

SEMINARIO DE LENGUAJES

OPCIÓN ANDROID

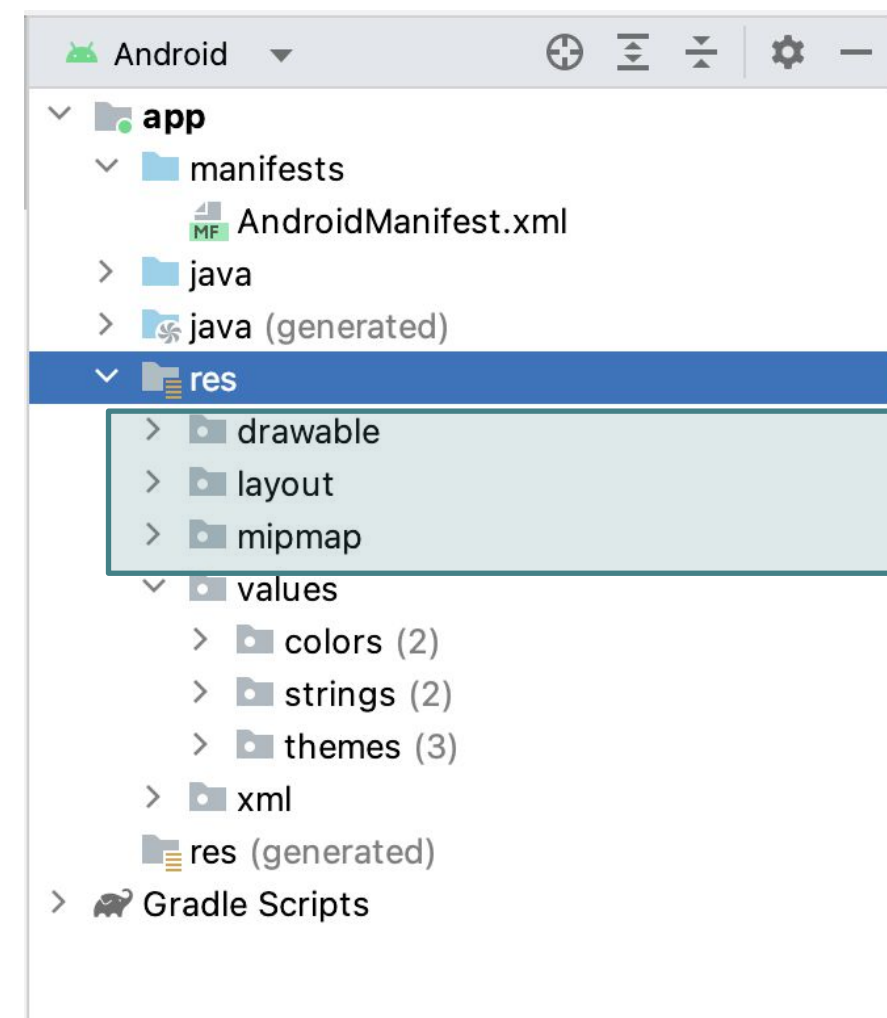


Recursos de archivo. Menús.

Esp. Fernández Sosa Juan Francisco

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

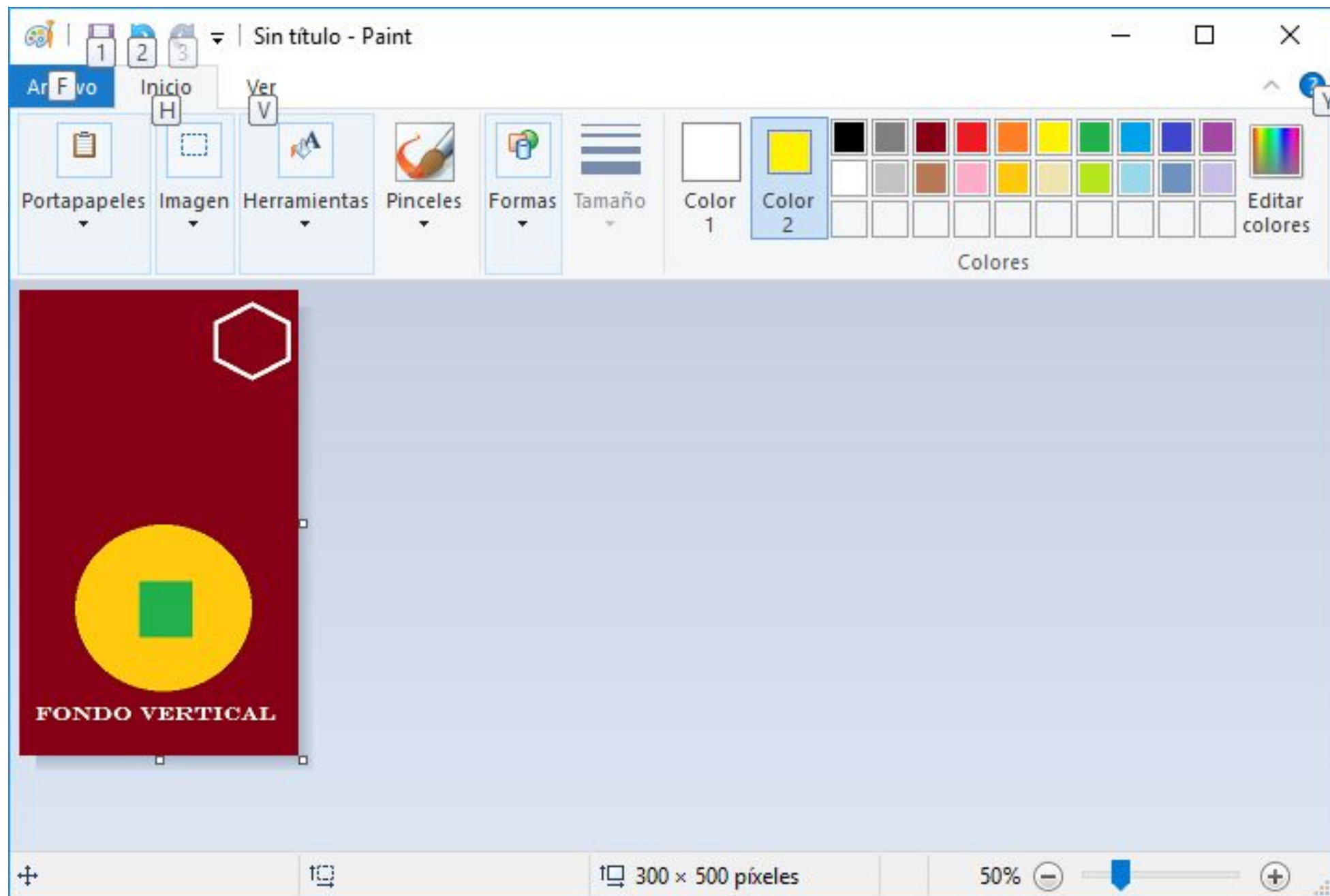
- Crear un nuevo proyecto **Android Studio** llamado **"RecursosDeArchivo"** basado en la siguiente **Empty View Activity**

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

</LinearLayout>
```

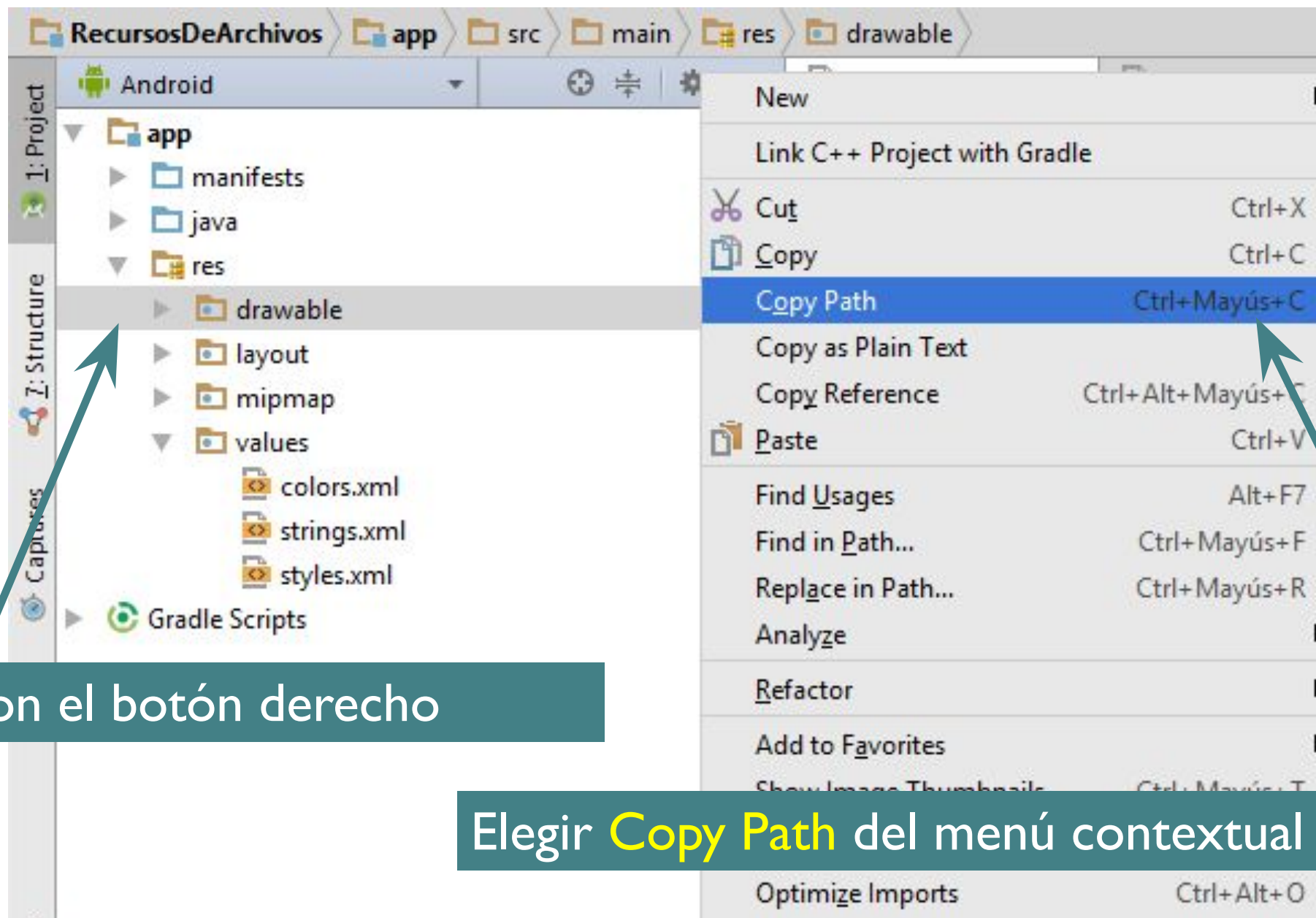
Actividad guiada

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



Click con el botón derecho

Elegir **Copy Path** del menú contextual

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"
```

