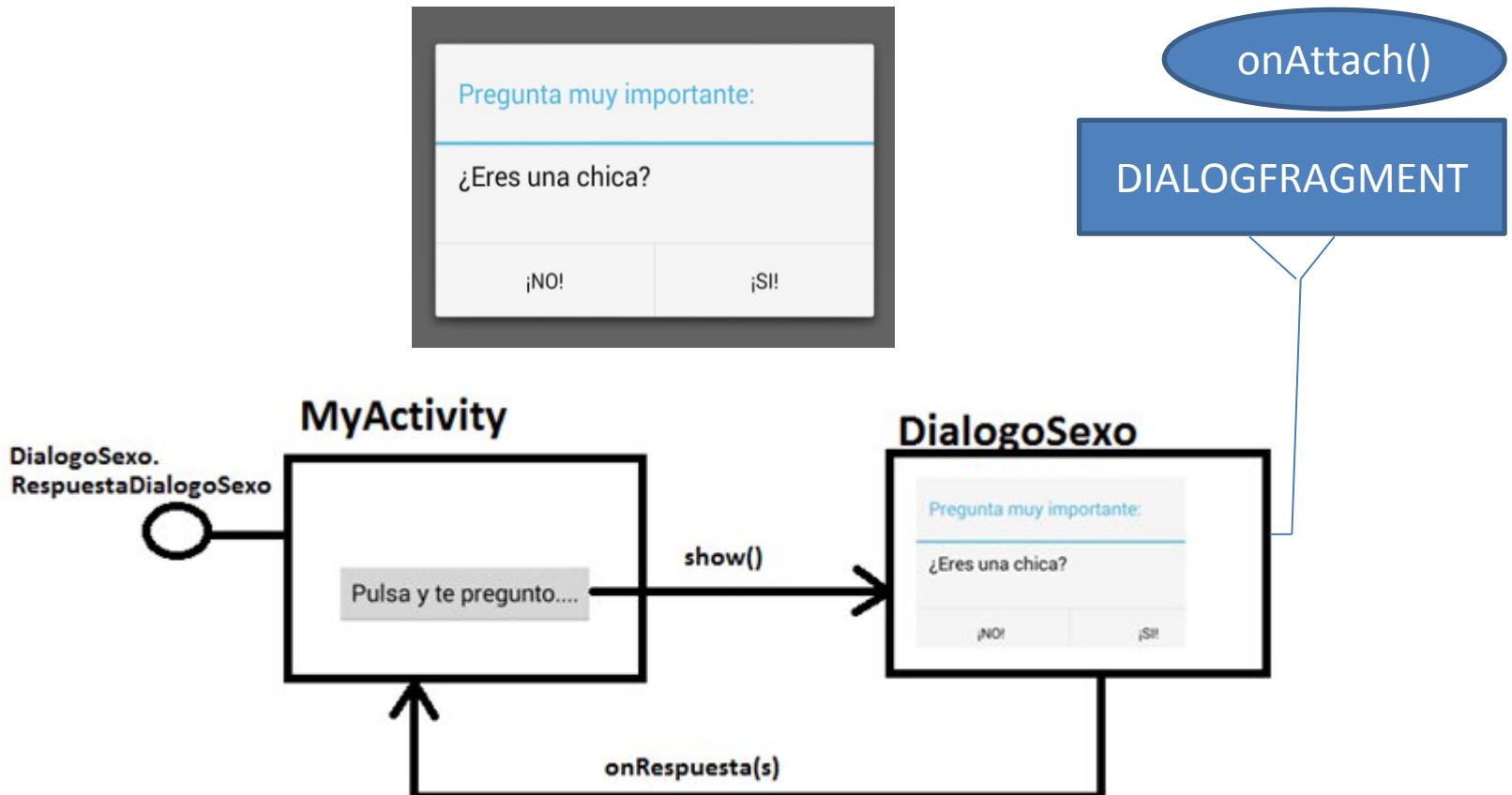


Diálogos



- Android recomienda usar la clase **DialogFragment** para contener diálogos
 - Un fragmento de una actividad es una parte de la App que controla una parte de la interfaz de usuario de la Activity
 - Los fragmentos tienen su propio ciclo de vida, reciben sus propios eventos y se pueden eliminar o añadir a la Activity mientras esta se ejecuta
 - Un DialogFragment es un tipo especial de fragmento usado para crear diálogos

Diálogos - Caso práctico



AlertDialog.Builder

```
→ builder.setPositiveButton("¡SI!", new DialogInterface.OnClickListener(){  
    public void onClick(DialogInterface dialog, int id) {  
        respuesta.onRespuesta("Es una chica!");  
    }  
});
```

[ficheros_tema2/DialogosConRespuesta.zip](#)

Diálogos

- Doc para diseño personalizado:

<http://developer.android.com/guide/topics/ui/dialogs.html#CustomLayout>

The screenshot shows the Android Developers website with a green header bar. The header includes the Android logo, the word 'Developers', and navigation tabs for 'DISEÑAR', 'DESARROLLAR', and 'DISTRIBUIR'. A search bar is on the right. On the left, there's a sidebar with links like 'Guías de la API', 'Menús', 'Configuración', 'Cuadros de diálogo' (which is highlighted in blue), 'Notificaciones', 'Avisos', 'Búsqueda', 'Compatibilidad con ventanas múltiples', 'Arrastrar y soltar', and 'Accesibilidad'. The main content area has a title 'Crear un diseño personalizado'. It contains text explaining how to create a custom dialog design by creating a layout XML file and setting it with `AlertDialog.setView()`. It also mentions that by default, the custom design occupies the entire dialog window but can still use `AlertDialog.Builder` for buttons and titles. An example XML code snippet for 'dialog_signin.xml' is shown:

