

Menús

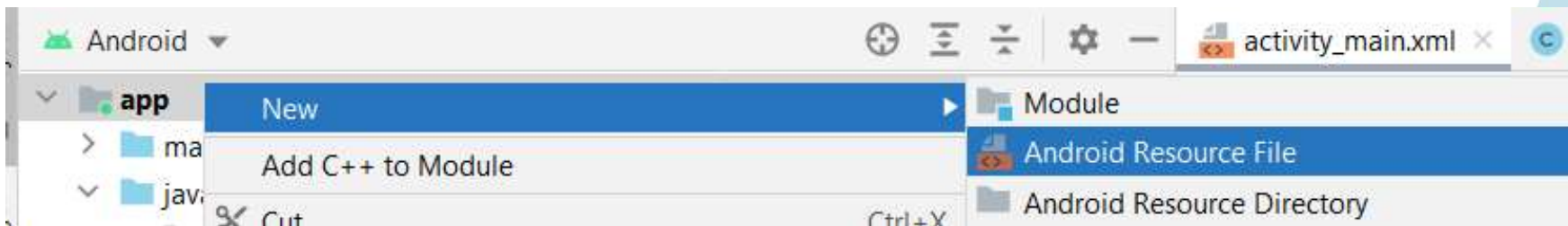


# Menú I

- ▶ Los menús son elementos gráficos de una aplicación móvil que nos sirven para poder navegar por las distintas opciones que nos ofrezca nuestra App.
- ▶ Tenemos tres tipos diferentes de menús:
  - ▶ **Menú de opciones:** es la colección principal de elementos de menú de una actividad. Es donde debes colocar las acciones que tienen un impacto global en la app, como "Buscar", "Redactar correo electrónico" y "Configuración".
  - ▶ **Menú contextual:** es un menú flotante que aparece cuando el usuario hace un clic largo en un elemento. Proporciona acciones que afectan al contenido seleccionado o al marco contextual.
  - ▶ **Menú emergente:** muestra una lista de elementos en una lista vertical que está anclada a la vista que invocó el menú. Es adecuado para proporcionar una ampliación de acciones relacionadas con contenido específico o para proporcionar opciones en una segunda parte de un comando.

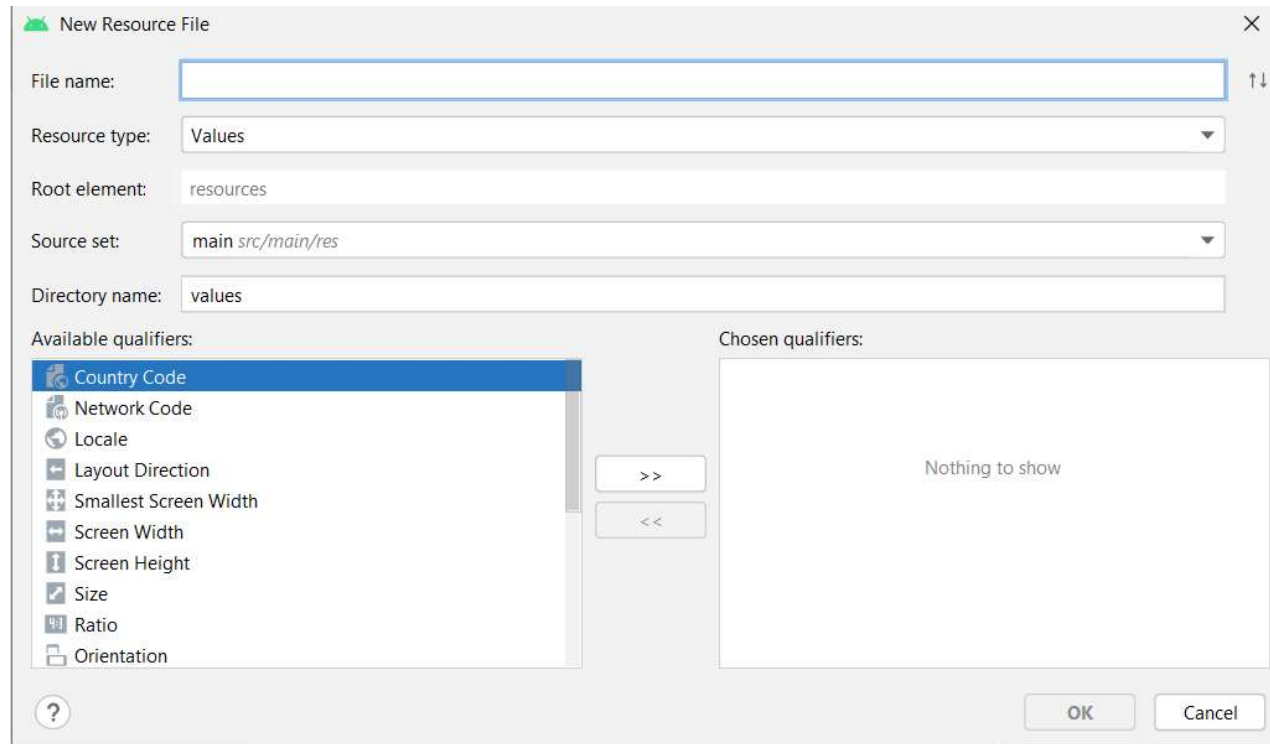
## Menú II

- ▶ Vamos a crear un MENÚ DE OPCIONES. Para ello vamos a generar un nuevo proyecto, al que llamaremos CreacionMenu.
- ▶ Una vez creado, vamos a crear nuestro menú. Para ello vamos a pulsar el botón derecho sobre la app y elegiremos la opción new. Dentro de new vamos a elegir la opción Android Resource File. De esta manera crearemos un XML de tipo menú dentro de nuestra aplicación.



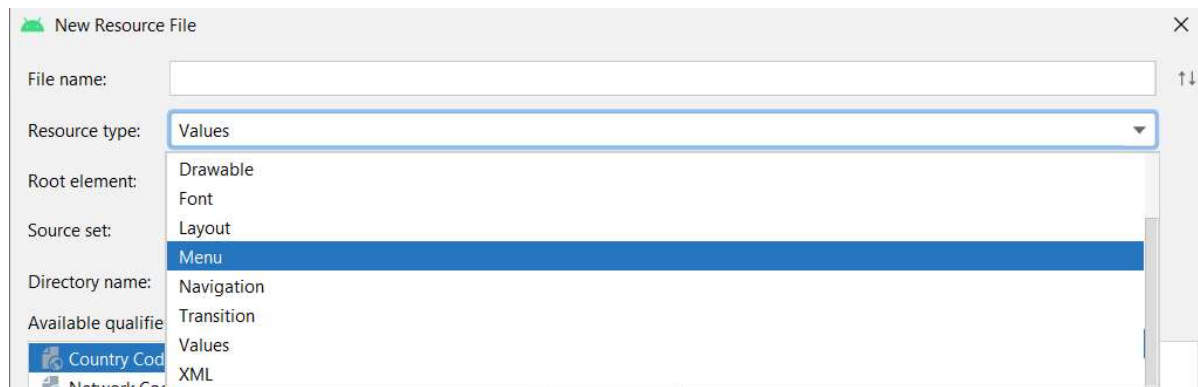
# Menú III

- Una vez elegida la opción aparecerá la siguiente pantalla:



## Menú IV

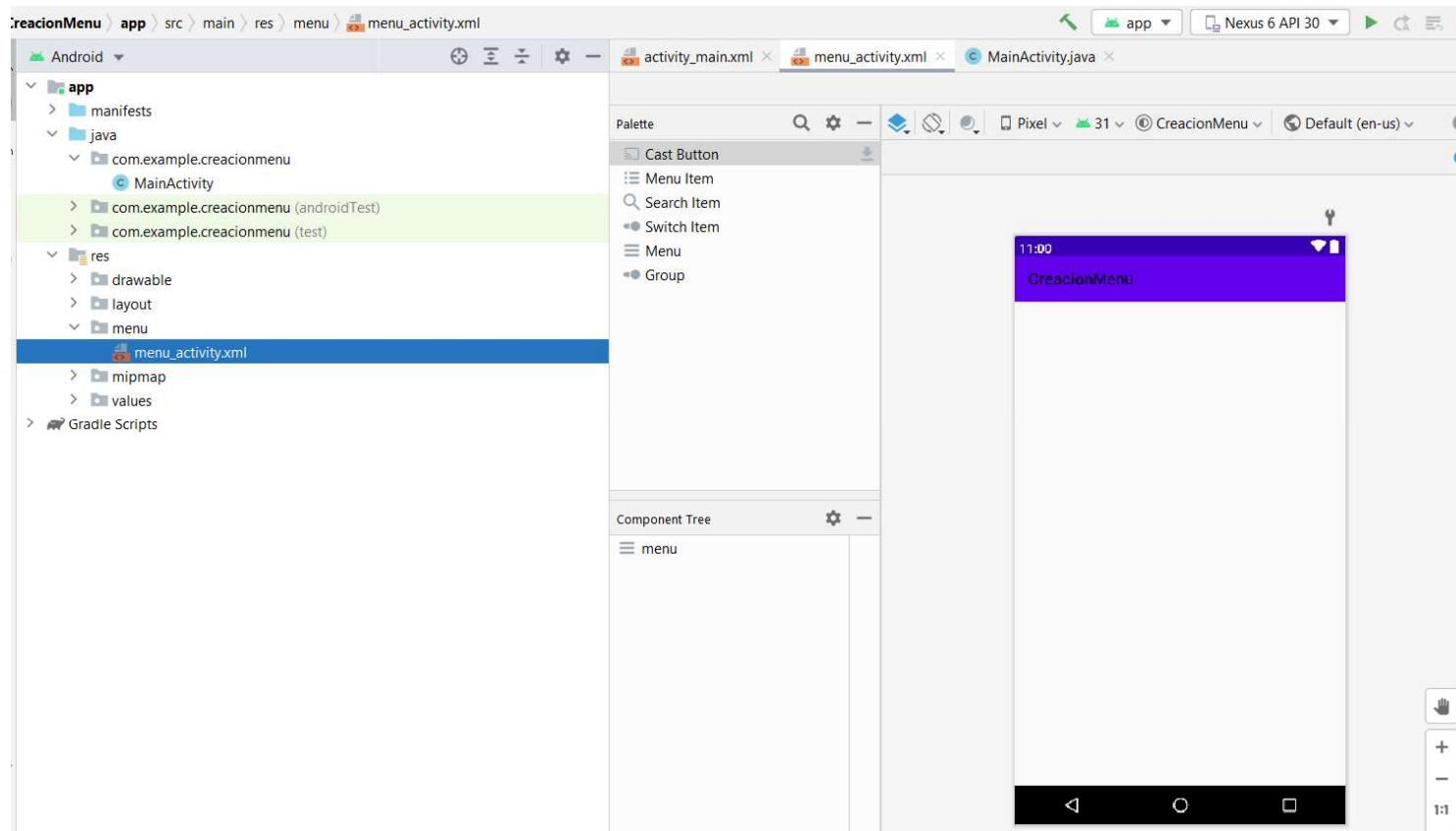
- ▶ Le daremos un nombre a nuestro menú en el campo File Name. En nuestro caso le llamaremos menu\_activity.
- ▶ Dentro de la opción Resource type, le tenemos que indicar que tipo de recurso vamos a crear. Tenemos las siguientes opciones:



- ▶ Nosotros elegiremos Menu.

# Menú V

- Se nos crea un fichero xml dentro de la carpeta res y en una subcarpeta llamada menu



## Menú VI

- ▶ Puede darse el caso de que automáticamente Android Studio te genere la subcarpeta menú y dentro el archivo xml en el que configuraremos el menú.
- ▶ Esto se produce cuando creamos un proyecto nuevo y en lugar de elegir un empty\_activity, elegimos por ejemplo un blank\_activity. En ese caso creará directamente la opción de menú.
- ▶ Si observáis la pantalla anterior, podréis ver que en la paleta aparecen diferentes opciones de las que aparecen habitualmente.
- ▶ En este caso aparecen elementos referidos al menú, ya que estamos en un xml específico para crearlo.
- ▶ Si accedemos a ver el fichero xml veremos lo siguiente:

```
1 |<?xml version="1.0" encoding="utf-8"?>
2 |<menu xmlns:android="http://schemas.android.com/apk/res/android">
3 |
4 |</menu>
```

## Menú VII

- ▶ Para indicar que el menú lo vamos a mostrar en la actividad principal, tenemos que incluir en el XML unos parámetros que necesita la etiqueta menú.

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

MainActivity (com.example.creacionmenu)

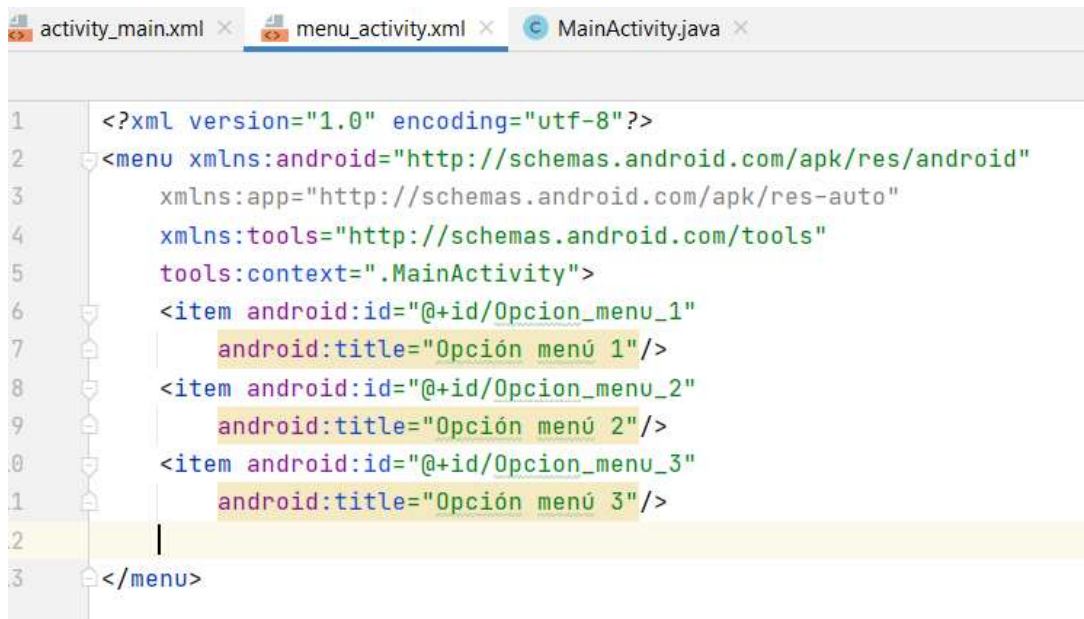
Ctrl+Abajo and Ctrl+Arriba will move caret down and up in the editor [Next Tip](#)

- ▶ En nuestro caso solo aparece la opción del MainActivity porque es la única actividad que tenemos. Si tuviéramos varias nos daría la opción de elegir.
- ▶ Con esto tenemos creado el menú, pero es un menú vacío.



# Menú VIII

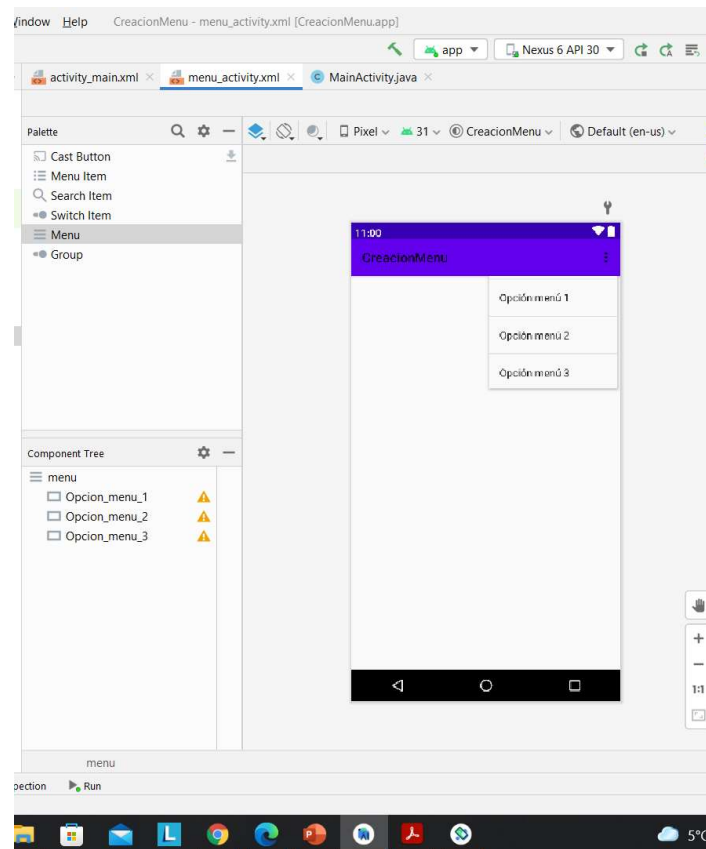
- ▶ El siguiente paso es confeccionar las opciones del menú.
- ▶ Para ello crearemos un elemento item por cada uno de los elementos que queremos que tenga el menú.



```
activity_main.xml x menu_activity.xml x MainActivity.java x
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     tools:context=".MainActivity">
6     <item android:id="@+id/Opcion_menu_1"
7         android:title="Opción menú 1"/>
8     <item android:id="@+id/Opcion_menu_2"
9         android:title="Opción menú 2"/>
10    <item android:id="@+id/Opcion_menu_3"
11        android:title="Opción menú 3"/>
12
13 </menu>
```

## Menú VIII

- Podemos añadir también todos los elementos a través del entorno gráfico. En la siguiente imagen podemos ver este entorno.



## Menú IX

- ▶ El siguiente paso es activar el menú en el MainActivity.java.
- ▶ Para ello crearemos dos métodos:
  - ▶ onCreateOptionsMenu: para crear el menú propiamente dicho.
  - ▶ onOptionsItemSelected: para configurar las acciones.
- ▶ Estos métodos ya están creados dentro de la clase Activity los estamos heredando. Lo que vamos a hacer es sobrescribirlos.
- ▶ Para sobrescribirlo vamos a utilizar @Override y a continuación ponemos el primer método que vamos a crear.
- ▶ En este caso creamos onCreateOptionsMenu. Este método devuelve un valor booleano y como parámetro tenemos un menú.
- ▶ Dentro de este método debemos incluir otro método que va a activar el menú.
- ▶ Ese método es el getMenuInflater().inflate(); Tendremos que poner el id del menú y el nombre del menú.