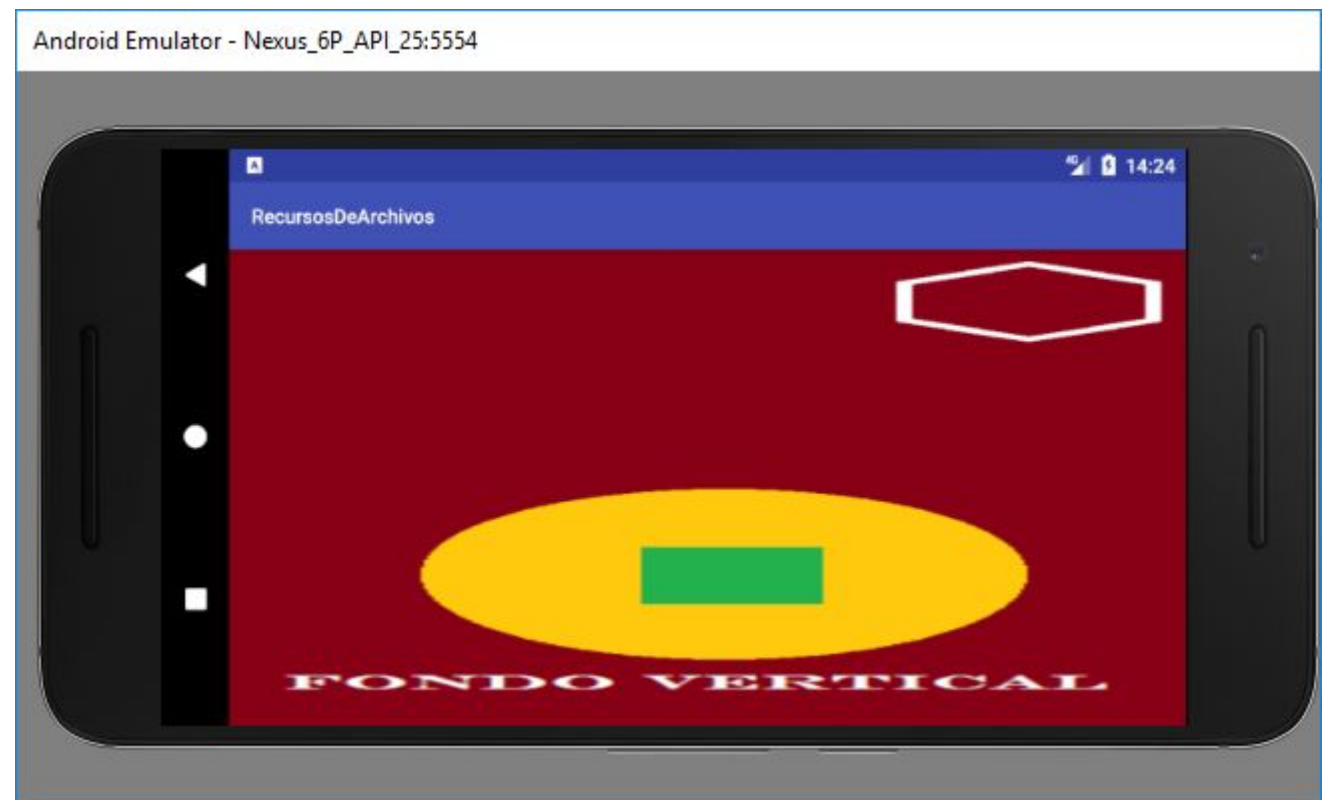
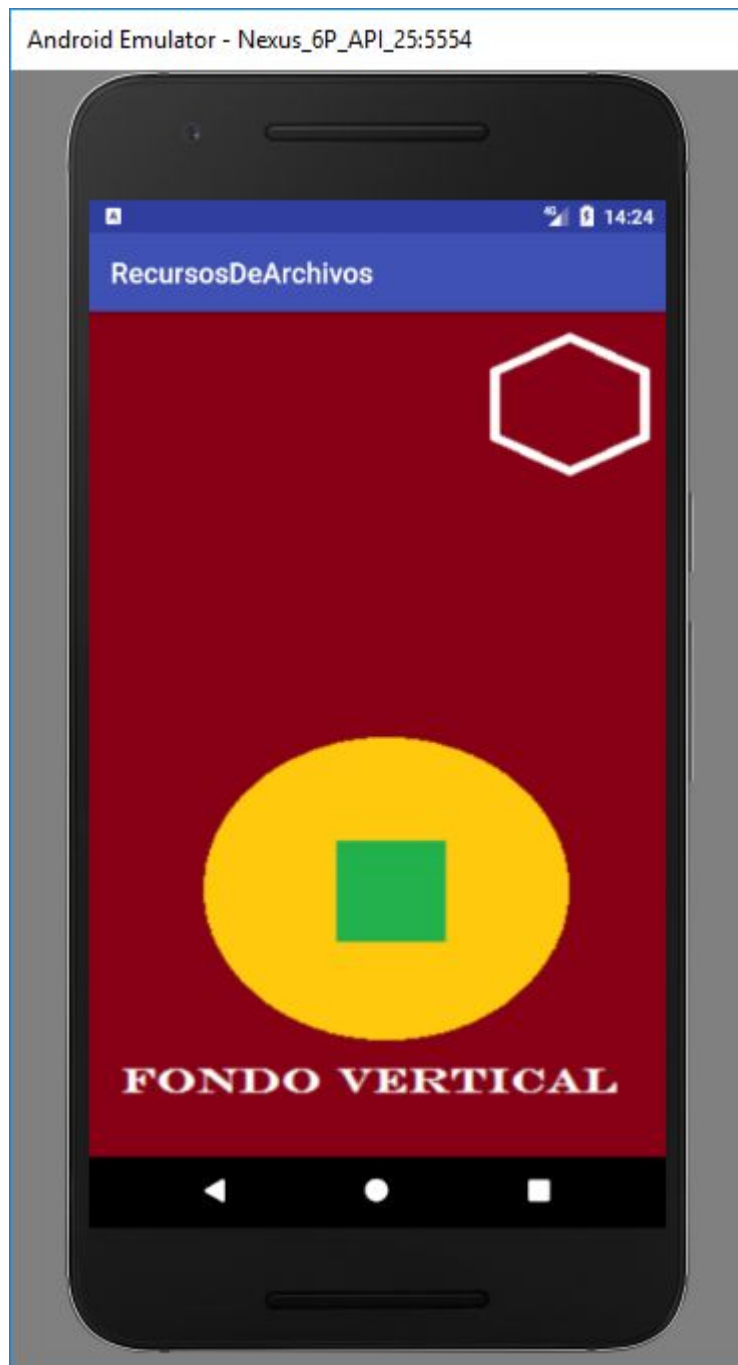
```
    android:background="@drawable/fondo"
```

```
>
```

```
</LinearLayout>
```