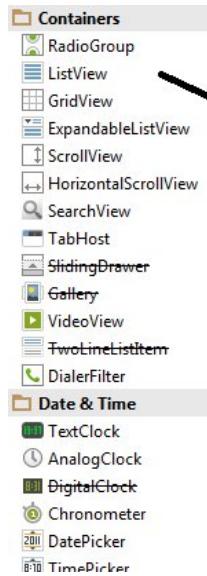
```
res/layout/dialog_signin.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ImageView
        android:src="@drawable/header_logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

To the right of the text, there's a visual representation of a custom dialog window titled 'ANDROID APP'. It has two text input fields labeled 'Username' and 'Password', and two buttons at the bottom labeled 'Cancel' and 'Sign in'.

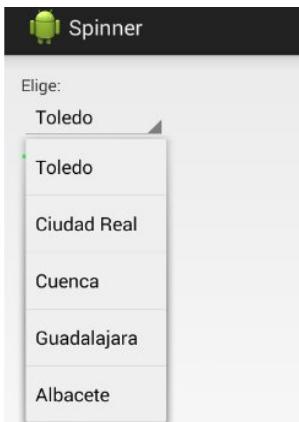
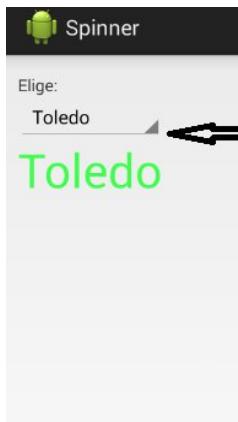
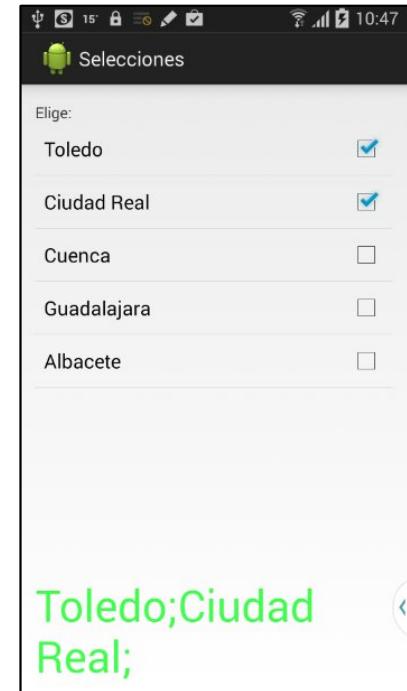
Figura 5: Diseño de diálogo personalizado.

WIDGETS DE SELECCIÓN



[ficheros_tema2/SeleccionMultiple.zip](#)

[ficheros_tema2/Selecciones.zip](#)



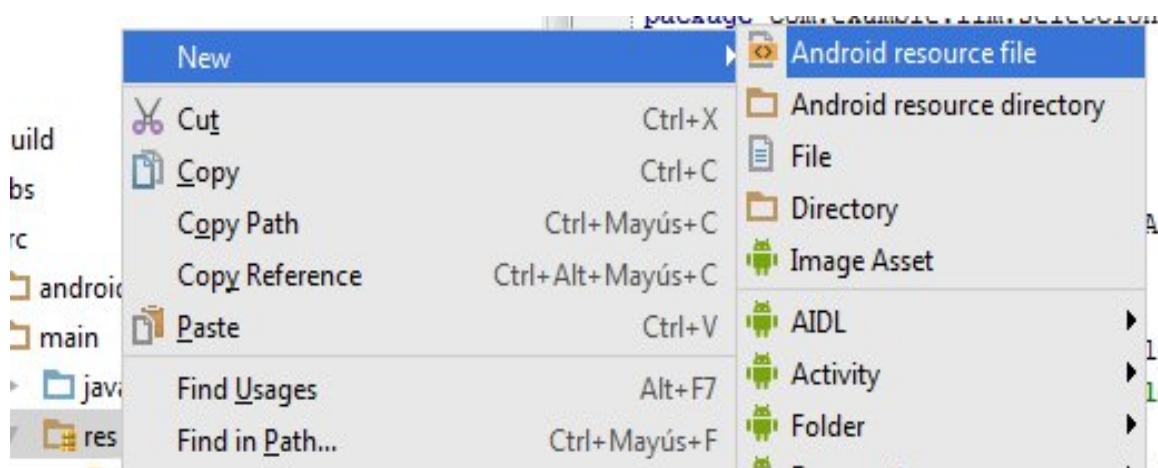
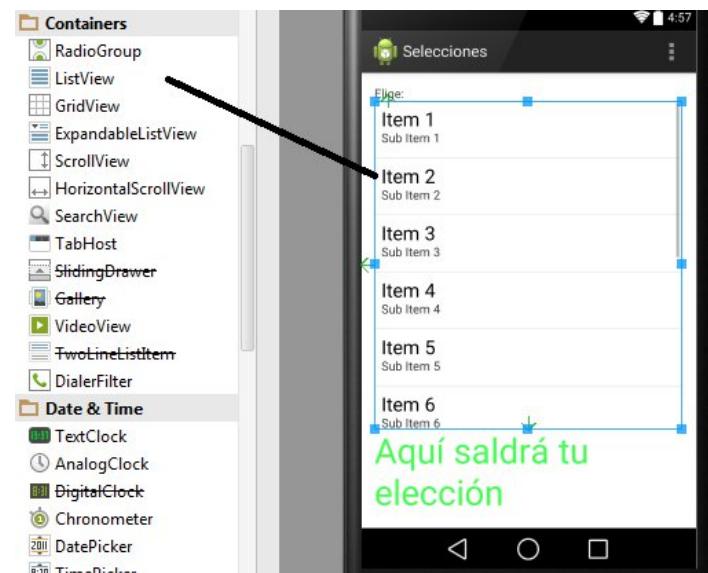
[ficheros_tema2/DatePicker.zip](#)



[ficheros_tema2/Spinner.zip](#)

ListView

- Necesitarás
 - Un adaptador
 - Un nuevo recurso a modo de layout file con la definición de un TextView que se usará como elemento simple de tu ListView.



Fila.xml

