

# SEMINARIO DE LENGUAJES

## OPCIÓN ANDROID

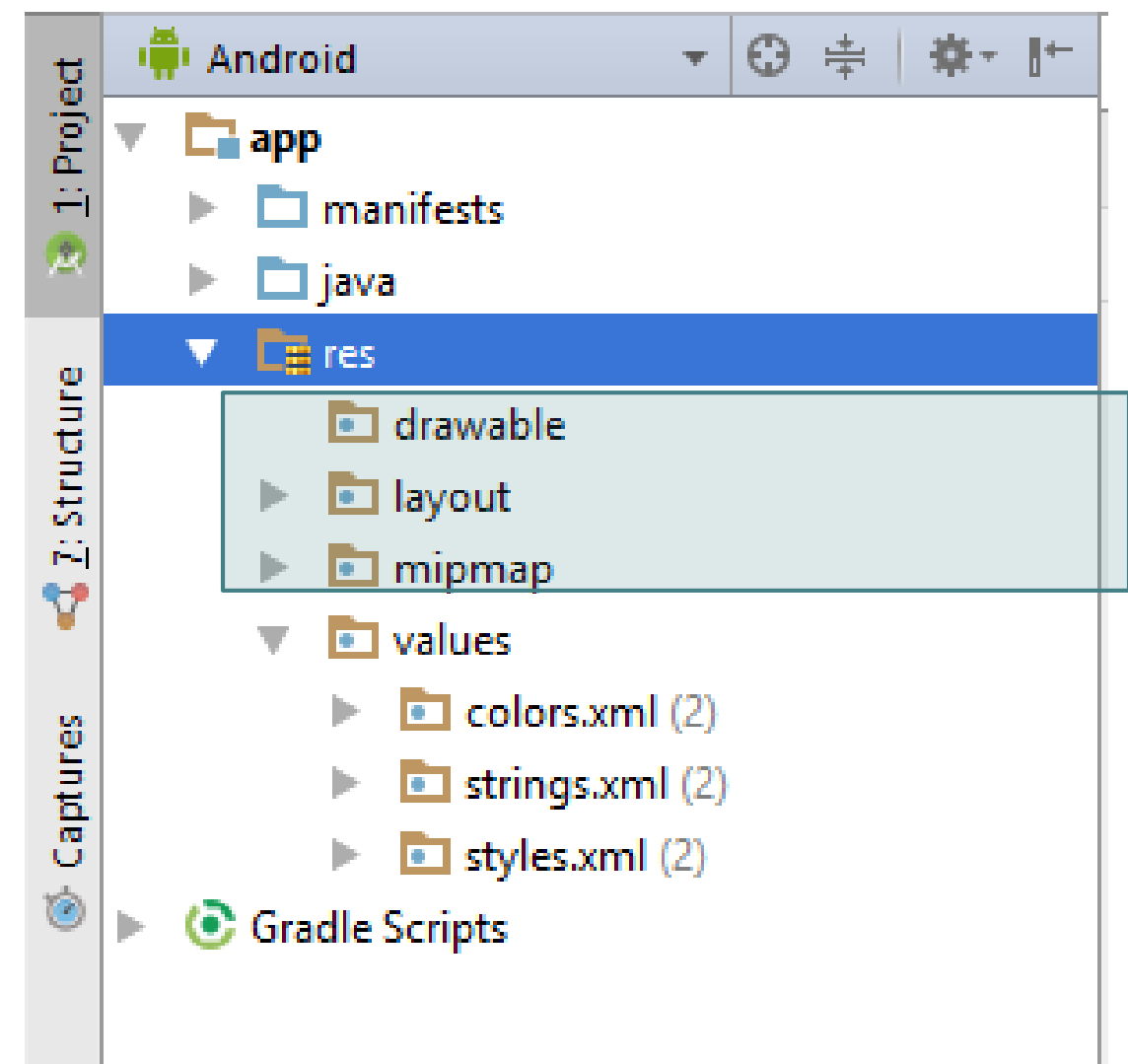


**Recursos de archivo. Menús.**

**Mg. Corbalán Leonardo, Esp. Delía Lisandro**

# Recursos de archivos

- En **Android Studio**, las carpetas contenidas en **res/** (a excepción de **res/value/** ) se utilizan para definir recursos de archivos.
- Para estos recursos se crea un **identificador** automático que coincide con el **nombre del archivo sin la extensión**.
- Según la carpeta donde se cree, se conocerá su tipo de recurso.



# Recursos de archivos

- Algunos de los tipos de recursos de archivos más utilizados son:
  - **drawable/** Archivos que definen recursos de imágenes: bitmaps (.png, .jpg o gif), XML con descriptores de gráficos (Ej. shape)
  - **layout/** Archivos XML que definen el diseño de una interfaz de usuario
  - **mipmap/** Archivos de elemento de diseño para diferentes densidades de los íconos lanzadores.
  - **menu/** Archivos XML que definen menús de aplicaciones, como un menú de opciones, un menú contextual o un submenú.
  - **raw/** Archivos arbitrarios para guardar sin procesar (Ej. audio o video)

# Actividad guiada

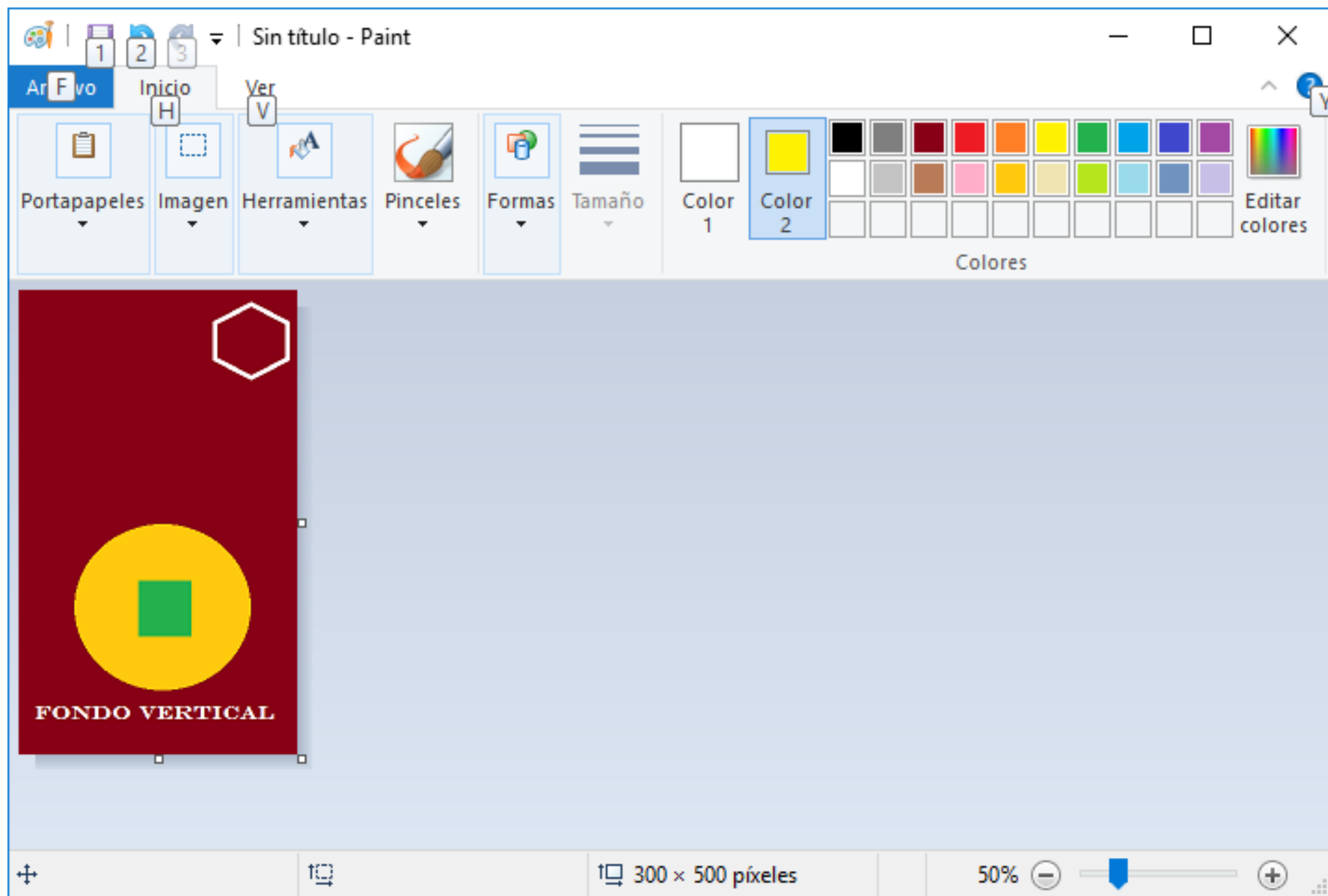
- Crear un nuevo proyecto **Android Studio** llamado **"RecursosDeArchivo"** basado en la siguiente **Empty Activity**

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

</LinearLayout>
```

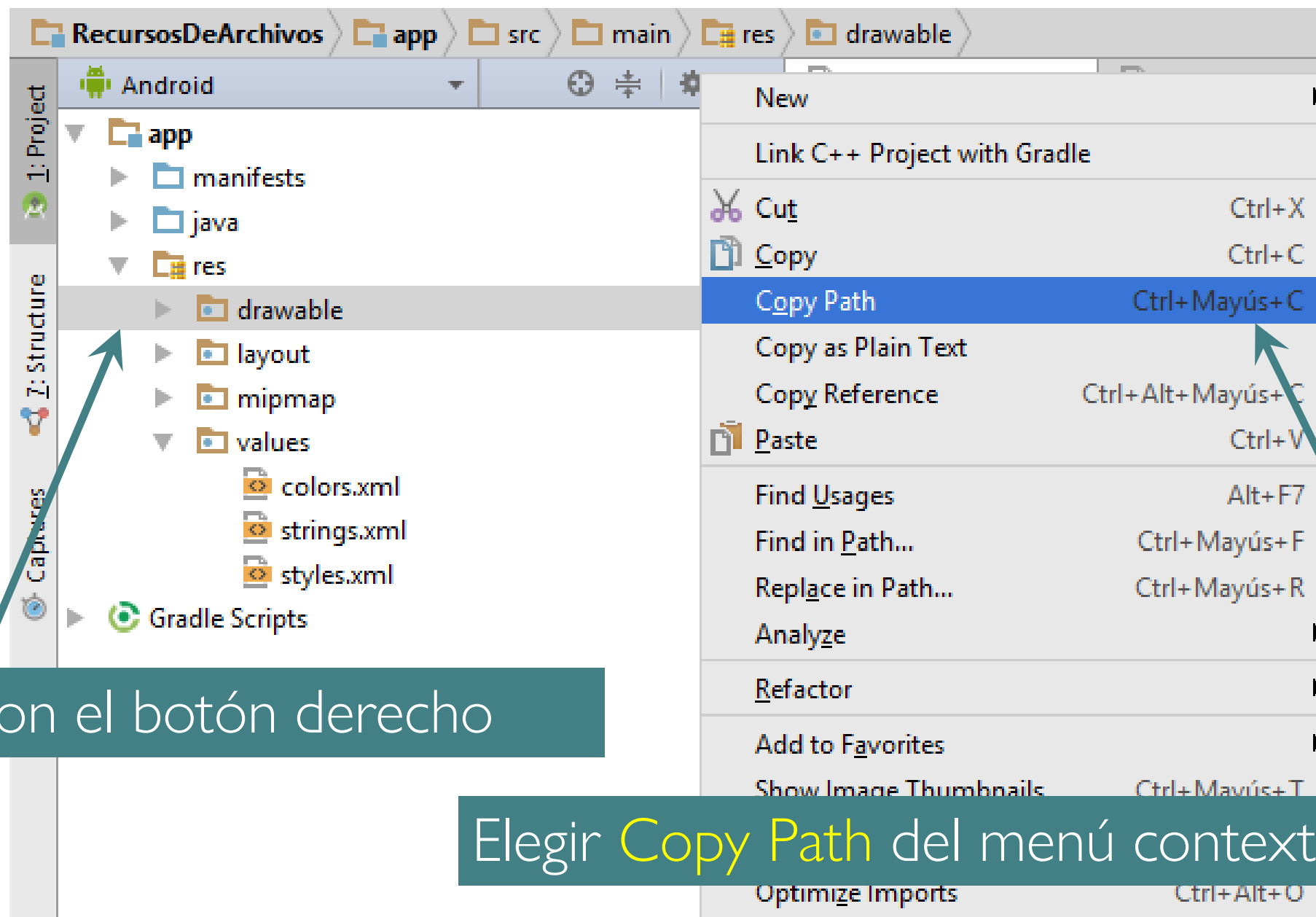
# Actividad guiada

Crear en el **Paint** una imagen con relación 3:5 (por ejemplo 300 x 500)



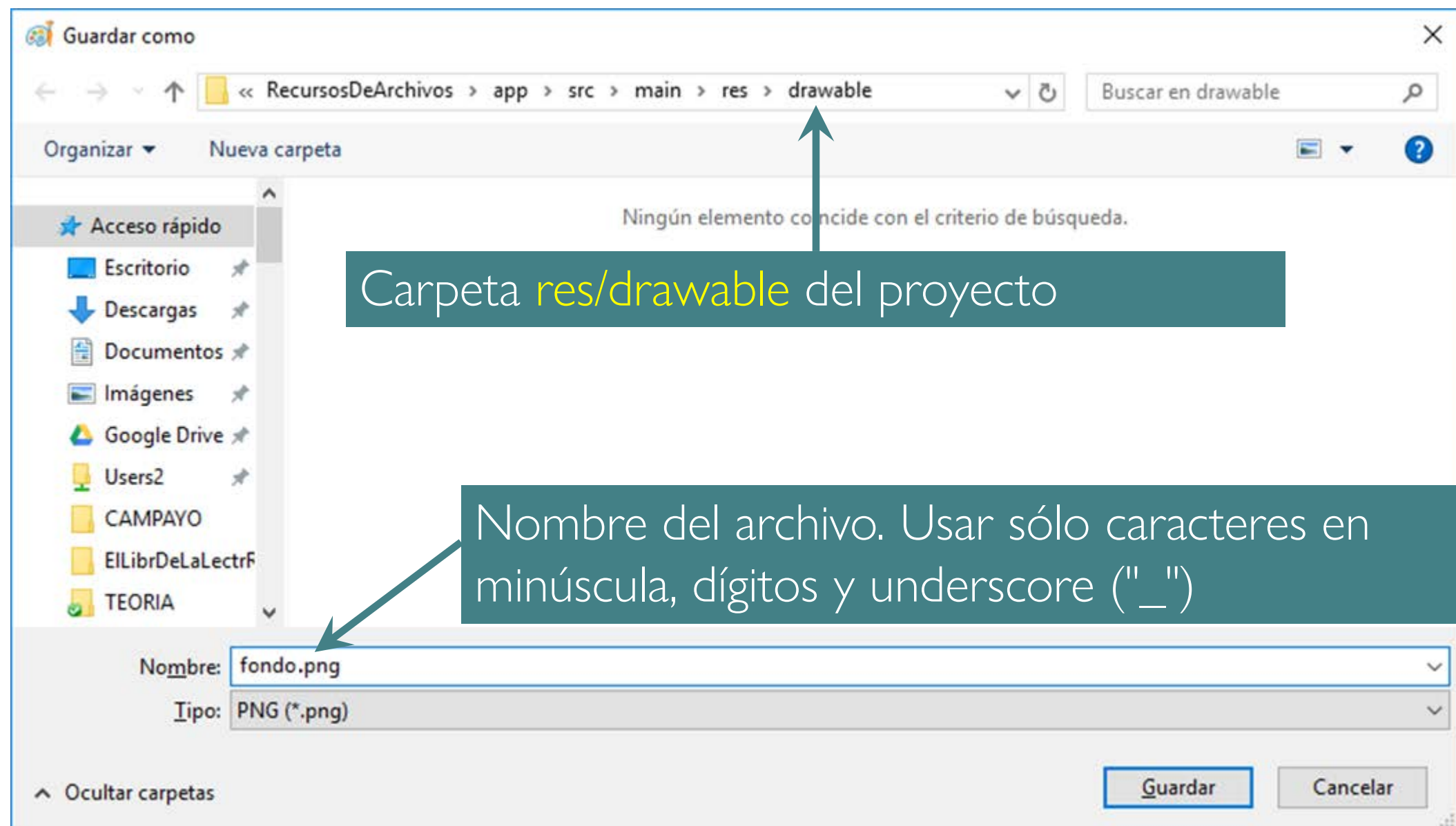
# Actividad guiada

Para guardar la imagen va a ser necesario ubicar el **path** completo de la carpeta **res/drawable** del proyecto



# Actividad guiada

Guardar imagen creada en Paint con el nombre **fondo.png** en la carpeta **res/drawable** del proyecto



# Actividad guiada

Establecer la imagen como fondo del **layout** de la **activity principal**