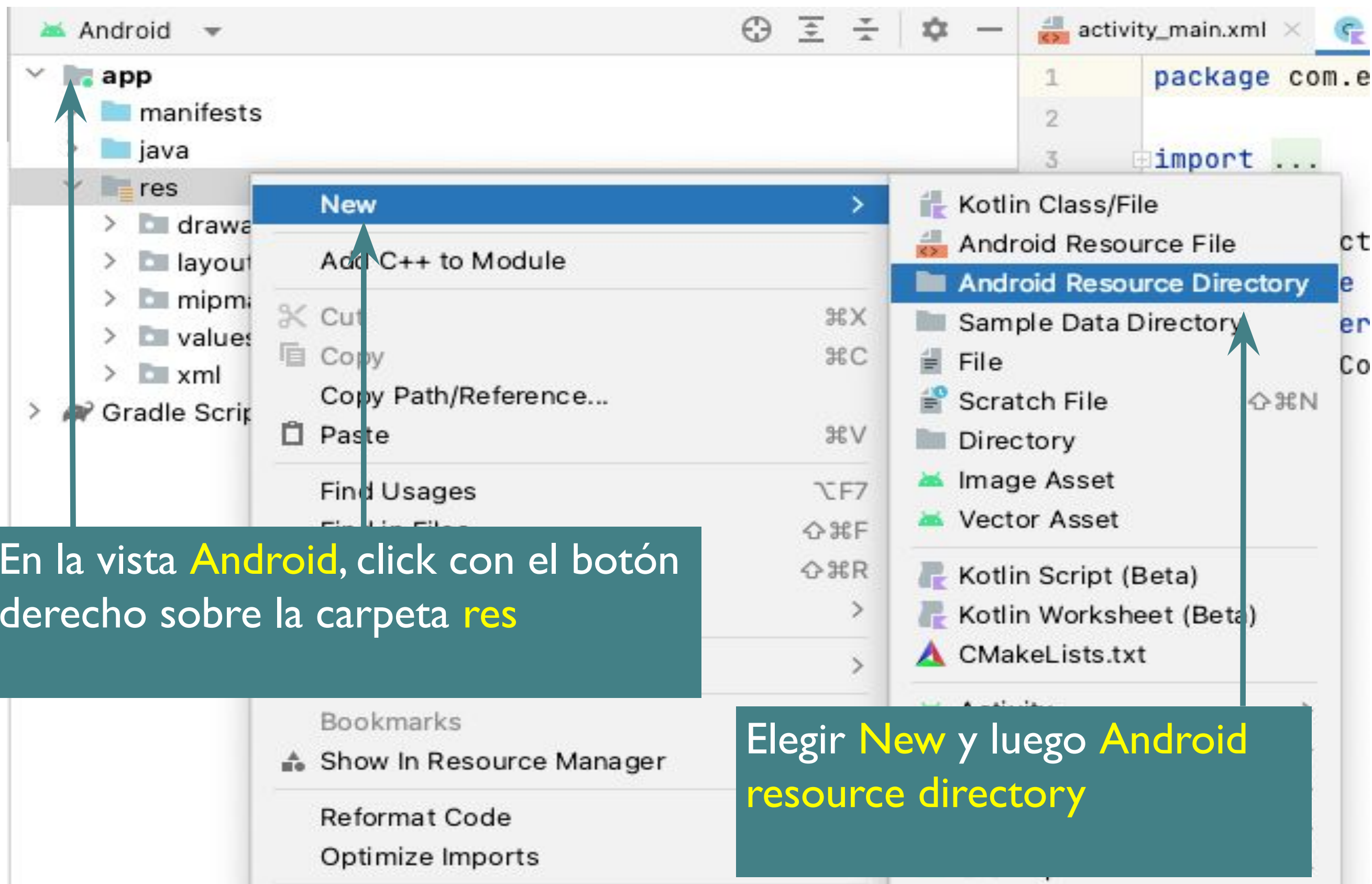

Actividad guiada

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



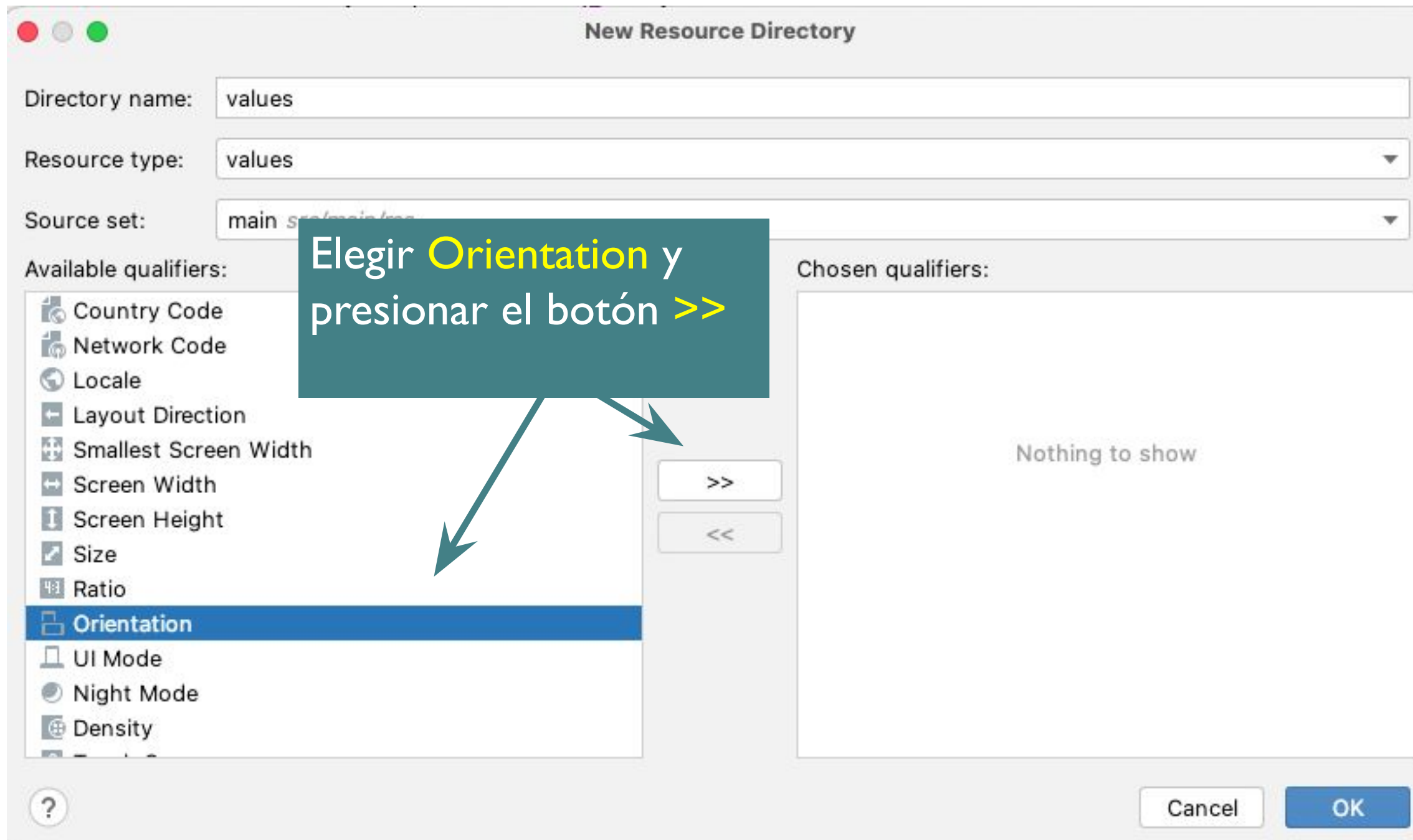
Actividad guiada

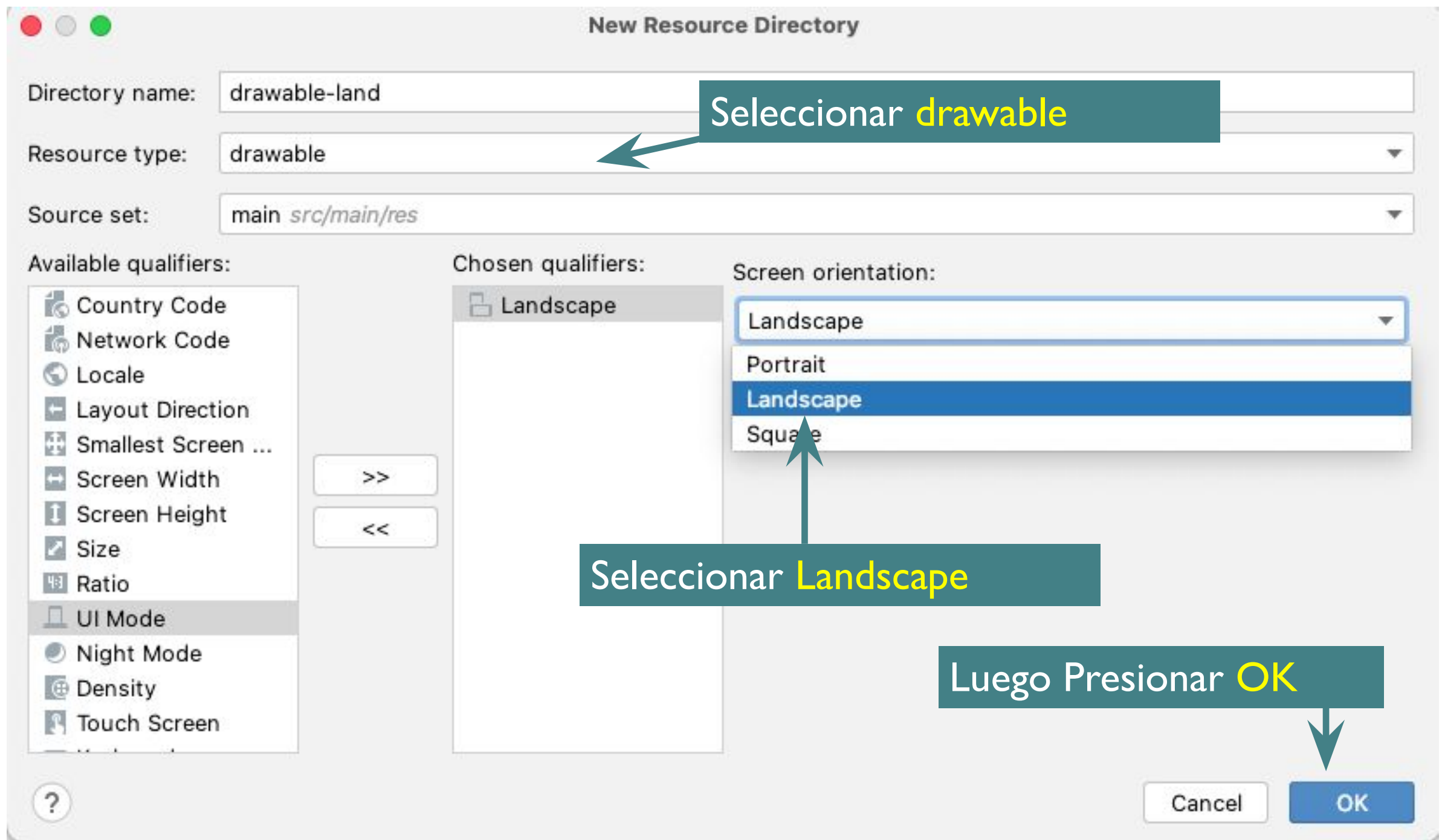
- Vamos a crear un recurso alternativo **drawable** para establecer otra imagen de fondo en la disposición horizontal (**landscape**)
- Para ello es necesario crear el directorio de recursos **res/drawble-land** para guardar en él una imagen que llamaremos también **fondo.png**
- El sufijo **-land** 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)



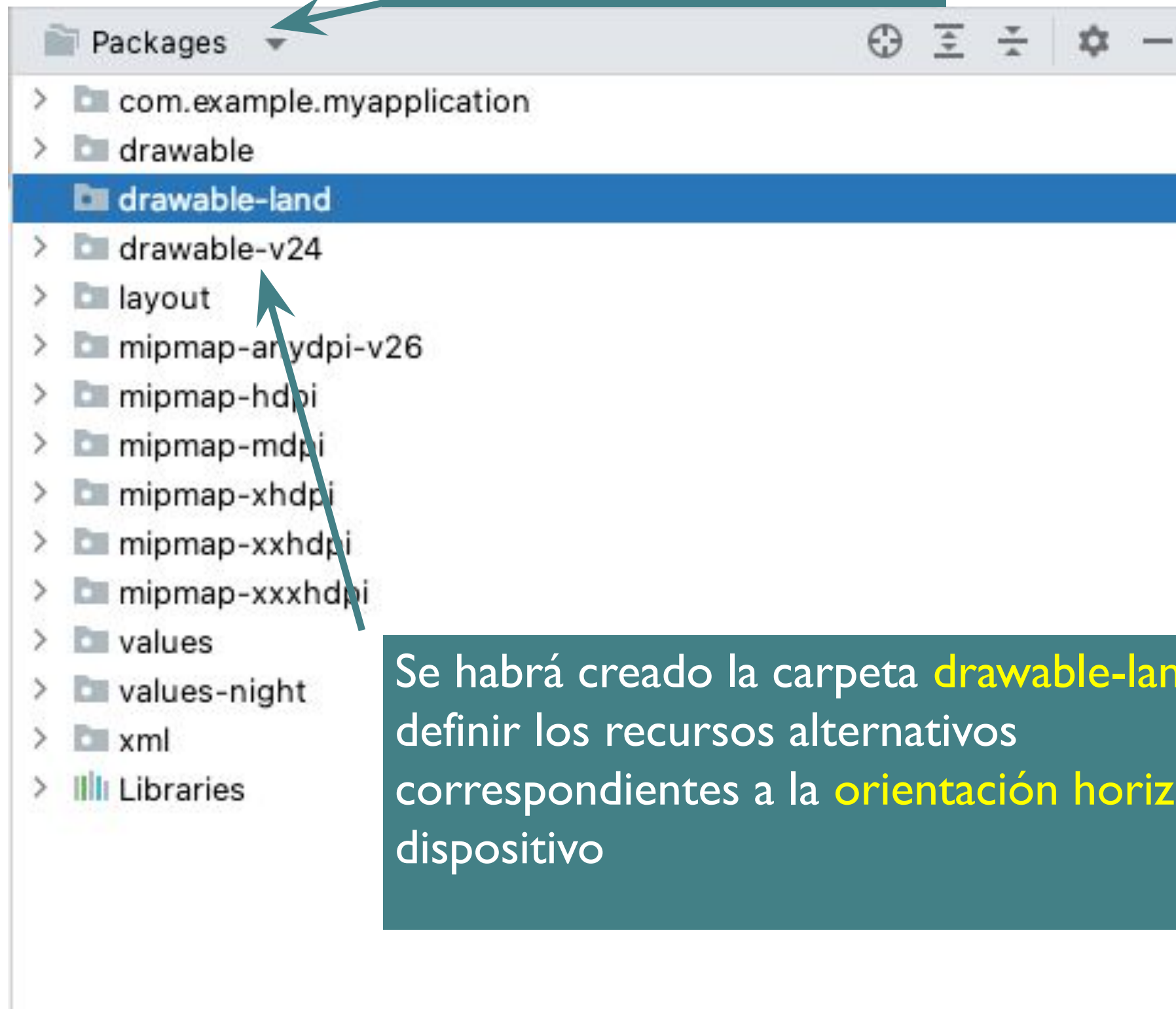
En la vista **Android**, click con el botón derecho sobre la carpeta **res**