# Menú X

```
activity_main.xml x menu_activity.xml x MainActivity.java x
package com.example.creacionmenu;

import ...

public class MainActivity extends AppCompatActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu mimenu){
        getMenuInflater().inflate(R.menu.menu_activity, mimenu);
        return true;
    }
}
```

## Menú XI

- ▶ El siguiente paso es indicar cual es la acción que realiza cada una de las opciones del menú.
- ▶ Para ello utilizaremos el método `onOptionsItemSelected`. Dentro de este método tendremos que capturar el valor del id del elemento del menú que seleccione el usuario.
- ▶ Una vez pulsada la opción del menú que se desee debemos de indicar en el programa que es lo que se debe hacer.
- ▶ El programa quedará de la siguiente manera:

## Menú XII

```
@Override
public boolean onOptionsItemSelected(MenuItem opcion_menu){
    int id=opcion_menu.getItemId();
    if (id==R.id.Opcion_menu_1){
        Toast.makeText( context: this, text: "Opción menu 1", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id==R.id.Opcion_menu_2){
        Toast.makeText( context: this, text: "Opción menu 2", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id==R.id.Opcion_menu_3){
        Toast.makeText( context: this, text: "Opción menu 3", Toast.LENGTH_LONG).show();
        return true;
    }
    return super.onOptionsItemSelected(opcion_menu);
}
```

# Ejemplo menú

AHORA OS TOCA TRABAJAR A VOSOTROS.

DEBEIS CREAR UN MENU CONTEXTUAL EN EL QUE APAREZCAN 3 COMUNIDADES AUTÓNOMAS. P.E. CATALUÑA, MADRID Y ARAGÓN

DENTRO DE CADA UNA DE ESTAS COMUNIDADES AUTÓNOMAS DEBEN APARECER LAS PROVINCIAS QUE LAS CONFORMAN. SI ES UNA COMUNIDAD AUTÓNOMA CON UNA SOLA PROVINCIA NO SE PONDRÁN OPCIONES DE SUBMENÚ.

CUANDO SE PULSE CADA UNA DE LAS PROVINCIAS, SE MOSTRARÁ UN MENSAJE EN EL QUE APAREZCAN 3 POBLACIONES DE LA PROVINCIA SELECCIONADA.

CUANDO HAYAS FINALIZADO AVISA A LA PROFESORA PARA CORREGIRLO.

## Menu XIII

- ▶ Dentro de los menús, podemos incluir los listados de elementos. Hay diferentes tipos de listados como por ejemplo:
  - ▶ ListView
  - ▶ GridView
  - ▶ RecyclerView
- ▶ Vamos a ver mediante ejemplos como funcionan cada uno de ellos.



# ListView I

- ▶ Vamos a crear un proyecto nuevo al que llamaremos ListViewSimple.
- ▶ De momento escogeremos una plantilla empty, aunque podríamos escoger una plantilla basic que contiene botón flotante y nos podría ayudar.
- ▶ Lo primero que vamos a hacer es crear un elemento ListView. Como siempre lo podemos hacer a nivel de código o en el entorno gráfico.
- ▶ Para hacerlo en el archivo XML ponemos la siguiente etiqueta con los siguientes atributos:

```
<ListView  
    android:id="@+id/Listviewejemplo"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
</ListView>
```

## ListView II

- ▶ Si lo queremos hacer de manera gráfica, accederemos a la legacy de la paleta y elegiremos la opción ListView.
- ▶ En cualquiera de los dos casos se nos genera una caja dentro del `activity_main` que tendremos que ocupa toda la pantalla y que tendremos que referenciar a los márgenes, como cualquier elemento que incluimos en el layout.
- ▶ Nuestra lista está vacía y tendremos que rellenarla con el listado de datos que necesitamos.
- ▶ Para ello tendremos que programar en el `MainActivity` para mostrar el listado. Definimos una variable de tipo `ListView` y la referenciamos a la lista generada en el `activity_main`.

## ListView III

- ▶ Lo siguiente será generar el listado que queremos que aparezca en la ListView.
- ▶ Vamos a definir una variable de lista de strings que vamos a llamar ejemplosList.
- ▶ Para poder rellenar esa lista de ejemplos tendremos que crear un elemento ArrayList que nos dará la opción de ir poniendo todos los elementos que necesitemos en nuestra lista.
- ▶ El MainActivity quedará de la siguiente manera:

# ListView IV

```
package com.example.listviewsimple;

import ...

public class MainActivity extends AppCompatActivity {

    private ListView lista;
    private List<String> ejemplolist;

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

        lista=(ListView) findViewById(R.id.Listviewejemplo);
        ejemplolist = new ArrayList<>();
        ejemplolist.add("Ejemplo 1");
        ejemplolist.add("Ejemplo 2");
        ejemplolist.add("Ejemplo 3");
        ejemplolist.add("Ejemplo 4");
    }
}
```

# ListView V

- ▶ Si probamos el código realizado hasta ahora en la aplicación, no aparecerá nada.
- ▶ Para que la lista aparezca necesitamos conectar el listview con los elementos string creados. Para realizar esta conexión necesitamos un componente llamado **ADPATER**.
- ▶ No hay una forma natural de unir la lista de elementos con la listview y por eso necesitamos los adaptadores.
- ▶ Los adaptadores toman la lista de elementos, toman la listview y toman una plantilla, es decir, layout que actúa como diseño de la lista de elementos. Nos permitirá indicar de que manera queremos mostrar cada elemento.
- ▶ En Android existe por defecto un adaptador (ArrayAdapter) y un layout, que es el que vamos a utilizar en este caso.

## ListView VI

- ▶ Vamos a crear un adaptador que llamaremos adaptadorEjemplo.
- ▶ Dentro del adaptador tendremos que indicar los parámetros que necesita:
  - ▶ El primero es el contexto, que en este caso será this.
  - ▶ El siguiente parámetro será el layout que voy a utilizar para dibujar cada elemento de la lista. Para ello utilizaremos un layout de la librería Android. El layout que elegiremos únicamente tiene un textView, por lo que solo podremos poner un texto en cada uno de los elementos.
  - ▶ Por último indicaremos el listado de elementos que queremos que nos muestre.
- ▶ El código quedará de la siguiente manera:

```

public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener {

    private ListView lista;
    private List<String> ejemploslist;

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

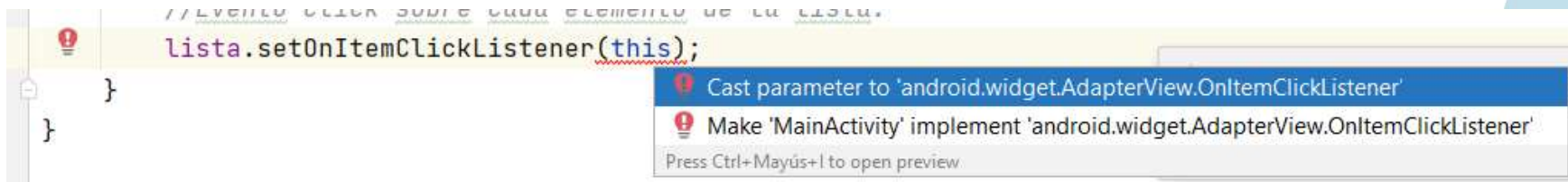
        lista=(ListView) findViewById(R.id.Listviewejemplo);
        ejemploslist = new ArrayList<>();
        ejemploslist.add(""); // pongo este elemento de la lista porque si no el nombre del activity t
        //apa el primer elemento de la lista.
        ejemploslist.add("Ejemplo 1");
        ejemploslist.add("Ejemplo 2");
        ejemploslist.add("Ejemplo 3");
        ejemploslist.add("Ejemplo 4");
        ArrayAdapter<String> adaptadorEjemplo = new ArrayAdapter<String>(
            context: this,
            android.R.layout.simple_list_item_1,
            ejemploslist
        );
        lista.setAdapter(adaptadorEjemplo);
        //Evento click sobre cada elemento de la lista.
        lista.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(context: this, text: "El ejemplo seleccionado es: "+ejemploslist.get(position), Toast.LENGTH_SHORT).show();
    }
}

```

# ListView VII

- ▶ Lo siguiente que tendremos que realizar es gestionar los eventos click de cada una de las opciones.
- ▶ Le vamos a indicar que la clase que va a implementar la interfaz es el propio MainActivity por lo que pondremos this. Nos va a devolver un error, porque el propio programa detecta que no puede resolverlo.
- ▶ Elegiremos la opción de hacer que el MainActivity implemente la interfaz. Para ello pulsaremos las teclas Alt+enter encima de this y elegiremos la segunda opción que nos aparece.





