

# SEMINARIO DE LENGUAJES

## OPCIÓN ANDROID



**Recursos.**

Mg. Corbalán Leonardo, Esp. Delía Lisandro

# Recursos

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

P: ¿ Con Cuales ?

R: Ids y Layout

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

# Recursos

## Por Ejemplo en:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_info );  
    TextView texto=(TextView)findViewById(R.id.texto );  
    . . .  
}
```

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

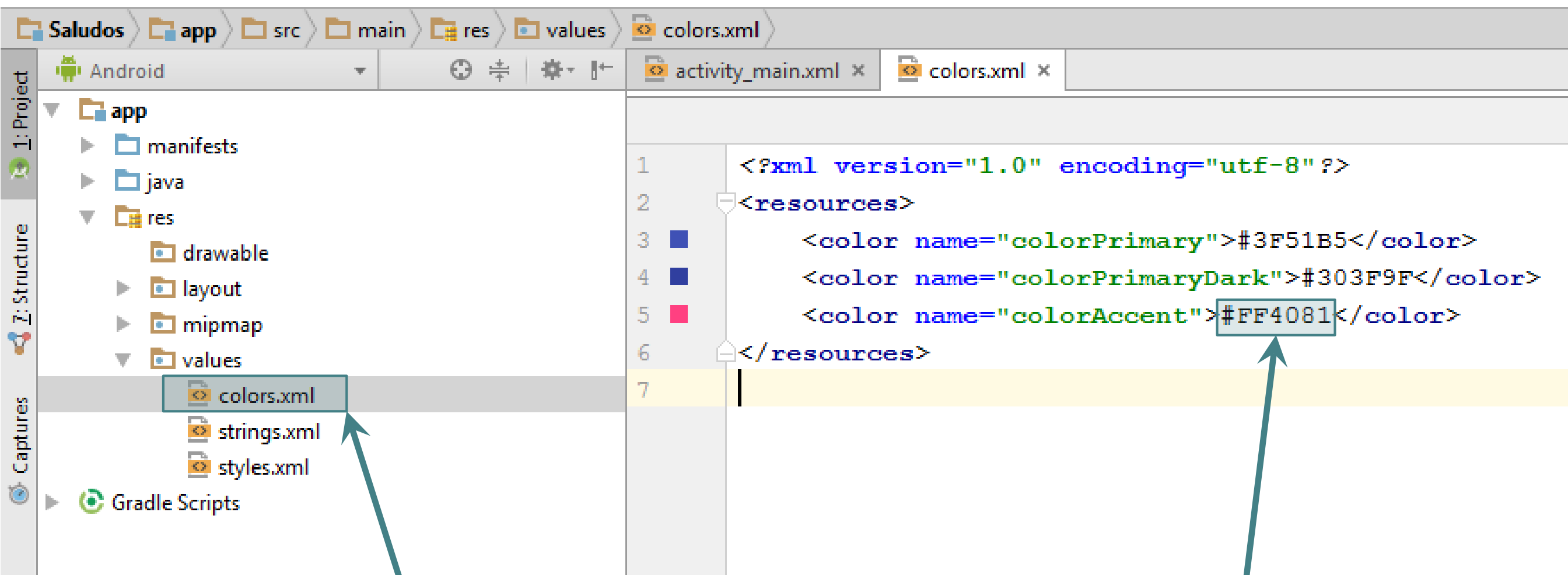
- Crear un nuevo proyecto Android Studio llamado "Saludos" basado en la siguiente Empty Activity

```
<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/colorAccent" >
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Saludar a todos"
    />
</LinearLayout>
```

The screenshot displays the Android Studio environment. On the left, the 'activity\_main.xml' file is open in the Text editor. The XML code defines a vertical LinearLayout with a blue background and a button labeled 'Saludar a todos'. A green box highlights the attribute `android:background="@color/colorAccent"`, with a green arrow pointing from a text box below to it. The right side shows the Preview window, which displays a mobile app interface. The interface has a blue header bar with the text 'Saludos', a grey button labeled 'SALUDAR A TODOS', and a large pink area below. The top status bar shows a Wi-Fi icon, a battery icon, and the time '7:00'. The bottom of the screen features the 'Design' and 'Text' tabs, and a status bar with '4: Run', 'TODO', 'Event Log', and 'Gradle Console'.

```
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="@color/colorAccent" >
6     <Button
7         android:layout_width="match_parent"
8         android:layout_height="wrap_content"
9         android:text="Saludar a todos"
10    />
11 </LinearLayout>
```

Dedicar dos minutos para descubrir y cambiar la definición del recurso **colorAccent**



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 **#FFAA00**

The screenshot displays the Android Studio IDE. On the left, the 'activity\_main.xml' file is open, showing the following XML code:

```
<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/colorAccent" >
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Saludar a todos"
    />
</LinearLayout>
```

On the right, the 'Preview' window shows a visual representation of the app. It features a blue header bar with the text 'Saludos', a white button labeled 'SALUDAR A TODOS', and a large orange area at the bottom. The status bar at the top right shows a Wi-Fi icon, a battery icon, and the time 7:00.

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



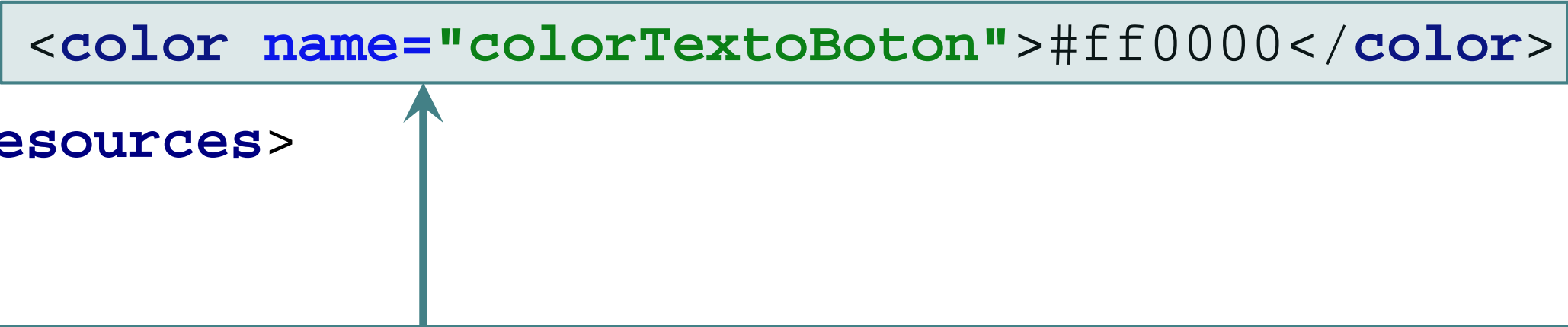
# Actividad guiada - continuación

- Agregue 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**
- Establezca la propiedad **textColor** de los botones con el recurso apropiado



# Solución

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FFAA00</color>
    <color name="colorTextBoton">#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 a todos"
    android:textColor="@color/colorTextoBoton"
/>
```