```
<TextView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:orientation="vertical"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:textStyle="italic"  
        android:textSize="20dp"  
        android:textColor="#FF0000"  
    >  
</TextView>
```

Código para la ListView

```
//Crear un array con los elementos seleccionables
String [] elementos={"Toledo","Ciudad Real",
                     "Cuenca","Guadalajara","Albacete"};  
  
//Declaras un adaptador de Texto (String)
ArrayAdapter<String> adaptador;  
  
//Obtienes una referencia a la lista
ListView l=(ListView)findViewById(R.id.listView);  
  
//Crees el adaptador
adaptador=new ArrayAdapter<String>(this,R.layout.fila,elementos);
//Le das el adaptador a la lista
l.setAdapter(adaptador);
```

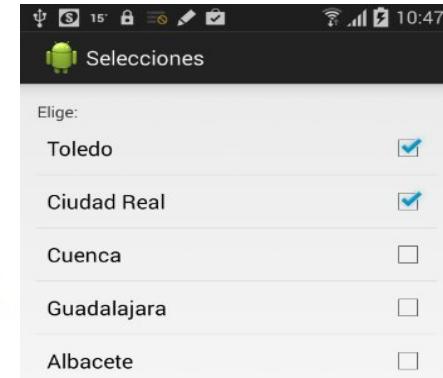


```
public class MyActivity extends Activity implements ListView.OnItemClickListener{  
  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id){  
        TextView t=(TextView) findViewById(R.id.textView);  
        t.setText("Has elegido:" + parent.getItemAtPosition(position).toString());  
    }  
    ...  
}
```

Selección de múltiples elementos

```
adaptador=new ArrayAdapter<String>(this,  
        android.R.layout.simple_list_item_multiple_choice, elementos);
```

```
public void onItemClick(AdapterView<?> a, View view, int position, long id) {  
    TextView t=(TextView)findViewById(R.id.textView);  
    ListView l=(ListView)findViewById(R.id.listView);  
    String seleccionado=new String();  
    SparseBooleanArray checked = l.getCheckedItemPositions();  
  
    for(int i=0;i<checked.size();i++)  
        if(checked.valueAt(i)){  
            seleccionado+=  
                a.getItemAtPosition(checked.keyAt(i)).toString()  
                +";";  
        }  
    t.setText(seleccionado);  
}
```



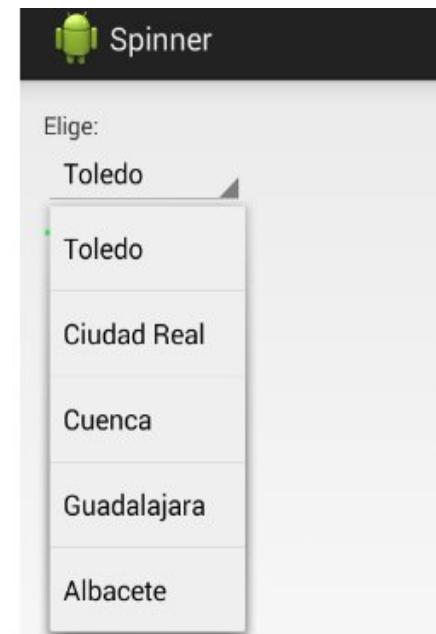
```
        android:choiceMode="multipleChoice"  
        o  
        setChoiceMode(CHOICE_MODE_MULTIPLE)
```

Toledo;Ciudad
Real;

Los spinners

```
adaptador = new ArrayAdapter<String>(this,  
        android.R.layout.simple_spinner_item, elementos);  
adaptador.setDropDownViewResource(  
        android.R.layout.simple_spinner_dropdown_item);
```

```
protected void onCreate(Bundle savedInstanceState) {  
    String[] elementos = {"Toledo", "Ciudad Real",  
        "Cuenca", "Guadalajara", "Albacete"};  
  
    ArrayAdapter<String> adaptador;  
  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_my);  
  