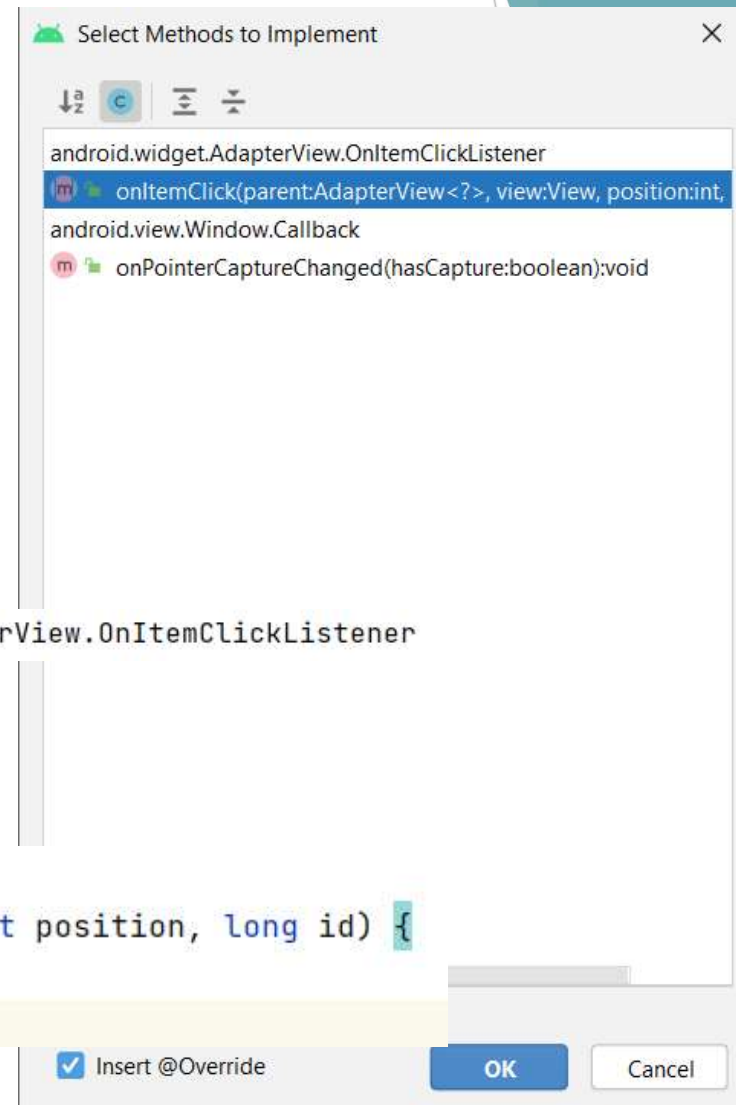
# ListView VIII

- ▶ Al pulsar la opción aparece la siguiente pantalla.
- ▶ Nos obliga a sobrecribir el método onItemClick.
- ▶ Y esto conlleva que se modifique el MainActivity en su cabecera para que pueda implementar el método onItemClickListener

```
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener
```

- ▶ Y nos aparece el método

```
@Override  
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
```



# ListView IX

- ▶ Este método va a ser lanzado cada vez que pulsemos uno de los elementos de la lista.
- ▶ Para que pueda identificar cual elemento hemos pulsado, al parámetro `int i` del método le vamos a cambiar el nombre y vamos a llamarlo posición.
- ▶ Como hemos creado un array, el primer elemento de la lista tiene la posición 0, el siguiente la posición 1 y así consecutivamente. Esas mismas posiciones están en nuestra lista.
- ▶ Como tenemos definida la lista a nivel de clase, vamos a poder acceder a ella desde el método sin problema.
- ▶ Para ver que accedemos a cada uno de los elementos vamos a poner un toast para visualizarlo.

```
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener {

    private ListView lista;
    private List<String> ejemploslist;

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

        lista=(ListView) findViewById(R.id.Listviewejemplo);
        ejemploslist = new ArrayList<>();
        ejemploslist.add(""); // pongo este elemento de la lista porque si no el nombre del activity t
        //apa el primer elemento de la lista.
        ejemploslist.add("Ejemplo 1");
        ejemploslist.add("Ejemplo 2");
        ejemploslist.add("Ejemplo 3");
        ejemploslist.add("Ejemplo 4");
        ArrayAdapter<String> adaptadorEjemplo = new ArrayAdapter<String>(
            context: this,
            android.R.layout.simple_list_item_1,
            ejemploslist
        );
        lista.setAdapter(adaptadorEjemplo);
        //Evento click sobre cada elemento de la lista.
        lista.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(context: this, text: "El ejemplo seleccionado es: "+ejemploslist.get(position), Toast.LENGTH_SHORT).show();
    }
}
```

# ListView personalizado

- ▶ Hasta ahora hemos realizado un ListView simple utilizando un layout existente, pero si queremos personalizar nuestro propio listado, tendremos que diseñar nuestro propio layout para nuestra ListView.
- ▶ Para ello vamos a generar un proyecto nuevo que llamaremos `listview_personalizado`.
- ▶ Vamos a crear un elemento `listview` en el archivo `activity_main.xml` como hemos hecho en el ejemplo anterior.
- ▶ También modificaremos el `MainActivity` definiendo una variable de tipo `listview`, como hicimos en el ejemplo anterior.
- ▶ En principio los archivos quedarán así:

# ListView personalizado

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listViewEjemplo"
        android:layout_width="398dp"
        android:layout_height="713dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

```
package com.example.listviewpersonalizado;

import ...

public class MainActivity extends AppCompatActivity {
    ListView lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lista=(ListView) findViewById(R.id.listViewEjemplo);
    }
}
```

# ListView personalizado

- ▶ Lo que hemos realizado en el ejercicio anterior, es generar un listado de elementos de tipo TextView, pero en este caso vamos a realizar un listado de elementos que tendrá una imagen o ruta url a un servidor donde esté la imagen, varios string y un número.
- ▶ En realidad podremos personalizarlo de la manera que necesitemos visualizar el listado de elementos.
- ▶ Para poder generar esta lista tendremos que definir una lista del tipo ejemplo que llamaremos ejemplo list. Esto lo añadiremos en nuestro ActivityMain.

```
list<Ejemplo> ejemplolist;
```

- ▶ Veremos que Ejemplo aparece como error. Eso es debido a que Ejemplo debe ser una clase que aún no existe, por lo que deberemos crearla.

# ListView personalizado

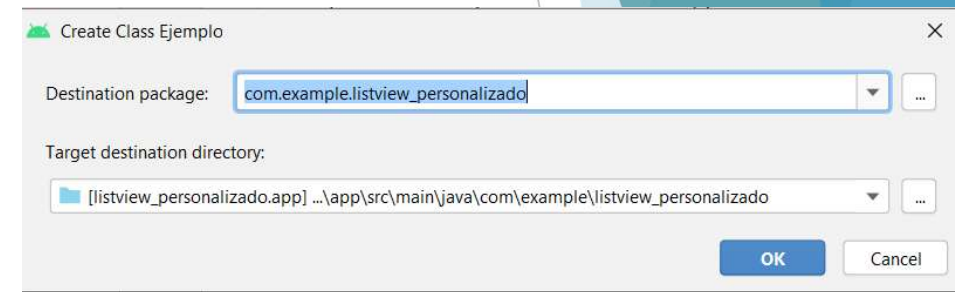
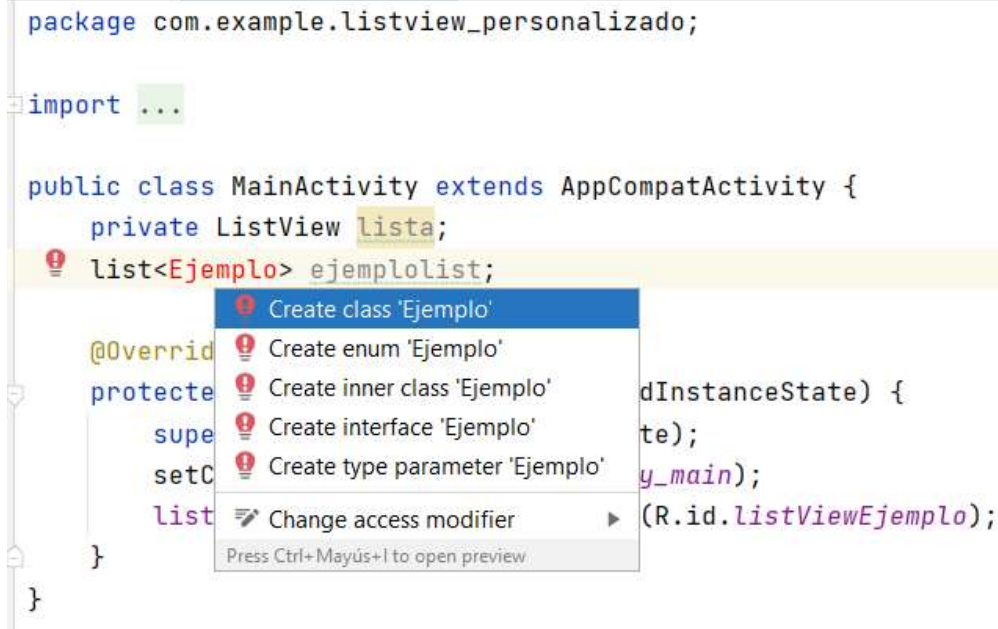
Crearemos la clase Ejemplo

```
package com.example.listview_personalizado;

import ...

public class MainActivity extends AppCompatActivity {
    private ListView lista;
    list<Ejemplo> ejemplolist;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        list = findViewById(R.id.listViewEjemplo);
    }
}
```



Creamos una clase java llamada ejemplo en la que definiremos las características que va a tener nuestro elemento de la vista.