Elegir **New** y luego **Android resource directory**



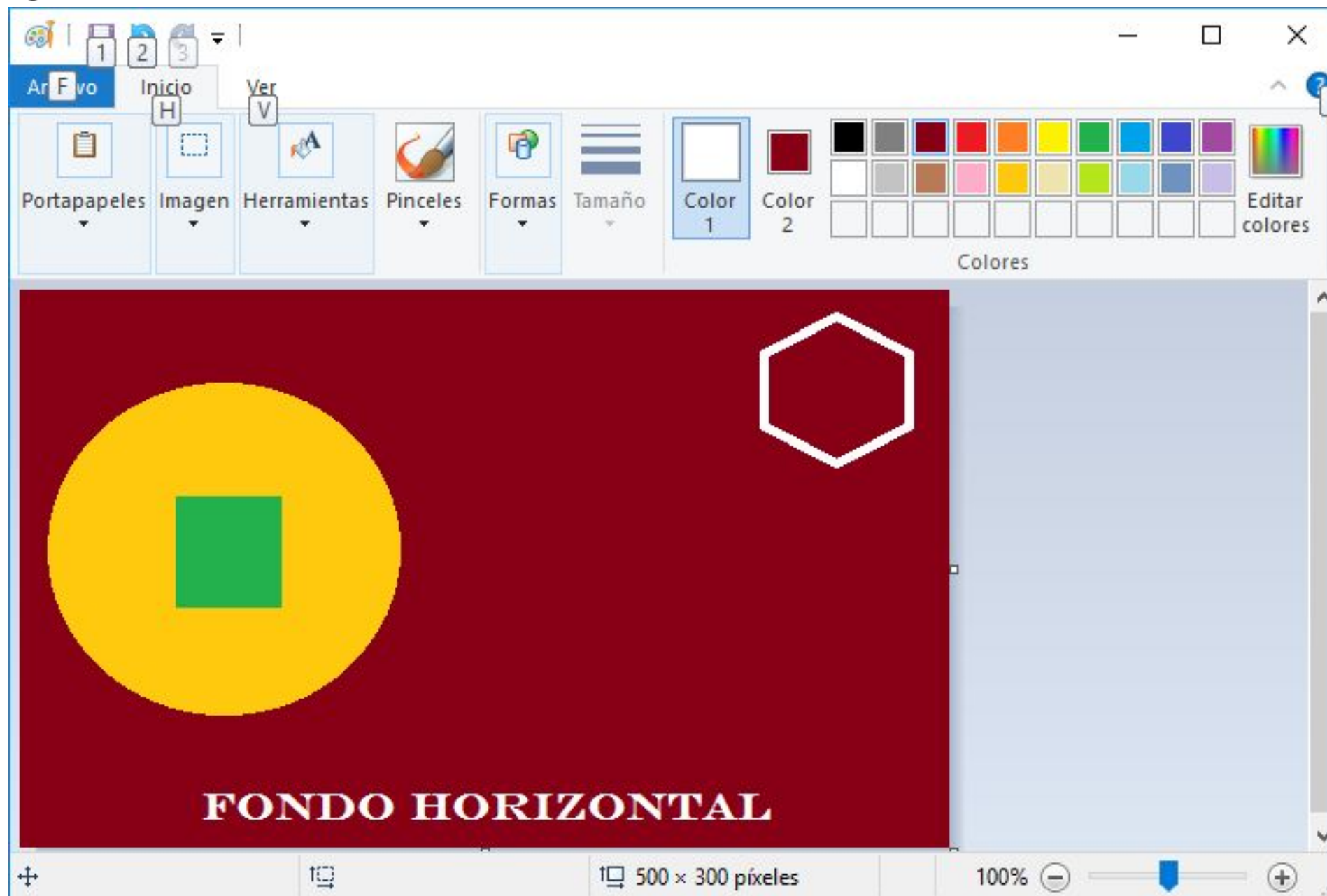


Vista Packages



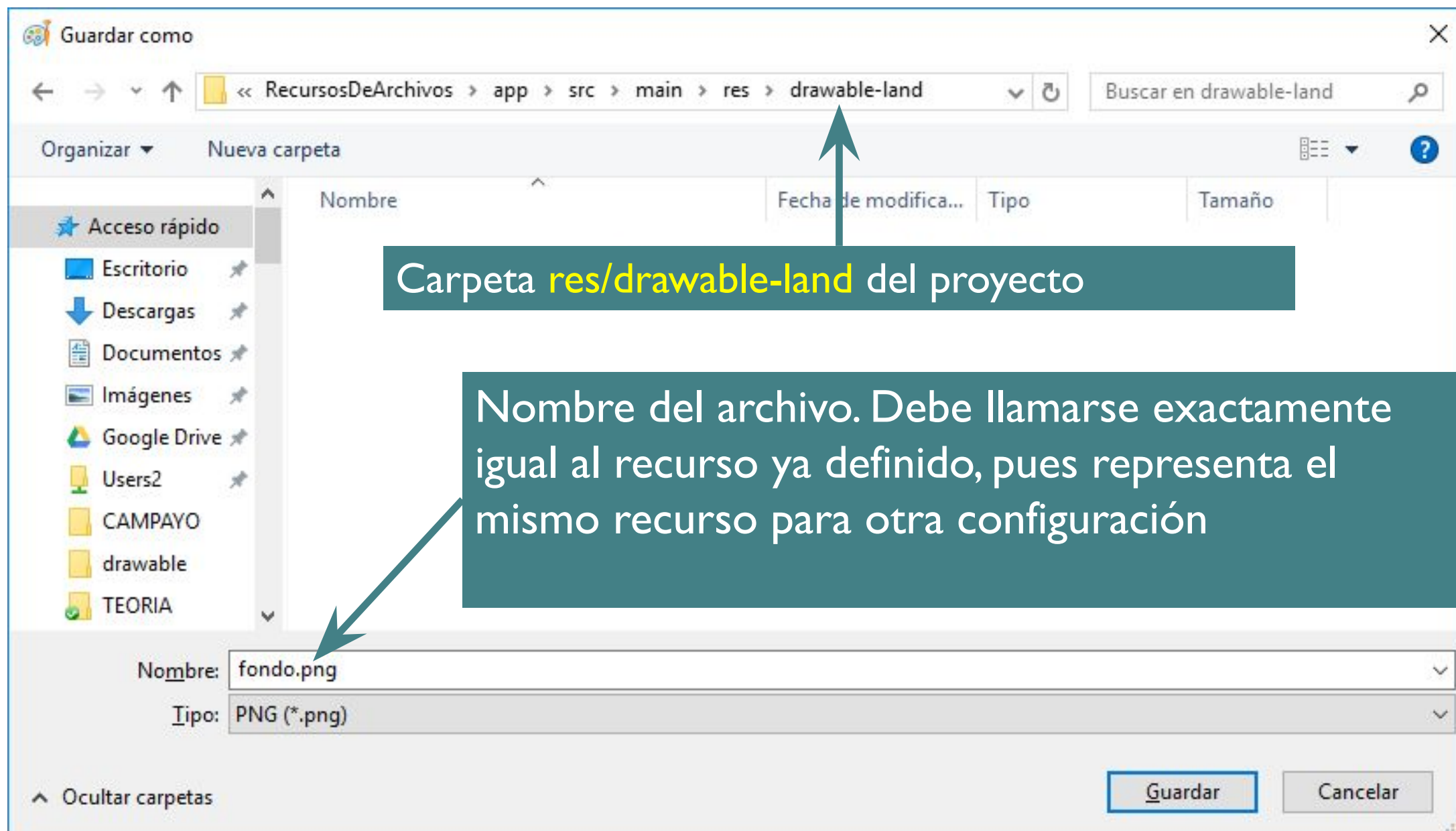
Actividad guiada

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



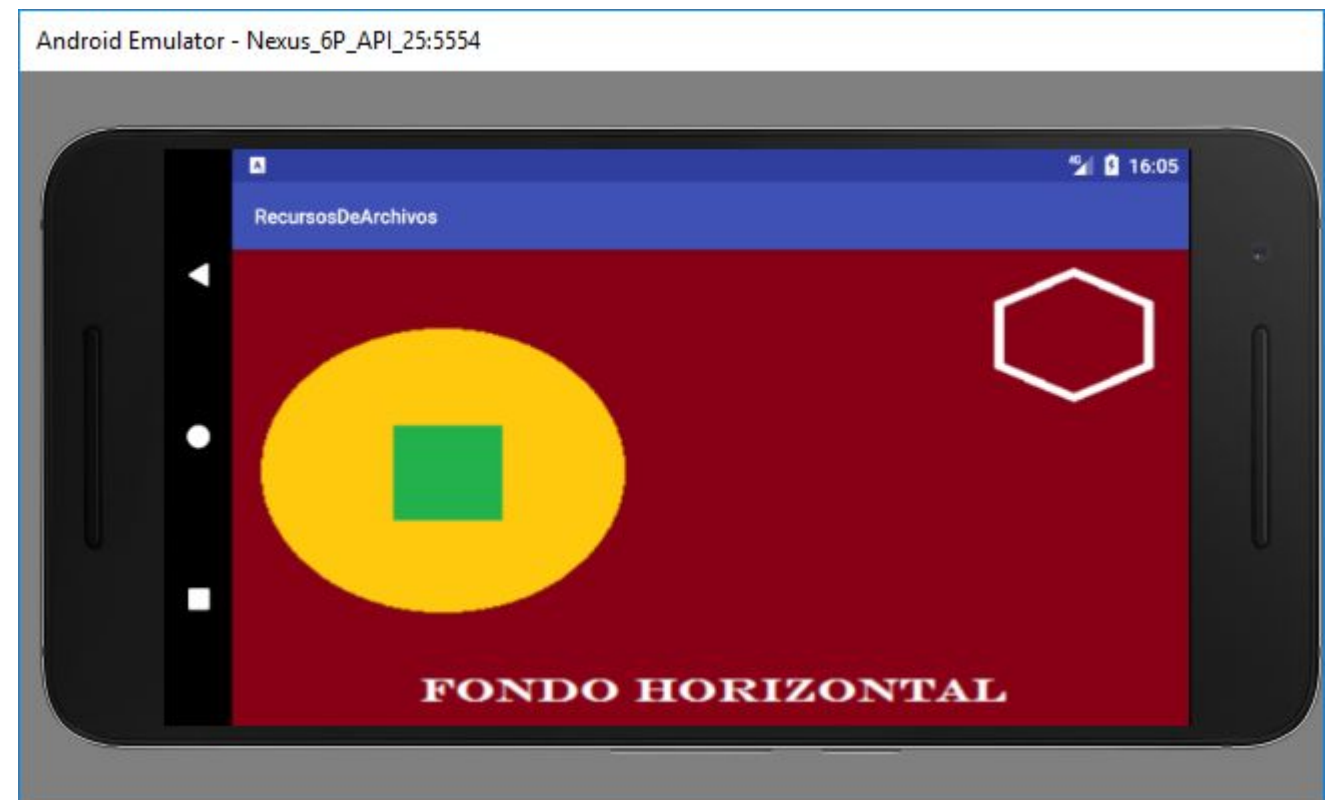
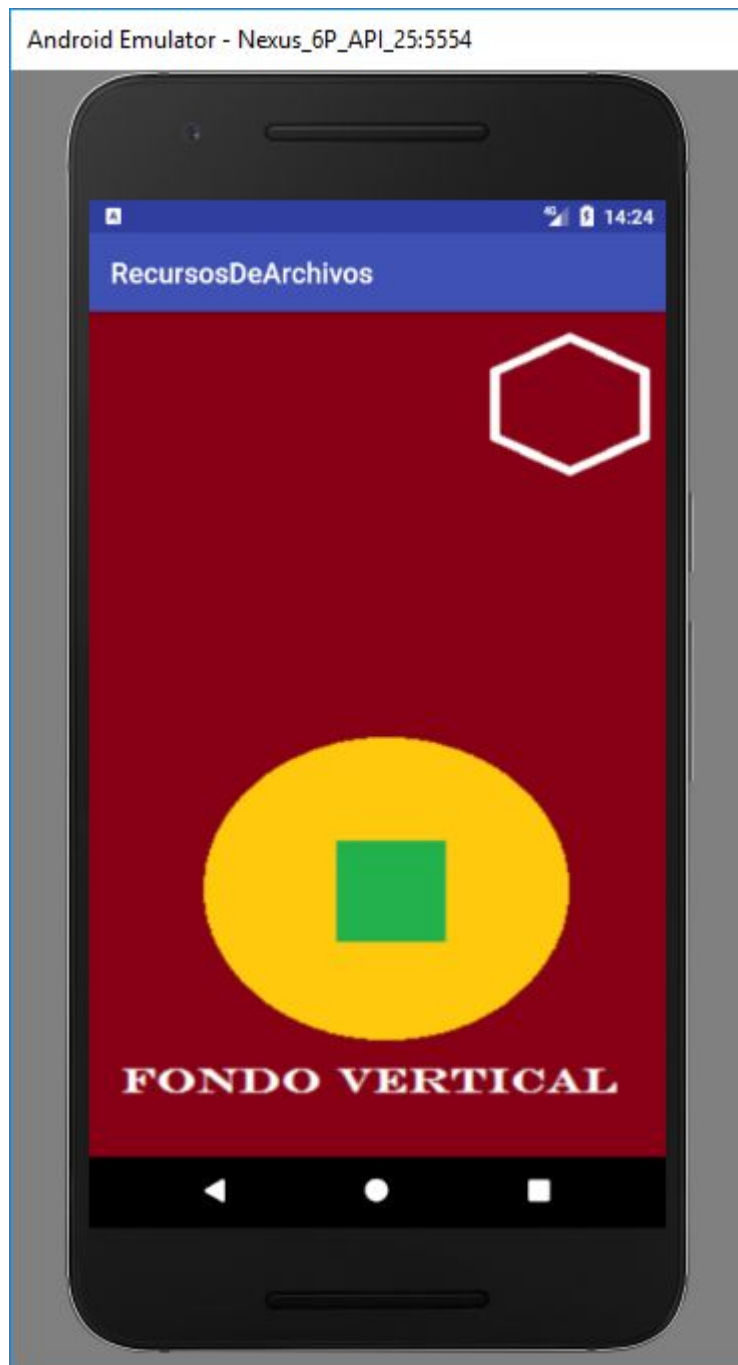
Actividad guiada

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

```
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>
```