    Spinner sp = (Spinner) findViewById(R.id.spinner);  
    adaptador = new ArrayAdapter<String>(this,  
        android.R.layout.simple_spinner_item, elementos);  
    adaptador.setDropDownViewResource(  
        android.R.layout.simple_spinner_dropdown_item);  
    sp.setAdapter(adaptador);  
    sp.setOnItemSelectedListener(this);  
}
```



Los Callbacks para Spinners

```
public class MyActivity extends Activity implements
    Spinner.OnItemSelectedListener{

    //Callback cuando se selecciona un elemento del Spinner
    public void onItemSelected(AdapterView<?> a,
        View view, int position, long id){

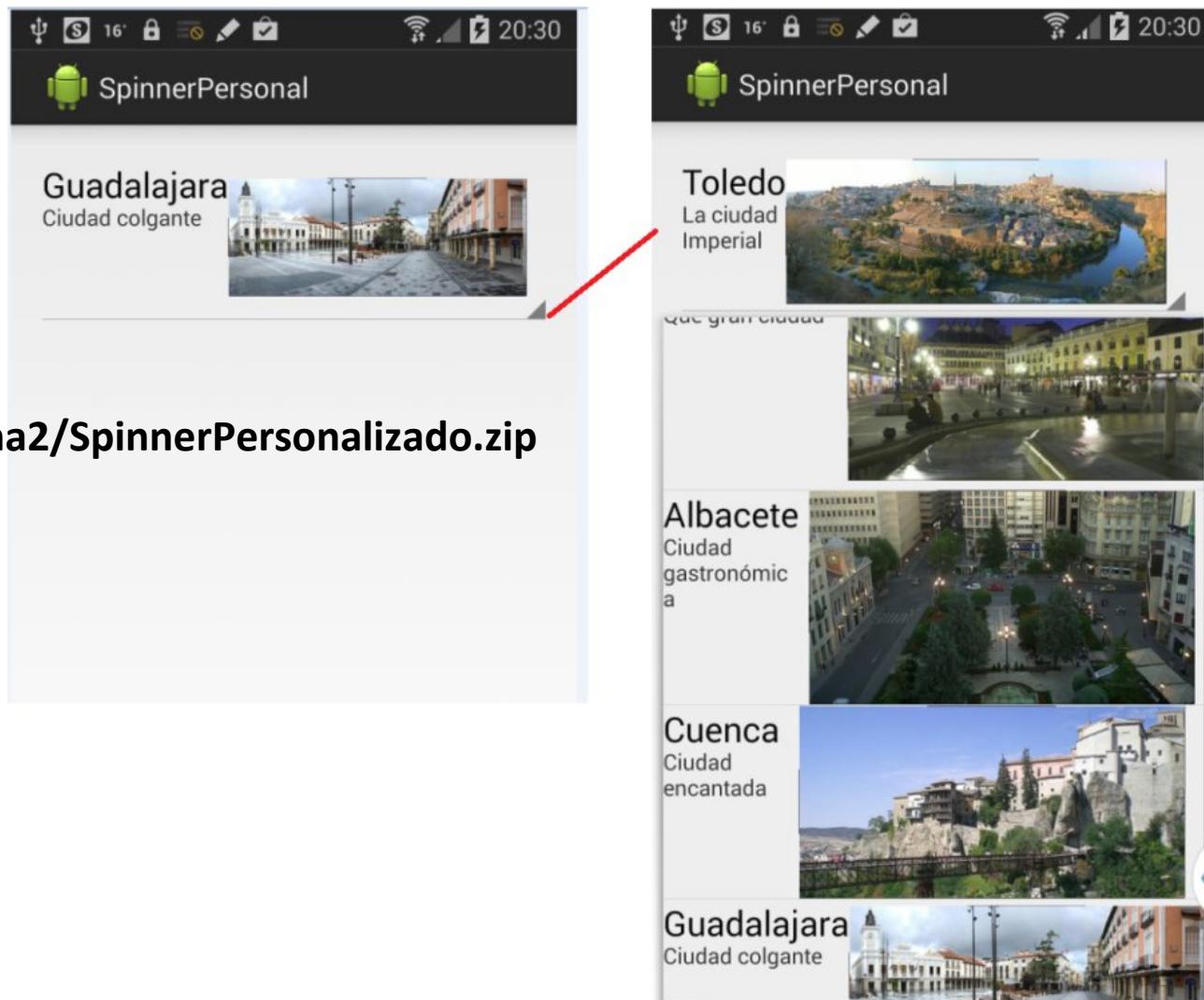
        TextView t=(TextView) findViewById(R.id.textView);
        Spinner sp = (Spinner) findViewById(R.id.spinner);

        t.setText(sp.getSelectedItem().toString());
    }

    //Callback cuando se no se selecciona un elemento del Spinner
    public void onNothingSelected(AdapterView<?> a){
        TextView t=(TextView) findViewById(R.id.textView);
        t.setText("No se ha seleccionado nada");
    }