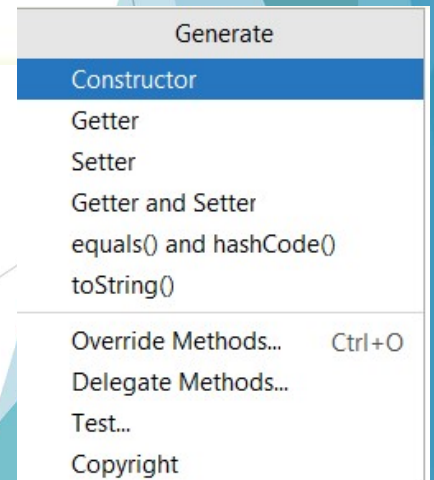
# ListView personalizado

- Un ejemplo podría ser:

```
package com.example.listview_personalizado;

public class Ejemplo {
    private String titulo;
    private String subtítulo;
    private String urlFoto;
    private int numeroEjemplo;
}
```

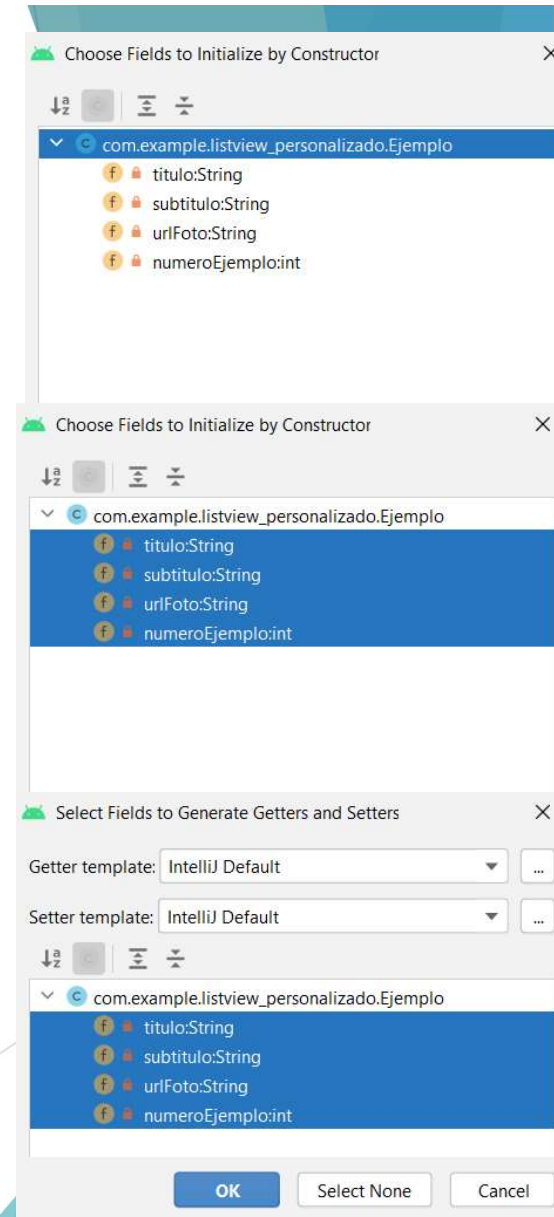
- Ahora vamos a generar el método constructor. Es una opción que nos facilita Android Studio. Para ello accederemos al menú code y dentro de este generate y elegimos la opción constructor





# ListView personalizado

- ▶ Vamos a dejar un constructor en blanco y para ello elegimos la primera opción.
- ▶ También vamos a generar otro constructor en el que recibimos todos los parámetros que tenemos como atributos en nuestra clase. Para ello volvemos a code\generate y seleccionamos todos los atributos y damos a OK.
- ▶ Por último nos queda generar todos los métodos para cambiar u obtener los valores de estas propiedades. Para ello volvemos a code\generate y esta vez elegimos getter and setter. Elegimos de nuevo todos los atributos y le damos a ok.
- ▶ El resultado en el fichero Ejemplo.java será el siguiente:



```
package com.example.listviewpersonalizado;
```

```
public class Ejemplo {  
    private String titulo;  
    private String subtítulo;  
    private String urlFoto;  
    private int numeroEjemplo;
```

```
    public Ejemplo() {  
    }
```

```
    public Ejemplo(String titulo, String subtítulo, String urlFoto, int numeroEjemplo) {  
        this.titulo = titulo;  
        this.subtítulo = subtítulo;  
        this.urlFoto = urlFoto;  
        this.numeroEjemplo = numeroEjemplo;  
    }
```

```
    public String getTitulo() {  
        return titulo;  
    }
```

```
    public void setTitulo(String titulo) {  
        this.titulo = titulo;  
    }
```

```
    public String getSubtítulo() {  
        return subtítulo;  
    }
```

```
    public void setSubtítulo(String subtítulo) {  
        this.subtítulo = subtítulo;  
    }
```

```
    public String getUrlFoto() {  
        return urlFoto;  
    }
```

```
    public void setUrlFoto(String urlFoto) {  
        this.urlFoto = urlFoto;  
    }
```

```
    public int getNumeroEjemplo() {  
        return numeroEjemplo;  
    }
```

```
    public void setNumeroEjemplo(int numeroEjemplo) {  
        this.numeroEjemplo = numeroEjemplo;  
    }
```

# Listview personalizado

- ▶ Volvemos al MainActivity e importamos la clase list para poder diseñar nuestro listado de elementos.
- ▶ Lo siguiente será generar nuestra lista de elementos y lo que haremos será crear un nuevo arraylist y después ir rellenando los elementos. Las líneas de código que debemos incluir son:

```
ejemploList = new ArrayList<>();
```

```
ejemploList.add(new Ejemplo("Título Ejemplo 1","Subtitulo Ejemplo 1","",2));
```

- ▶ Podemos incluir varios add para que nos salgan varios elementos y ver como funciona.
- ▶ El siguiente paso es incluir el adaptador. En este caso vamos a tener que generar un adaptador propio que tampoco existe, por lo que para generarlo tenemos que crear una clase que defina el adaptador.

```
MiAdapatorEjemplo adapatorEjemplo = new MiAdaptadorEjemplo;
```

# ListView personalizado

- ▶ Creamos la clase adaptador como lo hemos hecho anteriormente, pero en este caso la clase va a extender de la clase ArrayAdapter.

```
public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {  
}
```

- ▶ Nos aparece como error porque necesitamos que exista un constructor. El constructor lo crearemos como lo hemos hecho en la clase anterior (code\generate\constructor)
- ▶ Nos van a aparecer diferentes opciones pero nosotros elegiremos el constructor que tiene por parámetros el contexto, un layout para dibujar cada elemento de la lista y la lista de elementos propiamente dicha. El constructor será este:

  ArrayAdapter(context:Context, resource:int, objects:List<T>)

# ListView personalizado

- La clase programada quedará de la siguiente manera:

```
package com.example.listview_personalizado;

import android.content.Context;
import android.widget.ArrayAdapter;

import androidx.annotation.NonNull;

import java.util.List;

public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {
    public MiAdaptadorEjemplo(@NonNull Context context, int resource, @NonNull List<Ejemplo> objects) {
        super(context, resource, objects);
    }
}
```

activity\_main.xml x MainActivity.java x MiAdaptadorEjemplo.java x Ejemplo.java x

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    ListView lista;
    List<Ejemplo> ejemploList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lista = (ListView) findViewById(R.id.listViewEjemplo);
        ejemploList = new ArrayList<>();
        ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 1", subtitulo: "Subtitulo Ejemplo 1", urlFoto: "", numeroEjemplo: 1));
        ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 2", subtitulo: "Subtitulo Ejemplo 2", urlFoto: "", numeroEjemplo: 2));
        ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 3", subtitulo: "Subtitulo Ejemplo 3", urlFoto: "", numeroEjemplo: 3));
        ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 4", subtitulo: "Subtitulo Ejemplo 4", urlFoto: "", numeroEjemplo: 4));
    }
}
```

# ListView personalizado

- ▶ Volvemos al MainActivity y en el momento en el que estamos creando el adaptador, le tendremos que pasar los parámetros. El código sería:

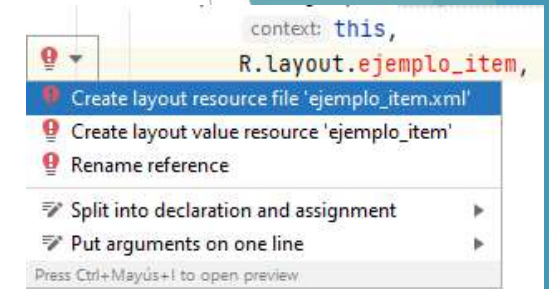
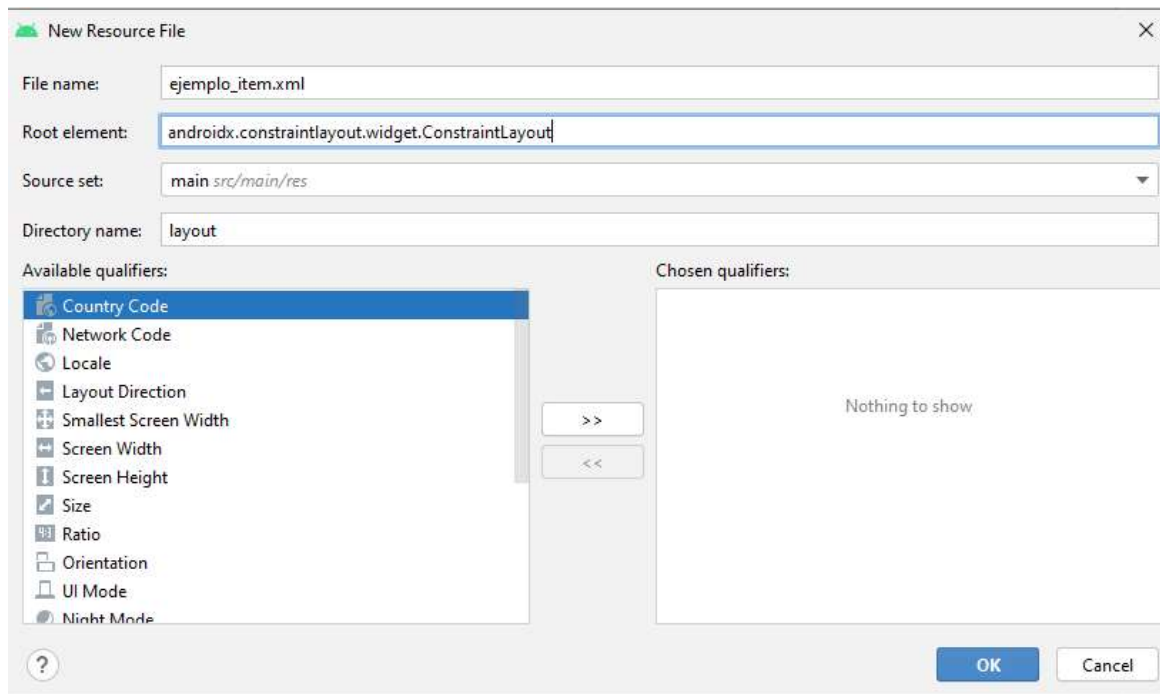
```
MiAdaptadorEjemplo adaptadorEjemplo = new MiAdaptadorEjemplo(  
    this,  
    R.layout.ejemplo_item,  
    ejemploList  
);
```

- ▶ Cuando pasamos el layout, nos va a dar un error porque no lo tenemos creado.
- ▶ Para solucionar el problema tendremos que crear el fichero de layout.



# ListView personalizado

- En este layout vamos a diseñar un elemento de la lista para mostrarlo.



- Le damos a OK y nos genera el layout de diseño del elemento de la lista.



# ListView personalizado

- ▶ Vamos a añadir entonces a este layout todos los elementos que necesitamos:
  - ▶ Imagen
  - ▶ 2 TextView
  - ▶ Número
- ▶ Para la imagen vamos a crear un nuevo Vector Asset y vamos a elegir una imagen de las que vienen por defecto en Android Studio.
- ▶ En el layout incluiremos un ImageView al que le asignaremos la imagen anterior. Su id será imageViewFoto.
- ▶ Después incluiremos un TextView. Su id será textViewTitulo. Lo uniremos por arriba y por el centro con la imagen. El texto estará en negro y negrita.
- ▶ Añadimos otro TextView. Su id será textViewSubtitulo y lo uniremos por arriba con la parte de abajo del otro TextView y también alineado por la izquierda.
- ▶ Por último nos falta otro TextView (en el que introduciremos el número). Estará unido por arriba con el TextView anterior y también alineado por la izquierda.

# ListView personalizado

- ▶ El resultado es:



- ▶ Lo siguiente que tenemos que realizar es el diseño de nuestro adaptador.
- ▶ Para ello, dentro de la clase MiAdaptadorEjemplo, vamos a definir una serie de variables que vamos a necesitar.
  - ▶ Context ctx;
  - ▶ int layoutTemplate;
  - ▶ List<Ejemplo> ejemploList;
- ▶ Son los tres parámetros que necesita nuestro método MiAdaptadorEjemplo.

# ListView personalizado

- ▶ Lo siguiente que vamos a hacer es ir guardando la información a medida que vamos recibiendo los parámetros en el adaptador. Utilizaremos el siguiente código:

```
this.ctx = context;  
this.layoutTemplate = resource;  
this.ejemploList = objects;
```

- ▶ Ahora vamos a sobrescribir el método getView (se genera de manera automática). Se lanza automáticamente como si fuera un bucle, una vez por cada elemento que recibimos en el adaptador. Es decir que como recibimos tres elementos se ejecutará tres veces.
- ▶ Cada vez que lo invocamos se va a recibir por parámetro la posición de la lista de elementos que está recorriendo en ese momento. Se repetirá hasta el último elemento del array que nos estén pasando por parámetro.

# ListView personalizado

- El código que tenemos hasta el momento en la clase MiAdaptadorEjemplo es:

```
package com.example.listview_personalizado;

import ...

public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {
    //Definición de variables.
    Context ctx;
    int layoutTemplate;
    List<Ejemplo> ejemploList;

    public MiAdaptadorEjemplo(@NonNull Context context, int resource, @NonNull List<Ejemplo> objects) {
        super(context, resource, objects);
        //A medida que recibo los parámetros, vamos a guardar las variables.
        this.ctx = context;
        this.layoutTemplate = resource;
        this.ejemploList = objects;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        return super.getView(position, convertView, parent);
    }
}
```

# ListView personalizado

- ▶ A continuación vamos a borrar la línea de código que tenemos dentro del `getView`, para hacer nuestra propia programación.
- ▶ El código que incluiremos nos va a permitir obtener el layout y cargarlo dentro del elemento padre que es el `listView` que hemos recibido a través del `activity`.
- ▶ De esta manera, a medida que vaya recorriendo los elementos de la lista, los irá incluyendo en el `listView`.
- ▶ La variable `v` tendrá en este momento el layout en el que hemos definido el elemento de la lista.
- ▶ Al final de las operaciones debemos devolverlo para que se incluya en el `listView` como un elemento hijo.

```
package com.example.listview_personalizado;

import ...

public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {
    //Definición de variables.
    Context ctx;
    int layoutTemplate;
    List<Ejemplo> ejemploList;

    public MiAdaptadorEjemplo(@NonNull Context context, int resource, @NonNull List<Ejemplo> objects) {
        super(context, resource, objects);
        //A medida que recibo los parámetros, vamos a guardar las variables.
        this.ctx = context;
        this.layoutTemplate = resource;
        this.ejemploList = objects;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        View v = LayoutInflater.from(ctx).inflate(layoutTemplate, parent, attachToRoot: false);

        return v;
    }
}
```

# ListView personalizado

- ▶ Nos quedan varias operaciones que realizar antes de devolver la variable `v`.
- ▶ Lo primero es obtener la información del elemento de la lista que estamos recorriendo en este momento. Para ello pondremos el siguiente código:
  - ▶ `Ejemplo elementoActual = ejemploList.get(position);`
- ▶ A continuación vamos a rescatar los elementos de la interfaz de usuario de la template (plantilla). La plantilla está cargada en la variable `v`, por lo que el código que necesitaremos será:
  - ▶ `TextView textViewTitulo = (TextView) v.findViewById(R.id.textViewTitulo);`
  - ▶ `TextView textViewSubtitulo = (TextView) v.findViewById(R.id.textViewSubtitulo);`
  - ▶ `TextView textViewNumero = (TextView) v.findViewById(R.id.textViewNumero);`
- ▶ De esta manera ya tenemos los elementos de la vista.

## ListView personalizado

- ▶ Lo siguiente es hacer un set de la información del elemento Actual en los elementos de la interface de usuario. Para ello haremos las siguientes líneas de programación:
  - ▶ `textViewTitulo.setText(elementoActual.getTitulo());`
  - ▶ `textViewSubtitulo.setText(elementoActual.getSubtitulo());`
  - ▶ `textViewNumero.setText(elementoActual.getNumeroEjemplo());`
- ▶ El resultado final de la clase programada será el siguiente:

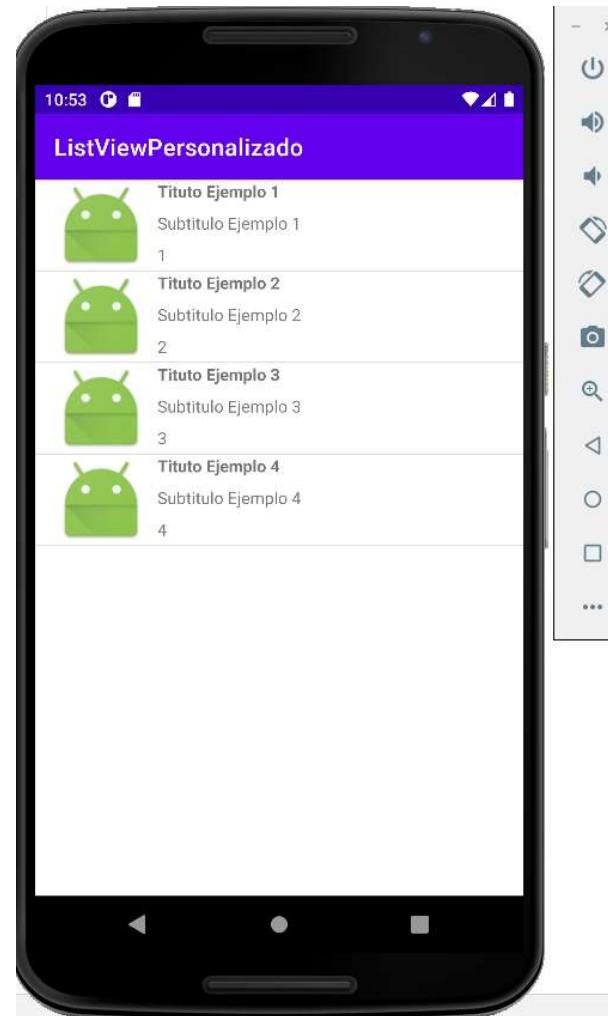


```
public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {
    //Definición de variables
    Context ctx;
    int layoutTemplate;
    List<Ejemplo> ejemploList;

    public MiAdaptadorEjemplo(@NonNull Context context, int resource, @NonNull List<Ejemplo> objects) {
        super(context, resource, objects);
        // A medida que recibo los parámetros, vamos a guardar las variables.
        this.ctx = context;
        this.layoutTemplate = resource;
        this.ejemploList = objects;
    }
    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        View v = LayoutInflater.from(ctx).inflate(layoutTemplate, parent, attachToRoot: false);
        //Obtener la información del elemento de la lista que estamos recorriendo en este momento.
        Ejemplo elementoActual = ejemploList.get(position);
        //Rescatar los elementos de la interfaz de usuario de la template (plantilla)
        TextView textViewTitulo = (TextView) v.findViewById(R.id.textviewTitulo);
        TextView textViewSubtitulo = (TextView) v.findViewById(R.id.textviewSubtitulo);
        TextView textViewNumero = (TextView) v.findViewById(R.id.textviewNumero);
        //Hacer un set de la información del elementoActual en los elementos de la interfaz del usuario.
        textViewTitulo.setText(elementoActual.getTitulo());
        textViewSubtitulo.setText(elementoActual.getSubtitulo());
        textViewNumero.setText(elementoActual.getNumeroEjemplo()+"");

        return v;
    }
}
```

# ListView personalizado



# activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listViewEjemplo"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

# MainActivity.java