<TextView

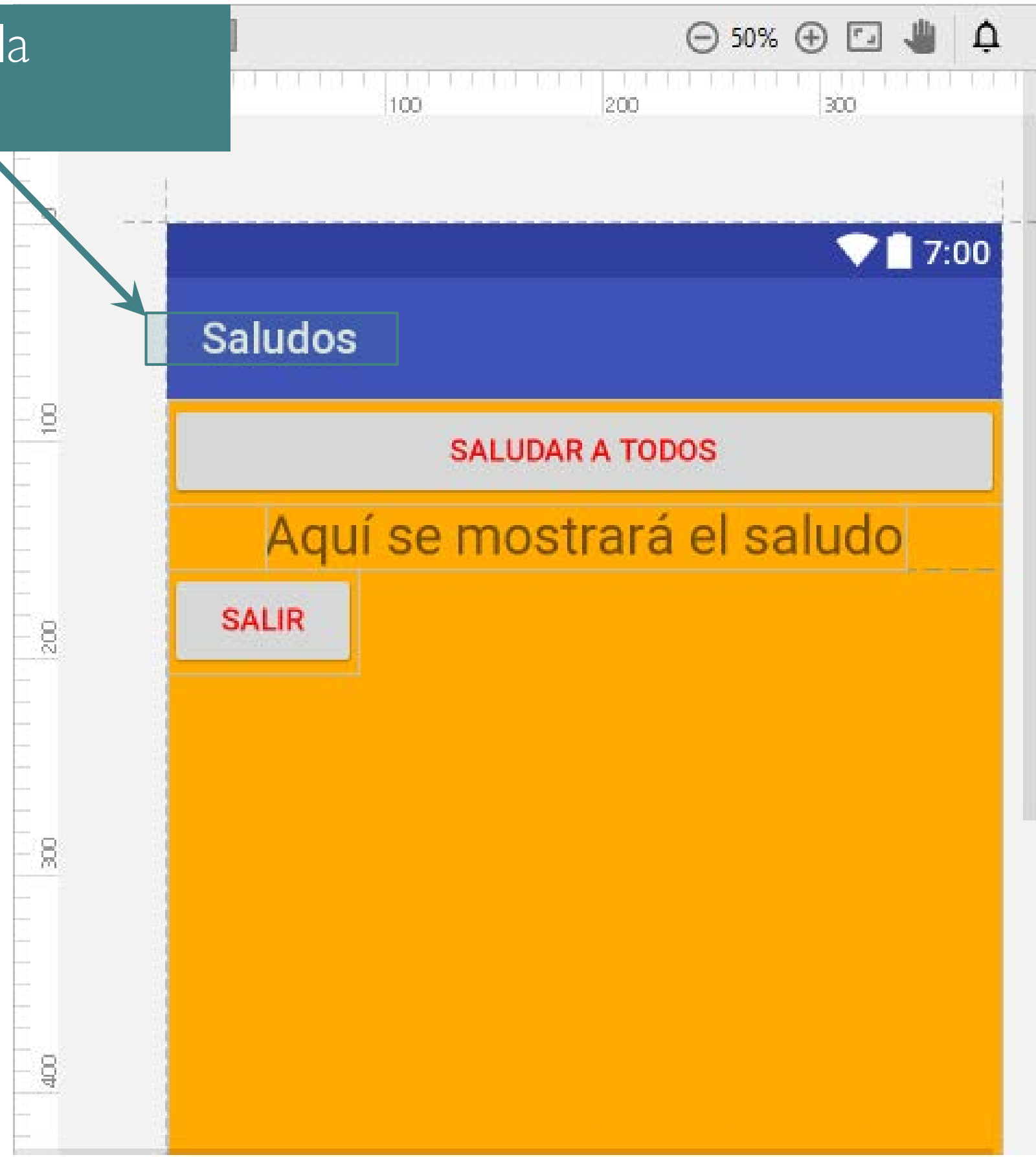
```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Aquí se mostrará el saludo"
    android:textSize="25sp"
    android:layout_gravity="center"
/>
```

<Button

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Salir"
    android:textColor="@color/colorTextoBoton"
/>
```

Aquí se visualiza el nombre de la aplicación

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



The screenshot shows the Android Studio interface. The top breadcrumb navigation shows the path: **Saludos** > **app** > **src** > **main** > **res** > **values** > **strings.xml**. The left sidebar has tabs for **1: Project**, **2: Favorites**, **Captures**, and **Structure**. The **Structure** tab is active, showing a tree view of the project files. The **res** folder is expanded, showing **drawable**, **layout**, **mipmap**, and **values**. The **values** folder is expanded, showing **colors.xml**, **strings.xml** (which is selected and highlighted), and **styles.xml**. The main editor area shows the content of **strings.xml** with the following XML code:

```
1 <resources>
2     <string name="app_name">Saludos</string>
3 </resources>
```

Line 2 is highlighted in yellow. A blue box highlights the word **Saludos** in the XML code. A blue arrow points from the **strings.xml** file in the **Structure** tab to the **Saludos** text in the XML code. Another blue arrow points from the **Saludos** text in the XML code to a text box on the right.

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**

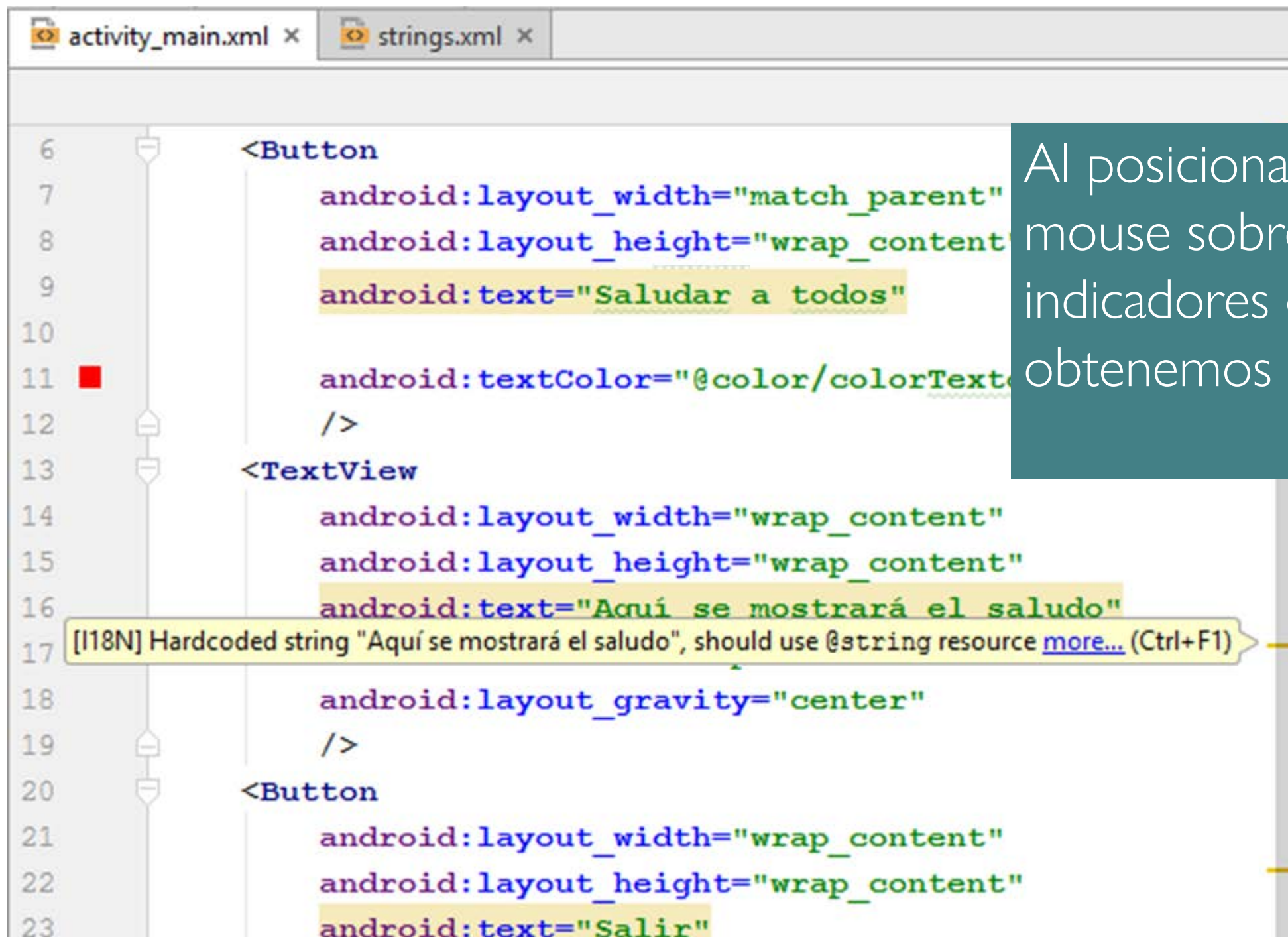
# Actividad guiada - continuación

```
6 <Button
7     android:layout_width="match_parent"
8     android:layout_height="wrap_content"
9     android:text="Saludar a todos"
10
11     android:textColor="@color/colorTextoBoton"
12 />
13 <TextView
14     android:layout_width="wrap_content"
15     android:layout_height="wrap_content"
16     android:text="Aquí se mostrará el saludo"
17     android:textSize="25sp"
18     android:layout_gravity="center"
19 />
20 <Button
21     android:layout_width="wrap_content"
22     android:layout_height="wrap_content"
23     android:text="Salir"
24     android:textColor="@color/colorTextoBoton"
25 />
26 </LinearLayout>
```

Qué significan  
estos warnings



# Actividad guiada - continuación



```
6 <Button
7     android:layout_width="match_parent"
8     android:layout_height="wrap_content"
9     android:text="Saludar a todos"
10
11     android:textColor="@color/colorText"
12 />
13 <TextView
14     android:layout_width="wrap_content"
15     android:layout_height="wrap_content"
16     android:text="Aquí se mostrará el saludo"
17
18     android:layout_gravity="center"
19 />
20 <Button
21     android:layout_width="wrap_content"
22     android:layout_height="wrap_content"
23     android:text="Salir"
```

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

# Actividad guiada - continuación

- Agregar los recursos de **string**
  - **saludarAtodos**
  - **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="saludarAtodos">Saludar a todos</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/saludarAtodos"
    android:textColor="@color/colorTextoBoton"
/>
```

<TextView

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/mostrarAqui"
    android:textSize="25sp"
    android:layout_gravity="center"
/>
```

<Button

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/salir"
    android:textColor="@color/colorTextoBoton"
/>
```

# 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 Java`. 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="saludarAtodos">Saludar a todos</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/saludarAtodos"  
    android:textColor="@color/colorTextoBoton"  
    android:onClick="saludar"
```

/>

<TextView

```
    android:id="@+id/leyenda"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/mostrarAqui"  
    android:textSize="25sp"  
    android:layout_gravity="center"
```

/>

# Solución

En el archivo  
MainActivity.java

```
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);  
    }  
}
```

# Actividad guiada - continuación

- En ocasiones es necesario acceder por **código java** 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 **getResources().getString(R.string.holaMundo)**
- Modificar la aplicación para que al presionar el botón superior, también muestre el mensaje utilizando un **Toast**

# Solución

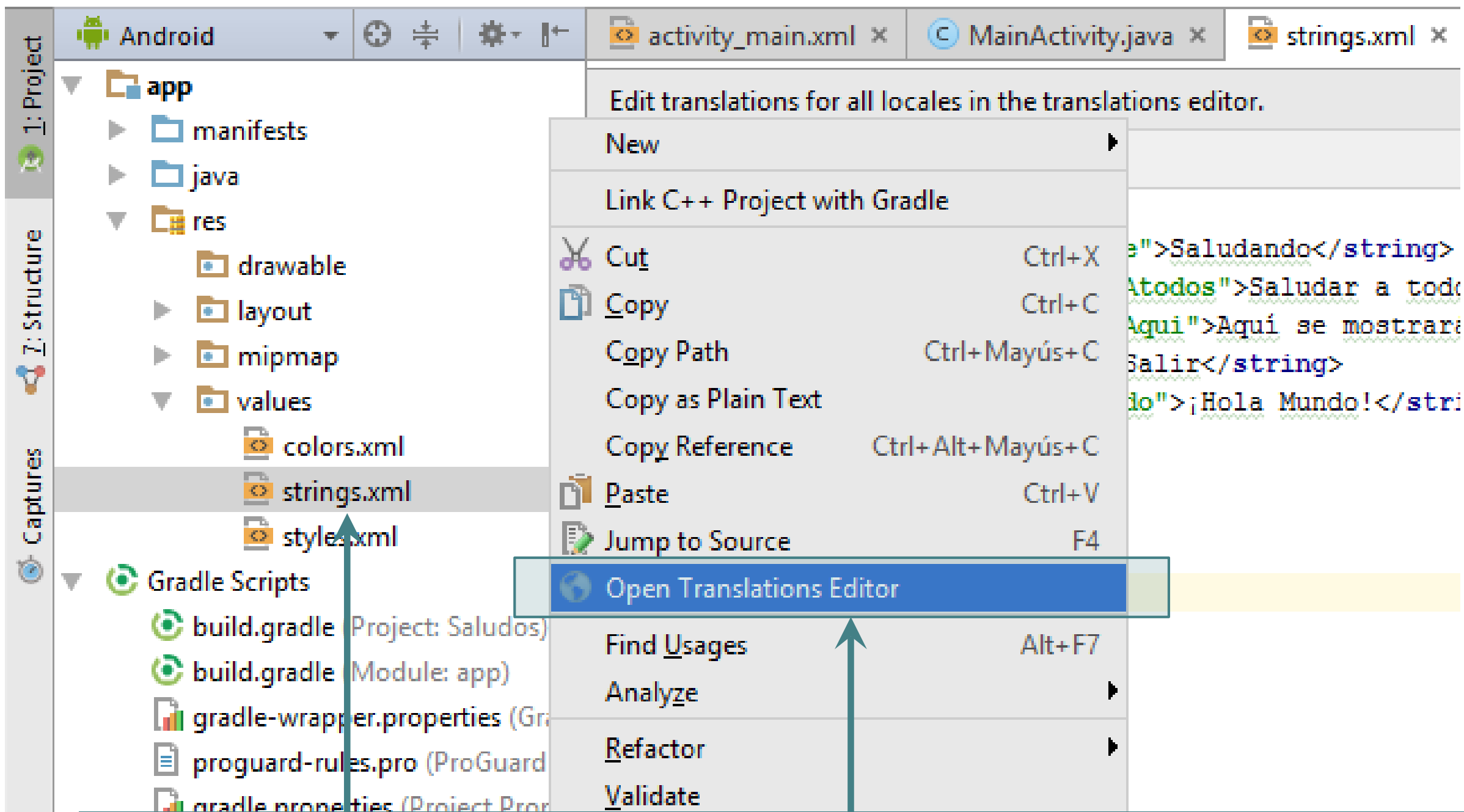
En el archivo  
MainActivity.java

