SEMINARIO DE LENGUAJES OPCIÓN ANDROID

Recursos.

Esp. Fernández Sosa Juan Francisco

Recursos

En este curso ya hemos estado trabajando con algunos tipos de recursos

P: ¿ Con Cuáles ?

R: Ids y Layout

Hemos utilizado la clase R para acceder a ellos a través de código Kotlin

Recursos

Por Ejemplo en:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main);
    val texto: TextView = findViewById(R.id.texto1);
}
```

Recurso de Layout

Recurso de Id

Recursos

¿Por qué es bueno utilizar recursos?

- Porque permite externalizar aspectos de la aplicación fuera del código y mantenerlos de forma independiente.
- Externalizar los recursos también permite proporcionar recursos alternativos que admiten configuraciones específicas de los dispositivos, como idiomas o tamaños de pantalla distintos.

Actividad guiada

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

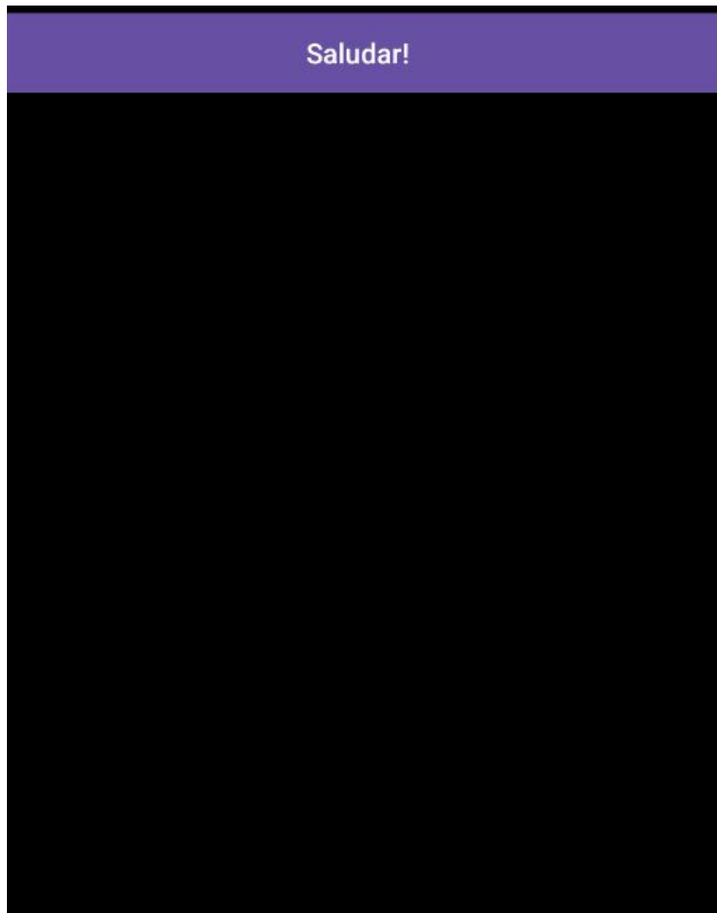
```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:background="@color/black">
  <Button
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="Saludar!"
     />
</LinearLayout>
```

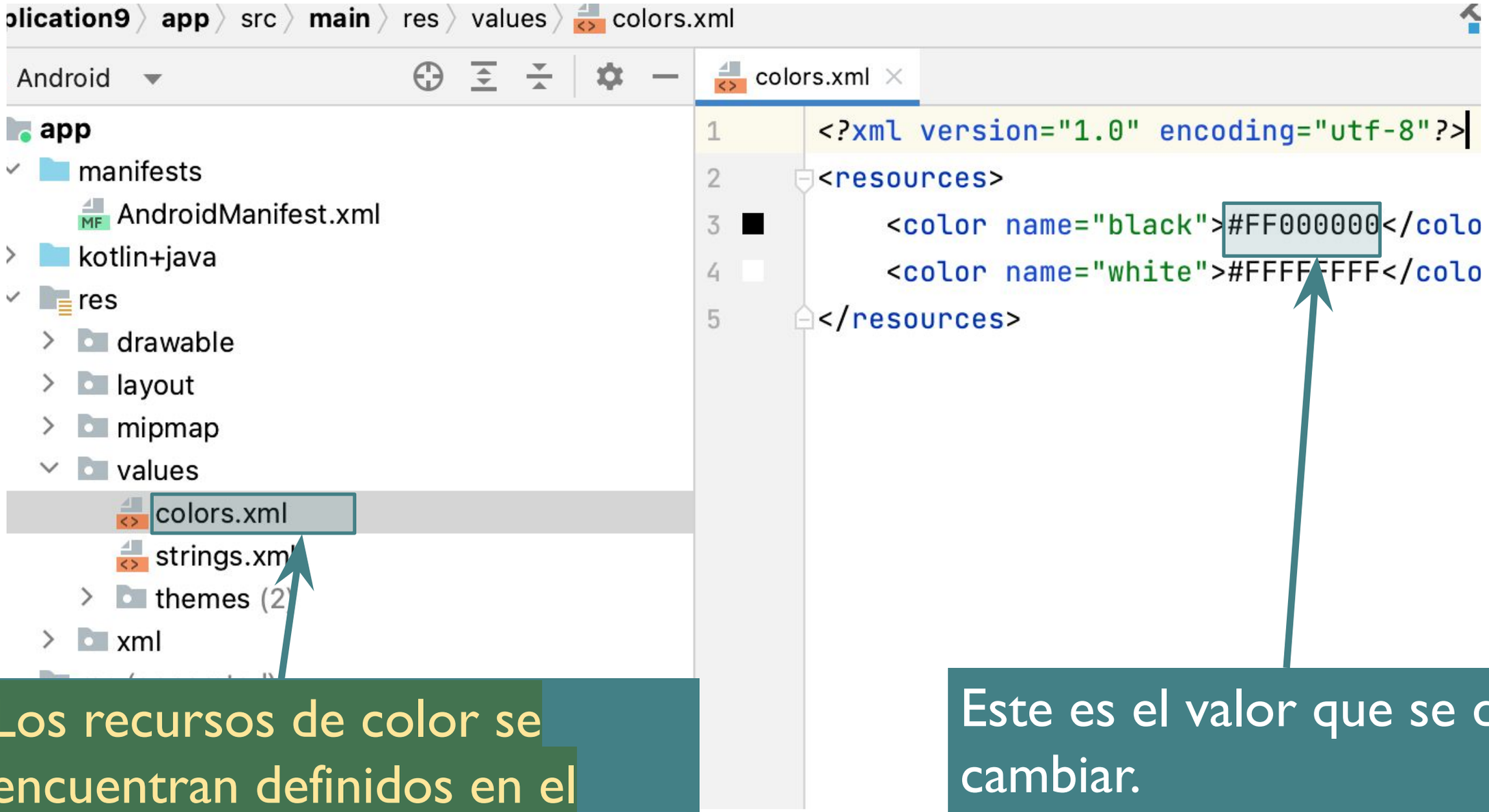
Actividad guiada

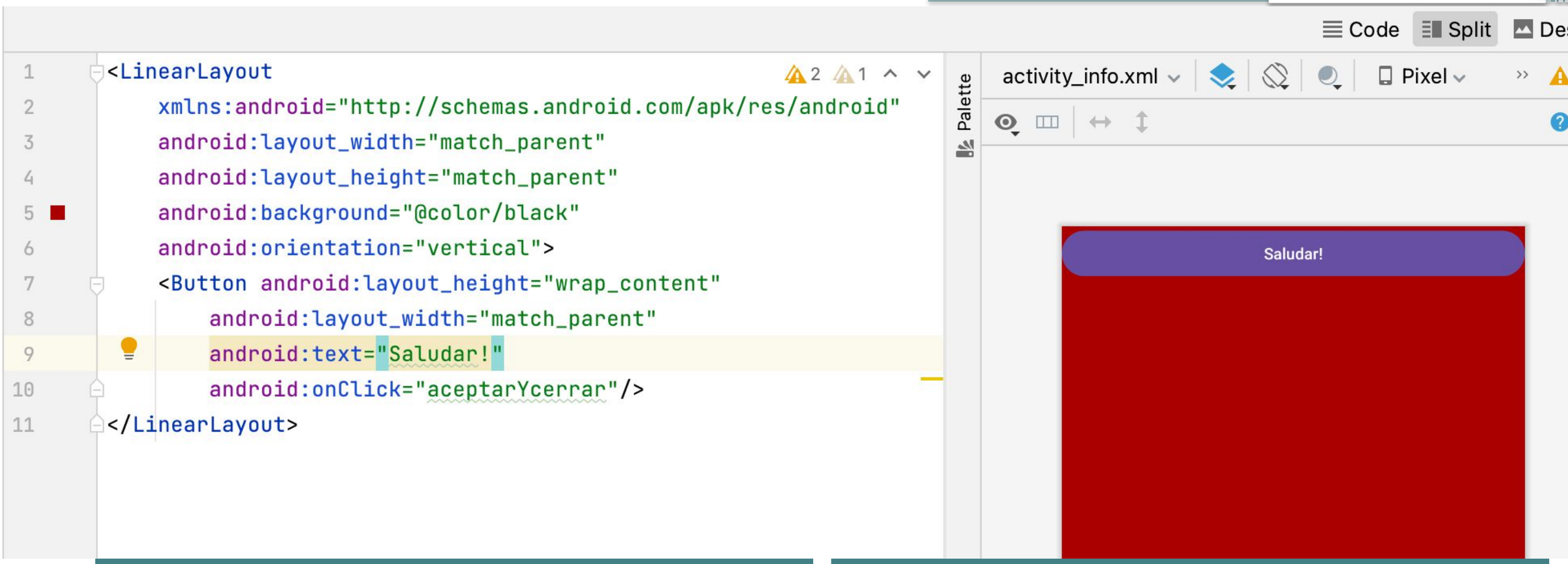
- Crear un nuevo proyecto Android Studio llamado "Saludos" basado en la siguiente Empty View Activity

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:background="@color/black">
  <Button
    android:layout_width="match_parent"
                                    ontent"
```

Dedicar dos minutos para descubrir y cambiar la definición del recurso `black`




plication9 app src main res values colors.xml
Android
app
manifests
AndroidManifest.xml
kotlin+java
res
drawable
layout
mipmap
values
colors.xml
strings.xml
themes (2)
xml
colors.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3 <color name="black">#FF000000</colo
4 <color name="white">#FFFFFFFF</colo
5 </resources>
Los recursos de color se encuentran definidos en el archivo colors.xml dentro de la carpeta values, dentro de la carpeta res
Este es el valor que se debe cambiar.
Cambiarlo por #FFAA0000



Obsérvese que se ha podido cambiar el color del fondo de la `Activity` sin necesidad de modificar el código fuente de la aplicación, tan sólo la definición del recurso apropiado.