    ...
}
```

WIDGETS DE SELECCIÓN – selecciones personalizables



ficheros_tema2/SpinnerPersonalizado.zip

Selecciones personalizables

- Cualquier tipo de control de selección es personalizable
- Hay que reescribir el funcionamiento de `ArrayAdapter<T>`
 - Cuando el selector demande los datos de las filas, devuelva una vista que sea la que nosotros queramos en cada momento
 - `getDropDownView(int position, View convertView, ViewGroup parent)`
 - `getView(int position, View convertView, ViewGroup parent)`
- El inflador (inflater)
 - se puede crear un widget personalizado e “inflarlo” (crearlo) habiendo definido su estructura mediante un Layout definido en XML

Lineaspinner.xml

```
<RelativeLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    xmlns:android="http://schemas.android.com/apk/res/android">  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Albacete, qué gran ciudad!"  
        android:id="@+id/descripcion"  
        android:layout_alignRight="@+id/nombre"  
        android:layout_alignEnd="@+id/nombre" />  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Albacete"/>  
  
    <ImageView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:src="@drawable/albacete"  
        android:id="@+id/imagenCiudad"/>  
/<RelativeLayout>
```

**textView
nombre**

**textView
descripción**

**ImageView
imagenCiudad**



```
String[] ciudades = { "Toledo", "Ciudad Real",
                      "Albacete", "Cuenca", "Guadalajara" };
String[] descripciones = { "La ciudad Imperial", "Qué gran ciudad",
                           "Ciudad gastronómica", "Ciudad encantada", "Ciudad colgante" };

int imagenes[] = { R.drawable.toledo, R.drawable.ciudadreal,
                   R.drawable.albacete, R.drawable.cuenca,
                   R.drawable.guadalajara};

public class AdaptadorPersonalizado extends ArrayAdapter<String> {
    public AdaptadorPersonalizado(Context ctx,
                                   int txtViewResourceId, String[] objects){
        super(ctx, txtViewResourceId, objects);
    }

    @Override
    public View getDropDownView(int position, View cnvtView, ViewGroup prnt){
        return crearFilaPersonalizada(position, cnvtView, prnt);
    }

    @Override
    public View getView(int pos, View cnvtView, ViewGroup prnt){
        return crearFilaPersonalizada(pos, cnvtView, prnt);
    }
}
```

.... A inflar...

OJO,
attachToRoot

```
public View crearFilaPersonalizada(int position,
        View convertView, ViewGroup parent){

    LayoutInflater inflater = getLayoutInflater();
    View miFila = inflater.inflate(R.layout.lineaspiner, parent, false);

    TextView nombre = (TextView) miFila.findViewById(R.id.nombre);
    nombre.setText(ciudades[position]);

    TextView descripcion = (TextView) miFila.findViewById(R.id.descripcion);
    descripcion.setText(descripciones[position]);

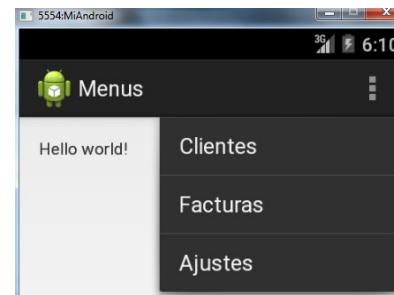
    ImageView imagen = (ImageView) miFila.findViewById(R.id.imagenCiudad);
    imagen.setImageResource(imagenes[position]);
    return miFila;

}
```

MENÚS

Menú de Opciones

- Action Bar / Settings



Contextuales

- Long-click

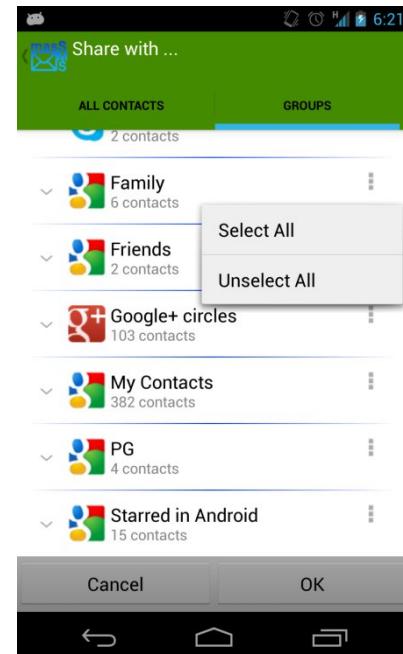


Submenús



ficheros_tema2/MenuDemo.zip

Pop-ups



Actionbar

Crea la carpeta *menu* en tu proyecto

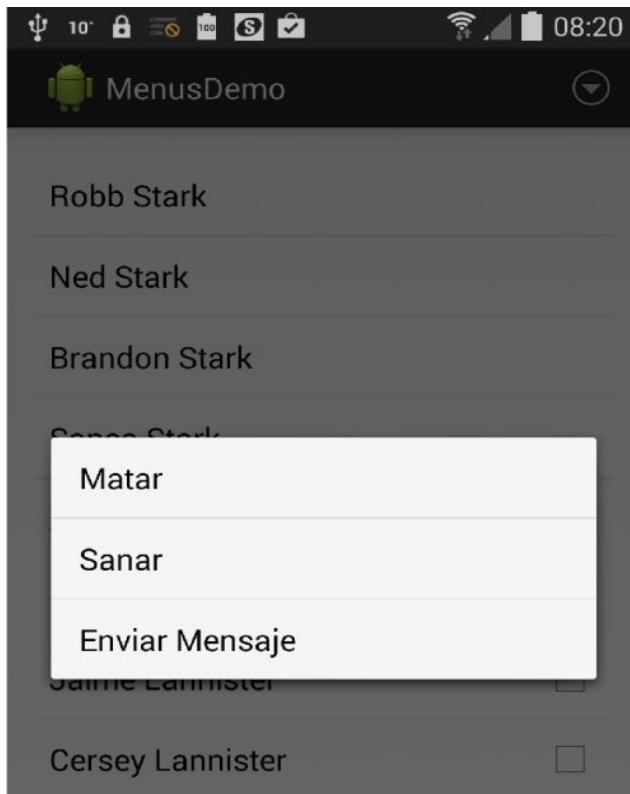