```
<LinearLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
```

```
    android:background="@drawable/fondo"
```

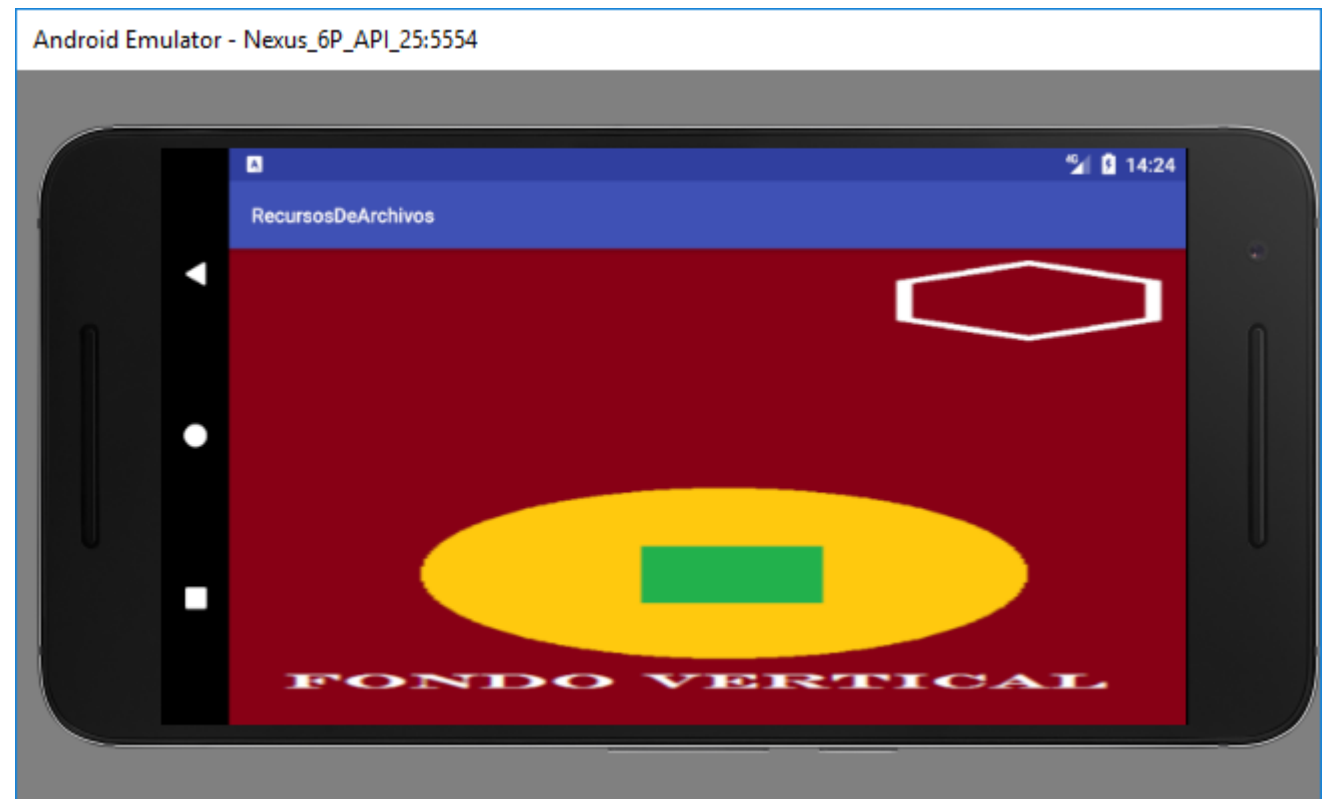
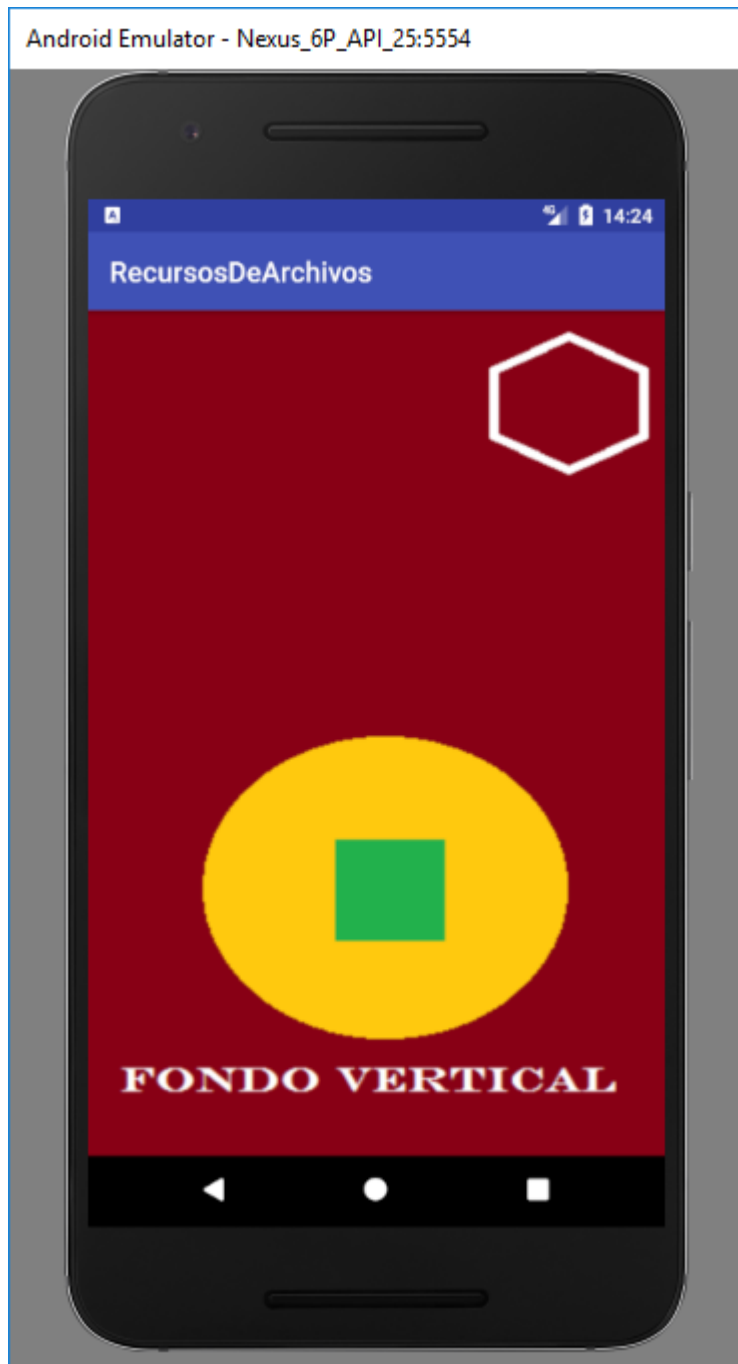
```
>
```