Además, todas las vistas de la aplicación que utilicen este recurso se verán actualizadas, automáticamente, facilitando los cambios y la consistencia visual

Actividad guiada - continuación

- Agregar un recurso de color llamado colorTextoBoton con el siguiente valor `#ff0000`

- Agregue un `TextView` con la leyenda "Aquí se mostrará el saludo"
- Agregue un botón para salir
- Establecer la propiedad `textColor` de los botones con el recurso apropiado

Solución

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="black">#FF000000</color>
  <color name="white">#FFFFFFFF</color>
  <color name="colorTextoBoton">#FF0000</color>
</resources>
```

Agregar esta definición de color en el archivo de recursos colors.xml

Solución

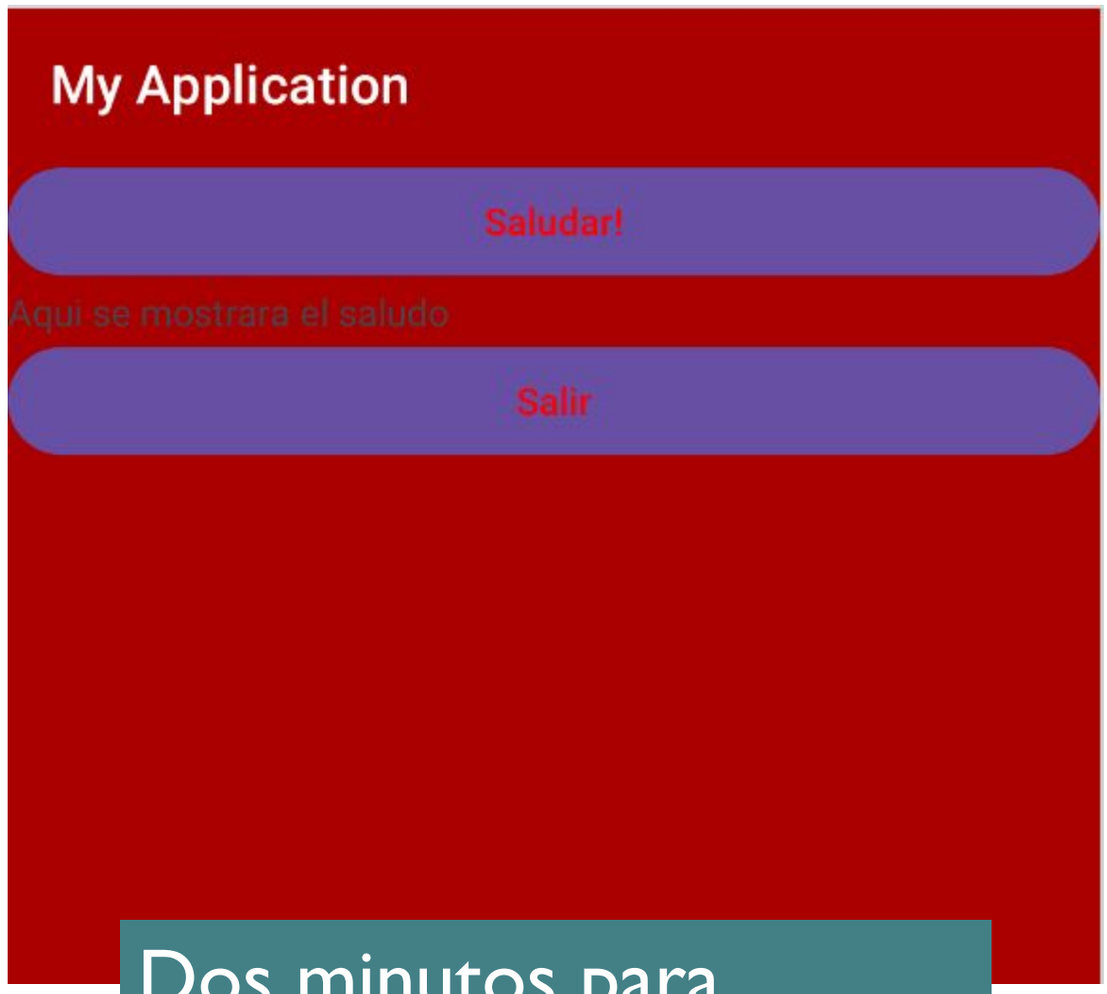
En el archivo
activity_main.xml

```
<Button
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Saludar!"
  android:textColor="@color/colorTextoBoton"
  />
<TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Aquí se mostrará el saludo"/>
<Button
  android:textColor="@color/colorTextoBoton"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Salir"/>
```

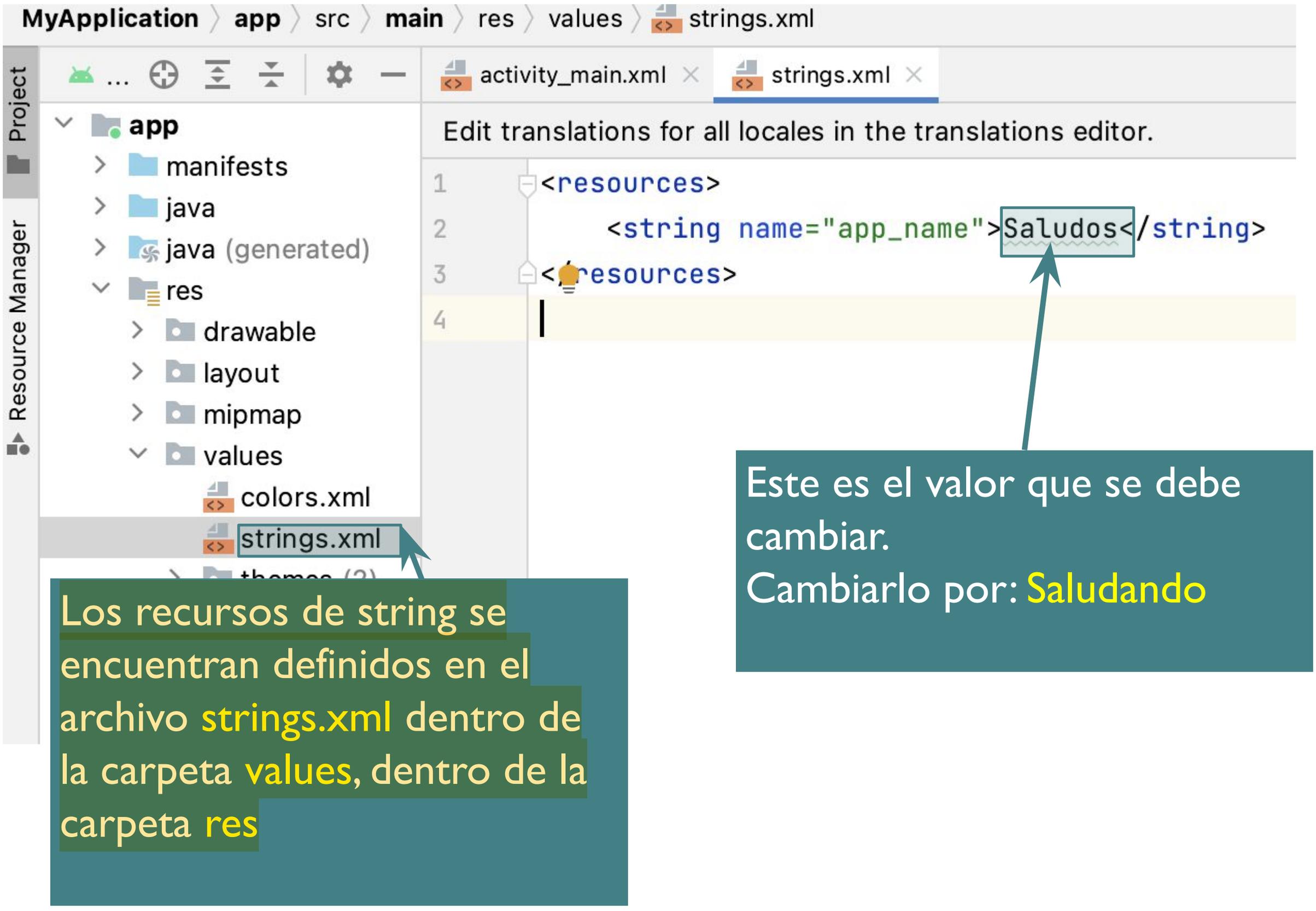
Actividad guiada - continuación

- Agregar al layout el componente **Toolbar** y establecer la propiedad title con el valor @string/app_name

- Establecer la propiedad titleTextColor del Toolbar con el recurso de color con valor blanco



Dos minutos para descubrir y cambiar la definición del recurso **app_name**


MyApplication 〉 app 〉 src 〉 main 〉 res 〉 values 〉 strings.xml
Project
Resource Manager
app
manifests
java
java (generated)
res
drawable
layout
mipmap
values
colors.xml
strings.xml
activity_main.xml
strings.xml
Edit translations for all locales in the translations editor.
1 <resources>
2 <string name="app_name">Saludos</string>
3 </resources>
4
Los recursos de string se encuentran definidos en el archivo strings.xml dentro de la carpeta values, dentro de la carpeta res
Este es el valor que se debe cambiar.
Cambiarlo por: Saludando

Actividad guiada - continuación