```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_height="wrap_content"
      android:layout_width="wrap_content"
      tools:context=".MainActivity" >
    <item android:id="@+id/ajustes"
          android:title="Ajustes"
          android:orderInCategory="100"
          android:icon="@drawable/new_york"
          app:showAsAction="ifRoom">
    ...
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.mimenu,menu);
        return true;
    }
```

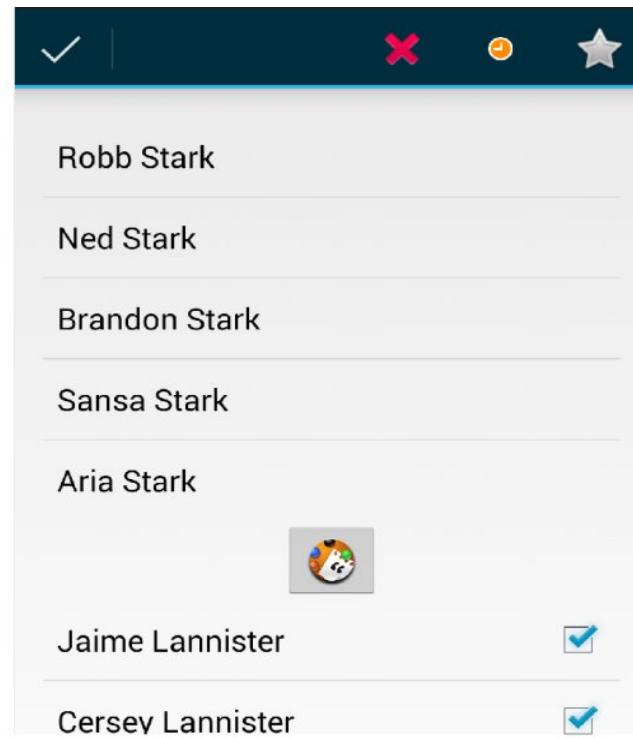
Respondiendo a las acciones

```
@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.BuscarCliente) {
        Toast.makeText(getApplicationContext(), "Se ha pulsado Buscar Cliente",Toast.LENGTH_LONG).show();
        return true;
    } else if (id == R.id.Cuentas) {
        Toast.makeText(getApplicationContext(),
            "Se ha pulsado Cuentas", Toast.LENGTH_LONG).show();
        return true;
    } else if (id == R.id.Facturas) {
        Toast.makeText(getApplicationContext(),
            "Se ha pulsado Facturas", Toast.LENGTH_LONG).show();
        return true;
    } else{
        Toast.makeText(getApplicationContext(), "Se ha pulsado otro elemento de
menu("+item.getTitle()+"")",Toast.LENGTH_LONG).show();
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Menús contextuales



Menú contextual



Menú en Contextual Action Bar (CAB)

starks

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:icon="@android:drawable/ic_delete"
          android:title="Matar"
          android:id="@+id/matar">
    </item>
    <item android:icon="@android:drawable/ic_menu_edit"
          android:title="Sanar"
          android:id="@+id/sanar">
    </item>
    <item android:icon="@android:drawable/sym_call_incoming"
          android:title="Enviar Mensaje"
          android:id="@+id/enviarmensjae">
    </item>
</menu>
```

Registrar menú contextual

```
//Creamos lista de starks para el menú contextual  
starks=(ListView)findViewById(R.id.listaStarks);  
registerForContextMenu(starks);
```

Crear menú y responder a evento

```
@Override  
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuItemInfo menuInfo) {  
    MenuInflater m=getMenuInflater();  
    m.inflate(R.menu.starks,menu);  
    super.onCreateContextMenu(menu, v, menuInfo);  
}  
  
@Override  
public boolean onContextItemSelected(MenuItem item) {  
    AdapterView.AdapterContextMenuInfo info =  
        (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();  
    int itemId = item.getItemId();  
    if (itemId == R.id.matar) {  
        Toast.makeText(getApplicationContext(), "Hemos matado a " +  
            starks.getItemAtPosition(info.position), Toast.LENGTH_LONG).show();  
        return true;  
    } else if (itemId == R.id.sanar) {  
        Toast.makeText(getApplicationContext(), "Hemos sanado a " +  
            starks.getItemAtPosition(info.position), Toast.LENGTH_LONG).show();  
        return true;  
    } else if (itemId == R.id.enviarmensaje) {  
        Toast.makeText(getApplicationContext(), "Le hemos enviado un mensaje a " +  
            starks.getItemAtPosition(info.position), Toast.LENGTH_LONG).show();  
        return true;  
    }  
    Toast.makeText(getApplicationContext(), "Le hemos hecho otra cosa a " +  
        starks.getItemAtPosition(info.position), Toast.LENGTH_LONG).show();  
    return true;  
}
```

Lannisters

The screenshot shows an Android XML menu editor window titled "lannisters.xml". The code in the editor is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:icon="@android:drawable/ic_delete"
          android:title="Aniquilar"
          android:id="@+id/aniquilar"
          ></item>
    <item android:icon="@android:drawable/presence_away"
          android:title="Encerrar"
          android:id="@+id/encerrar"
          ></item>
    <item android:icon="@android:drawable/btn_star"
          android:title="Salvar"
          android:id="@+id/salvar"
          ></item>
</menu>
```

The code consists of three menu items. The first item has an error icon (red X) and a warning icon (orange exclamation mark). The third item has a lightbulb icon, indicating a suggestion or warning.