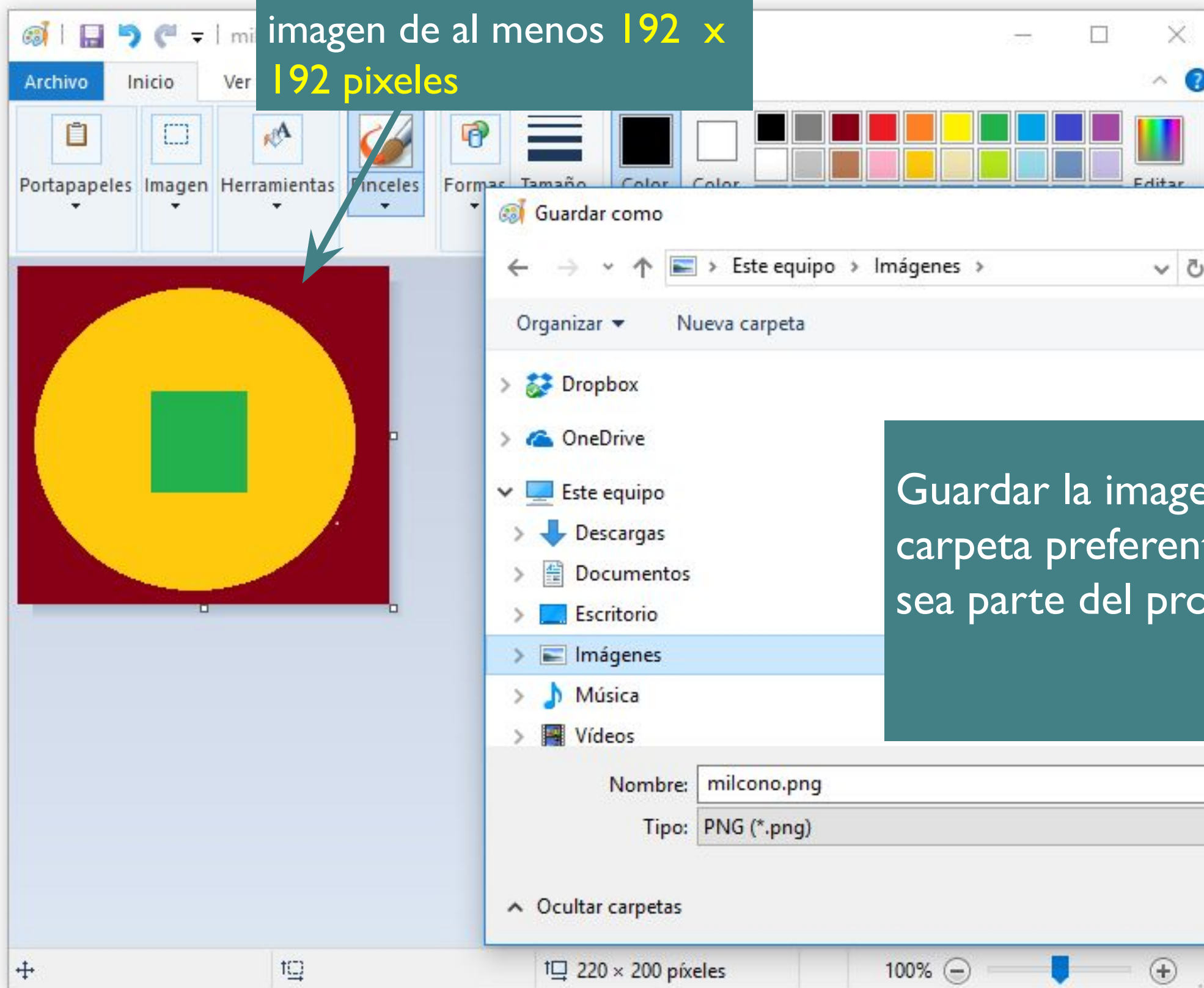
Los íconos lanzadores se establecen en el manifiesto de la aplicación

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

Tools VCS Window Help

Tasks & Contexts >


Generate JavaDoc...

Create Command-line Launcher...


XML Actions >


Markdown Converter >

JShell Console...


 Kotlin >


Cling >

 Groovy Console


 Device Manager

 SDK Manager

 Resource Manager

 Troubleshoot Device Connections

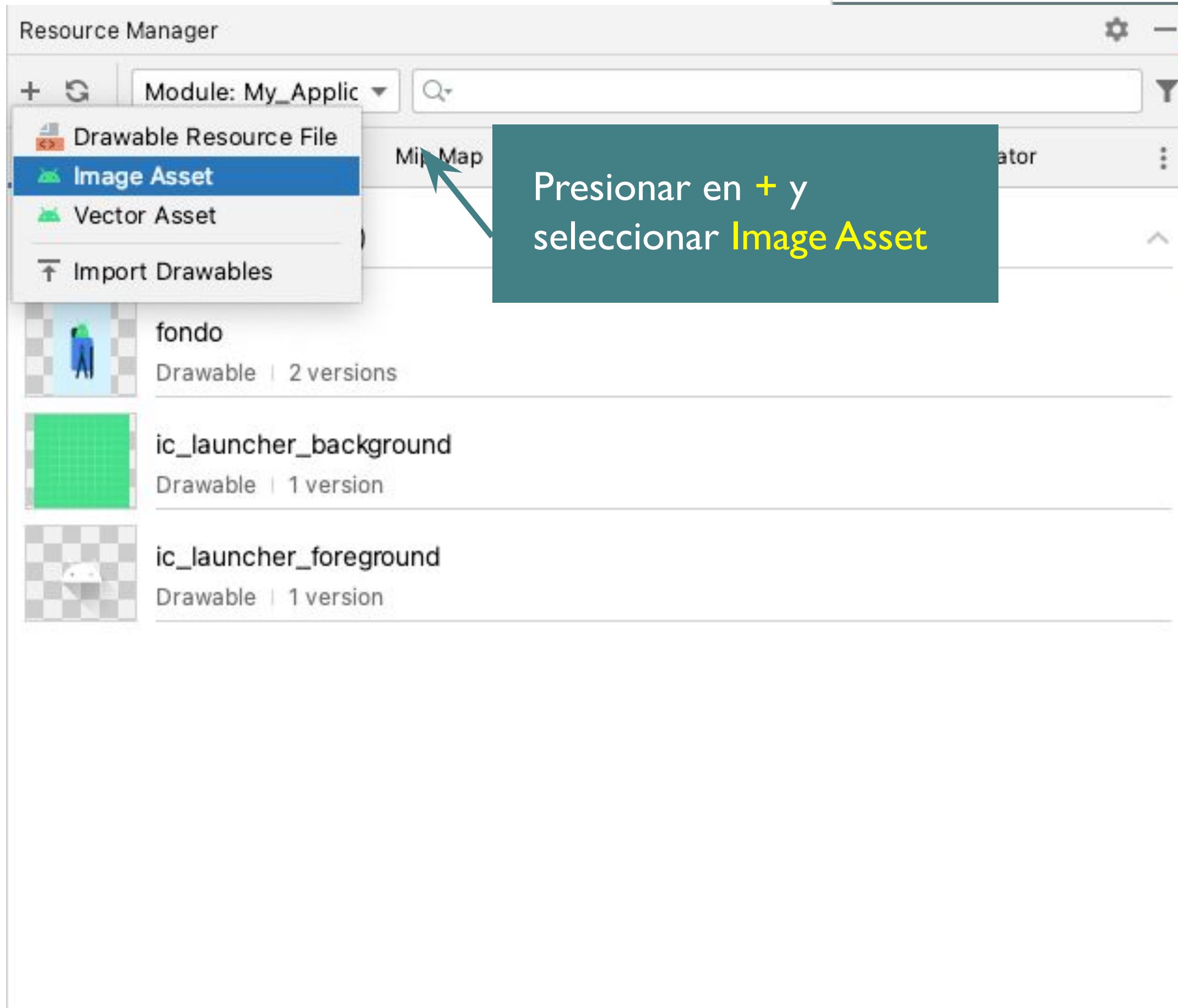
 App Links Assistant

 Firebase

 Layout Inspector

AGP Upgrade Assistant...

En Android Studio
seleccionar
Tools/Resource Manager

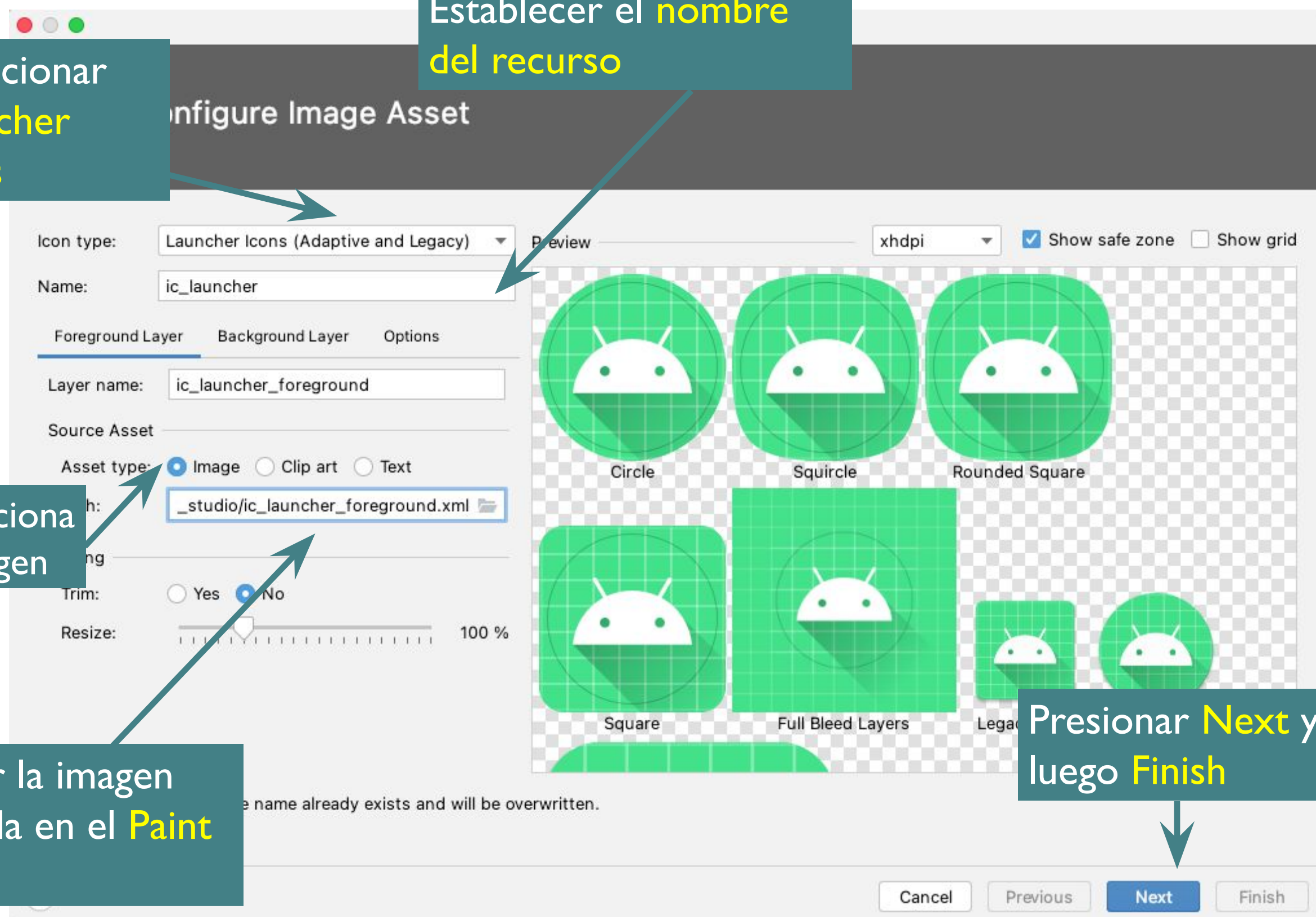


Seleccionar
Launcher
Icons

Establecer el **nombre**
del recurso

Seleccionar
Imagen

Elegir la imagen
creada en el **Paint**



Presionar **Next** y
luego **Finish**

The screenshot displays the Android Studio interface. On the left, the 'Resources' tab is active, showing a hierarchy of folders for different screen densities: **app**, **android**, **com.cursoandroid2017.recursosdearchivos**, **drawable**, **drawable-land**, **layout**, **mipmap-hdpi**, **mipmap-mdpi**, **mipmap-xhdpi**, **mipmap-xxhdpi**, and **mipmap-xxxhdpi**. Each density folder contains three launcher icons: **ic_launcher.png**, **ic_launcher2.png** (highlighted with a blue border), and **ic_launcher_round.png**. On the right, the **AndroidManifest.xml** file is open, showing the following XML code:

```
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:supportRtl="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 **</application>** tag on line 19.

Se crean los recursos alternativos para cada densidad

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