```
activity_main.xml x MainActivity.java x ejemplo_item.xml x MiAdaptadorEjemplo.java x Ejemplo.java x
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    ListView lista;
    List<Ejemplo> ejemploList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lista = (ListView) findViewById(R.id.listViewEjemplo);
        ejemploList = new ArrayList<>();
        ejemploList.add(new Ejemplo( titulo: "Tituto Ejemplo 1", subtitulo: "Subtitulo Ejemplo 1", urlFoto: "", numeroEjemplo: 1));
        ejemploList.add(new Ejemplo( titulo: "Tituto Ejemplo 2", subtitulo: "Subtitulo Ejemplo 2", urlFoto: "", numeroEjemplo: 2));
        ejemploList.add(new Ejemplo( titulo: "Tituto Ejemplo 3", subtitulo: "Subtitulo Ejemplo 3", urlFoto: "", numeroEjemplo: 3));
        ejemploList.add(new Ejemplo( titulo: "Tituto Ejemplo 4", subtitulo: "Subtitulo Ejemplo 4", urlFoto: "", numeroEjemplo: 4));
        MiAdaptadorEjemplo adaptadorEjemplo = new MiAdaptadorEjemplo(
            context: this,
            R.layout.ejemplo_item,
            ejemploList
        );
        lista.setAdapter(adaptadorEjemplo);
    }
}
```



# ejemplo\_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

<ImageView

```
    android:id="@+id/imageViewFoto"
    android:layout_width="79dp"
    android:layout_height="76dp"
    android:layout_marginStart="16dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.024"
    app:srcCompat="@android:mipmap/sym_def_app_icon" />
```

<TextView

```
    android:id="@+id/textViewTitulo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:text="Titulo ejemplo"
    android:textStyle="bold"
    app:layout_constraintStart_toEndOf="@+id/imageViewFoto"
    app:layout_constraintTop_toTopOf="@+id/imageViewFoto" />
```

<TextView

```
    android:id="@+id/textViewSubtitulo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Subtitulo Ejemplo"
    app:layout_constraintStart_toStartOf="@+id/textViewTitulo"
    app:layout_constraintTop_toBottomOf="@+id/textViewTitulo" />
```

<TextView

```
    android:id="@+id/textViewNumero"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Numero Ejemplo"
    app:layout_constraintStart_toStartOf="@+id/textViewSubtitulo"
    app:layout_constraintTop_toBottomOf="@+id/textViewSubtitulo" />
```

</androidx.constraintlayout.widget.ConstraintLayout>

# MiAdaptadorEjemplo.java

```
public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {  
    //Definición de variables  
    Context ctx;  
    int layoutTemplate;  
    List<Ejemplo> ejemploList;  
  
    public MiAdaptadorEjemplo(@NonNull Context context, int resource, @NonNull List<Ejemplo> objects) {  
        super(context, resource, objects);  
        // A medida que recibo los parámetros, vamos a guardar las variables.  
        this.ctx = context;  
        this.layoutTemplate = resource;  
        this.ejemploList = objects;  
    }  
    @NonNull  
    @Override  
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
        View v = LayoutInflater.from(ctx).inflate(layoutTemplate, parent, attachToRoot: false);  
        //Obtener la información del elemento de la lista que estamos recorriendo en este momento.  
        Ejemplo elementoActual = ejemploList.get(position);  
        //Rescatar los elementos de la interfaz de usuario de la template (plantilla)  
        TextView textViewTitulo = (TextView) v.findViewById(R.id.textViewTitulo);  
        TextView textViewSubtitulo = (TextView) v.findViewById(R.id.textViewSubtitulo);  
        TextView textViewNumero = (TextView) v.findViewById(R.id.textViewNumero);  
        //Hacer un set de la información del elementoActual en los elementos de la interfaz del usuario.  
        textViewTitulo.setText(elementoActual.getTitulo());  
        textViewSubtitulo.setText(elementoActual.getSubtitulo());  
        textViewNumero.setText(elementoActual.getNumeroEjemplo()+"");  
  
        return v;  
    }  
}
```

# Ejemplo.java

```
package com.example.listviewpersonalizado;

public class Ejemplo {
    private String titulo;
    private String subtitulo;
    private String urlFoto;
    private Integer numeroEjemplo;

    public Ejemplo() {
    }

    public Ejemplo(String titulo, String subtitulo, String urlFoto, Integer numeroEjemplo) {
        this.titulo = titulo;
        this.subtitulo = subtitulo;
        this.urlFoto = urlFoto;
        this.numeroEjemplo = numeroEjemplo;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getSubtitulo() {
        return subtitulo;
    }

    public void setSubtitulo(String subtitulo) {
        this.subtitulo = subtitulo;
    }
}
```

```
public String getUrlFoto() {
    return urlFoto;
}

public void setUrlFoto(String urlFoto) {
    this.urlFoto = urlFoto;
}

public Integer getNumeroEjemplo() {
    return numeroEjemplo;
}

public void setNumeroEjemplo(Integer numeroEjemplo) {
    this.numeroEjemplo = numeroEjemplo;
}
}
```

# ListView Personalizado

- ▶ Para finalizar, y tomando como ejemplo el listView simple que vimos anteriormente, vamos a mostrar un mensaje cada vez que pulsemos una de las opciones del ListView Personalizada.
- ▶ Mostraremos por ejemplo el título de cada una de las opciones.





# GridView personalizado

- ▶ Muestra los datos en forma de rejilla (generalmente se utiliza para imágenes)
- ▶ Incluye los scroll cuando los datos ocupan más del tamaño de la pantalla.
- ▶ Para mostrar los datos utilizaremos un adaptador personalizado como hicimos en la listview del ejemplo anterior.
- ▶ Vamos a crear un proyecto nuevo al que llamaremos GridViewPersonalizado. Utilizaremos como siempre una plantilla Empty.
- ▶ Lo siguiente que haremos será crear un Gridview. Como ya sabemos podemos hacerlo en el archivo xml o en el entorno gráfico.
- ▶ Para hacerlo en el entorno gráfico iremos a la opción Legacy y escogeremos el elemento GridView.

# GridView personalizado

- ▶ Tenemos que referenciarlo a los márgenes, asignarle un id (gridViewEjemplo).
- ▶ En GridView hay una propiedad que solo existe en este elemento que son las columnas. En esta propiedad pondremos el número de columnas (numcolumns) que queremos que tenga nuestro Grid.
- ▶ Existe un valor que podemos poner en esta propiedad que es auto\_fit que de manera automática detectará cuantas columnas se pueden incluir en la pantalla del dispositivo.
- ▶ Una vez realizado el layout vamos al MainActivity.java y definiremos un variable de tipo GridView que vamos a llamar view.
- ▶ A esta variable la enlazaremos con el elemento GridView generado en el layout.

```
activity_main.xml x MainActivity.java x
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <GridView
10         android:id="@+id/gridViewEjemplo"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"
13         android:numColumns="auto_fit"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
activity_main.xml x MainActivity.java x
package com.example.gridviewpersonalizado;

import ...

public class MainActivity extends AppCompatActivity {
    GridView view;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        view = (GridView) findViewById(R.id.gridViewEjemplo);
    }
}
```

# GridView Personalizado

- ▶ A partir de este momento, código va a ser muy parecido al que hemos realizado en el ListView personalizado.
- ▶ Tendremos que generar nuestro adaptador personalizado y nuestro tipo de datos.
- ▶ Podemos copiarlos del ejercicio anterior. El adaptador se llamaba MiAdaptadorEjemplo.java y el tipo de datos se llama Ejemplo.java.
- ▶ Puede ser que al copiarlo nos de algún error al referenciar Ejemplo.java. Simplemente lo modificamos para que se quede igual que en el código de listview.
- ▶ También va a dar error porque aún no hemos generado el layout de como queremos que aparezca el grid.

```
activity_main.xml x MainActivity.java x Ejemplo.java x MiAdaptadorEjemplo.java x
1 package com.example.gridviewpersonalizado;
2
3 public class Ejemplo {
4     private String titulo;
5     private String subtítulo;
6     private String urlFoto;
7     private Integer numeroEjemplo;
8
9     public Ejemplo() {
10    }
11    public Ejemplo(String titulo, String subtítulo, String urlFoto, Integer numeroEjemplo) {
12        this.titulo = titulo;
13        this.subtítulo = subtítulo;
14        this.urlFoto = urlFoto;
15        this.numeroEjemplo = numeroEjemplo;
16    }
17
18    public String getTitulo() { return titulo; }
21
22    public void setTitulo(String titulo) { this.titulo = titulo; }
25
26    public String getSubtítulo() { return subtítulo; }
29
30    public void setSubtítulo(String subtítulo) { this.subtítulo = subtítulo; }
33
34    public String getUrlFoto() { return urlFoto; }
37
38    public void setUrlFoto(String urlFoto) { this.urlFoto = urlFoto; }
41
42    public Integer getNumeroEjemplo() { return numeroEjemplo; }
45
46    public void setNumeroEjemplo(Integer numeroEjemplo) { this.numeroEjemplo = numeroEjemplo; }
49 }
```