```
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);  
        String st = getResources().getString(R.string.holaMundo);  
        Toast.makeText(this, st, Toast.LENGTH_SHORT).show();  
    }  
}
```



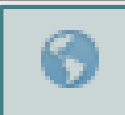
# Actividad guiada - continuación

- 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



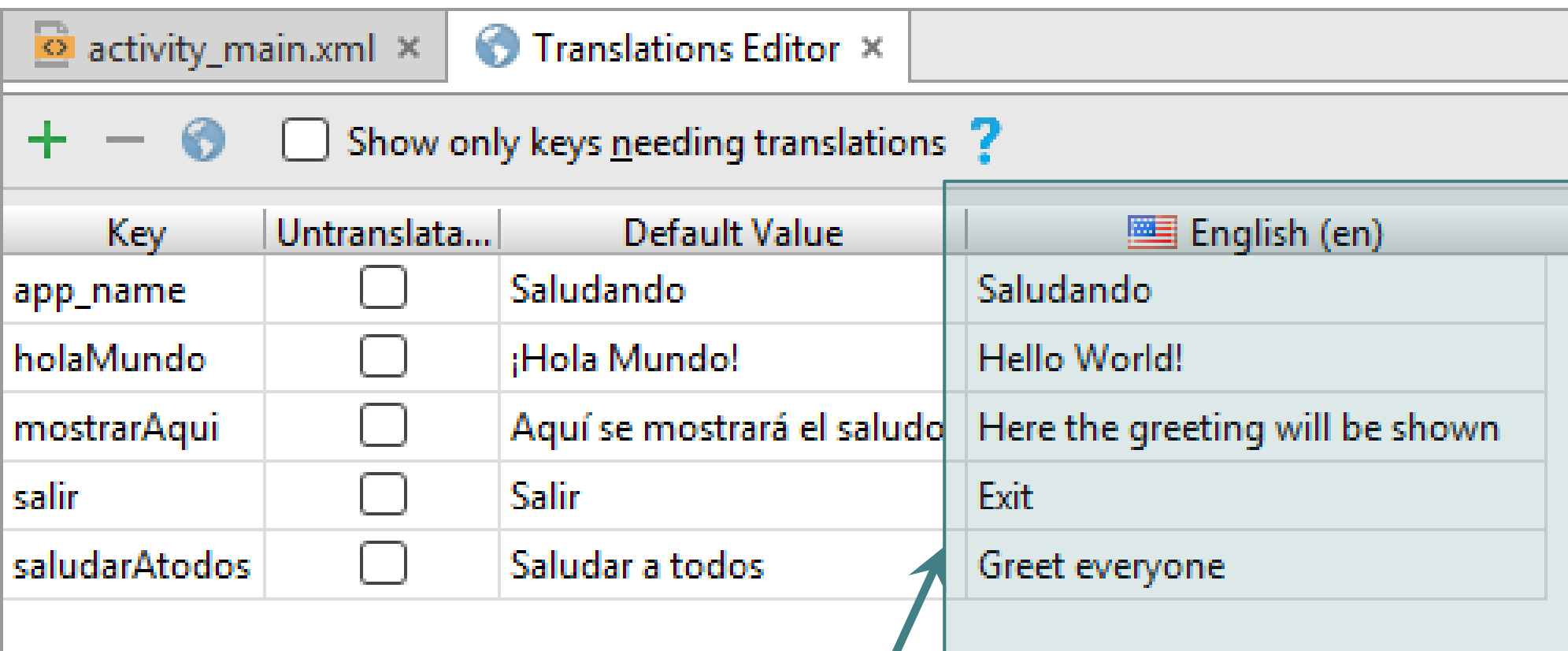
Hacer click con el botón derecho del mouse sobre **strings.xml** y seleccionar **Open Translations Editor** en el menú contextual

activity\_main.xml x MainActivity.java x Translations Editor x


+ -  ☐ Show only keys needing translations ?

Key	Untranslata...	Default Value
app_name	<input type="checkbox"/>	Saludando
holaMundo	<input type="checkbox"/>	¡Hola Mundo
mostrarAqui	<input type="checkbox"/>	Aquí se most
salir	<input type="checkbox"/>	Salir
saludarAtodos	<input type="checkbox"/>	Saludar a tod

Hacer clic en el mundo y elegir English(en)

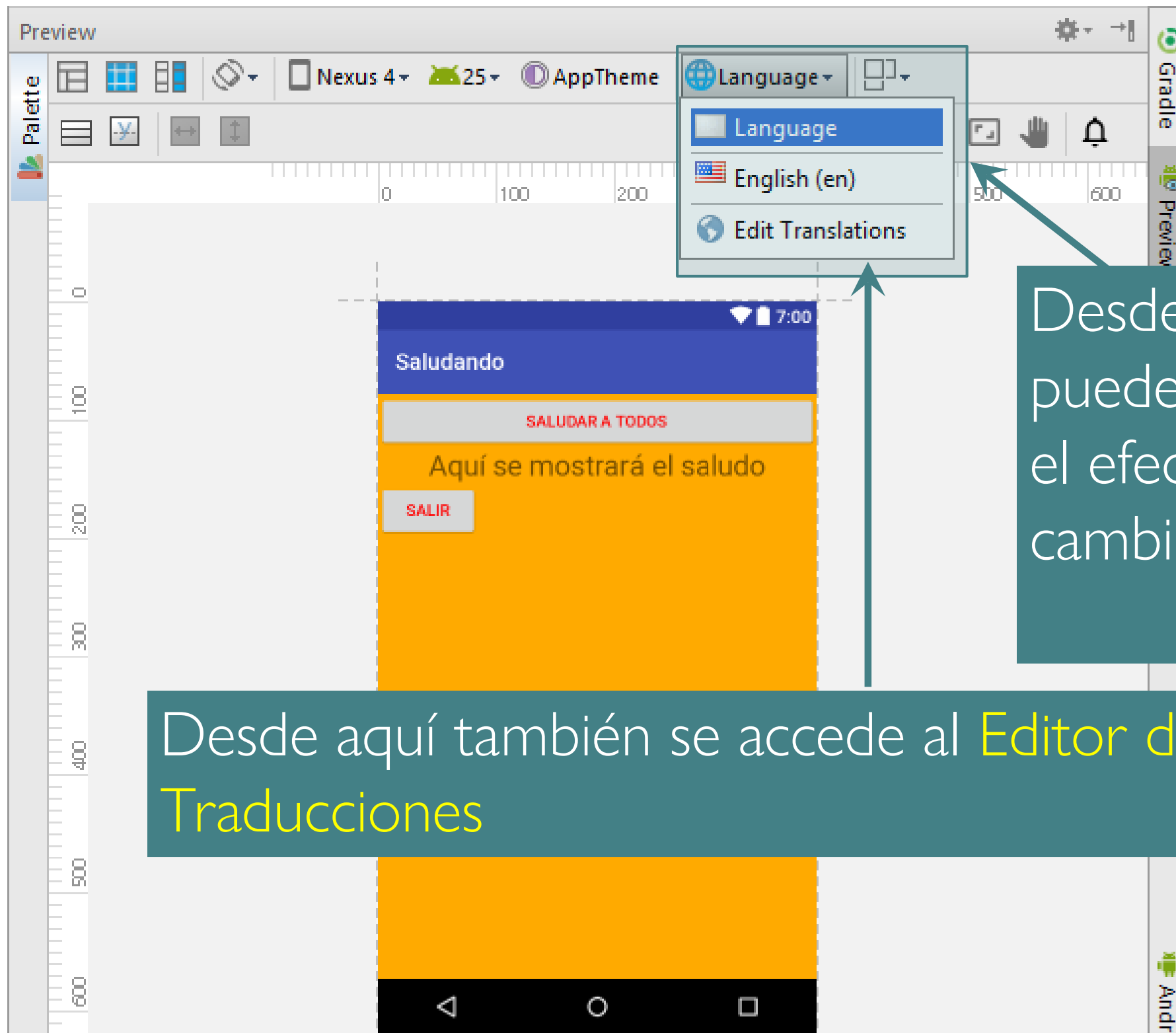


Key	Untranslata...	Default Value
app_name	<input type="checkbox"/>	Saludando
holaMundo	<input type="checkbox"/>	¡Hola Mundo!
mostrarAqui	<input type="checkbox"/>	Aquí se mostrará el saludo
salir	<input type="checkbox"/>	Salir
saludarAtodos	<input type="checkbox"/>	Saludar a todos

 English (en)  
Saludando  
Hello World!  
Here the greeting will be shown  