```
1  <LinearLayout                                   ⚠6 ⚠1
2      xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:background="@color/black"
6      android:orientation="vertical">
7      <Toolbar
8          android:layout_width="match_parent"
9          android:layout_height="wrap_content"
10         android:titleTextColor="@color/white"
11         android:title="@string/app_name"/>
12     <Button android:layout_height="wrap_content"
13         android:layout_width="match_parent"
14         android:text="Saludar!"
15         android:textColor="@color/colorTextoBoton"
16         android:onClick="aceptarYcerrar"/>
17     <TextView
18         android:layout_width="match_parent"
19         android:layout_height="wrap_content"
20         android:text="Aqui se mostrara el saludo"/>
21     <Button android:layout_height="wrap_content"
22         android:layout_width="match_parent"
23         android:text="Salir"
24         android:textColor="@color/colorTextoBoton"
25         android:onClick="aceptarYcerrar"/>
26 </LinearLayout>
```

Qué significan estos warnings

Actividad guiada - continuación

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:orientation="vertical">
    <Toolbar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:titleTextColor="@color/white"
        android:title="@string/app_name"/>
    <Button android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="Saludar!"
        android:textCol
        android:onClick
    <TextView
```

Hardcoded string "Saludar!", should use @string resource More... (⌘F1)

Extract string resource ⌥⇧⏎ More actions... ⌥⏎

Al posicionar el puntero del mouse sobre alguno de los indicadores del margen obtenemos la respuesta

Actividad guiada - continuación

- Agregar los recursos de string
  - saludar
  - mostrarAqui
  - salir
- Establecer la propiedad `text` de cada una de las vistas del layout con el recurso correspondiente
- Para referenciar a un recurso de string desde el archivo xml utilice `"@string/recurso"`

Solución

```
<resources>
  <string name="app_name">Saludando</string>
  <string name="saludar">Saludar!</string>
  <string name="mostrarAqui">Aquí se mostrará el saludo</string>
  <string name="salir">Salir</string>
</resources>
```

Agregar estas definiciones de recursos string en el archivo strings.xml

Solución

En el archivo
activity_main.xml

```
<Button
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/saludar"
  android:textColor="@color/colorTextoBoton"
  />
<TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/mostrarAqui"
  />
<Button
  android:textColor="@color/colorTextoBoton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/salir"
  />
```

Actividad guiada - continuación

- Codificar el `onClick` del botón superior para que al presionarlo aparezca la leyenda "¡Hola Mundo!" en el `TextView` debajo del mismo
- No utilizar el string "¡Hola Mundo!" directamente en el código Kotlin. En su lugar agregar un nuevo recurso de string llamado holaMundo y referenciarlo por medio de `R.string.holaMundo`

Solución

```
<resources>
  <string name="app_name">Saludando</string>
  <string name="saludar">Saludar!</string>
  <string name="mostrarAqui">Aquí se mostrará el saludo</string>
  <string name="salir">Salir</string>
  <string name="holaMundo">¡Hola Mundo!</string>
</resources>
```

Agregar estas definición de recurso string en el archivo strings.xml

Solución

En el archivo
activity_main.xml

```
<Button
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/saludar"
  android:textColor="@color/colorTextoBoton"
  android:onClick="saludar"
  />
<TextView
  android:id="@+id/leyenda"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/mostrarAqui"
  />
```

Solución

En el archivo
MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main);
    }
    fun saludar(v: View){
        val t: TextView = findViewById(R.id.leyenda);
        t.setText(R.string.holaMundo);
    }
}
```

Actividad guiada - continuación

- En ocasiones es necesario acceder por código Kotlin al string contenido en un recurso de tipo string. Utilizar `R.string.nombreRecurso` no es viable porque de esta forma obtenemos un entero que identifica al recurso pero no el string que se necesita
- En tales circunstancias se debe utilizar `resources.getString(R.string.holaMundo)`
- Modificar la aplicación para que al presionar el botón superior, también muestre el mensaje utilizando un Toast

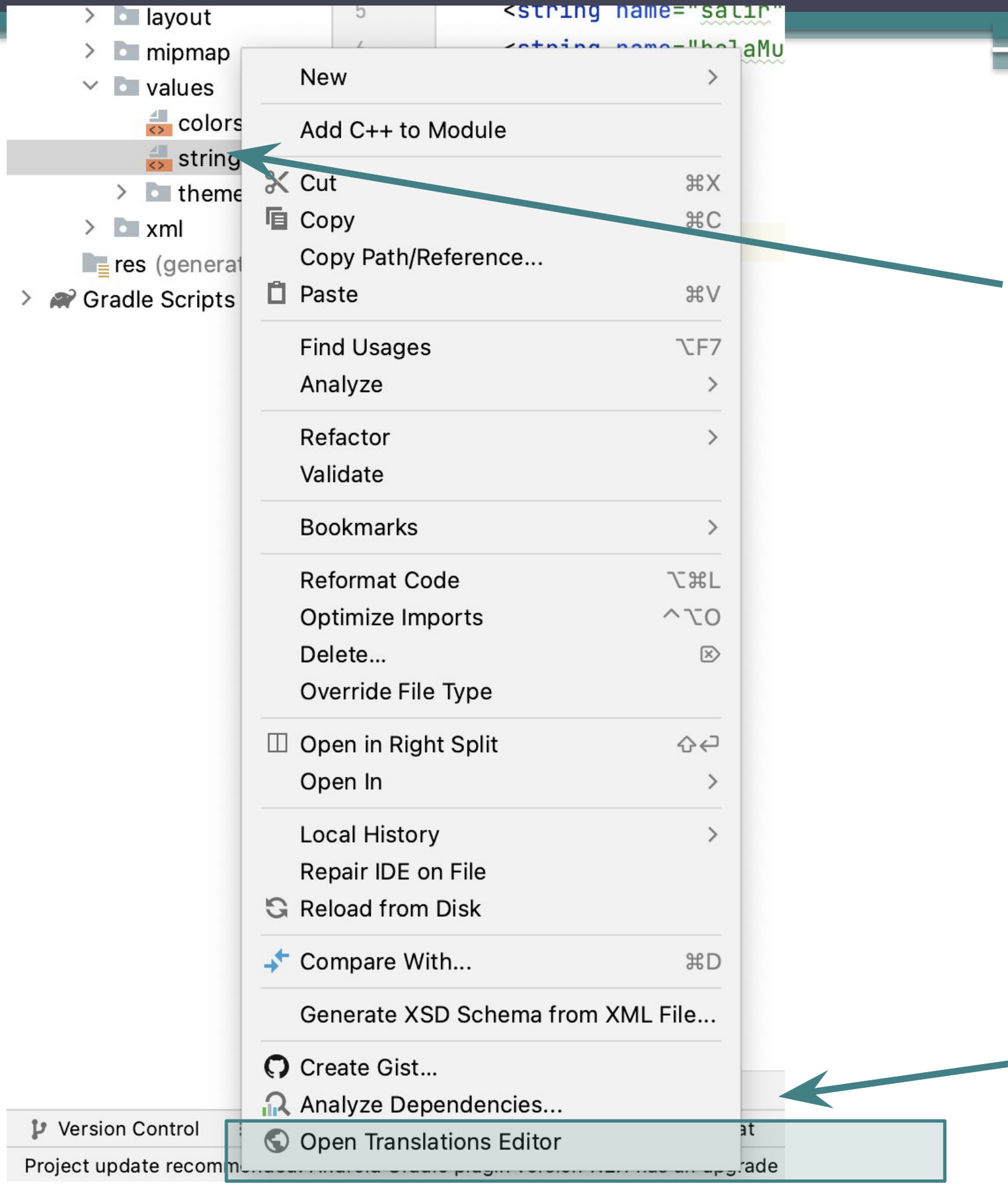
Solución

En el archivo
MainActivity.kt

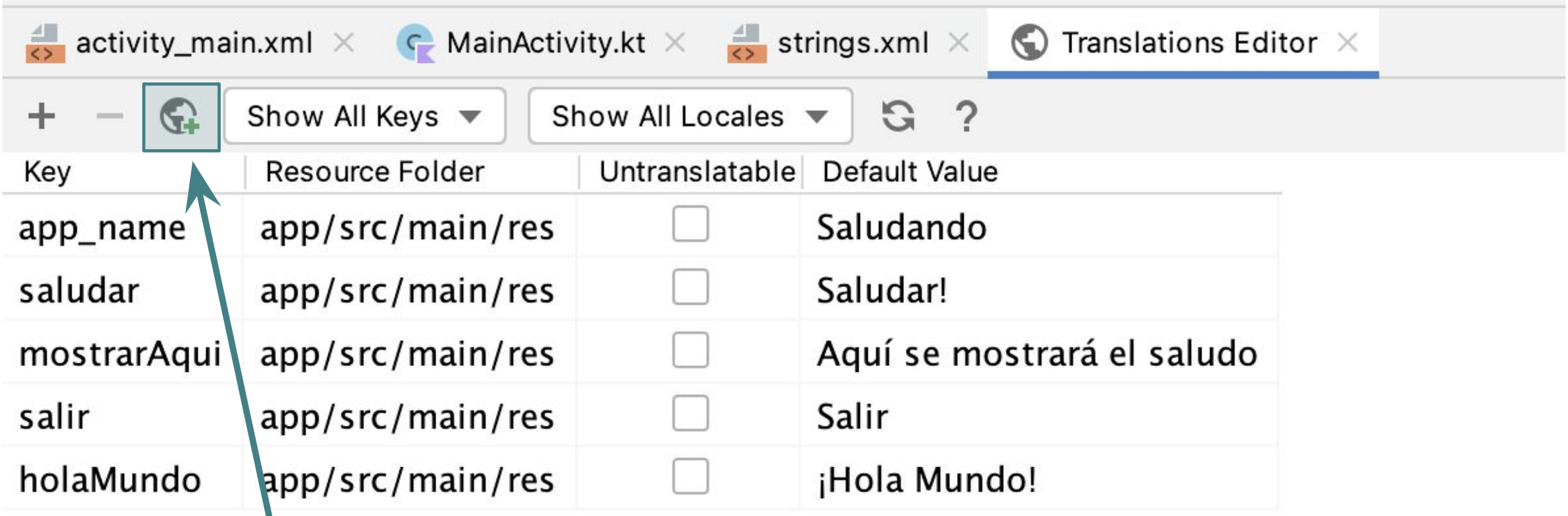
```
public class MainActivity extends AppCompatActivity {