```
</LinearLayout>
```



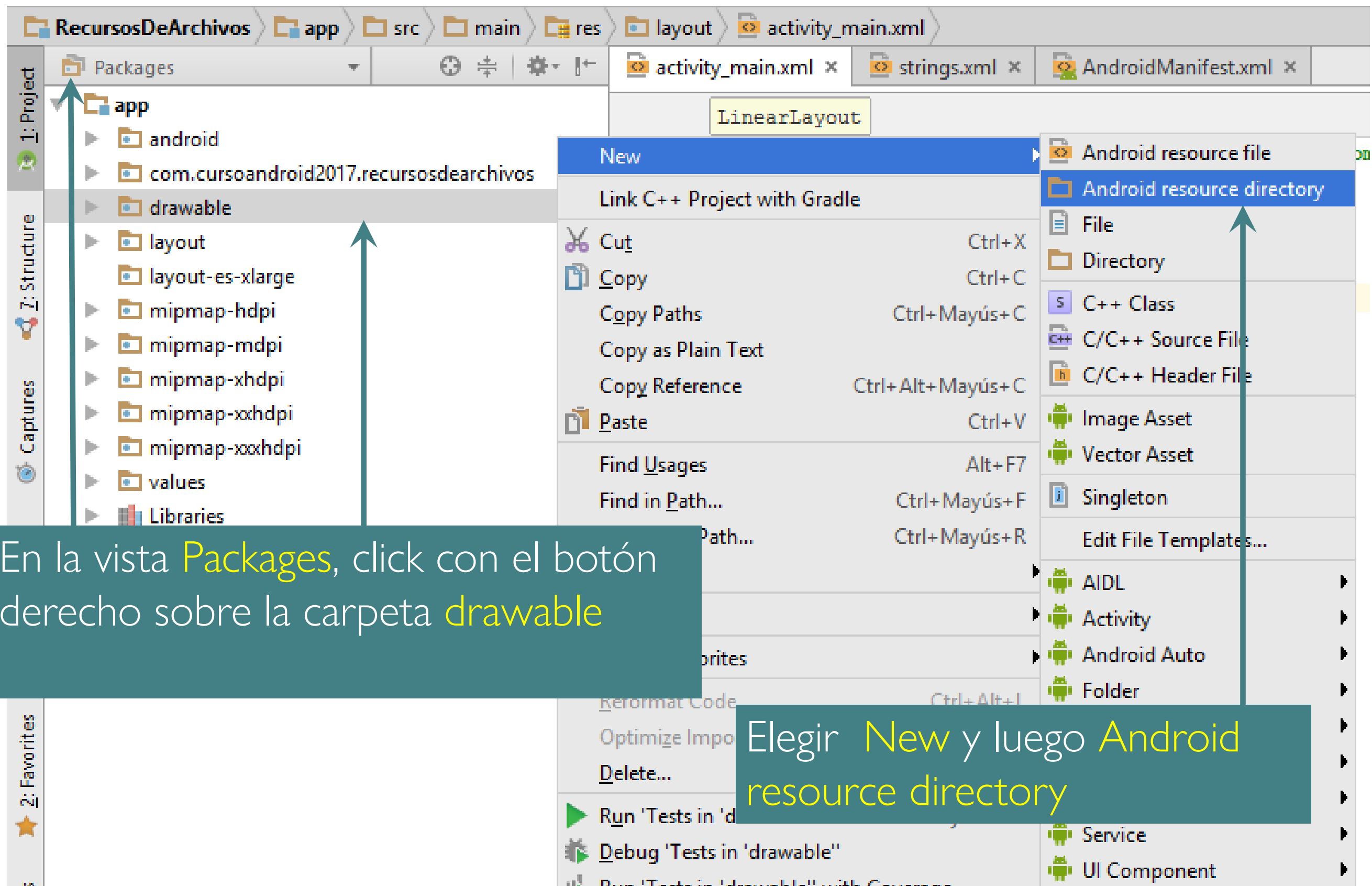
# Actividad guiada

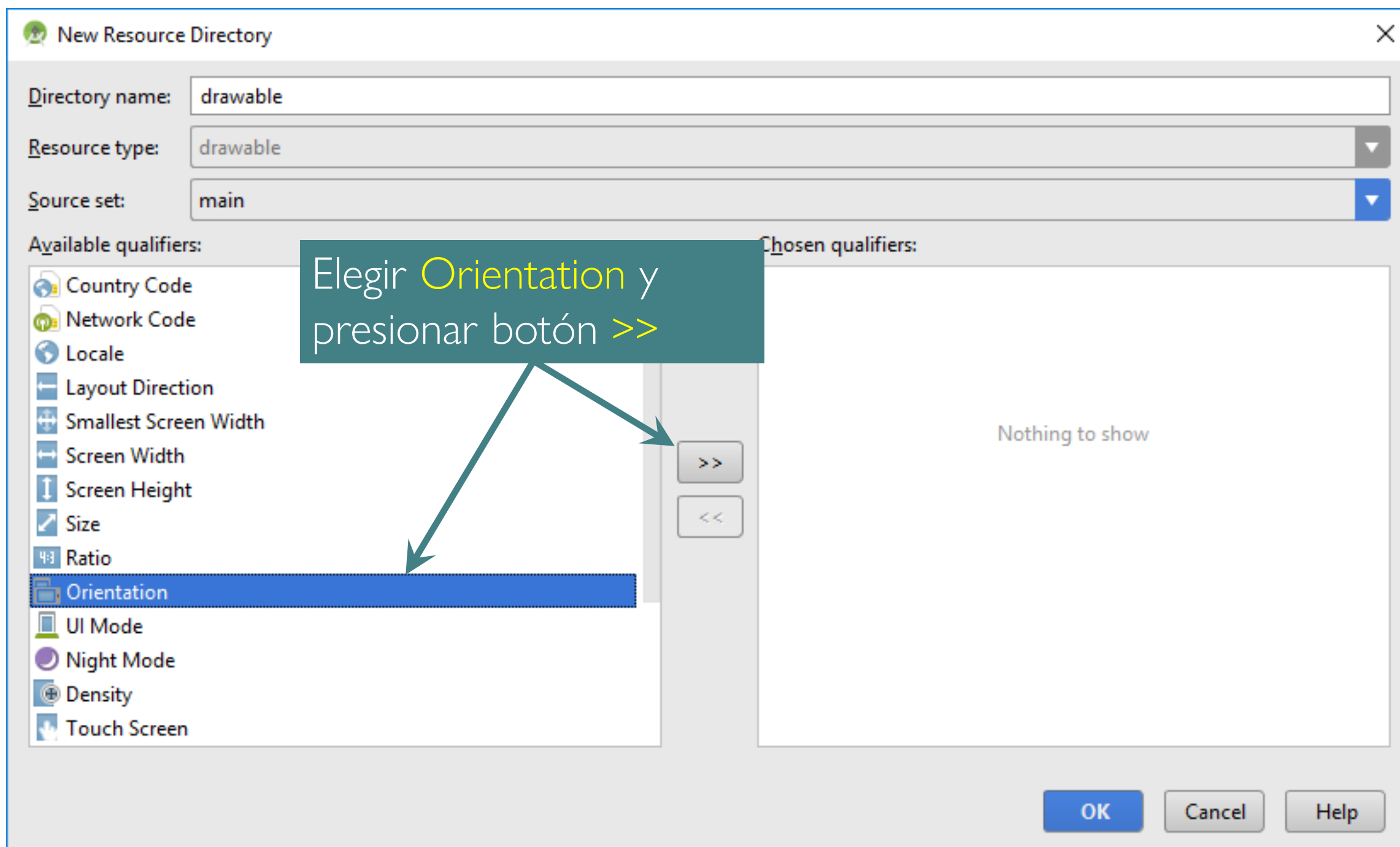
Observar que la imagen de fondo no se ve bien cuando el dispositivo se encuentra en posición horizontal

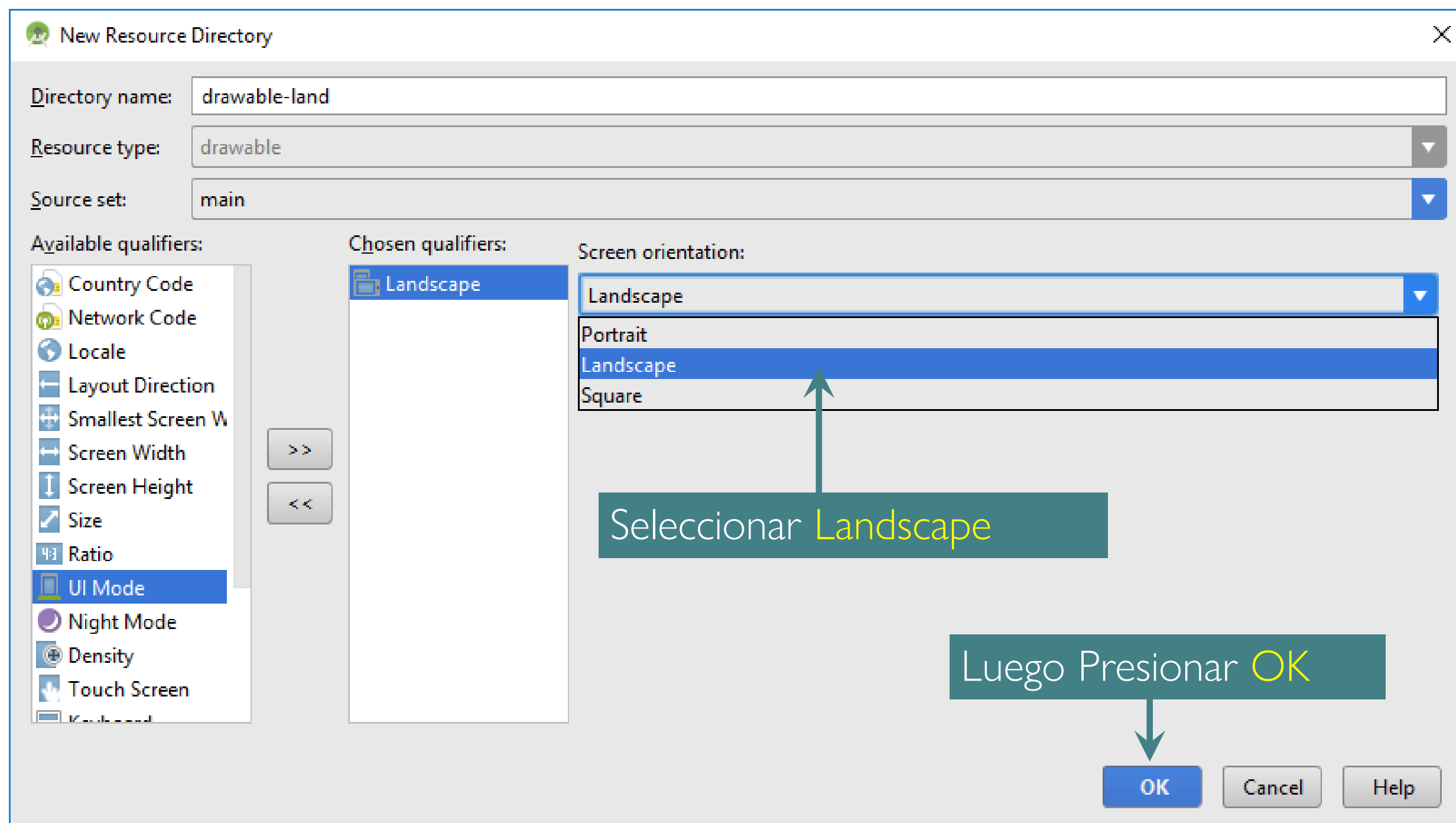


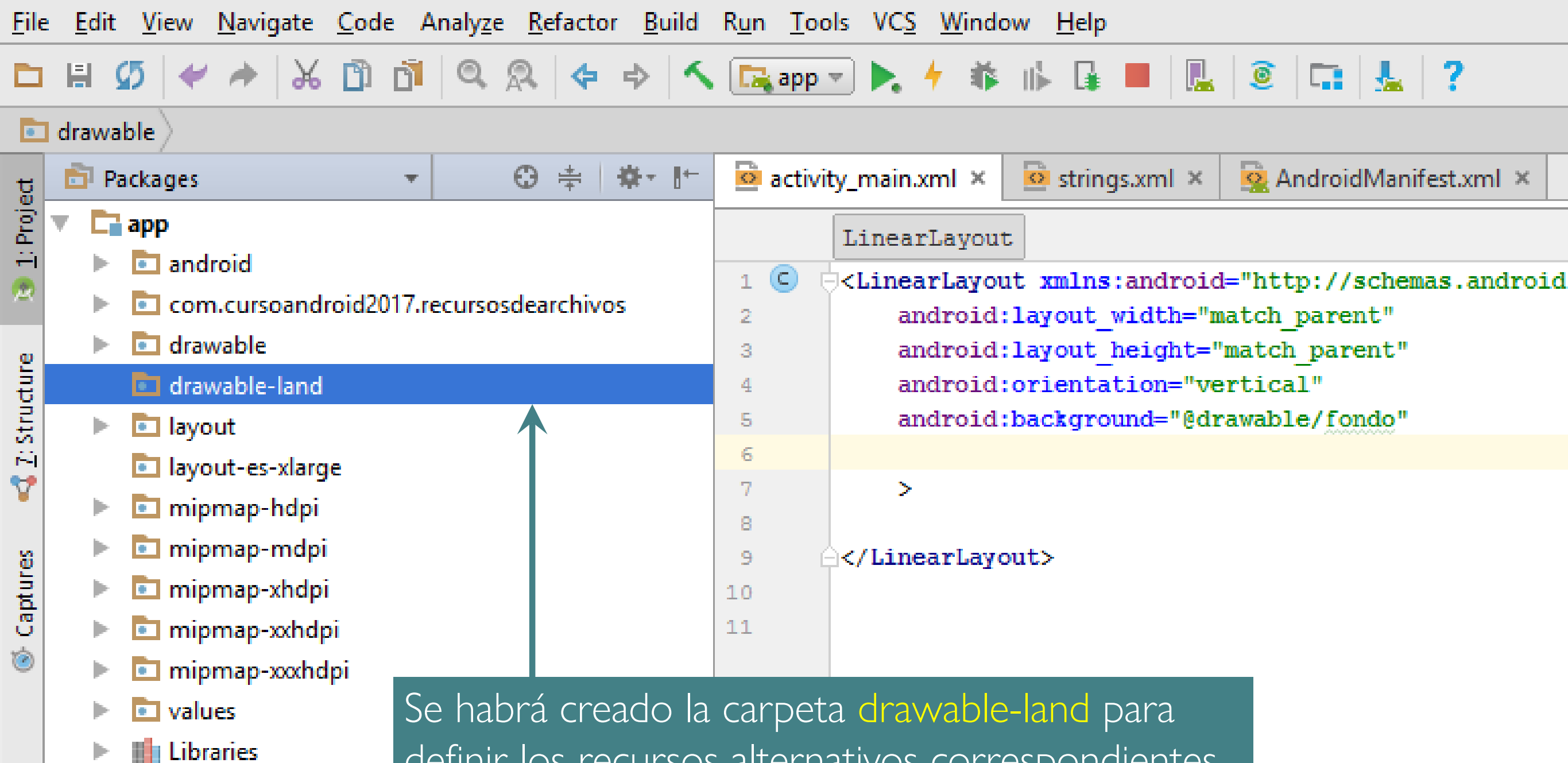
# Actividad guiada

- Vamos a crear un recurso alternativo **drawable** para establecer otra imagen de fondo en la disposición horizontal (**landscape**)
- Para ello es necesario crear el directorio de recursos **res/drawable-land** para guardar en él una imagen que llamaremos también **fondo.png**
- El sufijo **–land** es un calificador que hace referencia a los recursos alternativos para la disposición **landscape**, al igual que el sufijo **–en** lo hace para el **idioma inglés** (visto en la clase anterior)





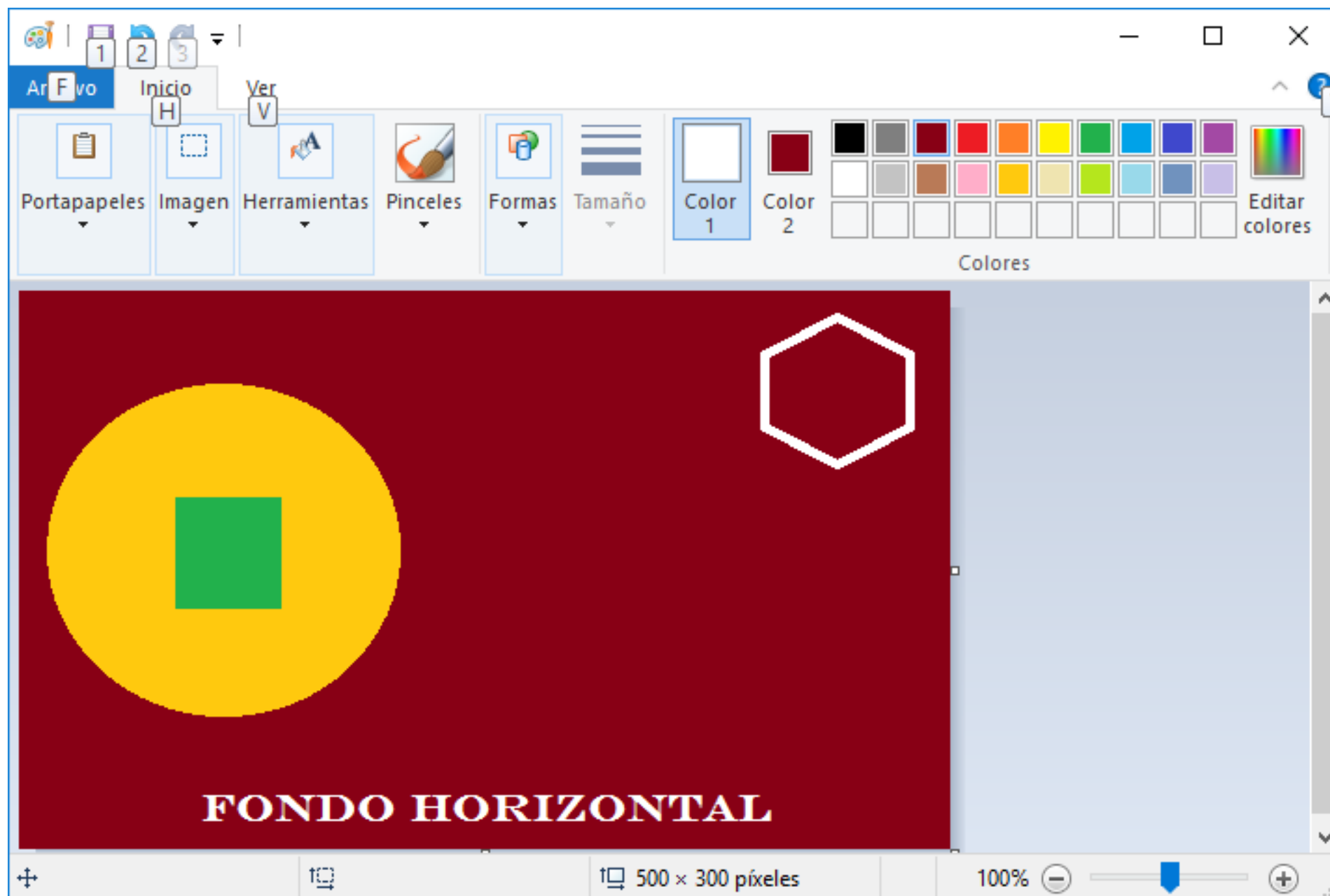




Se habrá creado la carpeta **drawable-land** para definir los recursos alternativos correspondientes a la **orientación horizontal** del dispositivo

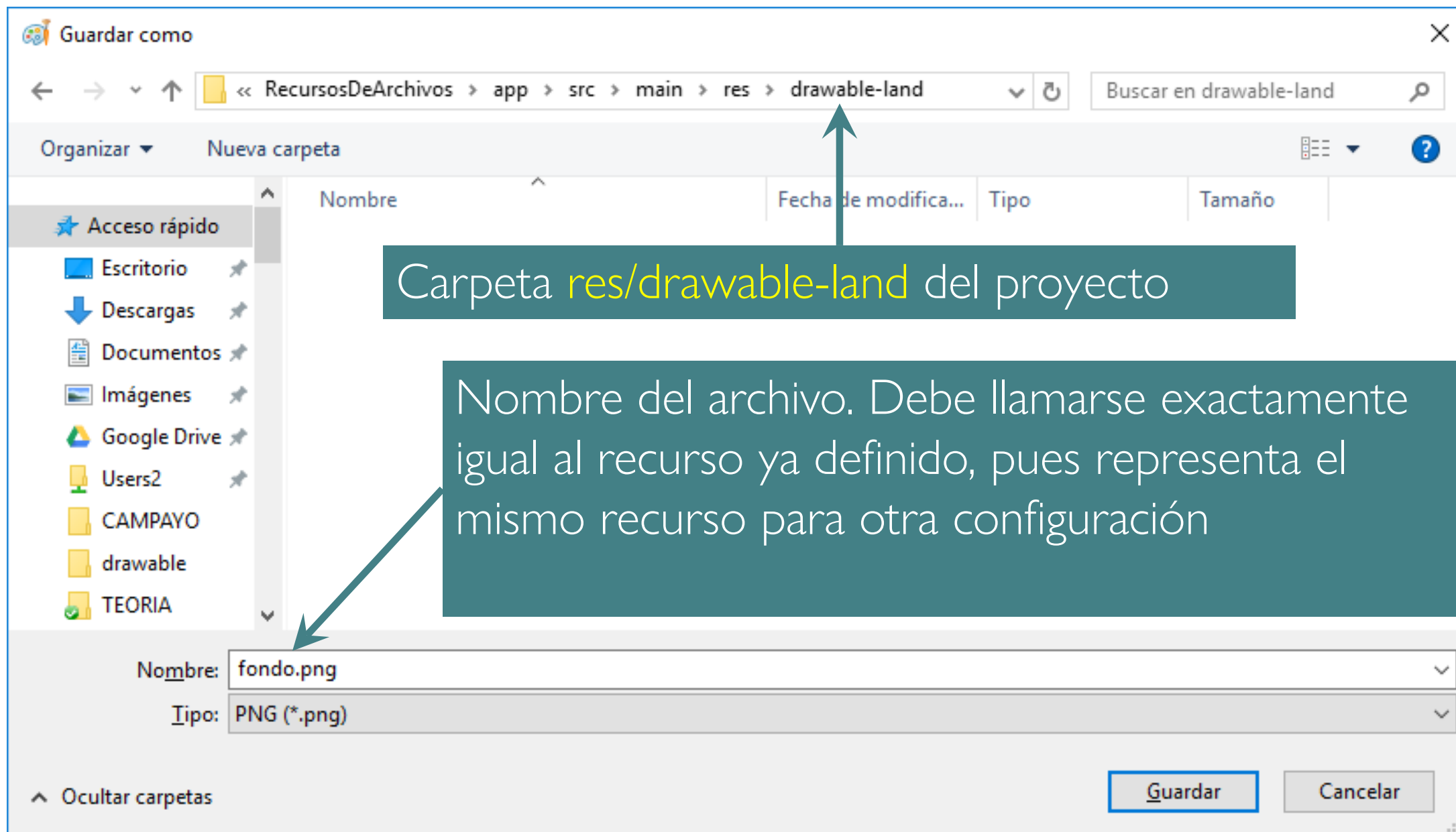
# Actividad guiada

Crear en el **Paint** una imagen con relación 5:3 (por ejemplo 500 x 300)



# Actividad guiada

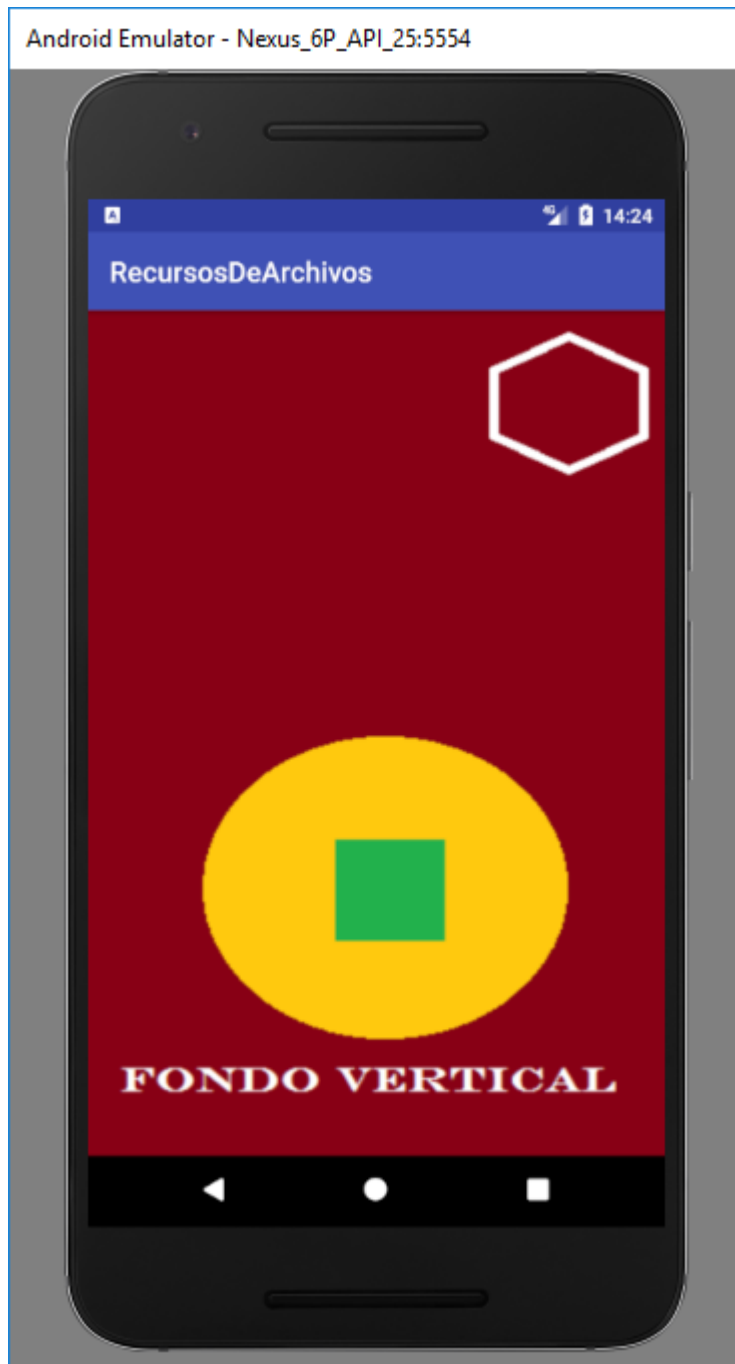
Guardar imagen creada en Paint con el nombre **fondo.png** en la carpeta **res/drawable-land** del proyecto





# Actividad guiada

Verificar ahora que al colocar el dispositivo en disposición horizontal cambia la imagen de fondo.



# Recursos mipmap

- En la carpeta **mipmap/** se colocan los archivos de imágenes que constituyen los recursos predeterminados de íconos lanzadores de la aplicación
- En la carpeta **mipmap-{calificadores}/** se colocan los archivos de imágenes para los recursos alternativos de íconos de la aplicación
- Al crear un proyecto, **Android Studio** establece recursos **mipmap** para cinco densidades distintas: **mdpi** (~160dpi), **hdpi**(~240dpi), **xhdpi** (~320dpi), **xxhdpi** (~480dpi) y **xxxhdpi**(~640dpi)

RecursosDeArchivos > app > src > main > AndroidManifest.xml

Packages

1: Project

2: Structure

Captures

Build Variants

app

- android
- com.cursoandroid2017.recursosdearchivos
- drawable
- drawable-land
- layout
- mipmap-hdpi
  - ic\_launcher.png
  - ic\_launcher\_round.png
- mipmap-mdpi
  - ic\_launcher.png
  - ic\_launcher\_round.png
- mipmap-xhdpi
  - ic\_launcher.png
  - ic\_launcher\_round.png
- mipmap-xxhdpi
  - ic\_launcher.png
  - ic\_launcher\_round.png
- mipmap-xxxhdpi
  - ic\_launcher.png
  - ic\_launcher\_round.png
- values
- Libraries

AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.cursoandroid2017.recursosdearchivos">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="RecursosDeArchivos"
9         android:roundIcon="@mipmap/ic_launcher"
10        android:supportsRtl="true"
11        android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15                 <category android:name="android.intent.category.LAUNCHER" />
16             </intent-filter>
17         </activity>
18     </application>
19
20 </manifest>
```