```
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_launcher2"
8         android:label="RecursosDeArchivos"
9         android:roundIcon="@mipmap/ic_launcher2"
10        android:supportRtl="true"
11        android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN">
15
16                 <category android:name="android.intent.category.LAUNCHER">
17
18             </intent-filter>
19         </activity>
20     </application>
```

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

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"
    >
```

activity_main.xml

Layout predeterminado de la
activity principal

```
<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"
    />
```

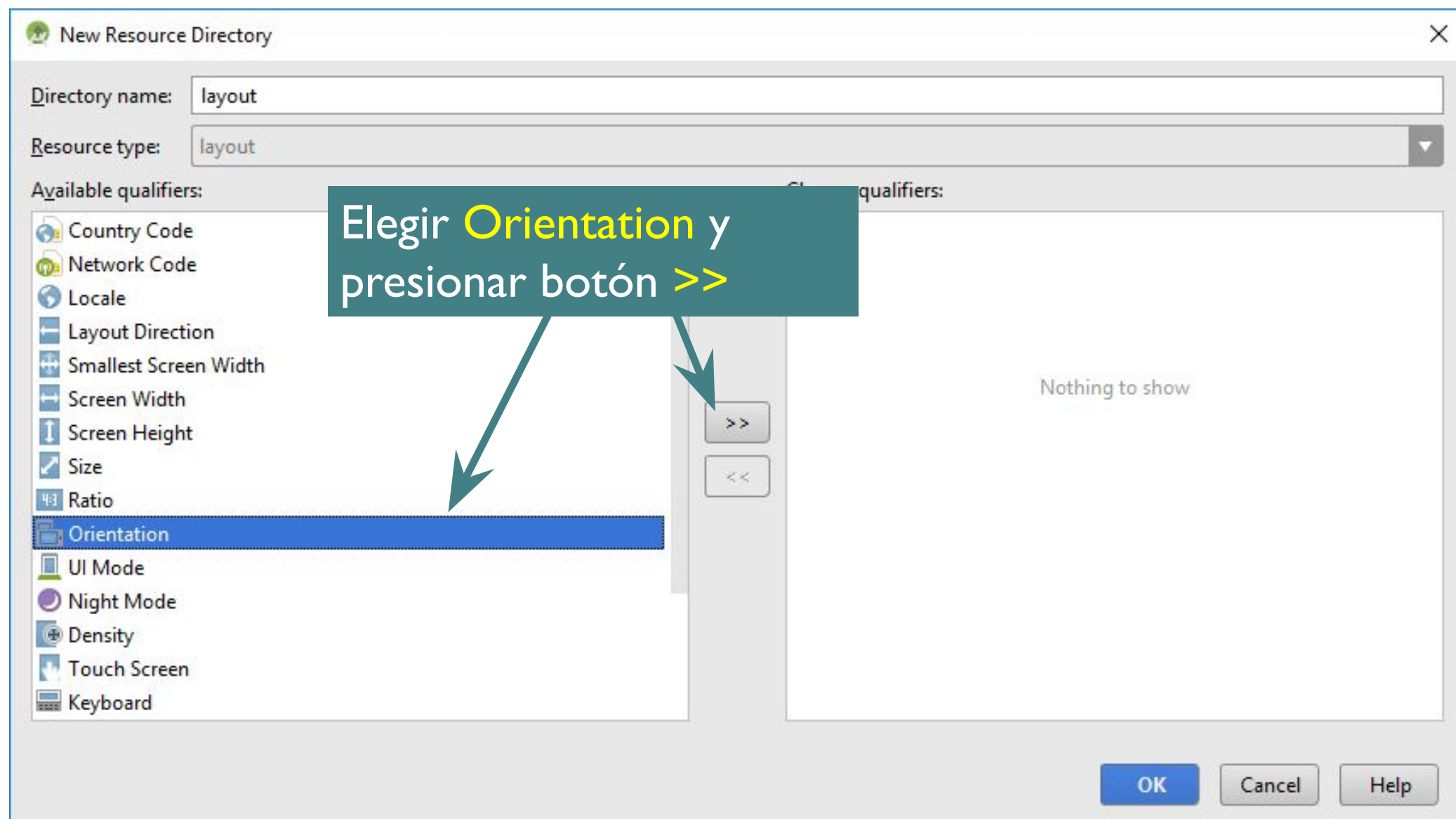
```
<Button
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Configuración"
    />
```

```
</LinearLayout>
```

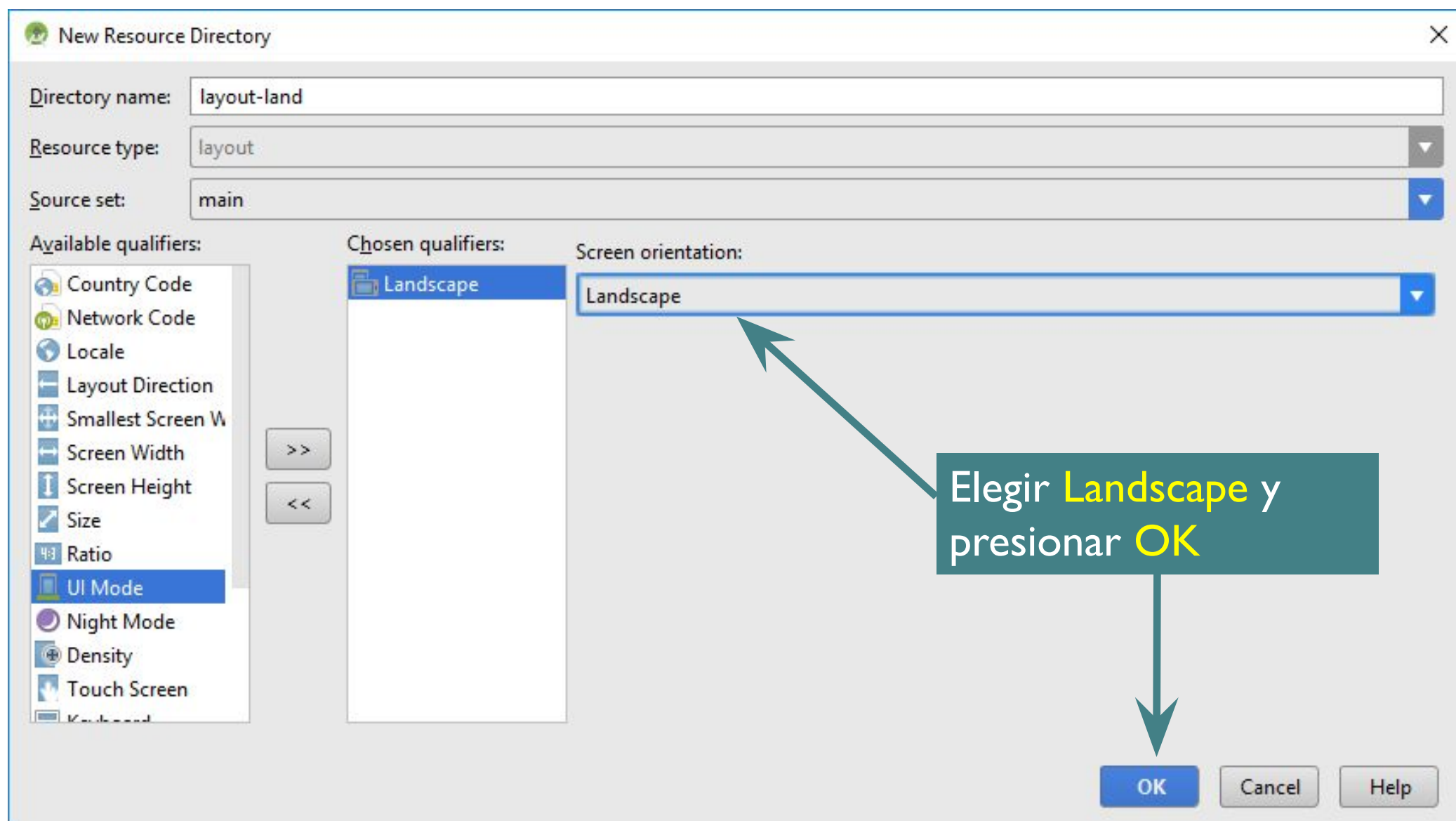
Actividad guiada - continuación

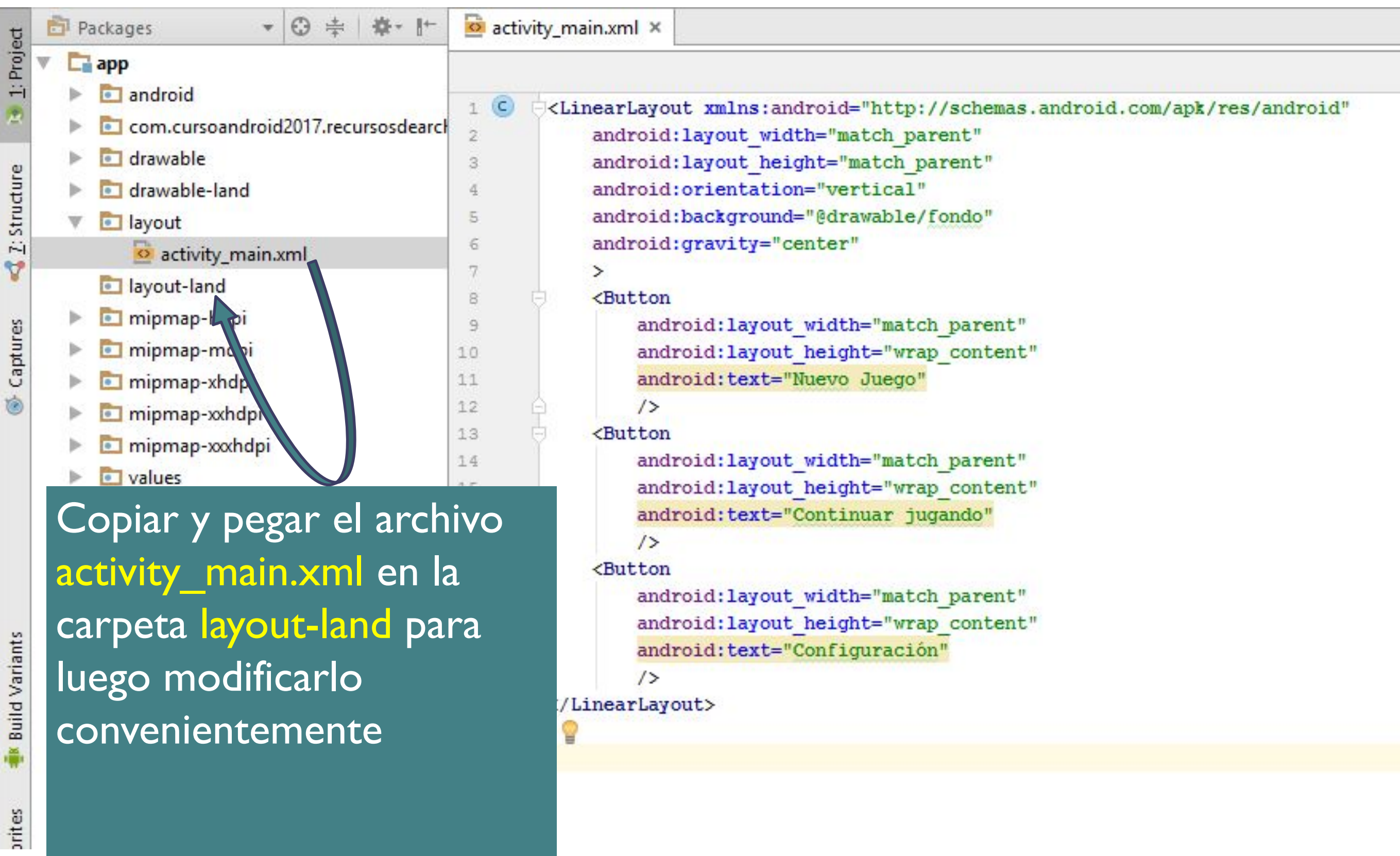
Crear un nuevo directorio de recursos **layout-land** para definir el **layout** de la **activity principal** en el caso de la **disposición horizontal** del dispositivo