Exit  
Greet everyone

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 cambio de idioma

Desde aquí también se accede al **Editor de Traducciones**

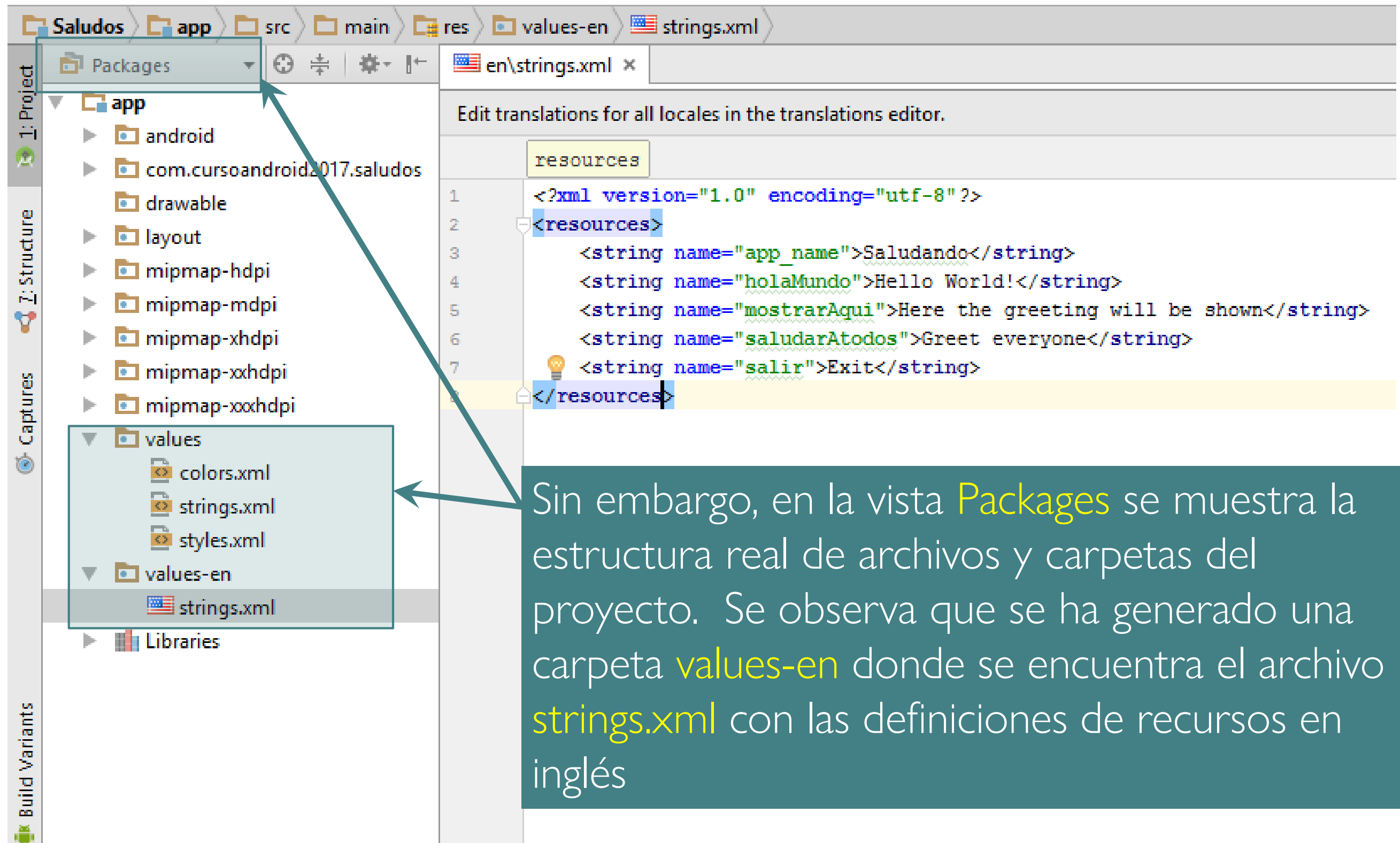
Dedicar un minuto para descubrir **en qué archivo** se están guardando los **recursos string** correspondientes al idioma **Inglés**.

The screenshot shows the Android Studio interface. On the left, the 'Project' view displays the project structure under the 'Android' tab. The 'app' folder is expanded, showing subfolders like 'manifests', 'java', and 'res'. The 'res' folder is further expanded, showing 'drawable', 'layout', 'mipmap', and 'values'. The 'strings.xml (2)' file is highlighted, with a sub-menu showing 'strings.xml' and 'strings.xml (en)'. A blue box highlights this sub-menu, and a green arrow points from it to the 'en\strings.xml' file in the main editor. The main editor shows the content of 'en\strings.xml', which is an XML file for string resources. The text in the editor is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="app_name">Saludando</string>
4     <string name="holaMundo">Hello World!</string>
5     <string name="mostrarAqui">Here the greeting will be shown</string>
6     <string name="saludarAtodos">Greet everyone</string>
7     <string name="salir">Exit</string>
8 </resources>
```