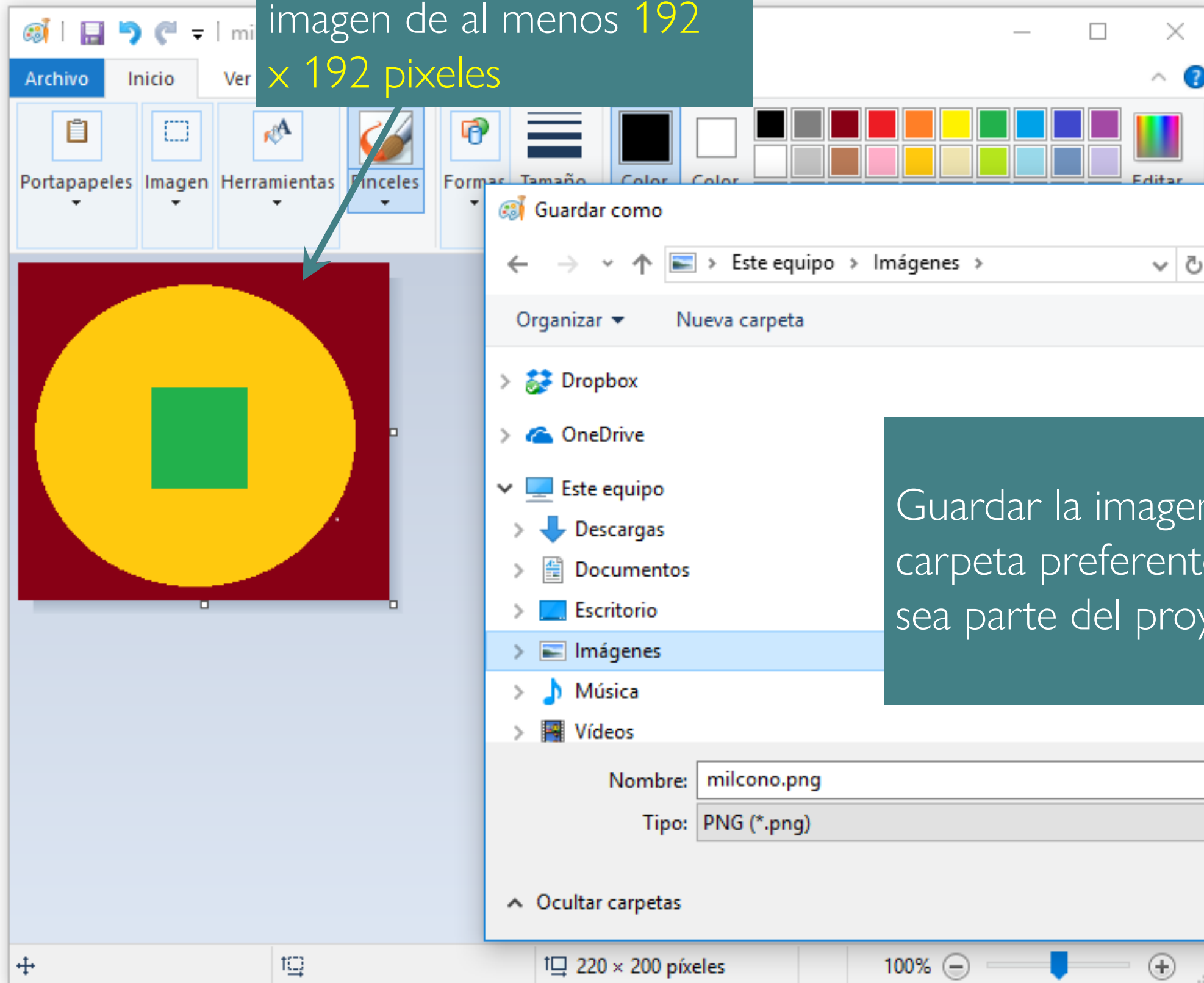
Los íconos lanzadores se establecen en el manifiesto de la aplicación