Crear modo de acción y asociarlo al OnItemClick listView

```
ActionMode mActionMode;  
  
public void onItemClick(AdapterView<?> p, View v, int position, long id){  
  
    mActionMode = MyActivity.this.startActionMode(mActionModeCallback);  
    v.setSelected(true);  
  
}
```



```
private ActionMode.Callback mActionModeCallback = new ActionMode.Callback() {
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        // Inflate a menu resource providing context menu items
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.lannisters, menu);
        return true;
    }
    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false; // Return false if nothing is done
    }
    // Se llama a este método cuando se ha pulsado en la lista de los lannisters
    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        int itemId = item.getItemId();
        if (itemId == R.id.aniquilar) { //hay que crear un Aniquilar() para
            //recorrer todos los elementos seleccionado (checked) en la listView
            Toast.makeText(getApplicationContext(), "Hemos aniquilado a algún
                Lannister", Toast.LENGTH_LONG).show();
            return true;
        } else if (itemId == R.id.encerrar) {
            Toast.makeText(getApplicationContext(), "Hemos encerrado a algún
                Lannister", Toast.LENGTH_LONG).show();
            return true;
        } else if (itemId == R.id.salvar) {
            Toast.makeText(getApplicationContext(), "Hemos salvado a algún Lannister",
                Toast.LENGTH_LONG).show();
            return true;
        } return false;
    }
    @Override
    public void onDestroyActionMode(ActionMode mode) {
        mActionMode = null;
    }
};
```

Más temas que van más allá del temario



EL APPBAR / ACTION BAR

- Acciones siempre visibles en tu Activity



- Permite anexar menús contextuales



<https://developer.android.com/training/appbar/?hl=es-419>

- Lecciones

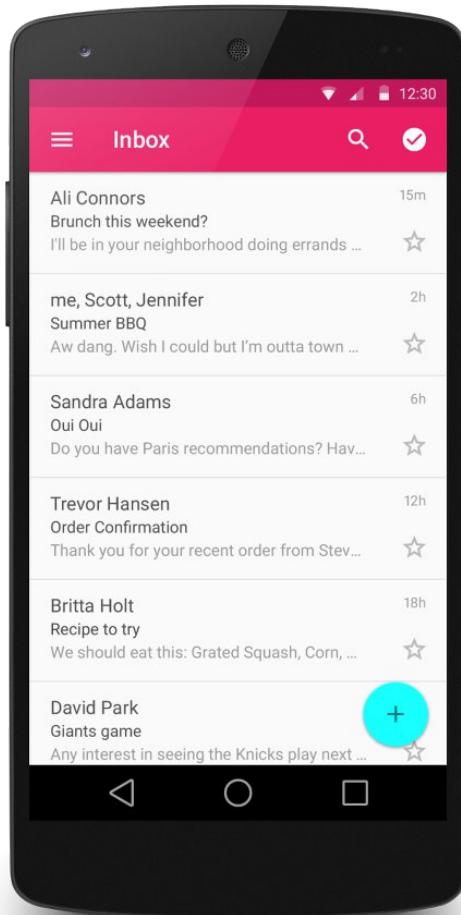
OTROS WIDGETS

Common	Ab TextView	Common	Ab TextView	Common	Button	Common	View
Text	■ Button	Text	Ab Plain Text	Text	■ ImageButton	Text	■ ImageView
Buttons	■ ImageView	Buttons	Ab Password	Buttons	✓ CheckBox	Buttons	○ WebView
Widgets	■ RecyclerView	Widgets	Ab Password (Numeric)	Widgets	○ RadioGroup	Widgets	■ VideoView
Layouts	<> <fragment>	Layouts	Ab E-mail	Layouts	○ RadioButton	Layouts	■ CalendarView
Containers	■ ScrollView	Containers	Ab Phone	Containers	■ ToggleButton	Containers	○ ProgressBar
Google	■ Switch	Google	Ab Postal Address	Google	■ Switch	Google	■ ProgressBar (Horizontal)
Legacy		Legacy	Ab Multiline Text	Legacy	+ FloatingActionButton	Legacy	■ SeekBar
	Common		Ab Time				○ SeekBar (Discrete)
	Text		Ab Date				★ RatingBar
	Buttons		Ab Number				🔍 SearchView
	Widgets		Ab Number (Signed)				☒ TextureView
	Layouts		Ab Number (Decimal)				⋮ SurfaceView
	Containers		Ab AutoCompleteTextView				□ Horizontal Divider
	Google		☒ MultiAutoCompleteTextView				□ Vertical Divider
			☒ CheckedTextView				
			Ab TextInputLayout				
Common	.ConstraintLayout	Common	Spinner	Common	GridLayout	Common	
Text	I---I Guideline (horizontal)	Text	■ RecyclerView	Text	■ ListView	Text	
Buttons	I Guidelines (vertical)	Buttons	■ ScrollView	Buttons	■ TabHost	Buttons	
Widgets	■ LinearLayout (horizontal)	Widgets	■ HorizontalScrollView	Widgets	■ RelativeLayout	Widgets	
Layouts	■ LinearLayout (vertical)	Layouts	■ NestedScrollView	Layouts	■ GridView	Layouts	
Containers	■ FrameLayout	Containers	■ ViewPager	Containers		Containers	
Google	■ TableLayout	Google	■ CardView	Google		Google	
Legacy	■ TableRow	Legacy	■ Tabs	Legacy		Legacy	
	I---I Space		■ AppBarLayout				
			■ NavigationView				
			■ BottomNavigationView				
			■ Toolbar				
			■ TabLayout				
			■ TabItem				
			⋮ ViewStub				
			↳ <include>				
			⟨⟩ <fragment>				
			□ <view>				
			☒ <requestFocus>				