At the bottom right, a green text box contains the following text:

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



Saludos app src main res values-en strings.xml

1: Project

2: Structure

Captures

Build Variants

Resources

Edit translations for all locales in the translations editor.

```
resources
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="app_name">Saludando</string>
4     <string name="holaMundo">Hello World!</string>
5     <string name="mostrarAqui">Here the greeting will be shown</string>
6     <string name="saludarAtodos">Greet everyone</string>
7     <string name="salir">Exit</string>
8 </resources>
```

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

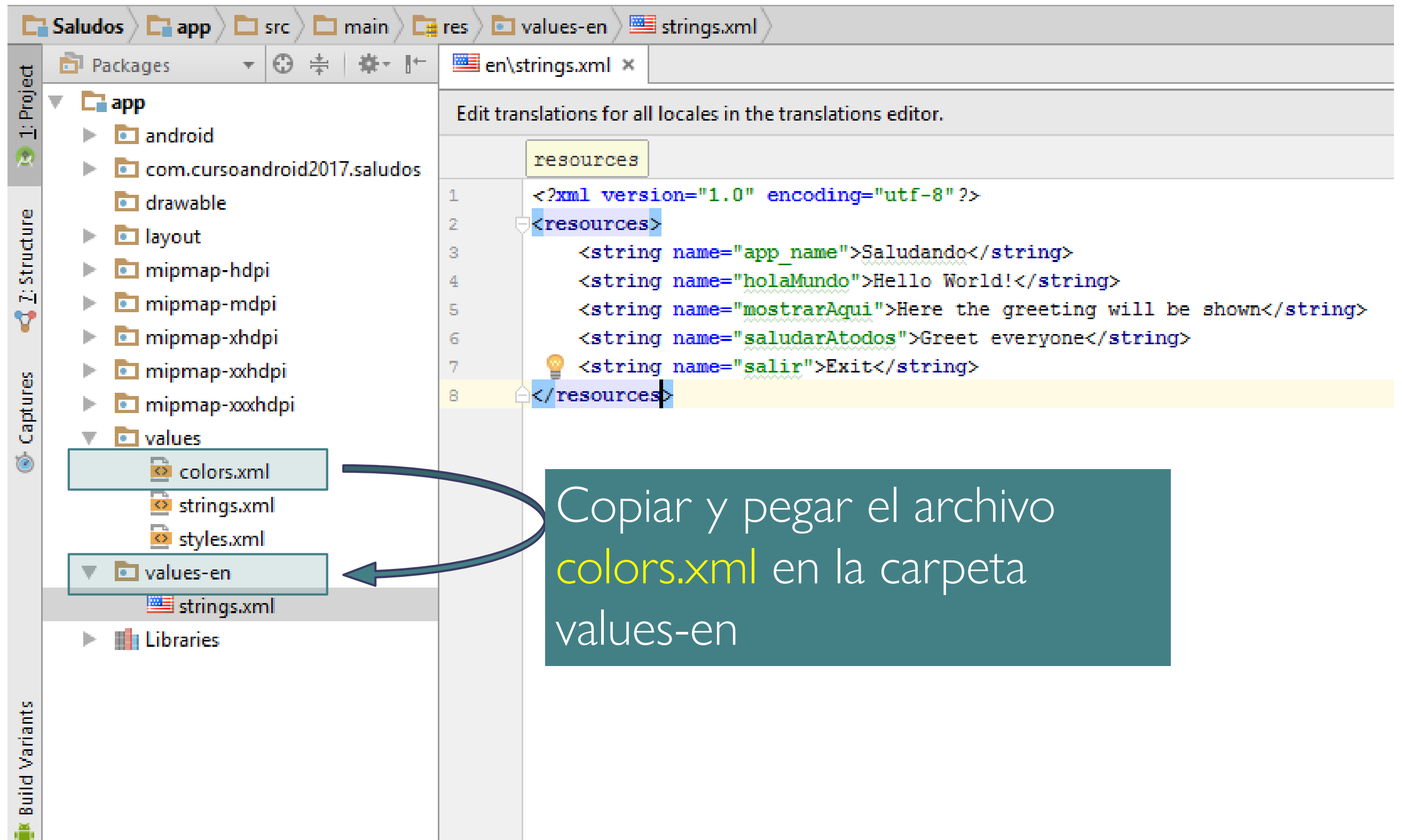
# Actividad guiada - continuación

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-en**





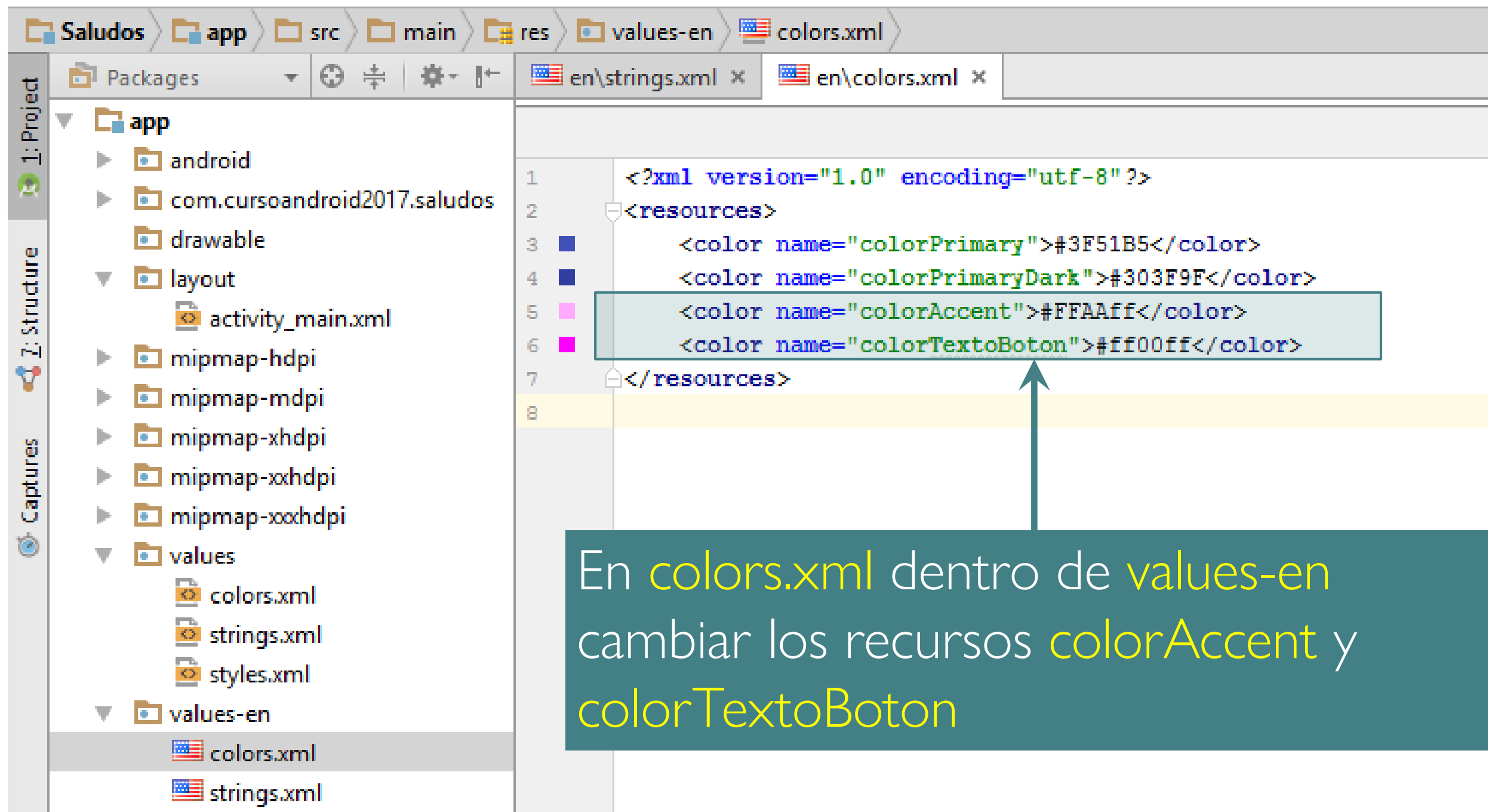
The screenshot shows the Android Studio interface. The top toolbar includes buttons for Packages, Run, Stop, Settings, and a zoom slider. The breadcrumb navigation at the top reads: Saludos > app > src > main > res > values-en > strings.xml. The left sidebar contains the Project, Structure, Captures, and Build Variants tabs. The Project tab shows the following structure:

- app
  - android
  - com.cursoandroid2017.saludos
    - drawable
    - layout
    - mipmap-hdpi
    - mipmap-mdpi
    - mipmap-xhdpi
    - mipmap-xxhdpi
    - mipmap-xxxhdpi
    - values
      - colors.xml
      - strings.xml
      - styles.xml
    - values-en
      - strings.xml
  - Libraries