Recursos alternativos para cinco densidades distintas

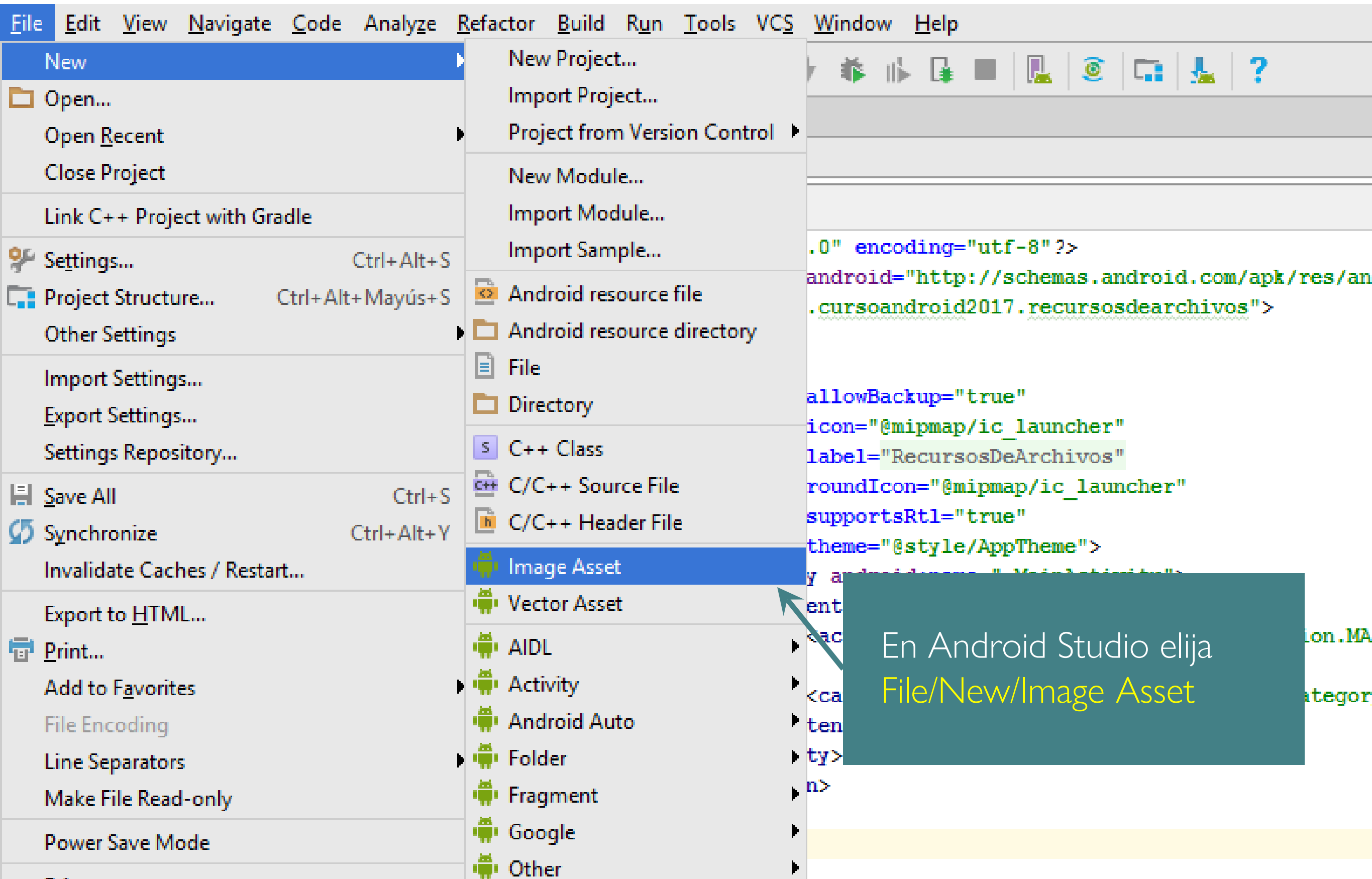
# Recursos mipmap

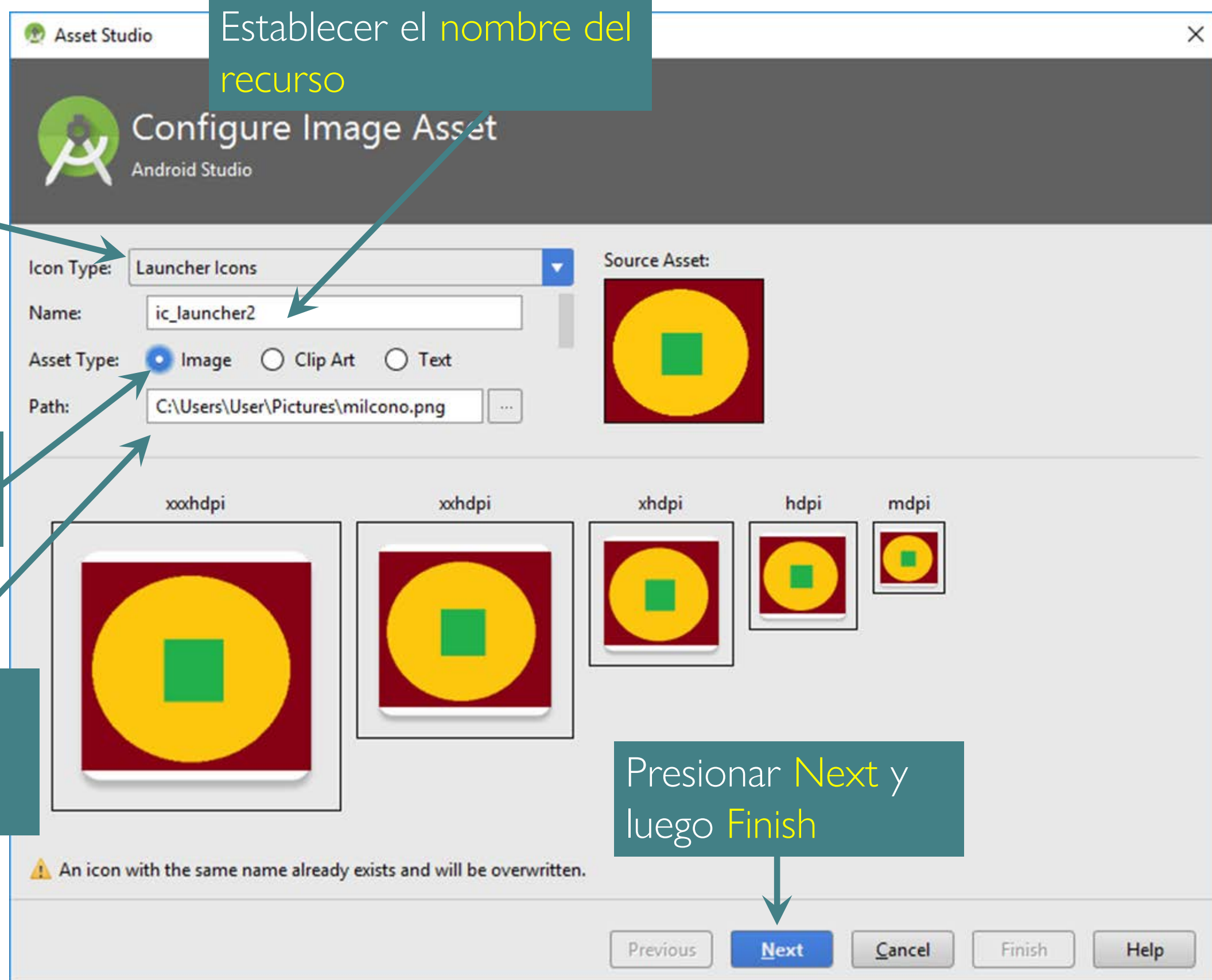
- Vamos a crear un nuevo **ícono lanzador** para nuestra aplicación
- Luego utilizaremos una herramienta provista por **Android Studio** para generar las distintas versiones alternativas para cada densidad

Crear en el Paint una imagen de al menos 192 x 192 píxeles



Guardar la imagen en alguna carpeta preferentemente que no sea parte del proyecto





The screenshot displays the Android Studio interface. On the left, the 'Project' tab shows the project structure under the 'app' folder. It includes subfolders for 'android', 'com.cursoandroid2017.recursosdearchivos', 'drawable', 'drawable-land', 'layout', and 'mipmap-hdpi'. The 'mipmap-hdpi' folder is expanded, showing three launcher icons: 'ic\_launcher.png', 'ic\_launcher2.png' (highlighted), and 'ic\_launcher\_round.png'. Below this, other mipmap folders (mdpi, xhdpi, xxhdpi, xxxhdpi) are also expanded, each containing the same three launcher icons. The 'Build Variants' tab at the bottom left shows 'avorites'.

The main editor displays the 'AndroidManifest.xml' file. The 'manifest' tab is active. The XML code is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/a
3     package="com.cursoandroid2017.recursosdearchivos">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="RecursosDeArchivos"
9         android:roundIcon="@mipmap/ic_launcher"
10        android:supportsRtl="true"
11        android:theme="@style/AppTheme">
12        <activity android:name=".MainActivity">
13            <intent-filter>
14                <action android:name="android.intent.action.M
15
16                <category android:name="android.intent.catego
17            </intent-filter>
18        </activity>
19    </application>
20
```

A lightbulb icon is visible next to the closing tag of the application element on line 19.

Se crean los recursos alternativos para cada densidad



En `AndroidManifest.xml` establecer los íconos de lanzamiento con el nuevo recurso definido

Ejecutar en el emulador y verificar que ha cambiado el ícono de lanzamiento de la aplicación

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cursoandroid2017.recursosdearchivos">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher2"
        android:label="RecursosDeArchivos"
        android:roundIcon="@mipmap/ic_launcher2"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN">
                <category android:name="android.intent.category.LAUNCHER">
            </intent-filter>
        </activity>
    </application>


```

# Actividad guiada - continuación

Con la misma estrategia utilizada al establecer el fondo de la aplicación, implemente dos **layout** distintos para la **activity** principal