```
activity_main.xml x MainActivity.java x Ejemplo.java x MiAdaptadorEjemplo.java x
16 public class MiAdaptadorEjemplo extends ArrayAdapter<Ejemplo> {
17     //Definición de variables
18     Context ctx;
19     int layoutTemplate;
20     List<Ejemplo> ejemploList;
21
22     public MiAdaptadorEjemplo(@NonNull Context context, int resource, @NonNull List<Ejemplo> objects) {
23         super(context, resource, objects);
24         // A medida que recibo los parámetros, vamos a guardar las variables.
25         this.ctx = context;
26         this.layoutTemplate = resource;
27         this.ejemploList = objects;
28     }
29     @NonNull
30     @Override
31     public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
32         View v = LayoutInflater.from(ctx).inflate(layoutTemplate, parent, attachToRoot: false);
33         //Obtener la información del elemento de la lista que estamos recorriendo en este momento.
34         Ejemplo elementoActual = ejemploList.get(position);
35         //Rescatar los elementos de la interfaz de usuario de la template (plantilla)
36         TextView textViewTitulo = (TextView) v.findViewById(R.id.textViewTitulo);
37         TextView textViewSubtitulo = (TextView) v.findViewById(R.id.textViewSubtitulo);
38         TextView textViewNumero = (TextView) v.findViewById(R.id.textViewNumero);
39         ImageView imageViewFoto = (ImageView) v.findViewById(R.id.imageViewFoto);
40         //Hacer un set de la información del elementoActual en los elementos de la interfaz del usuario.
41         textViewTitulo.setText(elementoActual.getTitulo());
42         textViewSubtitulo.setText(elementoActual.getSubtitulo());
43         textViewNumero.setText(elementoActual.getNumeroEjemplo()+"");
44
45         return v;
46     }
47 }
```

# GridView Personalizado

- ▶ Lógicamente el Grid no se ve igual que una lista listview por lo que el layout no podrá ser igual.
- ▶ En listview los elementos ocupan todo el ancho de la pantalla del dispositivo.
- ▶ En el caso del grid ocupa un cuadro de la columna en que haya dividido la pantalla.
- ▶ De todas formas los pasos para crearlo son iguales. Tenemos que:
  - ▶ crear la lista,
  - ▶ incluir en el MainActivity.java la lista de ejemplo como lo pusimos en listview.
- ▶ Lo podemos copiar del ejercicio anterior y hacer unas leves modificaciones.
- ▶ Quedará así:

activity\_main.xml x MainActivity.java x Ejemplo.java x MiAdaptadorEjemplo.java x

```
package com.example.gridviewpersonalizado;
```

```
import ...
```

```
public class MainActivity extends AppCompatActivity {
```

```
    GridView view;
```

```
    List<Ejemplo> ejemploList;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    view = (GridView) findViewById(R.id.gridViewEjemplo);
```

```
    ejemploList = new ArrayList<>();
```

```
    ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 1", subtitulo: "Subtitulo Ejemplo 1", urlFoto: "", numeroEjemplo: 1));
```

```
    ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 2", subtitulo: "Subtitulo Ejemplo 2", urlFoto: "", numeroEjemplo: 2));
```

```
    ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 3", subtitulo: "Subtitulo Ejemplo 3", urlFoto: "", numeroEjemplo: 3));
```

```
    ejemploList.add(new Ejemplo( titulo: "Titulo Ejemplo 4", subtitulo: "Subtitulo Ejemplo 4", urlFoto: "", numeroEjemplo: 4));
```

```
    MiAdaptadorEjemplo adaptadorEjemplo = new MiAdaptadorEjemplo(
```

```
        context: this,
```

```
        R.layout.ejemplo_item,
```

```
        ejemploList
```

```
    );
```

```
    //lista.setAdapter(adaptadorEjemplo);
```

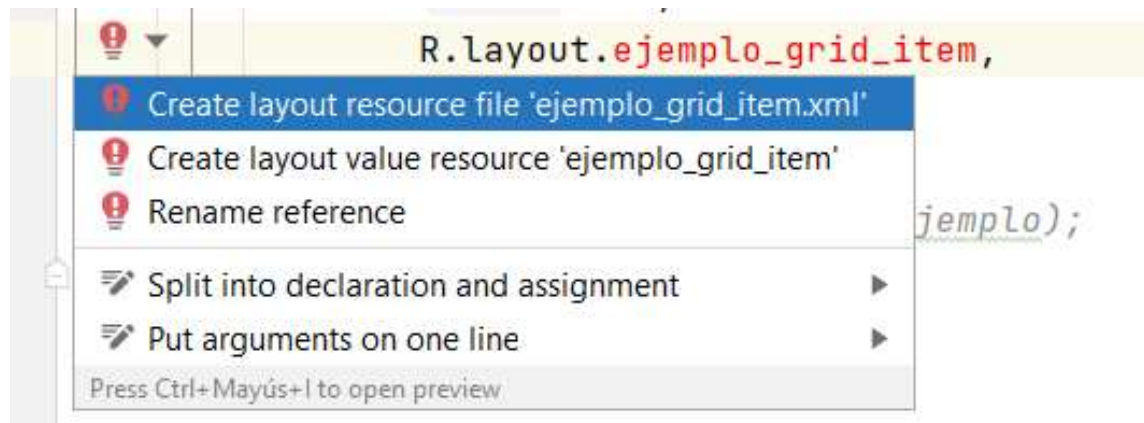
```
}
```

```
}
```



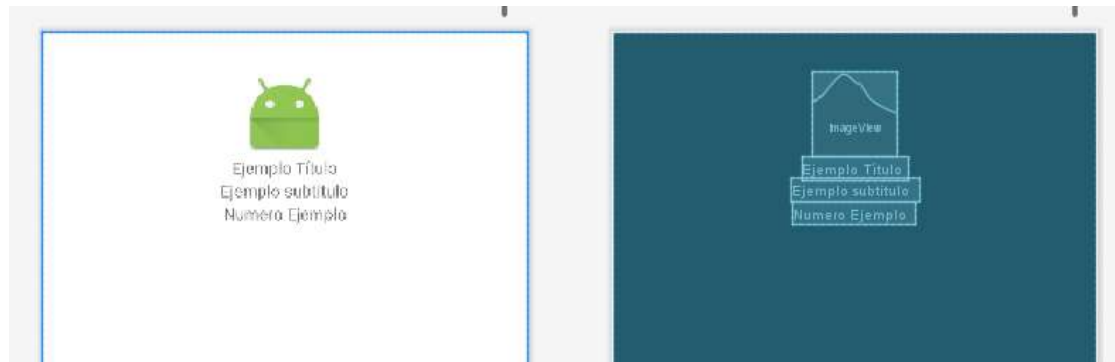
# GridView Personalizado

- ▶ Como veis he dejado comentada la última línea en la que llamamos al adaptador y tenemos un error donde llama al layout donde teníamos diseñado el aspecto de los elementos de listview.
- ▶ Vamos a cambiar el nombre a ese layout y lo vamos a llamar ejemplo\_grid\_item y lo vamos a crear desde cero.



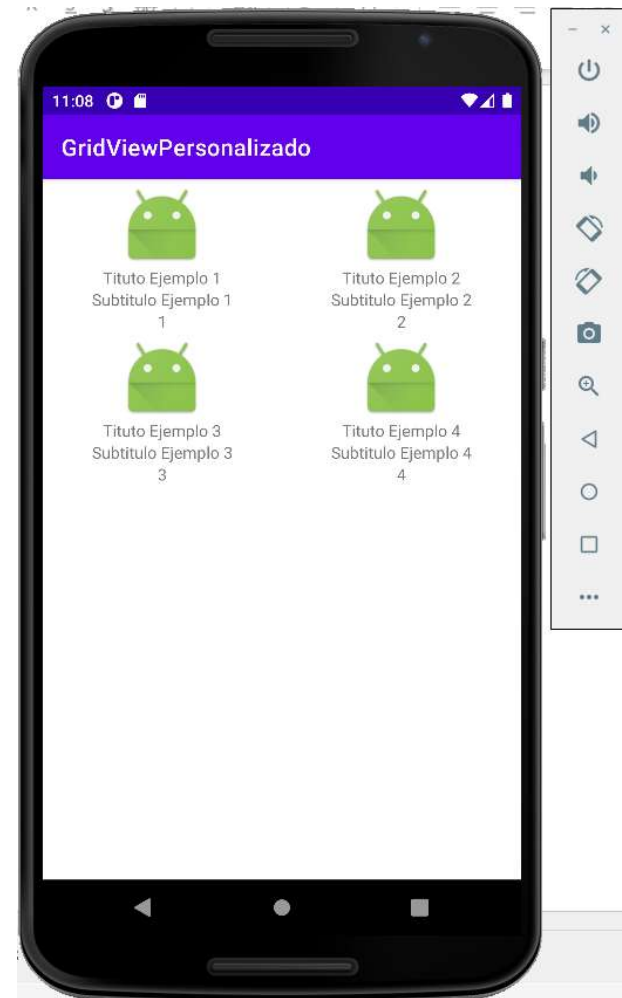
# GridView Personalizado

- ▶ En este archivo xml, vamos a generar nuestro elemento del grid.
- ▶ Vamos a incluir los mismos elementos que en la listview pero organizados de otra manera.
- ▶ Se quedaría de la siguiente manera:



- ▶ Es muy importante mantener los mismos id que en el ejemplo anterior porque son los que hemos copiado para reutilizar las líneas de código.

El resultado será el siguiente:



## GridView Personalizado

- ▶ Como ya hemos realizado en los ejercicios anteriores cada vez que pulsemos en una de las opciones saldrá un texto que nos indicará en que opción estamos.

# Ejemplo repaso

- ▶ Desde cualquiera de los menús/listas/grid que hemos creado en ejemplos anteriores, al pulsar cualquier opción del menú/lista/grid nos va a abrir un activity secundario con el texto de la opción que es.
- ▶ Desde cualquiera de los menús/listas/grid que hemos creado en ejemplos anteriores, al pulsar la primera opción del menú vamos a pasar por parámetro el número del ejemplo a un activity secundario y lo vamos a multiplicar por 3 y en la segunda opción del menú vamos a abrir el activity secundario creado en el apartado anterior.
- ▶