The Structure tab shows the 'resources' folder selected. The main editor displays the content of 'en\strings.xml':

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="app_name">Saludando</string>
4     <string name="holaMundo">Hello World!</string>
5     <string name="mostrarAqui">Here the greeting will be shown</string>
6     <string name="saludarAtodos">Greet everyone</string>
7     <string name="salir">Exit</string>
8 </resources>
```

A callout box with a teal background and white text contains the instruction: "Copiar y pegar el archivo colors.xml en la carpeta values-en". A blue arrow points from the 'values' folder in the Project tab to the 'values-en' folder in the same tab.



The screenshot shows the Android Studio interface with the following components:

- Top Breadcrumbs:** Saludos > app > src > main > res > values-en > colors.xml
- Left Panel (Project Structure):** Shows the project hierarchy. The 'values' folder is expanded, showing 'colors.xml', 'strings.xml', and 'styles.xml'. The 'values-en' folder is also visible.
- Right Panel (Code Editor):** Displays the content of 'colors.xml' in the 'en' locale. The code is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <color name="colorPrimary">#3F51B5</color>
4   <color name="colorPrimaryDark">#303F9F</color>
5   <color name="colorAccent">#FFAAff</color>
6   <color name="colorTextoBoton">#ff00ff</color>
7 </resources>
```

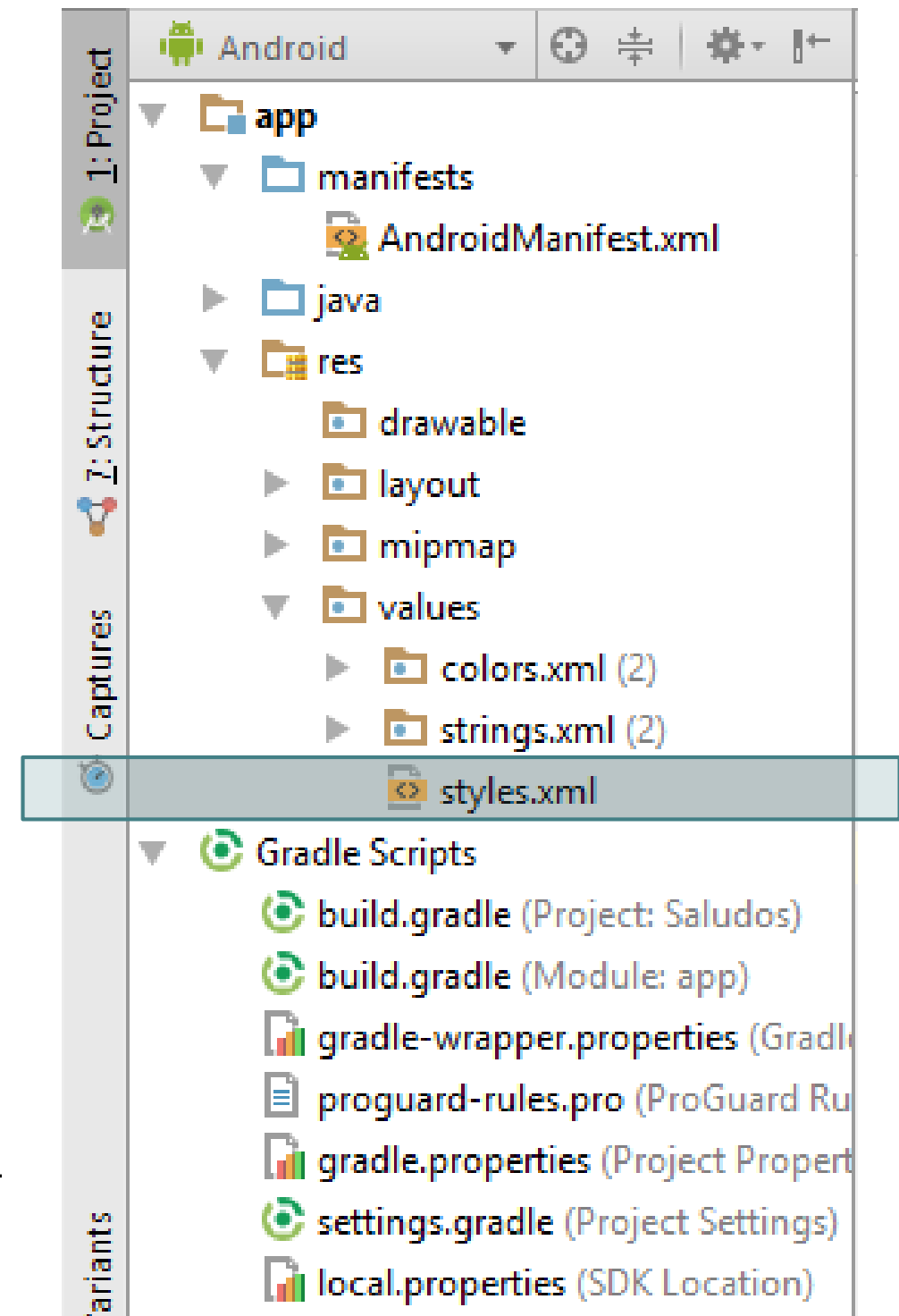
A blue rectangular box highlights lines 5 and 6 of the code, which define 'colorAccent' and 'colorTextoBoton'. A blue arrow points from this box to a text box below.