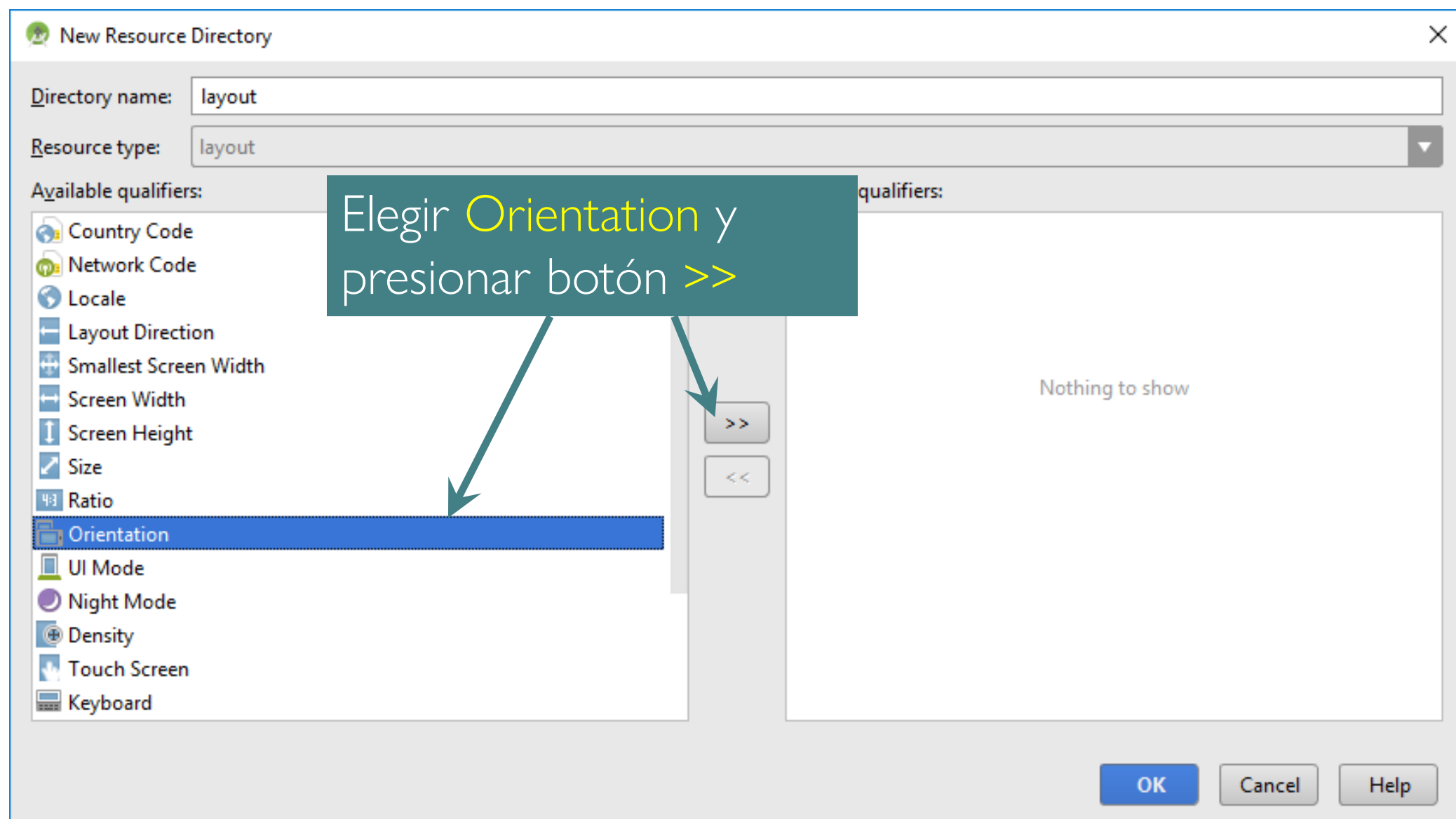
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/fondo"
    android:gravity="center"
>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Nuevo Juego"
    />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Continuar jugando"
    />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Configuración"
    />
</LinearLayout>
```

activity\_main.xml

Layout predeterminado de la activity principal

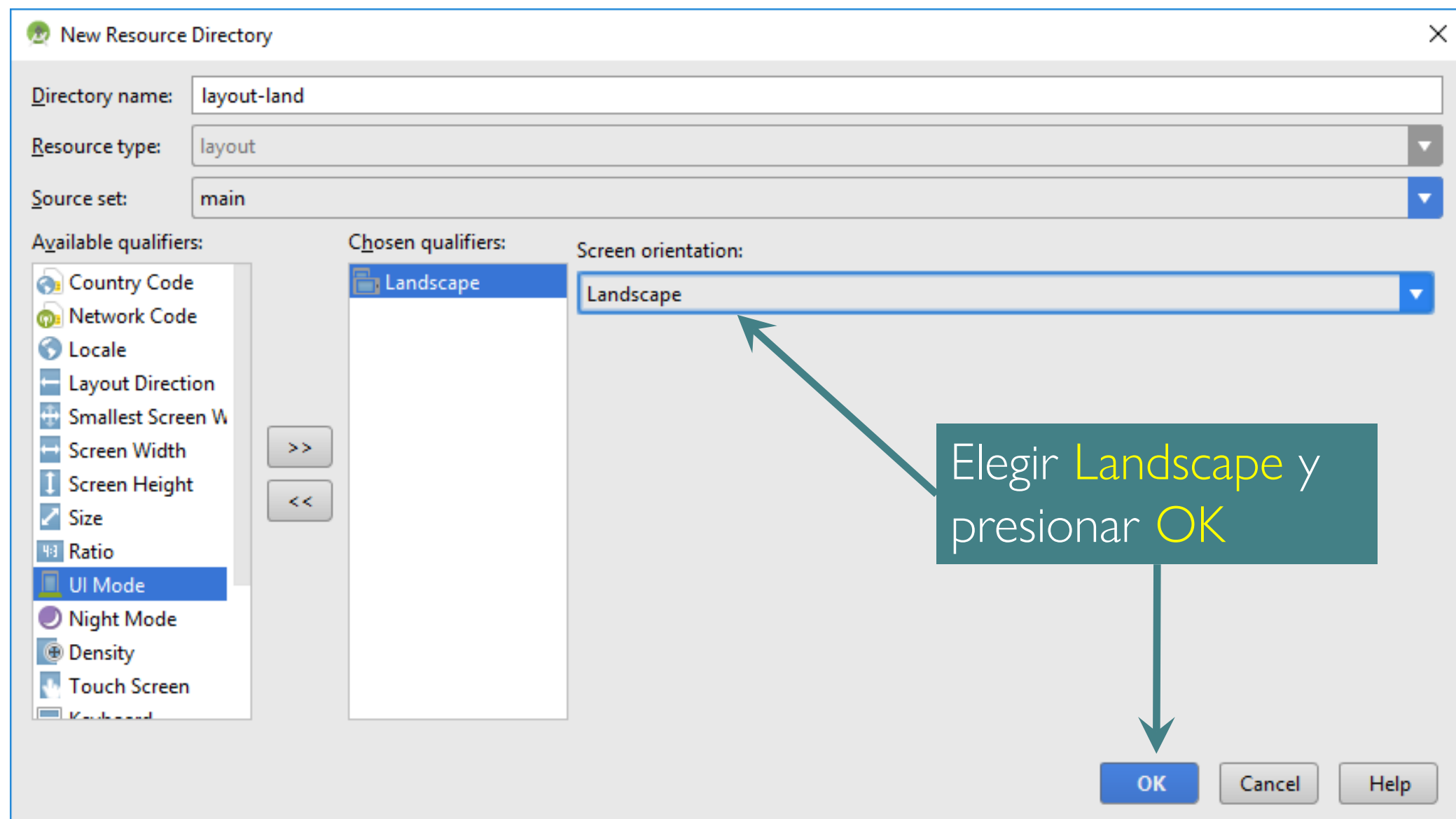
# Actividad guiada - continuación

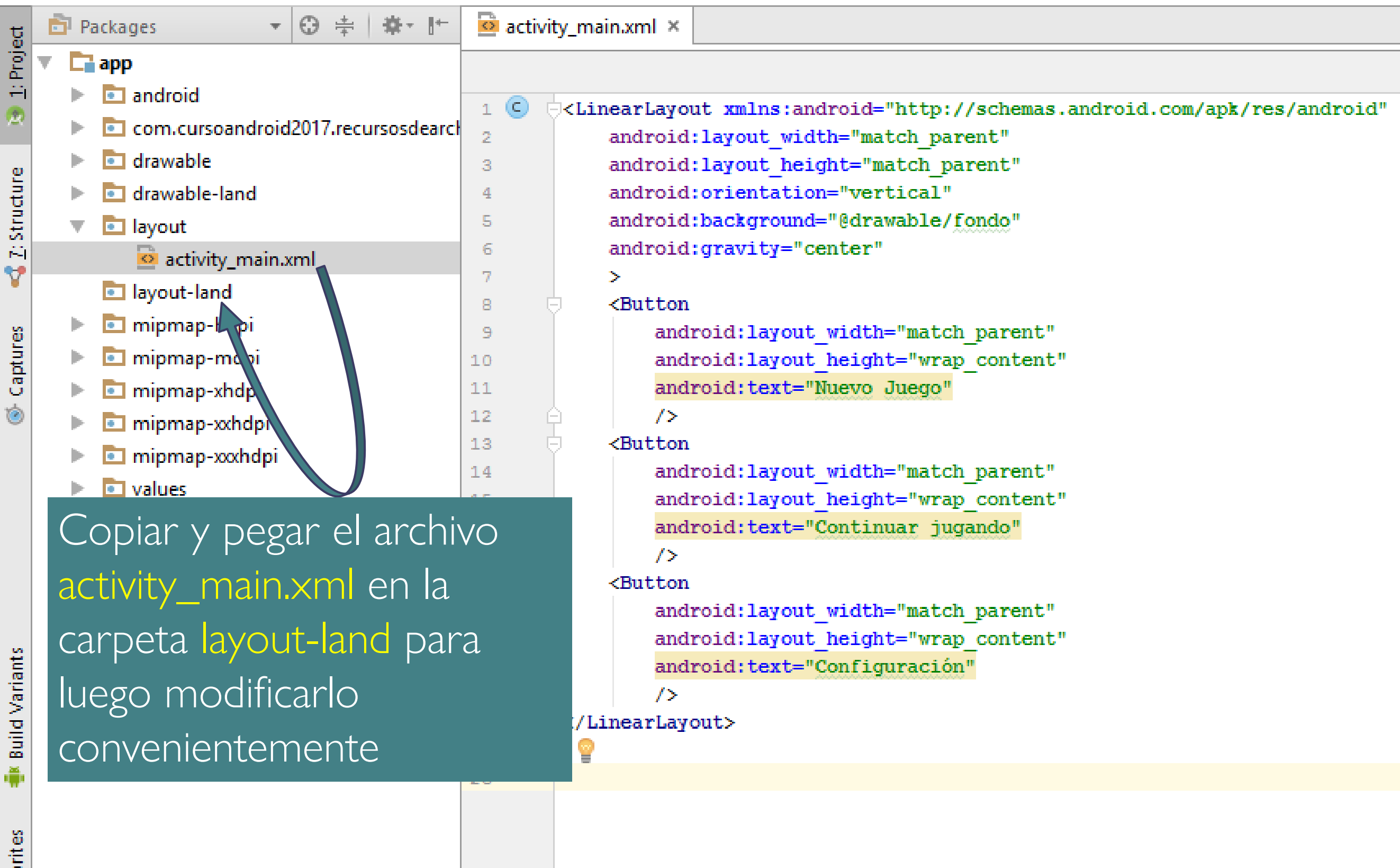
Crear un nuevo directorio de recursos **layout-land** para definir el **layout** de la **activity principal** en el caso de la **disposición horizontal** del dispositivo



# Actividad guiada - continuación

Crear un nuevo directorio de recursos **layout-land** para definir el **layout** de la **activity principal** en el caso de la **disposición horizontal** del dispositivo





Project Structure:

- app
  - android
  - com.cursoandroid2017.recursosdearchivos
  - drawable
  - drawable-land
  - layout
    - activity\_main.xml
    - layout-land
    - mipmap-hdpi
    - mipmap-mdpi
    - mipmap-xhdpi
    - mipmap-xxhdpi
    - mipmap-xxxhdpi
    - values

activity\_main.xml content:

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     android:layout_width="match_parent"
3     android:layout_height="match_parent"
4     android:orientation="vertical"
5     android:background="@drawable/fondo"
6     android:gravity="center"
7 >
8     <Button
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:text="Nuevo Juego"
12    />
13    <Button
14        android:layout_width="match_parent"
15        android:layout_height="wrap_content"
16        android:text="Continuar jugando"
17    />
18    <Button
19        android:layout_width="match_parent"
20        android:layout_height="wrap_content"
21        android:text="Configuración"
22    />
23 </LinearLayout>
```

Copiar y pegar el archivo **activity\_main.xml** en la carpeta **layout-land** para luego modificarlo convenientemente

```

<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:background="@drawable/fondo"
    android:stretchColumns="*">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Nuevo Juego" />
    <TableRow>
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Continuar jugando" />
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Configuración" />
    </TableRow>
</TableLayout>

```

activity\_main.xml en la carpeta layout-land  
Recurso de layout alternativo para la orientación landscape

Ejecutar en el emulador

# Para poner en práctica - Ejercicio:

Crear un fondo alternativo para la activity principal para el caso en que el dispositivo esté dispuesto horizontalmente con la configuración del idioma en francés.