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





1: Project
I: Structure
Captures
Build Variants

Packages

- app
 - android
 - com.cursoandroid2017.recursosdearch
 - drawable
 - drawable-land
 - layout
 - activity_main.xml
 - layout-land
 - mipmap-ldpi
 - mipmap-mdpi
 - mipmap-xhdpi
 - mipmap-xxhdpi
 - mipmap-xxxhdpi
 - values

activity_main.xml x

```
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>

```

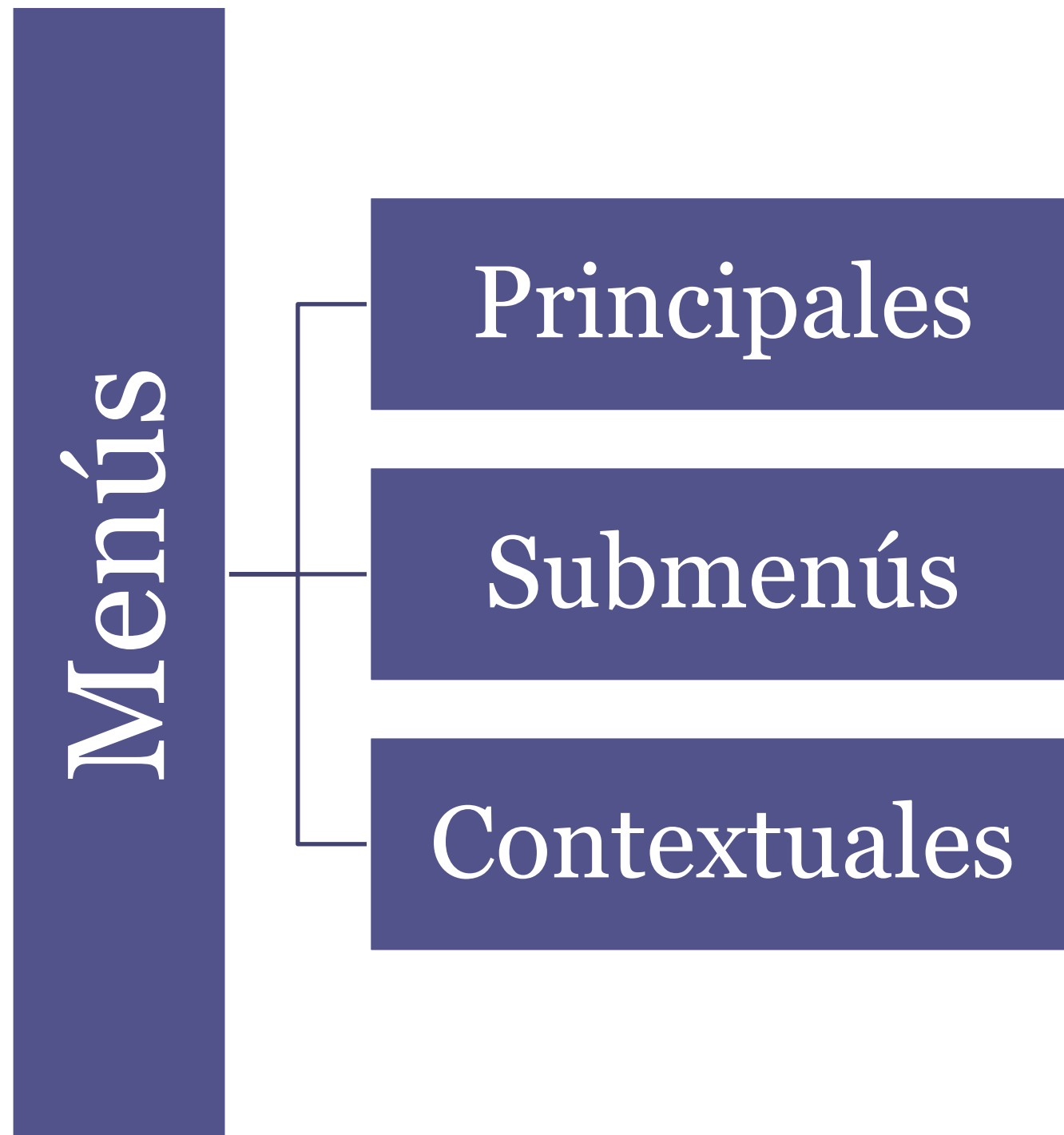
activity_main.xml en la carpeta layout-land
Recurso de layout alternativo para la orientación landscape

Ejecutar en el emulador



Menús

Tres tipos de Menús



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

Actividad guiada - creación de menú principal

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



Creación de un menú principal de una *activity*

1. Creación del menú

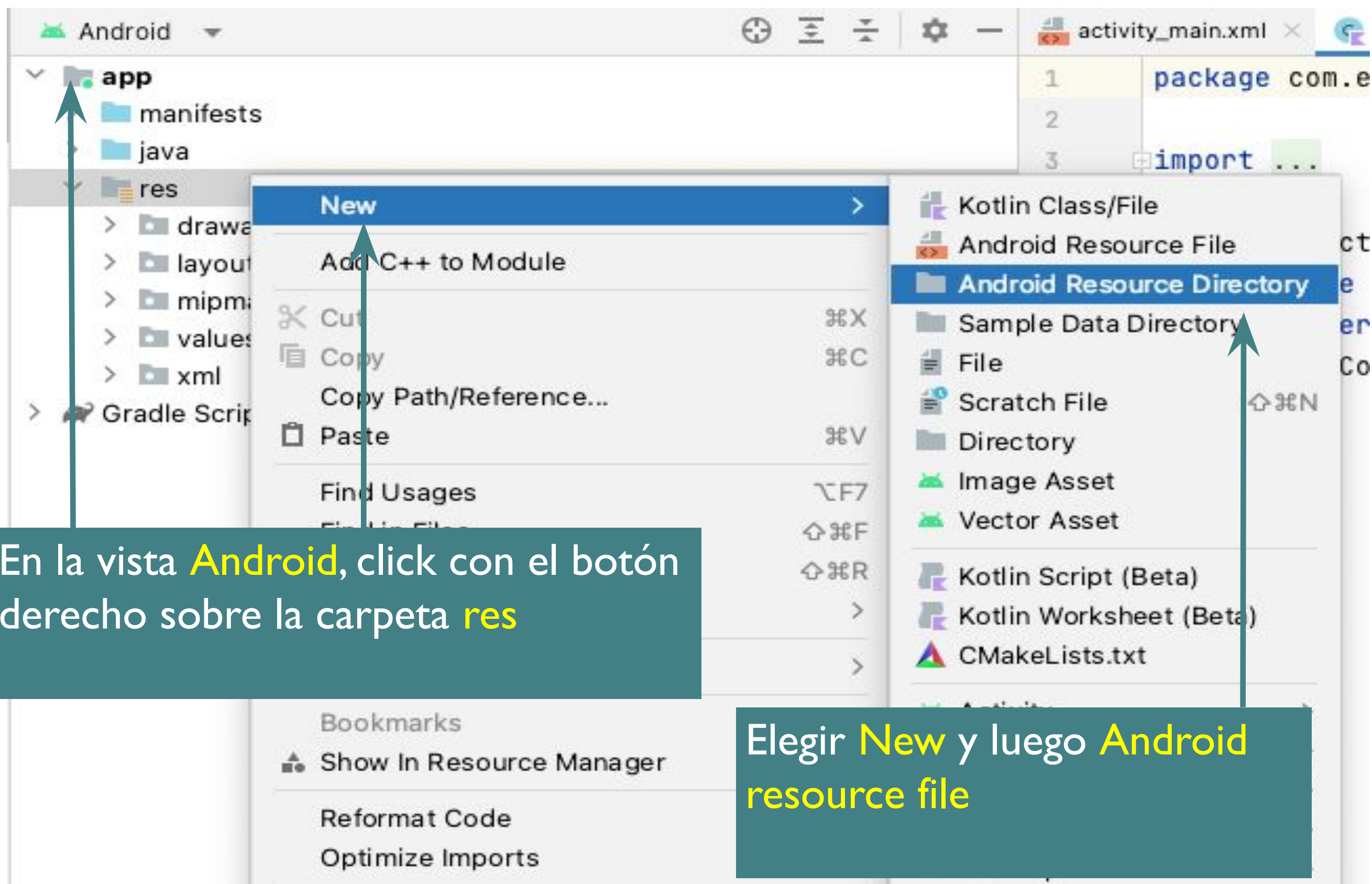
- a) Crear un nuevo archivo de recursos XML de tipo menú
- b) Configurar las opciones de menú: Id y título

2. Agregar una Toolbar en el layout de la actividad

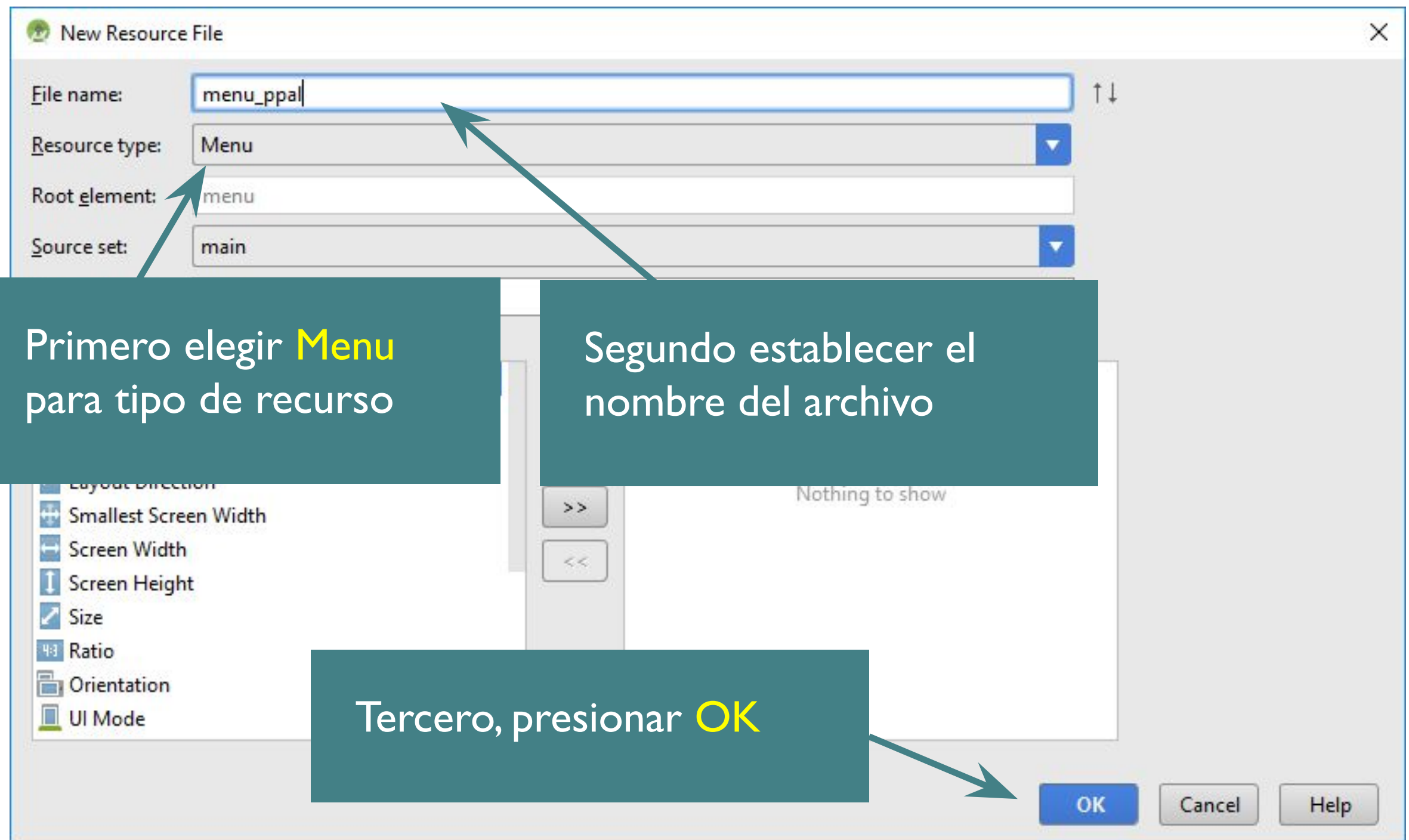
- a) Asegurarse de utilizar **androidx.appcompat.widget.Toolbar**
- b) Setear la Toolbar como ActionBar en el código