Android JetPack

RecyclerView + CardView

- Listview es ahora “Legacy”
- RecyclerView es más flexible que la ListView
- CardView está de moda



MÁS FUNCIONES DE CALLBACK

The screenshot shows an Android Studio code editor with Java code for an Activity. A dropdown menu is open over the word 'On' in the interface 'View.On' block, listing various View.OnClickListener methods. The first method in the list is highlighted.

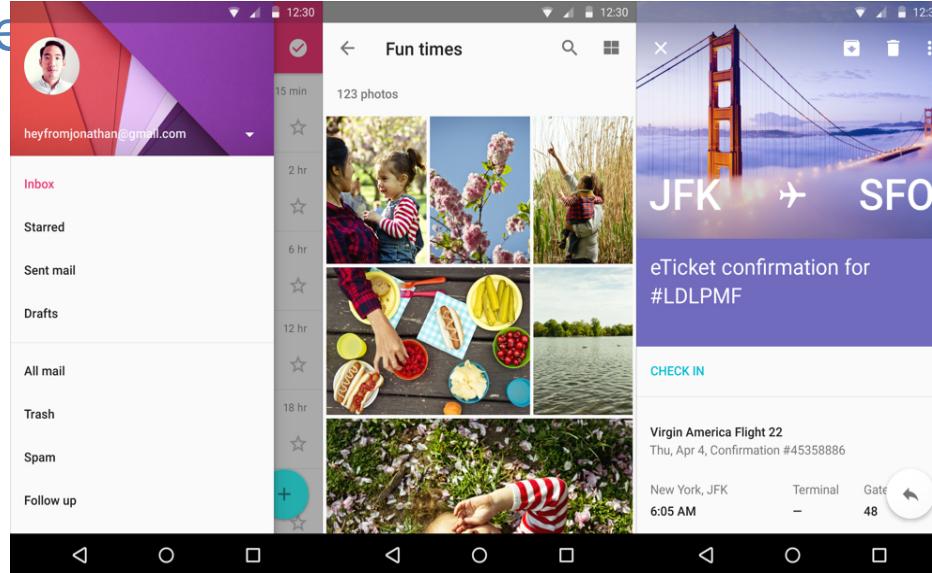
```
public class MyActivity extends Activity implements View.OnClickLister {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_my);  
    }  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; this adds items to the action bar.  
        getMenuInflater().inflate(R.menu.my, menu);  
        return true;  
    }  
  
    @Override  
    public boolean onOptionsItemSelected(MenuItem item) {  
        // Handle action bar item clicks here.  
        // automatically handle clicks on the Hamburger icon  
        // as you specify a parent activity in AndroidManifest.xml.  
        int id = item.getItemId();  
        if (id == R.id.action_settings) {  
            return true;  
        }  
    }  
}
```

View.OnClickListener (android.view.View) (android.view)
View.OnCreateContextMenuListener (android.view.View) (android.view)
View.OnApplyWindowInsetsListener (android.view.View) (android.view)
View.OnAttachStateChangeListener (android.view.View) (android.view)
View.OnDragListener (android.view.View) (android.view)
View.OnGenericMotionListener (android.view.View) (android.view)
View.OnFocusChangeListener (android.view.View) (android.view)
View.OnHoverListener (android.view.View) (android.view)
View.OnKeyListener (android.view.View) (android.view)
View.OnLayoutChangeListener (android.view.View) (android.view)
View.OnLongClickListener (android.view.View) (android.view)
View.OnSystemUiVisibilityChangeListener (android.view.View) (android.view)
View.OnTouchListener (android.view.View) (android.view)
View.KEEP_SCREEN_ON (android.view) int
View.SCREEN_STATE_ON (android.view) int

Press Ctrl+Punto to choose the selected (or first) suggestion and insert a dot afterwards >>

ESTILOS, TEMAS Y MATERIAL DESIGN

- Estilo
 - colección de propiedades que especifican cómo se dibujar una vista (View) o ventana
- Temas
 - Colección de estilos
- Material Design
 - Filosofía de...



Más plantillas