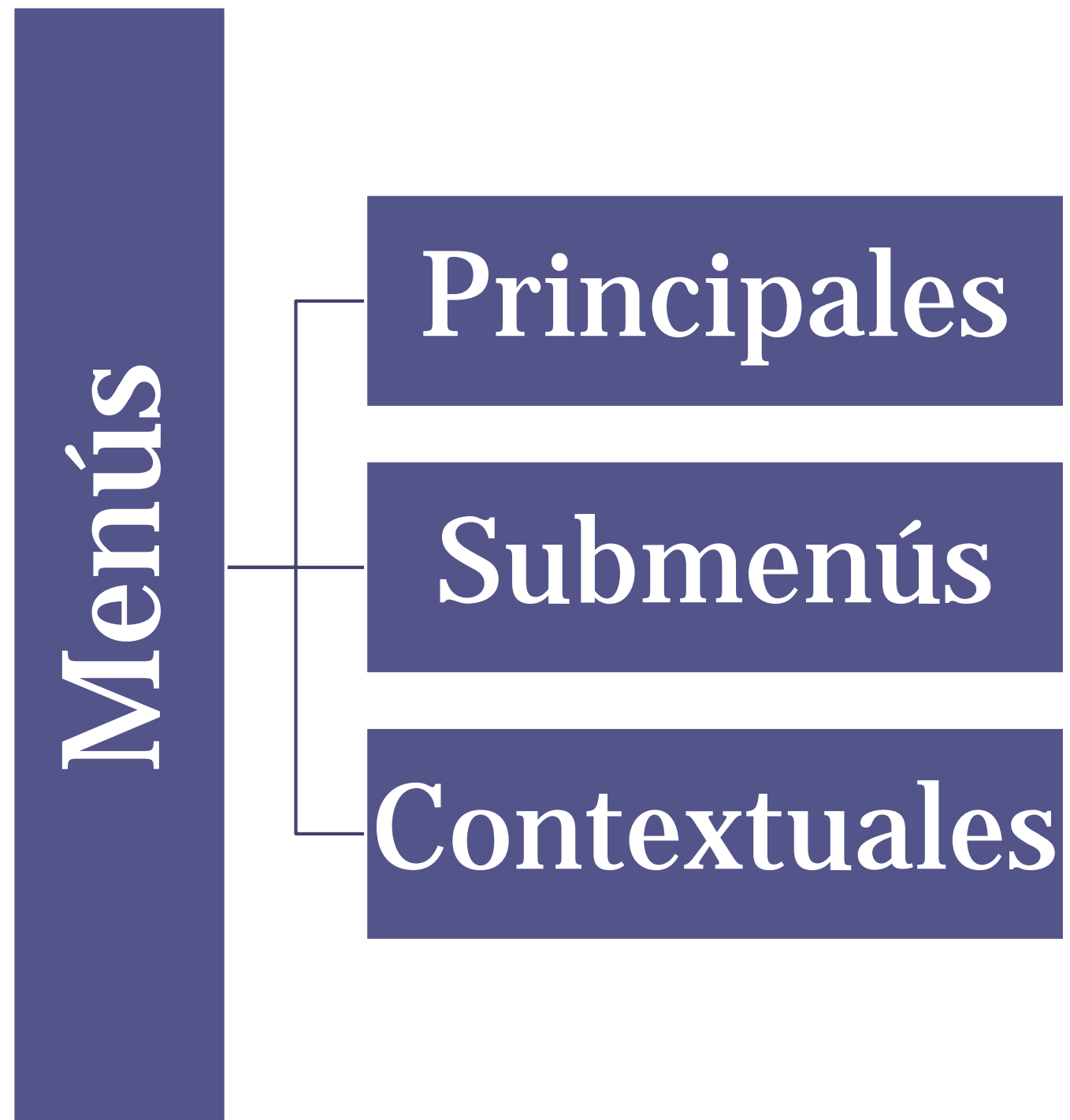
**TIP:** Se debe crear el directorio de recursos **drawable-fr-land/**





# Menús

# Tres tipos de Menús



# Menú principal de una *activity*

Si se desarrolla una aplicación para **Android 2.3.x (nivel de API 10) o versiones anteriores**, el contenido del menú de opciones aparece en la parte inferior de la pantalla cuando el usuario presiona el botón *Menú* del dispositivo



Menú de opciones del navegador, en Android 2.3.

# Menú principal de una *activity*

Si se desarrolla una aplicación para Android 3.0 (nivel de API 11) y versiones posteriores, los elementos del menú de opciones están disponibles a la derecha sobre la barra de app



Algunos ítems del menú pueden visualizarse directamente sobre la barra

Los otros ítems se visualizan por medio del ícono de acciones adicionales

# Creación de un menú principal de una *activity*

## 1. Creación del menú

- a) Crear un nuevo archivo de recursos XML de tipo menú
- b) Configurar las opciones de menú: Id y título

## 2. Activación del menú

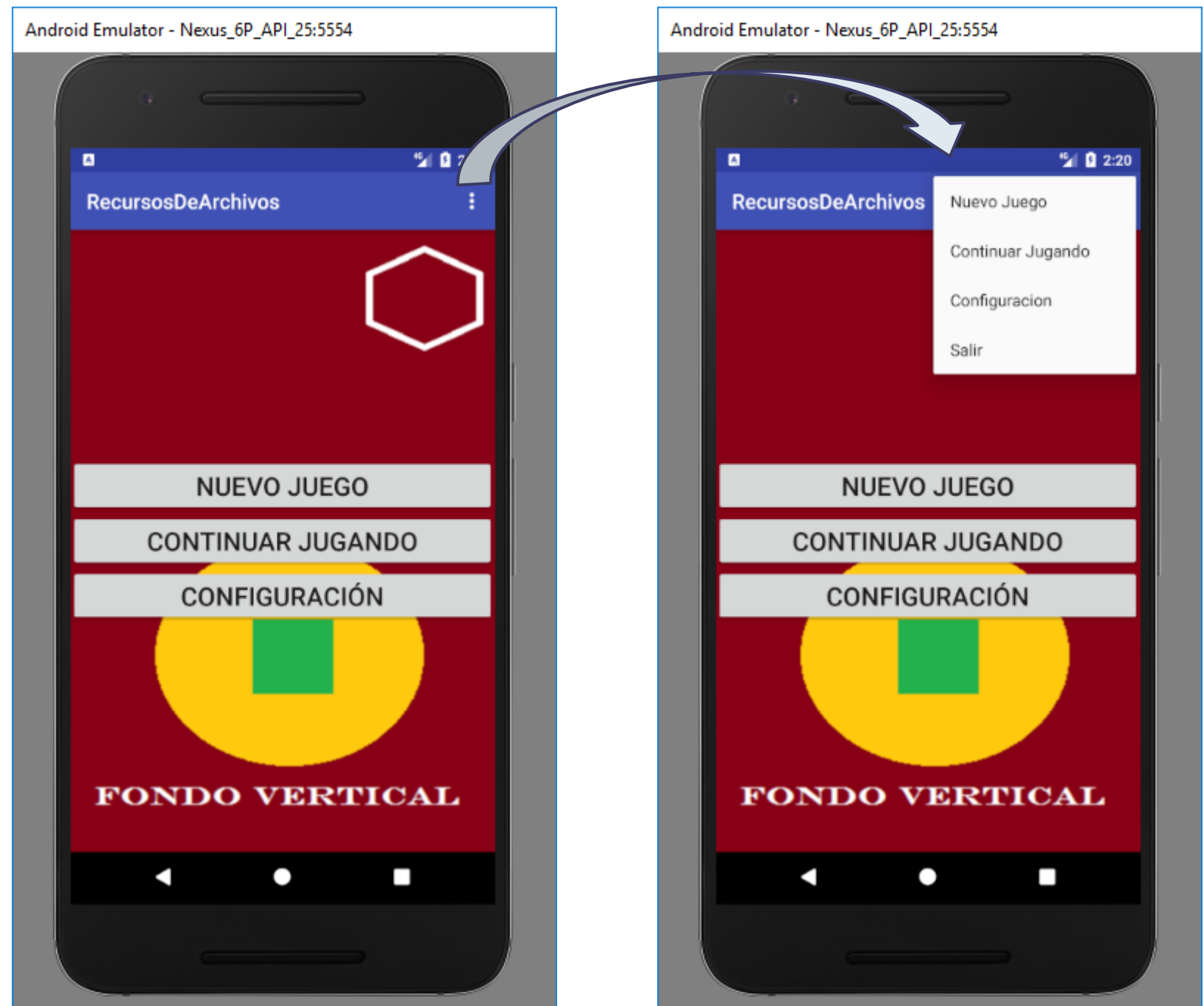
1. Agregar el menú a la activity
2. Configurar la acción según las opciones elegidas

`onOptionsItemSelected()`

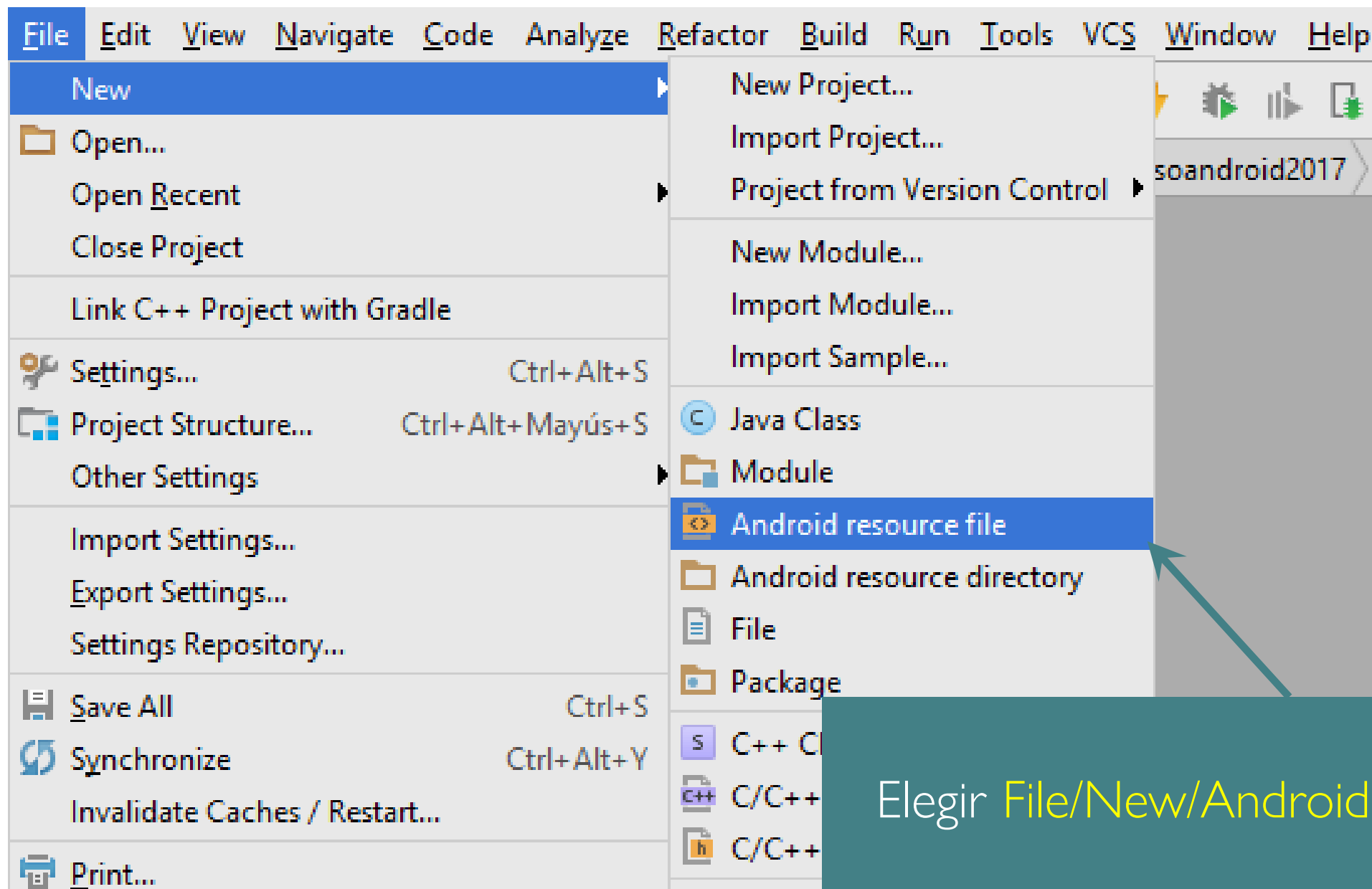
`onOptionsItemSelected()`

# Actividad guiada - creación de menú principal

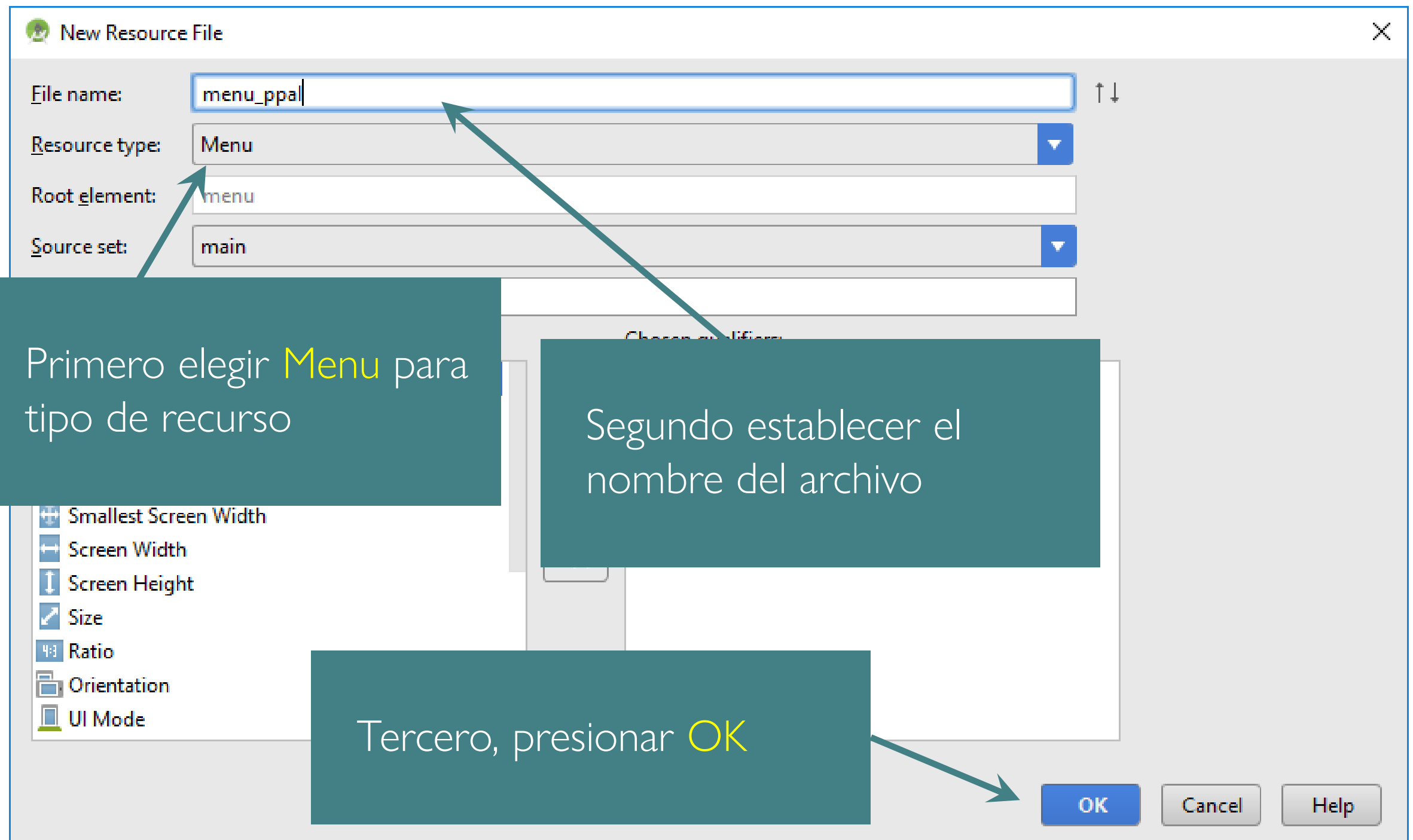
Vamos a crear un menú principal para la **activity** definida en nuestra aplicación.