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

    public void saludar(View v)
    {
        TextView t = (TextView)findViewById(R.id.leyenda);
        t.setText(R.string.holaMundo);
        val st: String = resources.getString(R.string.holaMundo);
        Toast.makeText(this,st,Toast.LENGTH_SHORT).show();

    }
}
```

Actividad guiada - continuación

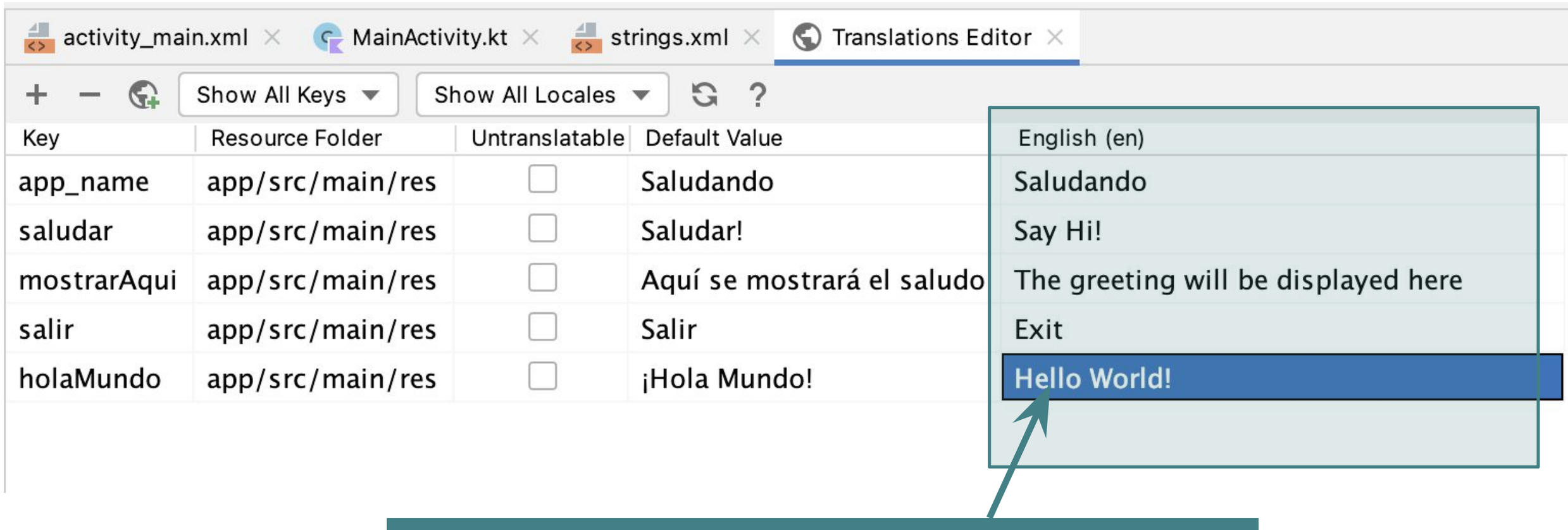
- La utilización de recursos string nos permite construir fácilmente aplicaciones multi-idioma.
- Los strings que muestra una aplicación multi-idioma cambian automáticamente según el idioma del dispositivo



Hacer click con el botón derecho del mouse sobre strings.xml y seleccionar Open Translations Editor en el menú contextual

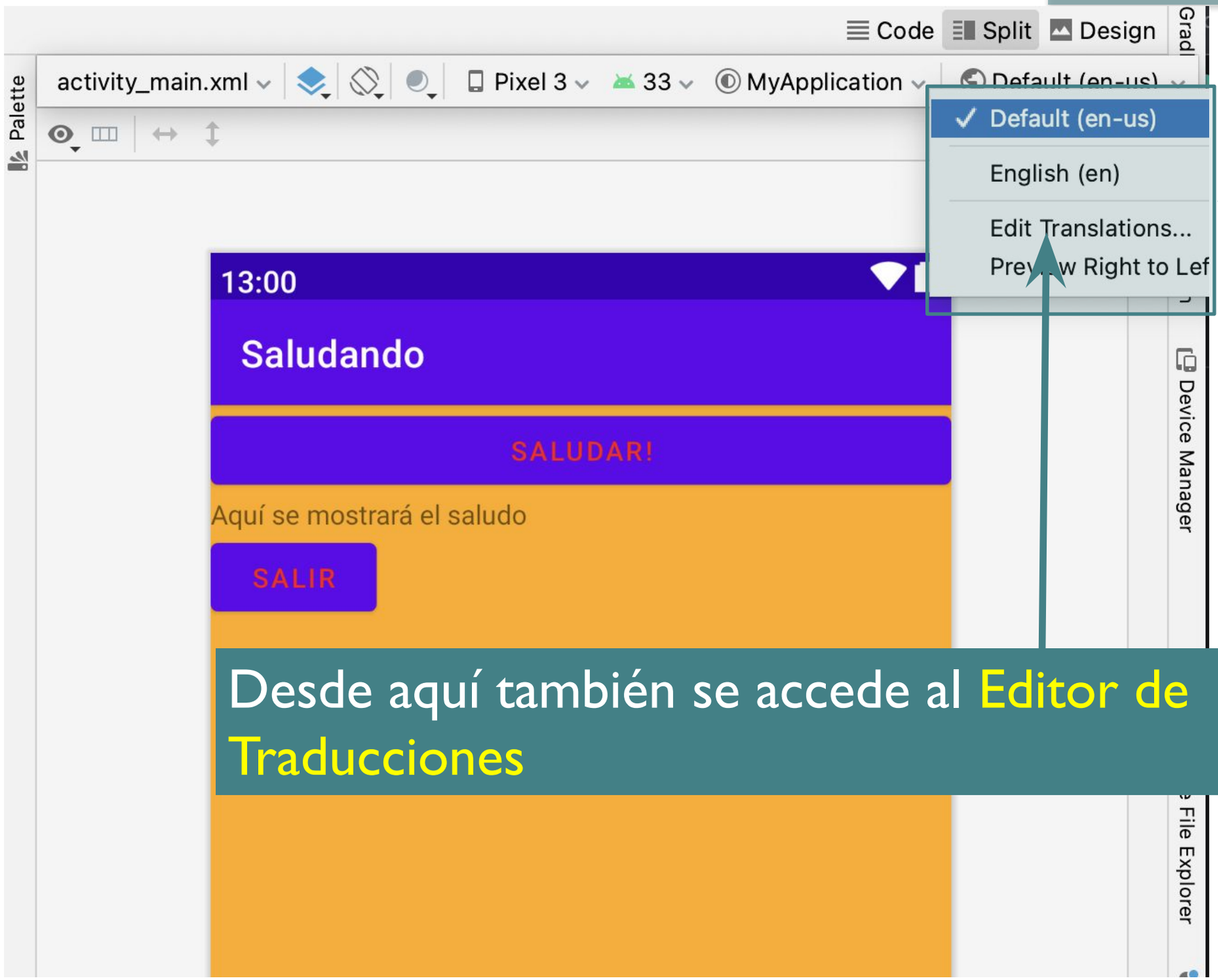


Hacer clic en el mundo y elegir English(en)



Completar las traducciones de cada string al inglés

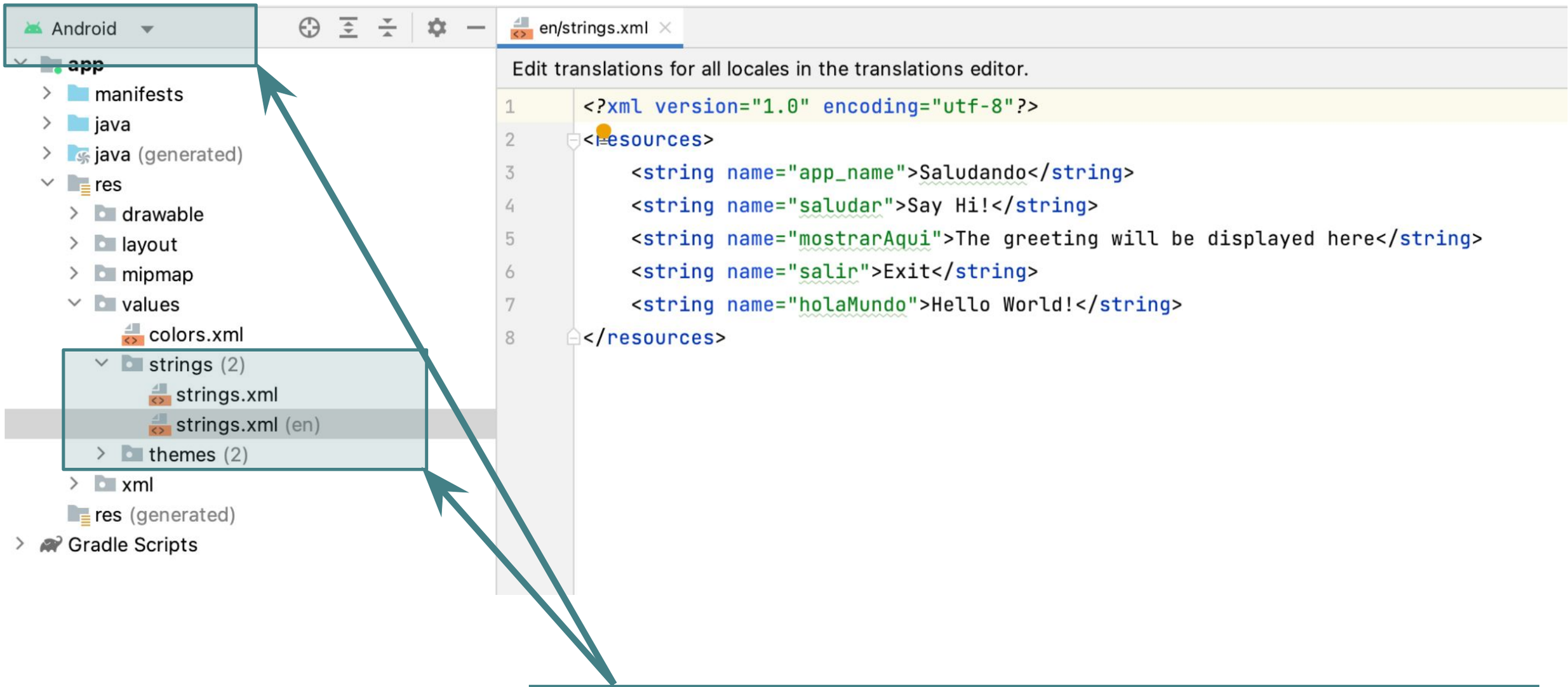
Compilar y ejecutar en el emulador. Verificar cambiando la configuración de idiomas. ¿Qué ocurre si se elige francés?



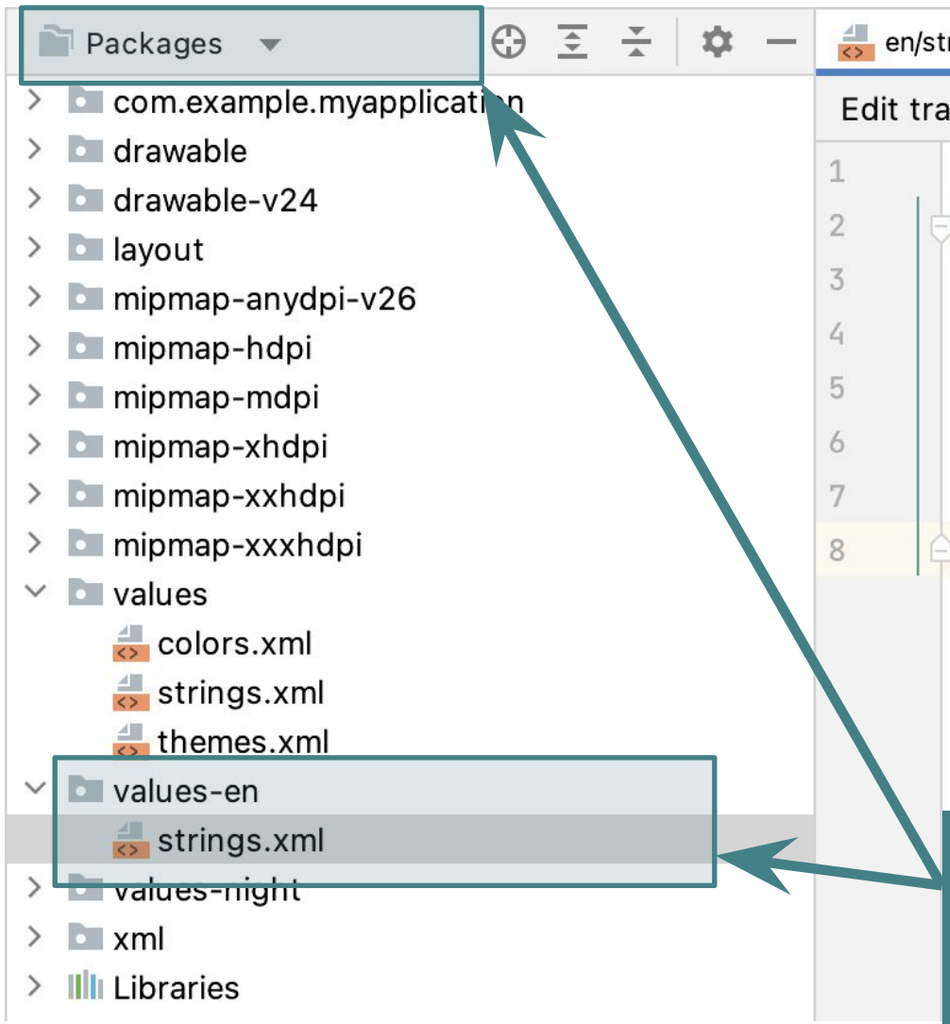
Desde el panel Preview puede visualizar fácilmente el efecto sobre la vista del

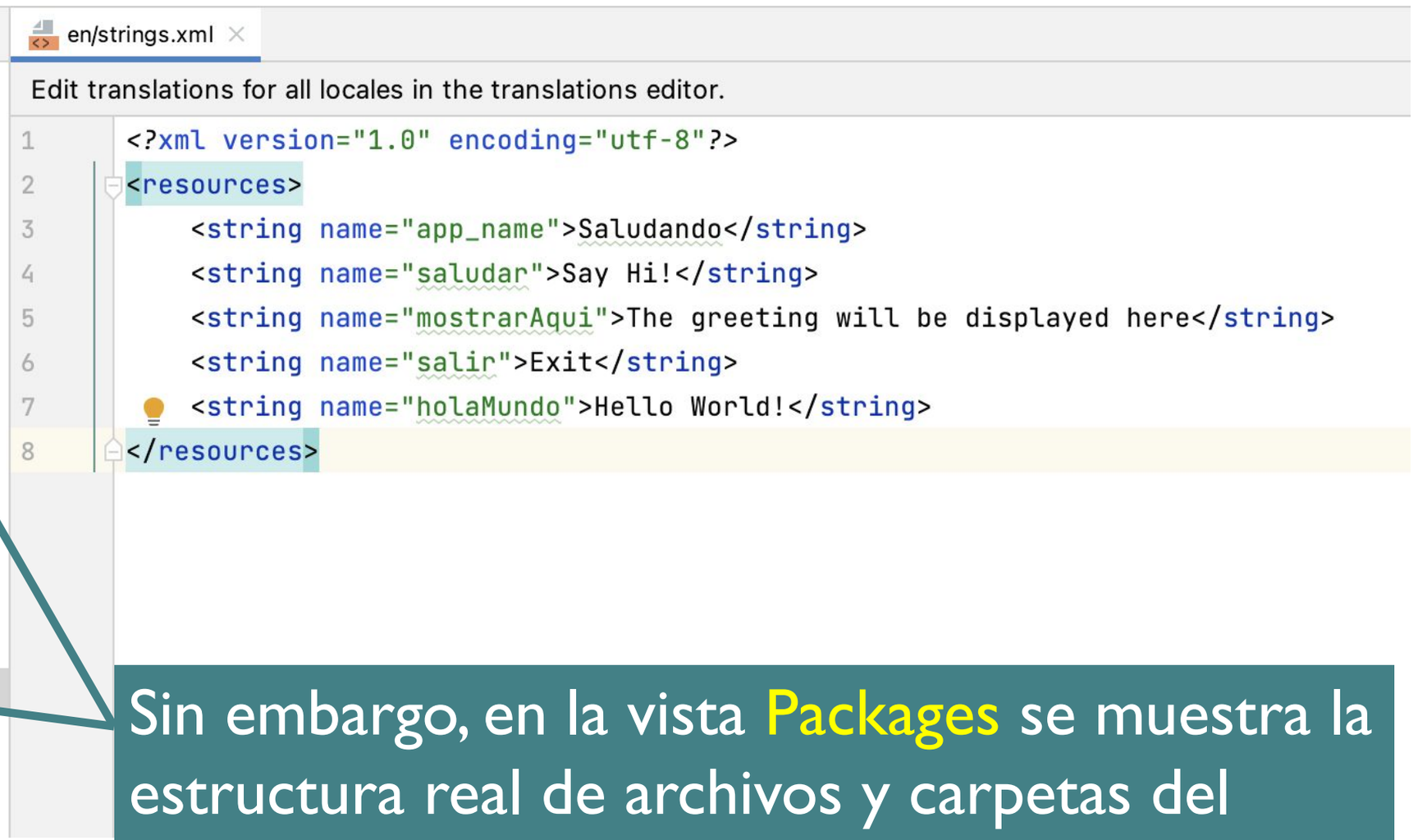
Desde aquí también se accede al Editor de Traducciones