3. Activación del menú

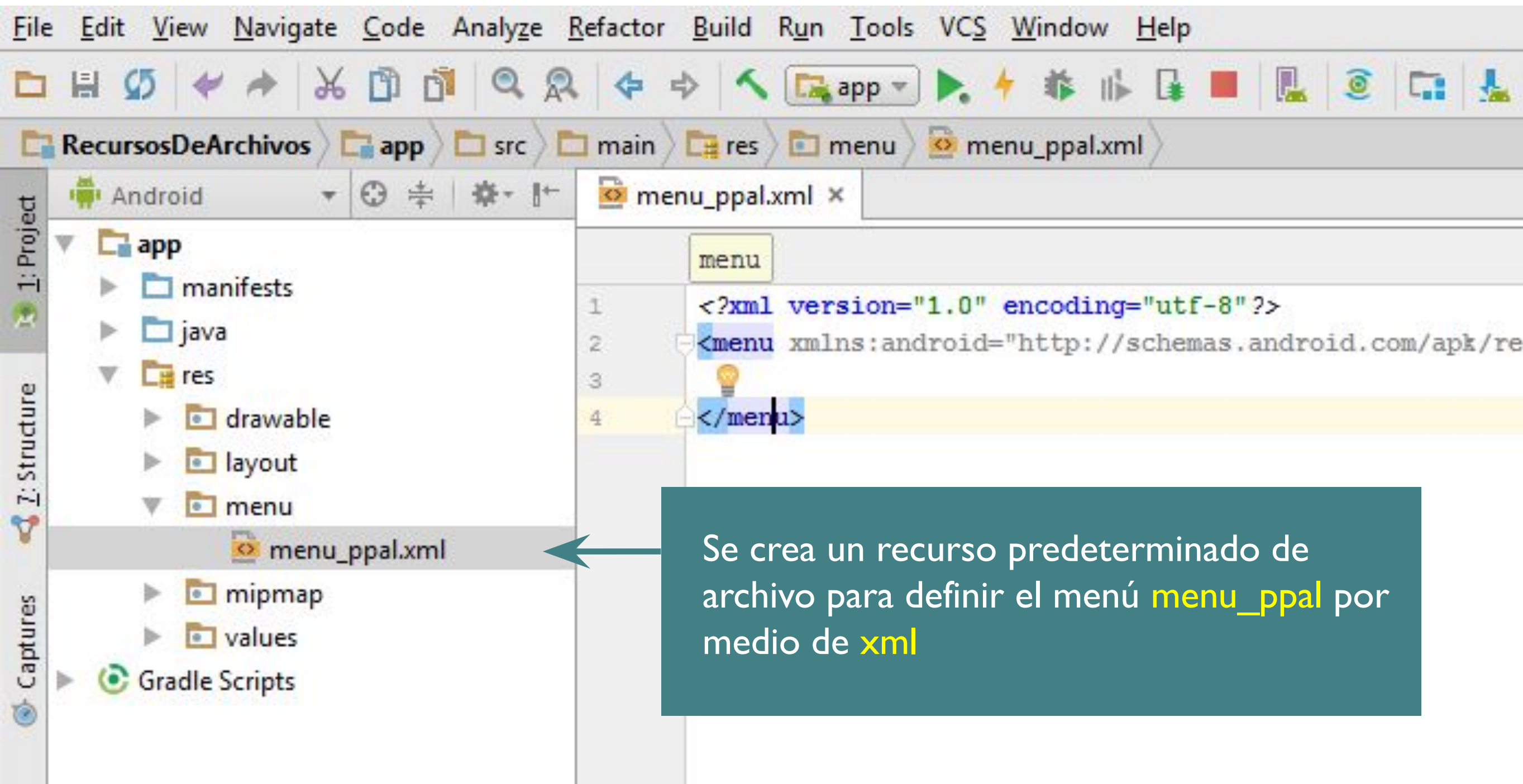
- a) Agregar el menú en la actividad usando MenuProvider en el método onCreate()
- b) Configurar la acción según las opciones elegidas



Actividad guiada - creación de menú principal



Actividad guiada - creación de menú principal



The screenshot shows an IDE interface with the following components:

- Top Menu Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Toolbar:** Contains icons for file operations (new, open, save, delete), navigation (back, forward), and development (run, debug, test).
- 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'. The code is as follows:

```
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 following 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

Agregar Toolbar en la activity:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical">
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:background="?attr/colorPrimary"
        app:titleTextColor="@color/white"
        app:title="@string/app_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

Actividad guiada - creación de menú principal

Agregar Toolbar en la activity:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res/app"
    android:orientation="vertical">
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:background="?attr/colorPrimary"
        app:titleTextColor="@color/white"
        app:title="@string/app_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
```

Usar el widget Toolbar de la biblioteca **androidX**

Coloca como color de fondo el color primario definido en theme de la app

El título y su color los define en el namespace **app**

Actividad guiada - creación de menú principal

Establecer el Toolbar como un **ActionBar**, para poder asignarle luego un MenuProvider

```
import androidx.appcompat.widget.Toolbar
...
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main) ;

    val toolbar = findViewById<Toolbar>(R.id.toolbar)
    setSupportActionBar(toolbar)
    ...
```

Actividad guiada - creación de menú principal

Agregar el menú a la actividad usando la clase

MenuHost y MenuProvider

Convierte el archivo XML en el menú visible para el usuario

```
val menuHost: MenuHost = this
menuHost.addMenuProvider(object : MenuProvider {
    override fun onCreateMenu(menu: Menu, menuInflater: MenuInflater) {
        menuInflater.inflate(R.menu.menu_ppal, menu)
    }

    override fun onMenuItemSelected(menuItem: MenuItem): Boolean {
        return when (menuItem.itemId) {
            R.id.menuSalir -> {
                finish()
                true
            }
            else -> false
        }
    }
}, this);
```

Establecer la lógica de las opciones

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main);
```

```
        val toolbar = findViewById<Toolbar>(R.id.toolbar)  
        setSupportActionBar(toolbar)
```

```
        val menuHost: MenuHost = this  
        menuHost.addMenuProvider(object : MenuProvider {  
            override fun onCreateMenu(menu: Menu, menuInflater: MenuInflater) {  
                menuInflater.inflate(R.menu.menu_ppal, menu)  
            }  
  
            override fun onMenuItemSelected(menuItem: MenuItem): Boolean {  
                return when (menuItem.itemId) {  
                    R.id.menuSalir -> {  
                        finish()  
                        true  
                    }  
                    else -> false  
                }  
            }  
        }, this);
```

Probar en
el
emulador

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

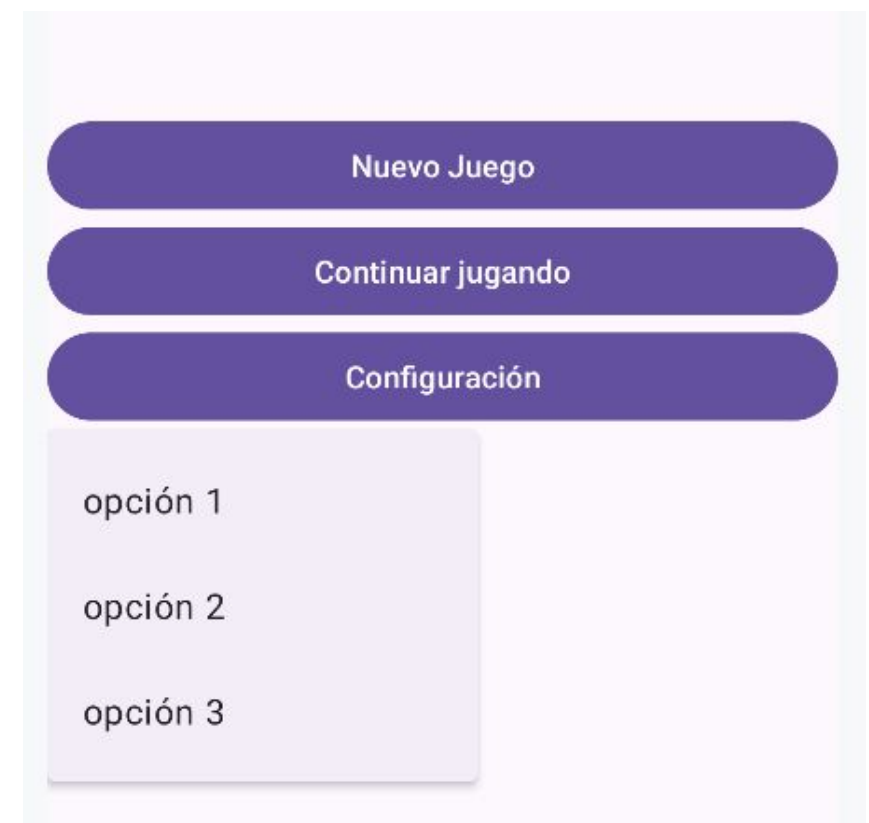
En la **ActionBar** las opciones pueden visualizarse por medio de íconos

```
<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 PopUp

- Un PopupMenu es un pequeño menú contextual que se muestra junto a un botón o vista específica cuando el usuario toca esa vista.
- Presenta una lista de opciones en forma de ventana flotante y desaparece automáticamente cuando el usuario selecciona una opción o toca afuera.
- 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.



Actividad guiada - Menú popup

Vamos a definir un menú tipo PopUp que mostraremos al clickear el boton con ID "btnOpciones"

Agregue el recurso de menú **menu_popup.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 popup

- Agregar este código el método `onCreate()` de `MainActivity`

```
val boton = findViewById<Button>(R.id.btnOpciones)
boton.setOnClickListener {
    val popup = PopupMenu(this, boton)
    popup.menuInflater.inflate(R.menu.menu_popup,
popup.menu)

    popup.setOnMenuItemClickListener { item ->
        when (item.itemId) {
            R.id.op1 -> { /* acción 1 */ true }
            R.id.op2 -> { /* acción 2 */ true }
            R.id.op3 -> { /* acción 3 */ true }
            else -> false
        }
    }

    popup.show()
}
```

Actividad guiada - Menú popup

- Agregar este código el método `onOptionsItemSelected()` en `MainActivity`

```

val boton = findViewById<Button>(R.id.opciones)
boton.setOnClickListener {
    val popup = PopupMenu(this, boton)
    popup.menuInflater.inflate(R.menu.menu_popup,
        popup.menu)

    popup.setOnMenuItemClickListener { item ->
        when (item.itemId) {
            R.id.op1 -> { /* acción 1 */ true }
            R.id.op2 -> { /* acción 2 */ true }
            R.id.op3 -> { /* acción 3 */ true }
            else -> false
        }
    }

    popup.show()
}
}

```

El menú se mostrará al lado de ese botón.

Se convierte el archivo XML `menu_popup.xml` en un objeto `Menu` visible en pantalla.

Se define qué hacer cuando el usuario toca una de las opciones del menú.