# Actividad guiada - creación de menú principal

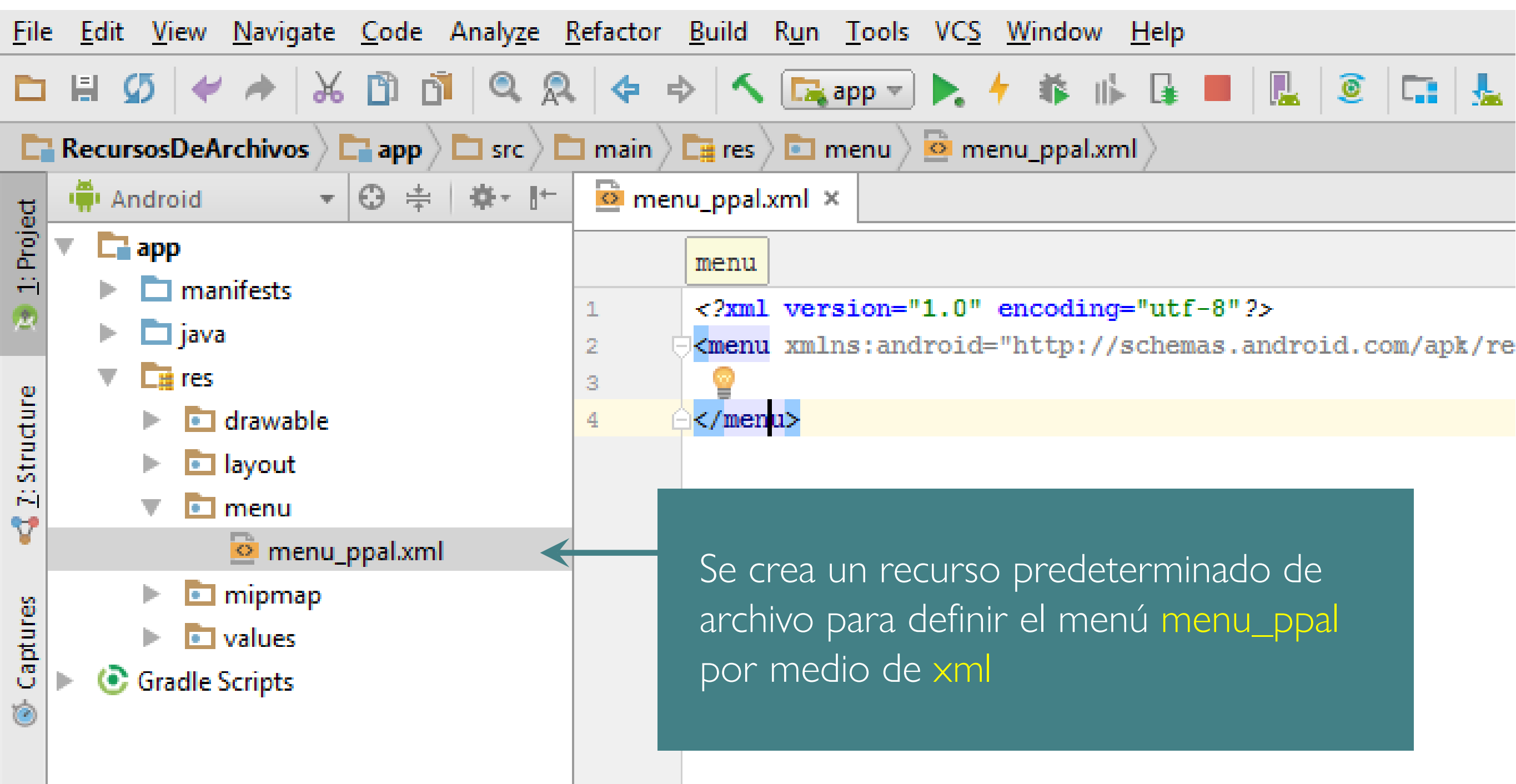


# Actividad guiada - creación de menú principal





# Actividad guiada - creación de menú principal



The screenshot shows an IDE interface with the following components:

- Menu Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Toolbar:** Contains icons for file operations (open, save, copy, paste, delete), navigation (back, forward, search), and development (run, debug, stop, refresh, undo, redo).
- Breadcrumb:** RecursosDeArchivos > app > src > main > res > menu > menu\_ppal.xml.
- Project Explorer (Left):** Shows the project structure under '1: Project'. The 'app' folder is expanded, showing 'manifests', 'java', 'res', and 'Gradle Scripts'. The 'res' folder is further expanded to show 'drawable', 'layout', and 'menu'. The file 'menu\_ppal.xml' is highlighted in the 'menu' folder.
- Code Editor (Right):** Displays the content of 'menu\_ppal.xml' with line numbers 1 to 4. The code is:

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

A teal callout box with a white arrow pointing to the 'menu\_ppal.xml' file in the Project Explorer contains the text:

Se crea un recurso predeterminado de archivo para definir el menú **menu\_ppal** por medio de **xml**

# Actividad guiada - creación de menú principal

Defina el siguiente menú:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menuNuevo"
        android:title="Nuevo Juego" />
    <item
        android:id="@+id/menuContinuar"
        android:title="Continuar Jugando" />
    <item
        android:id="@+id/menuConfiguracion"
        android:title="Configuracion" />
    <item
        android:id="@+id/menuSalir"
        android:title="Salir" />
</menu>
```

# Actividad guiada - creación de menú principal

Redefina el método `onCreateOptionsMenu()` de la clase `MainActivity`

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

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


Probar en el  
emulador

`true` indica que debe visualizarse

# Actividad guiada - creación de menú principal

Para establecer las acciones de las opciones elegidas debemos redefinir el método `onOptionsItemSelected()` de la clase `MainActivity`

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.menuSalir)
    {
        this.finish();
    }
    return true;
}
```



Probar en el emulador

Retornar `true` finaliza el procesamiento de la selección de menú. Si se devuelve `false`, y el ítem seleccionado posee un Intent, se prosigue el procesamiento lanzando la actividad correspondiente.

# Actividad guiada - Sub menú

Modifique **menu\_ppal.xml** y verifique el funcionamiento

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menuNuevo"
        android:title="Nuevo Juego">
        <menu>
            <item android:id="@+id/ES1"
                android:title="Escenario 1" />
            <item android:id="@+id/ES2"
                android:title="Escenario 2" />
        </menu>
    </item>
    <item
        android:id="@+id/menuContinuar"
        android:title="Continuar Jugando" />
    <item
```

# Actividad guiada - Sub menú



La opción Nuevo Juego  
ahora despliega un  
submenú



# Opciones en la ActionBar

Modifique **menu\_ppal.xml** y verifique el funcionamiento

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/menuNuevo"
        android:title="Nuevo Juego"
        app:showAsAction="ifRoom"
    >
        <menu>
            <item android:id="@+id/ES1"
                android:title="Escenario 1" />
            <item android:id="@+id/ES2"
                android:title="Escenario 2" />
        </menu>
    </item>
    <item
        android:id="@+id/menuContinuar"
        android:title="Continuar Jugando"
        app:showAsAction="ifRoom" />
    <item
```

# Opciones en la ActionBar





# Opciones en la ActionBar

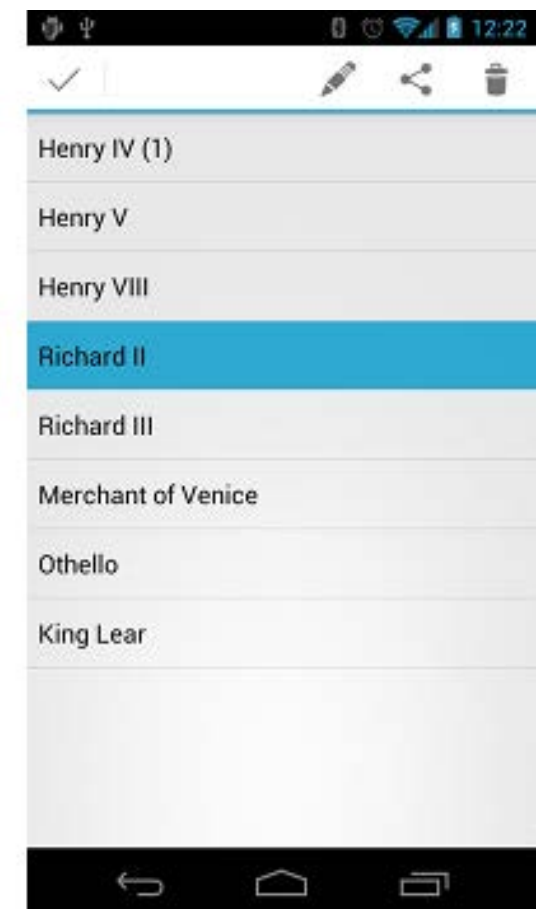
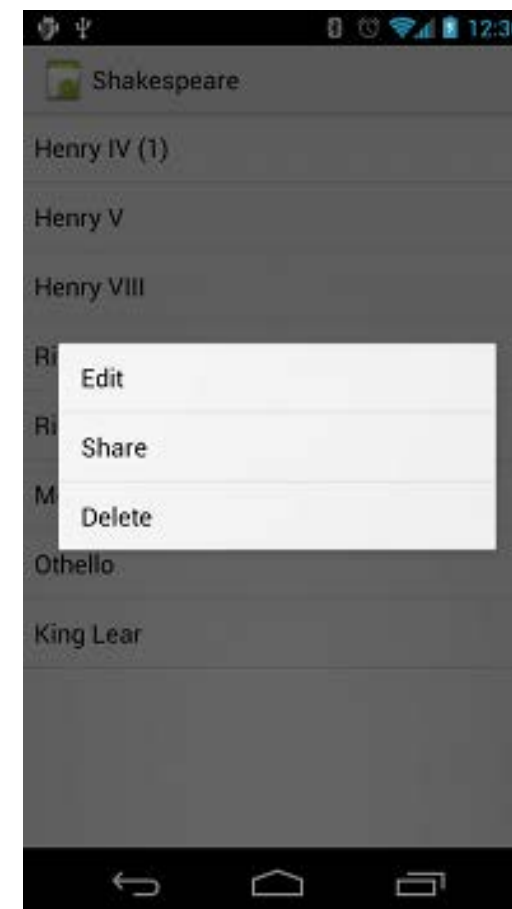
En la **ActionBar** las opciones pueden visualizarse por medio de íconos

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
    android:id="@+id/menuNuevo"
    android:title="Nuevo Juego"
    app:showAsAction="ifRoom"
    android:icon="@android:drawable/star_big_on"
  >
  <menu>
    <item android:id="@+id/1"
          android:title="Escer"
    <item android:id="@+id/1"
          android:title="Escer"
    </menu>
  </item>
```



# Menús contextuales

- Un **menú contextual** se asocia a un control concreto de la pantalla y se muestra al realizar una **pulsación larga** sobre éste.
- Suele mostrar **opciones específicas** disponibles únicamente para el elemento pulsado.
- La creación y utilización de este tipo de menús es muy parecida a lo que ya vimos para los menús y submenús básicos.
- Métodos claves:  
**`onCreateContextMenu()`** y **`onContextItemSelected()`**



# Actividad guiada – Menú contextual

Vamos a definir un menú contextual para el **layout** contenedor de la activity de nuestra aplicación.

Agregue el recurso de menú **menu\_contextual.xml** con la siguiente definición

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/op1"
        android:title="opción 1" />
    <item android:id="@+id/op2"
        android:title="opción 2" />
    <item android:id="@+id/op3"
        android:title="opción 3" />
</menu>
```

# Actividad guiada – Menú contextual

- Agregar `android:id="@+id/fondo"` al `layout` contenedor en ambos recursos (predeterminado y alternativo (`-land`))
- Modificar el método `onCreate()` de `MainActivity`

`@Override`

```
protected void onCreate(Bundle  
savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    View fondo = findViewById(R.id.fondo);  
    registerForContextMenu(fondo);  
}
```

Se indica que la vista con `id = fondo` tendrá un menú contextual

# Actividad guiada – Menú contextual

Agregar los siguiente dos métodos en **MainActivity**

@Override

```
public void onCreateContextMenu(ContextMenu menu, View v,  
                                ContextMenu.ContextMenuInfo menuInfo) {  
    MenuInflater mi = getMenuInflater();  
    mi.inflate(R.menu.menu_contextual, menu);  
}
```

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {  
    if (item.getItemId() == R.id.op2) {  
        Toast.makeText(this, "Se eligió la opción 2",  
                        Toast.LENGTH_SHORT).show();  
    }  
    return true;  
}
```

Probar en el emulador