Dedicar un minuto para descubrir en qué archivo se están guardando los recursos string correspondientes al idioma Inglés.



En la vista Android del proyecto se visualizan agrupados todos los archivos de recursos strings de los diferentes idiomas





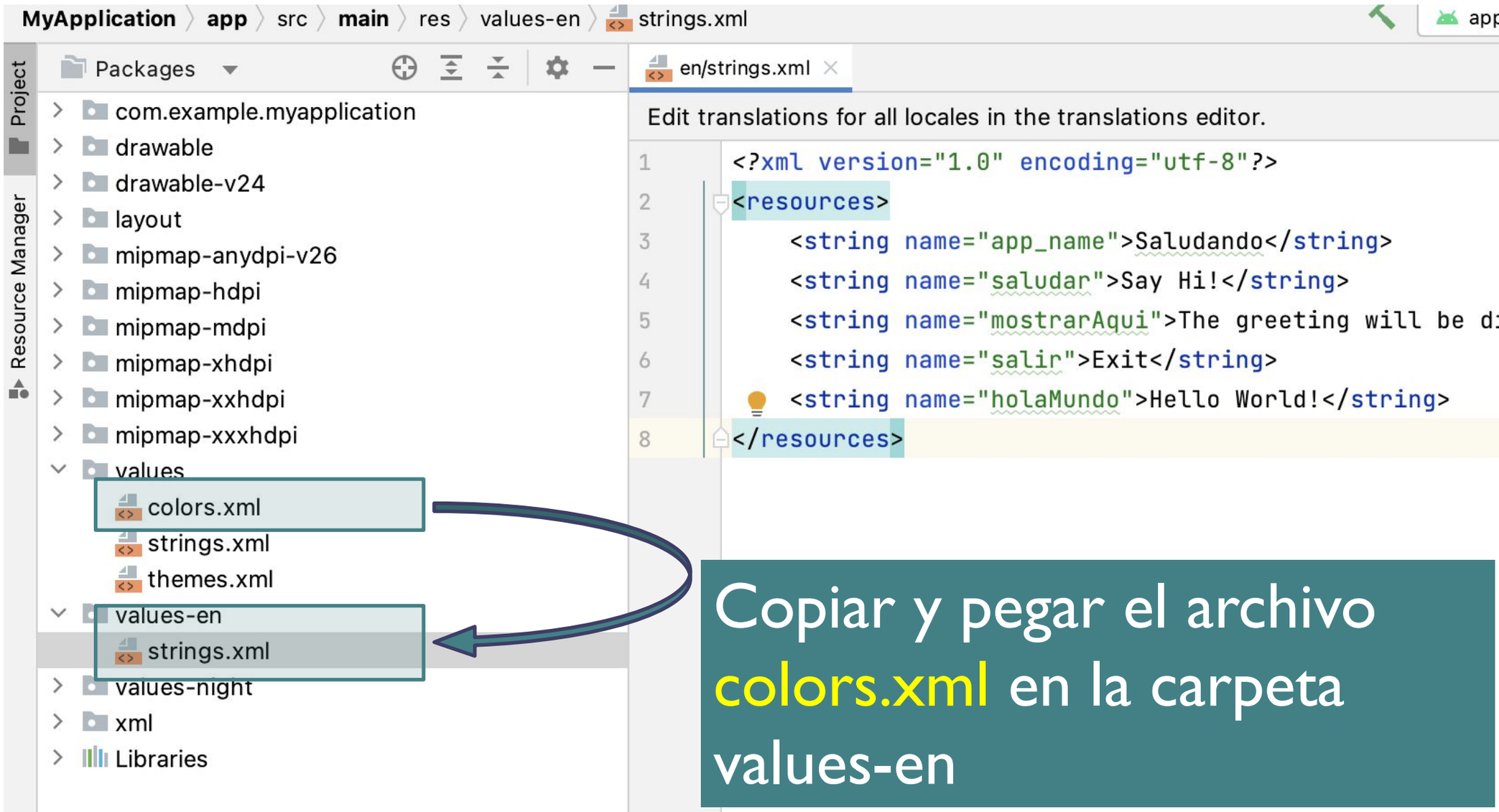
Sin embargo, en la vista Packages se muestra la estructura real de archivos y carpetas del proyecto. Se observa que se ha generado una carpeta values-en donde se encuentra el archivo strings.xml con las definiciones de recursos en inglés

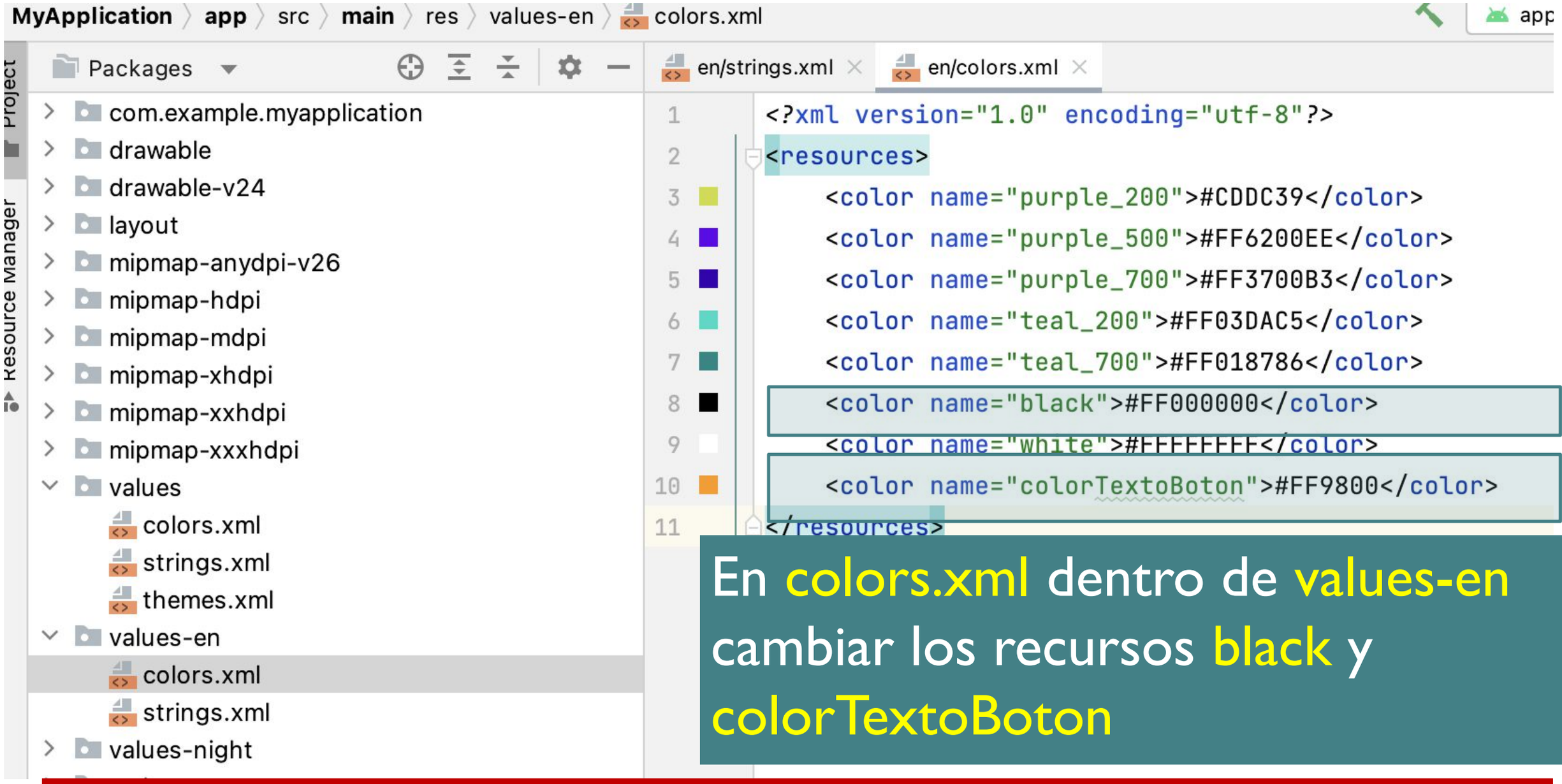
Actividad guiada - continuación

Vamos a definir recursos de color alternativos para la configuración del idioma en inglés

P: Teniendo en cuenta lo visto para recursos strings, cómo se imagina se podrán definir recursos de color para el idioma inglés?

R: Efectivamente definiendo el archivo xml correspondiente dentro de la carpeta values

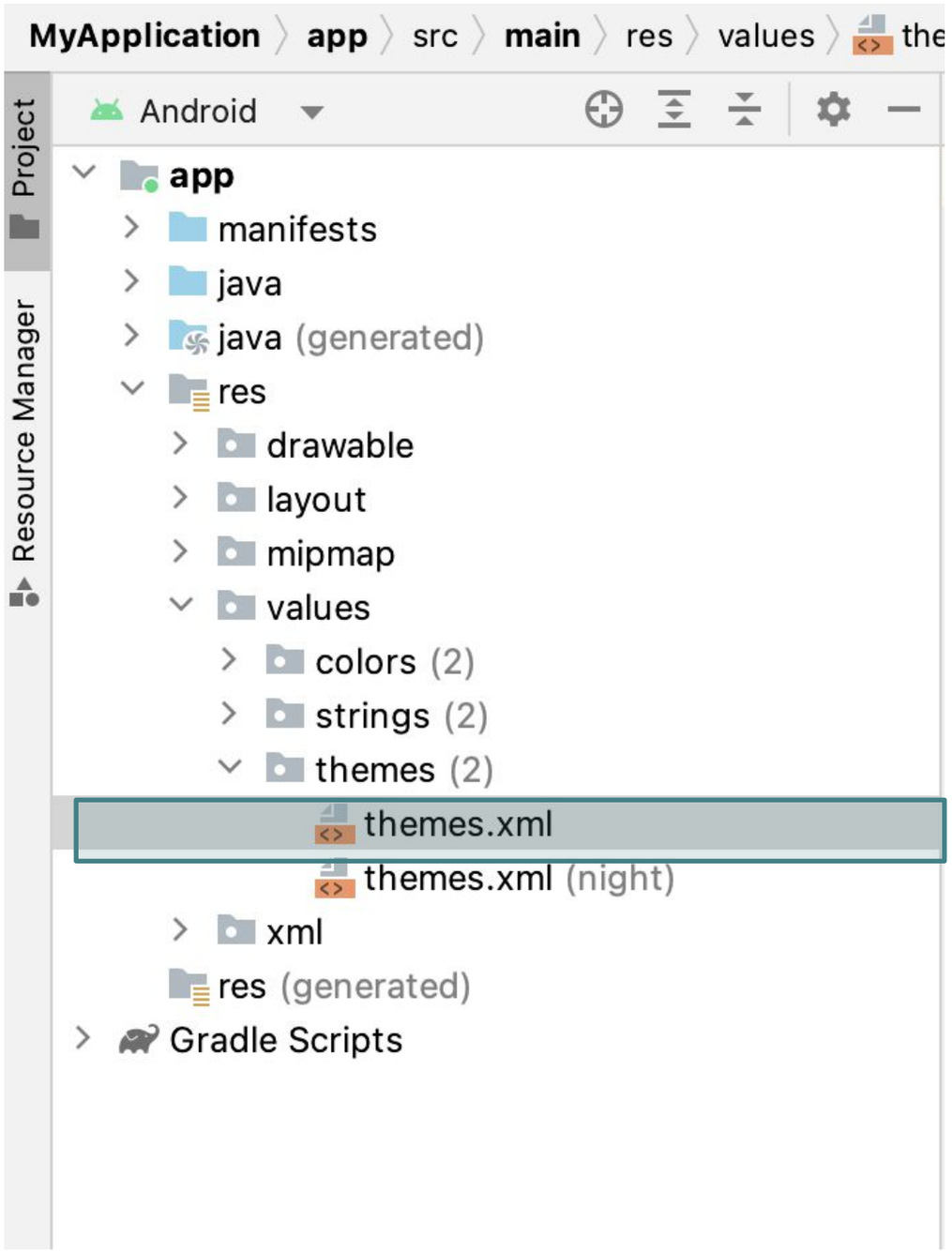

MyApplication app src main res values-en strings.xml
app
Emulador nuevo API 32
Project
Resource Manager
Packages
en/strings.xml
Edit translations for all locales in the translations editor.
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3 <string name="app_name">Saludando</string>
4 <string name="saludar">Say Hi!</string>
5 <string name="mostrarAqui">The greeting will be displayed here</string>
6 <string name="salir">Exit</string>
7 <string name="holaMundo">Hello World!</string>
8 </resources>
com.example.myapplication
drawable
drawable-v24
layout
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
colors.xml
strings.xml
themes.xml
values-en
strings.xml
values-night
xml
Libraries
Copiar y pegar el archivo colors.xml en la carpeta values-en



Verificar en el emulador o en el panel Preview que el texto y los colores cambian en función del idioma configurado en el dispositivo

Estilos

- Un estilo es una colección de propiedades que definen el formato y apariencia que tendrá una vista. Podemos especificar cosas como tamaño, márgenes, color, fuentes, etc.
- Los estilos se definen en archivos de recursos al igual que los colores o strings.



Actividad guiada - continuación

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.MyApplication" parent="Theme.Material3.DayNight.NoActionBar">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>

  <style name="Theme.MyApplication" parent="Base.Theme.MyApplication" />

  <style name="MiEstilo">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textSize">50sp</item>
  </style>
</resources>
```

Agregrar el siguiente estilo en themes.xml

Actividad guiada - continuación

```
<Button
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/saludar"
  android:textColor="@color/colorTextoBoton"
  android:onClick="saludar"
  />
<TextView
  android:id="@+id/leyenda"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/mostrarAqui"
  />
<Button
  style="@style/MiEstilo"
  android:textColor="@color/colorTextoBoton"
  android:text="@string/salir"
  />
```

En el archivo activity_main.xml

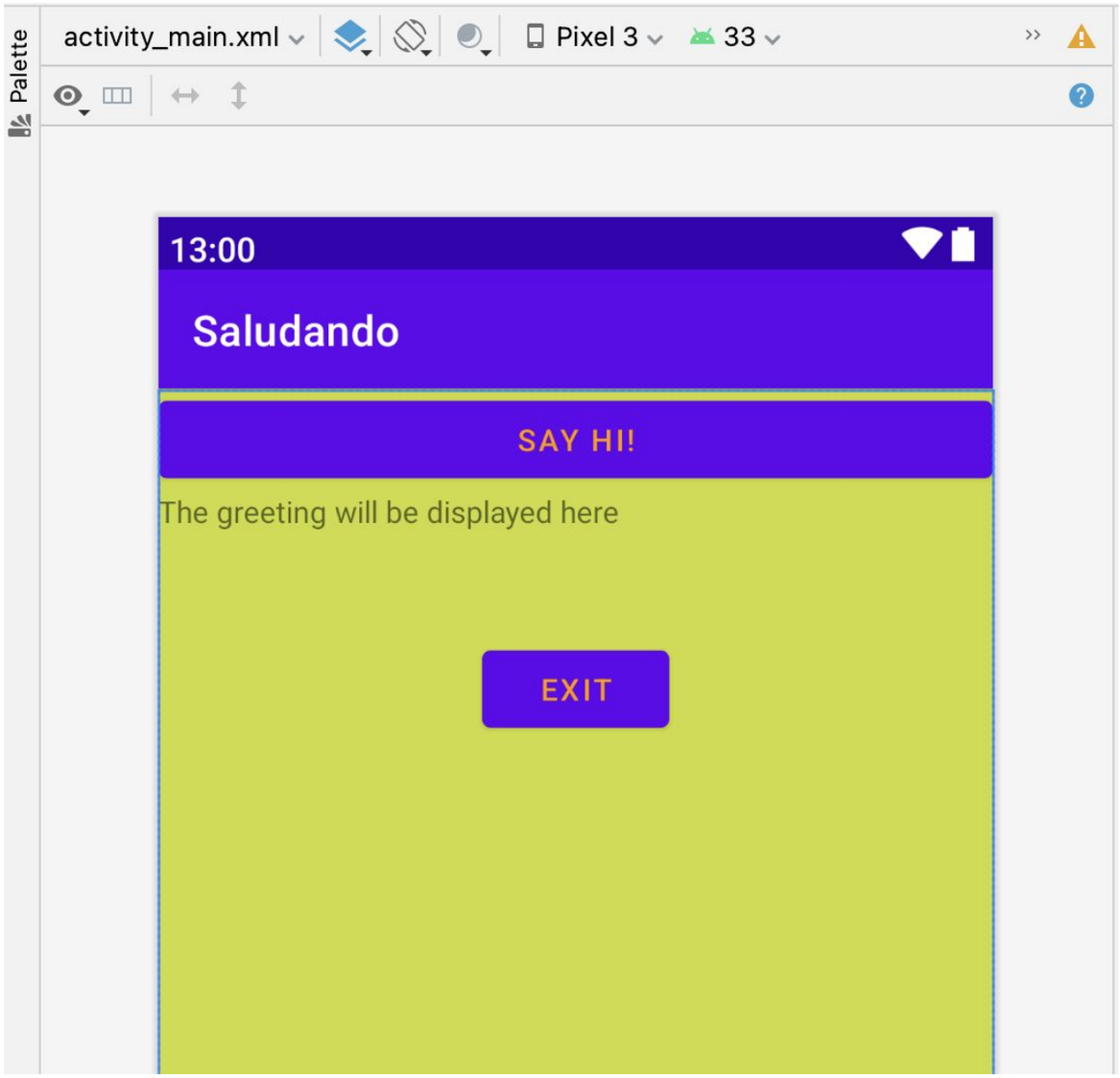
Eliminar atributos `layout_width` y `layout_height` Establecer el atributo `style`

Verificar efecto en el panel Preview

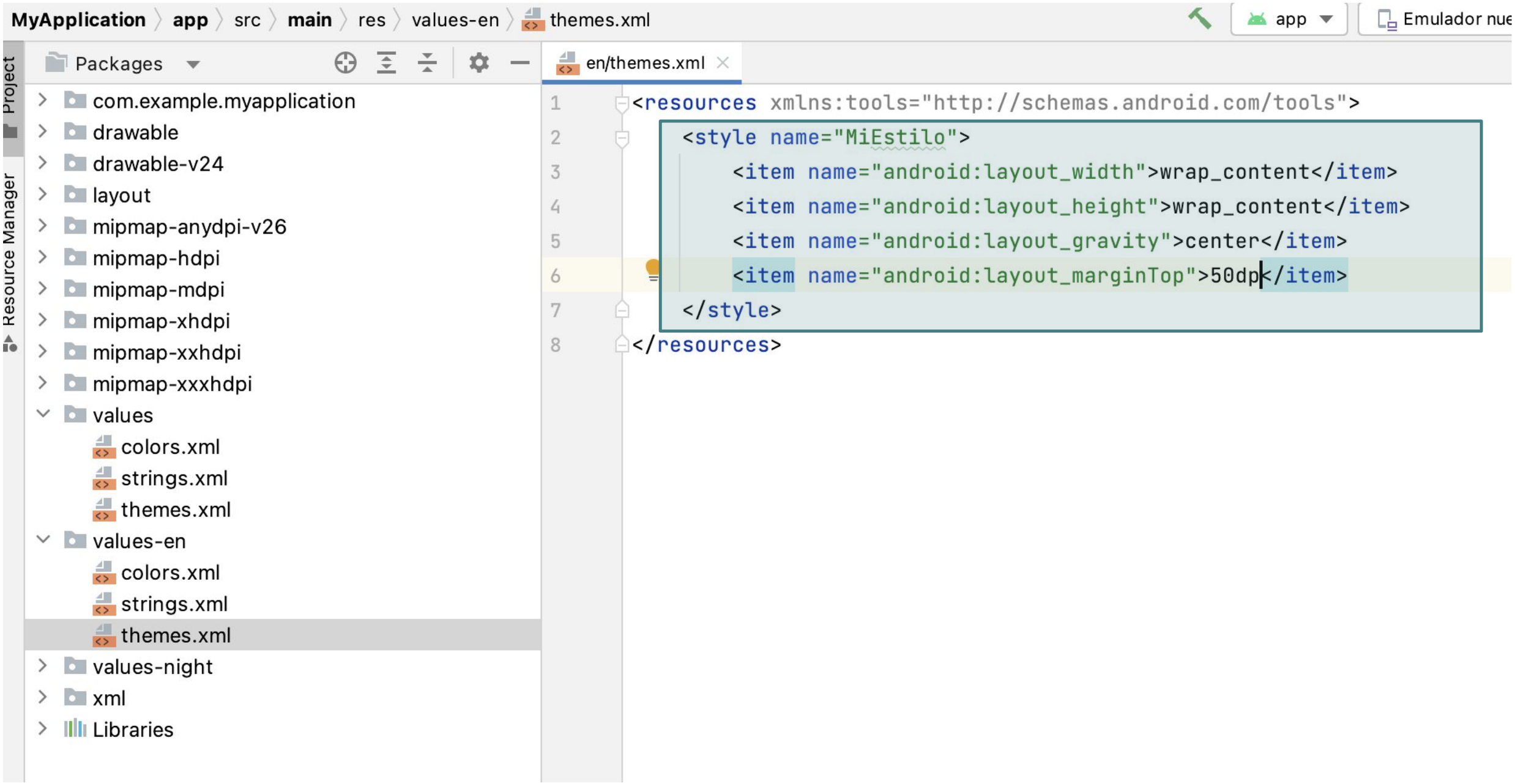
Actividad guiada - continuación

Agregar el estilo MiEstilo en un recurso alternativo para el idioma inglés, tal que la vista se vea como en la imagen de la derecha.

(El botón EXIT está separado por un margen superior de 50dp)



Solución


MyApplication 〉 app 〉 src 〉 main 〉 res 〉 values-en 〉 themes.xml
app
Emulador nue
Project
Resource Manager
Packages
com.example.myapplication
drawable
drawable-v24
layout
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
colors.xml
strings.xml
themes.xml
values-en
colors.xml
strings.xml
themes.xml
values-night
xml
Libraries
en/themes.xml
1 <resources xmlns:tools="http://schemas.android.com/tools">
2 <style name="MiEstilo">
3 <item name="android:layout_width">wrap_content</item>
4 <item name="android:layout_height">wrap_content</item>
5 <item name="android:layout_gravity">center</item>
6 <item name="android:layout_marginTop">50dp</item>
7 </style>
8 </resources>

Heredando de un estilo

Utilizando el mismo nombre de un estilo ya creado y completando el nombre con un punto más un sufijo, se obtiene un nuevo estilo que hereda todas las características del primero y agrega las nuevas definidas, Por ejemplo:

```
<style name="MiEstilo.negrita">
    <item name="android:textStyle">bold</item>

</style>
```

En este ejemplo se obtiene un nuevo estilo que sería igual a *MiEstilo* más la propiedad *textStyle* en *bold*.

Temas

Un tema es un estilo aplicado a toda una actividad o aplicación, en lugar de a un View individual. Cada elemento del estilo sólo se aplicará a aquellos elementos donde sea posible

Para aplicar un tema a toda una aplicación debe establecerse en el archivo AndroidManifest.xml agregando el atributo `android:theme` en la etiqueta `<application>` :

```
<application android:theme="@style/MiTema">
```

También, en AndroidManifest.xml se puede aplicar un tema a una activity determinada

```
<activity android:theme="@style/MiTema">
```

Temas – Manifiesto de la aplicación recién construida

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  package="com.example.myapplication">
  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.MyApplication"
    tools:targetApi="31">
    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

Temas – themes.xml predeterminado de la aplicación recién construida

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.MyApplication" parent="Theme.Material3.DayNight.NoActionBar">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>

  <style name="Theme.MyApplication" parent="Base.Theme.MyApplication" />

  <style name="MiEstilo">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textSize">50sp</item>
  </style>
</resources>
```

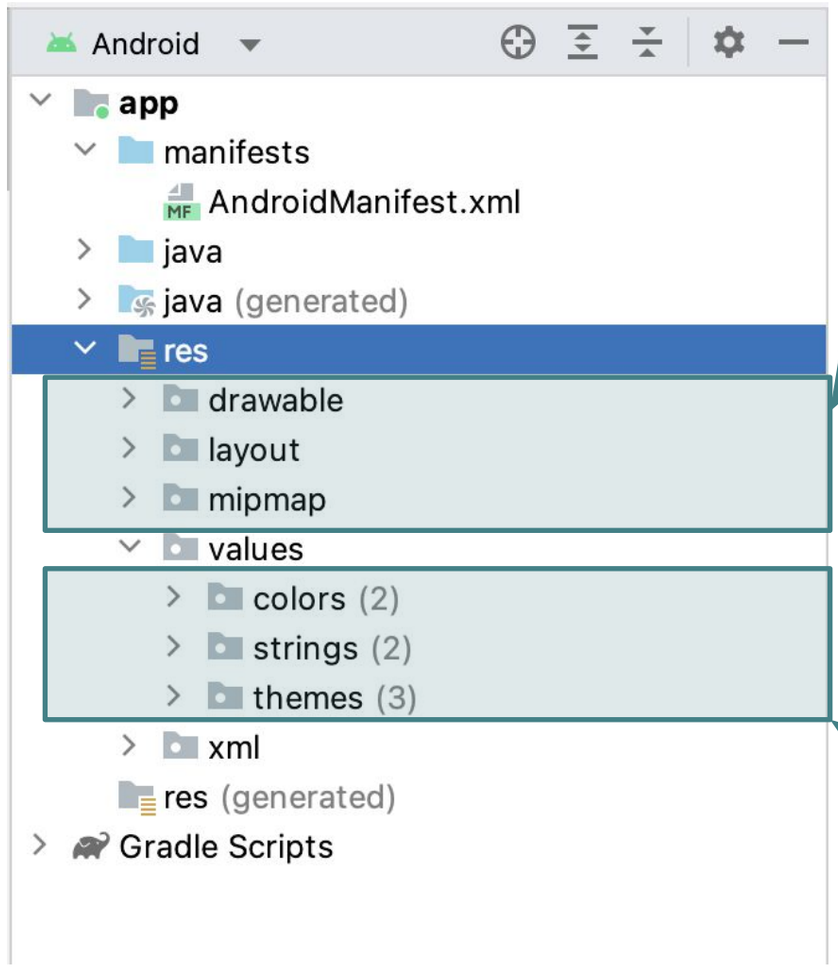
Hereda de un estilo ya definido

Recapitulando

- Para cualquier tipo de recursos pueden definirse recursos predeterminados y varios alternativos
- Los recursos predeterminados son los que se usan sin importar la configuración del dispositivo o cuando no hay recursos alternativos que coincidan con la configuración actual.
- Los recursos alternativos son los que se utilizan con una configuración específica. A fin de especificar que un grupo de recursos es para una configuración específica, se debe agregar un calificador de configuración apropiado al nombre del directorio.

Tipos de recurso

- Se pueden distinguir dos tipos de recursos.
  - Recursos de archivo
  - Recursos de valor



Recursos de archivo: Cada archivo en estas carpetas representa un recurso cuyo nombre coincide con el nombre del archivo. Por ejemplo el recurso R.layout.activity_main está definido en el archivo activity_main.xml dentro de la carpeta layout

Recursos de valor: Cada archivo en estas carpetas es un documento xml que define un conjunto de recursos por medio de las etiquetas correspondientes

Recursos de valor

- Ya hemos trabajado con tres tipos de recursos de valor:
  - Recursos string (strings.xml)
  - Recursos color (colors.xml)
  - Recursos style (themes.xml)
- Otros tipos de recursos valor son:
  - Recursos dimen (dimensions.xml)
  - Recursos integer
  - Recursos bool
  - Recursos id
  - Recursos array

Aunque, por cuestiones de organización, se recomienda definir los recursos dentro del xml correspondiente, es posible mezclar los recursos en cualquier archivo nombrado arbitrariamente

Recursos de valor

Recursos dimen: definen un tamaño por medio de un número seguido de una unidad.

Ejemplo, archivo res/values/nombre_archivo.xml:

```
<dimen name="alto">2.2mm</dimen>
<dimen name="tamano_fuente">16sp</dimen>
```

Se accede como:

`R.dimen.alto` en código Kotlin

`"@dimen/alto"`

Recursos de valor

Recursos integer: definen un valor entero.

Ejemplo, archivo res/values/nombre_archivo.xml:

```
<integer name="max_cant">5</integer>
```

Se accede como:

`R.integer.max_cant` en código Kotlin

`"@integer/max_cant"`

Recursos de valor

Recursos bool: definen un valor booleano.

Ejemplo, archivo res/values/nombre_archivo.xml:

```
<bool name="lunesAbierto">true</bool>
```

Se accede como:

`R.bool.lunesAbierto` en código Kotlin

`"@bool/lunesAbierto"`

Recursos de valor

Recursos id: Define un recurso de id único. Aunque habitualmente los id se definen utilizando el atributo `id="@+id/nombre"` en algunos casos es conveniente disponer de un id previamente creado, para que los elementos así nombrados cumplan un rol específico.

Ejemplo, archivo res/values/nombre_archivo.xml:

```
<item type="id" name="boton_ok"/>
```

Se accede como:

`R.id.boton_ok` en código Kotlin

`"@id/boton_ok"`

Recursos de valor

Recursos Array: Una serie ordenada de elementos. Pueden ser de strings, de enteros o de recursos.

Ejemplo, archivo res/values/nombre_archivo.xml:

```
<integer-array name="primos">
  <item>2</item><item>3</item><item>5</item>
</integer-array>
```

Se accede desde el código Kotlin de una activity como:

```
var v: IntArray = resources.getIntArray(R.array.primos);
```