En **colors.xml** dentro de **values-en** cambiar los recursos **colorAccent** y **colorTextoBoton**

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>**

```
<!-- Base application theme. -->  
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
  <!-- Customize your theme here. -->  
  <item name="colorPrimary">@color/colorPrimary</item>  
  <item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
  <item name="colorAccent">@color/colorAccent</item>  
</style>
```

```
<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>**

Agregar el siguiente estilo en **styles.xml**

# Actividad guiada - continuación

<Button

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/saludarAtodos"
android:textColor="@color/colorTextoBoton"
android:onClick="saludar"
/>
```

<TextView

```
android:id="@+id/leyenda"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/mostrarAqui"
android:textSize="25sp"
android:layout_gravity="center"
/>
```

<Button

```
style="@style/MiEstilo"
android:text="@string/salir"
android:textColor="@color/colorTextoBoton"
/>
```

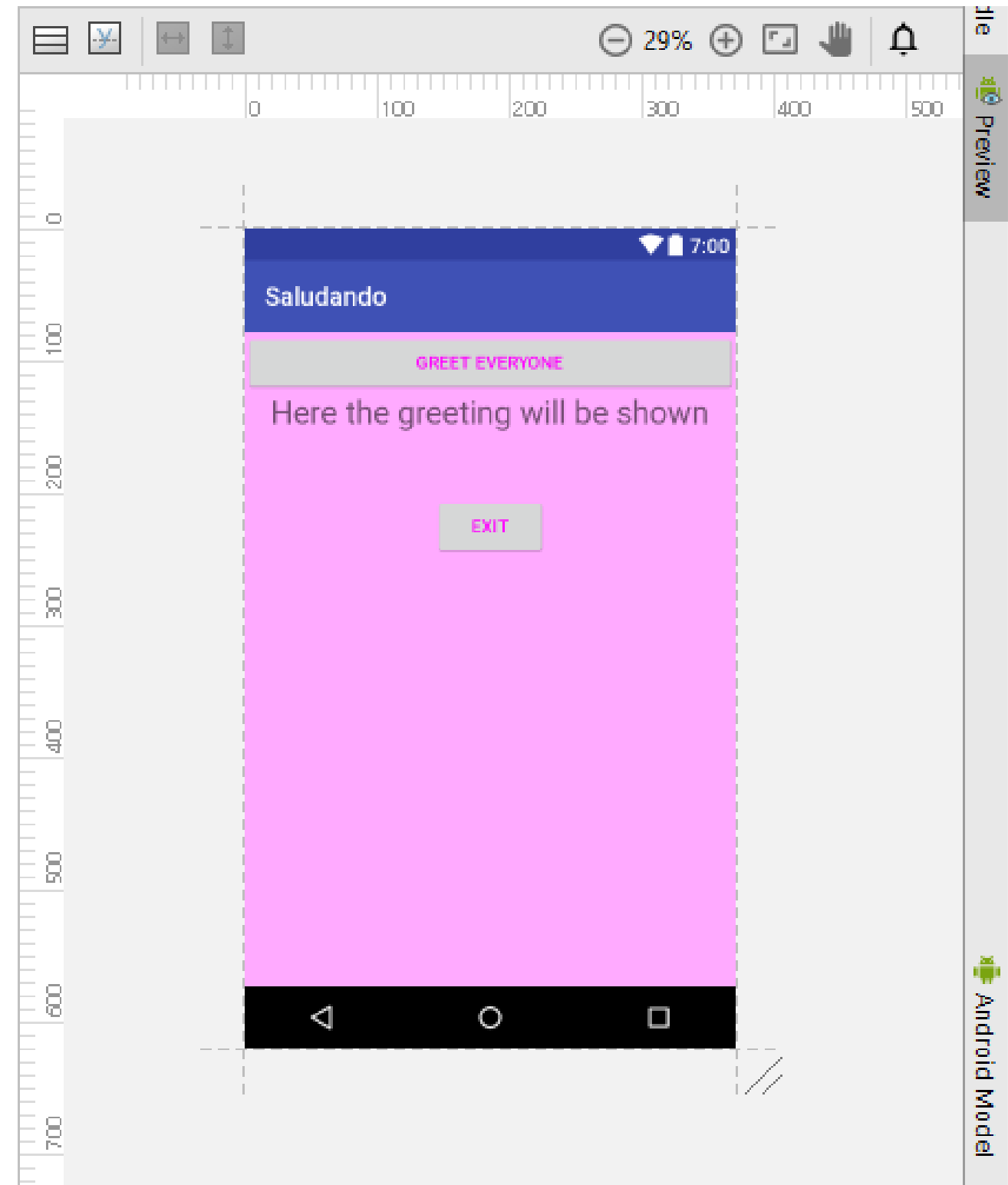
En el archivo  
activity\_main.xml

Eliminar atributos  
**layout\_width** y  
**layout\_height**  
Establecer el atributo  
**style**

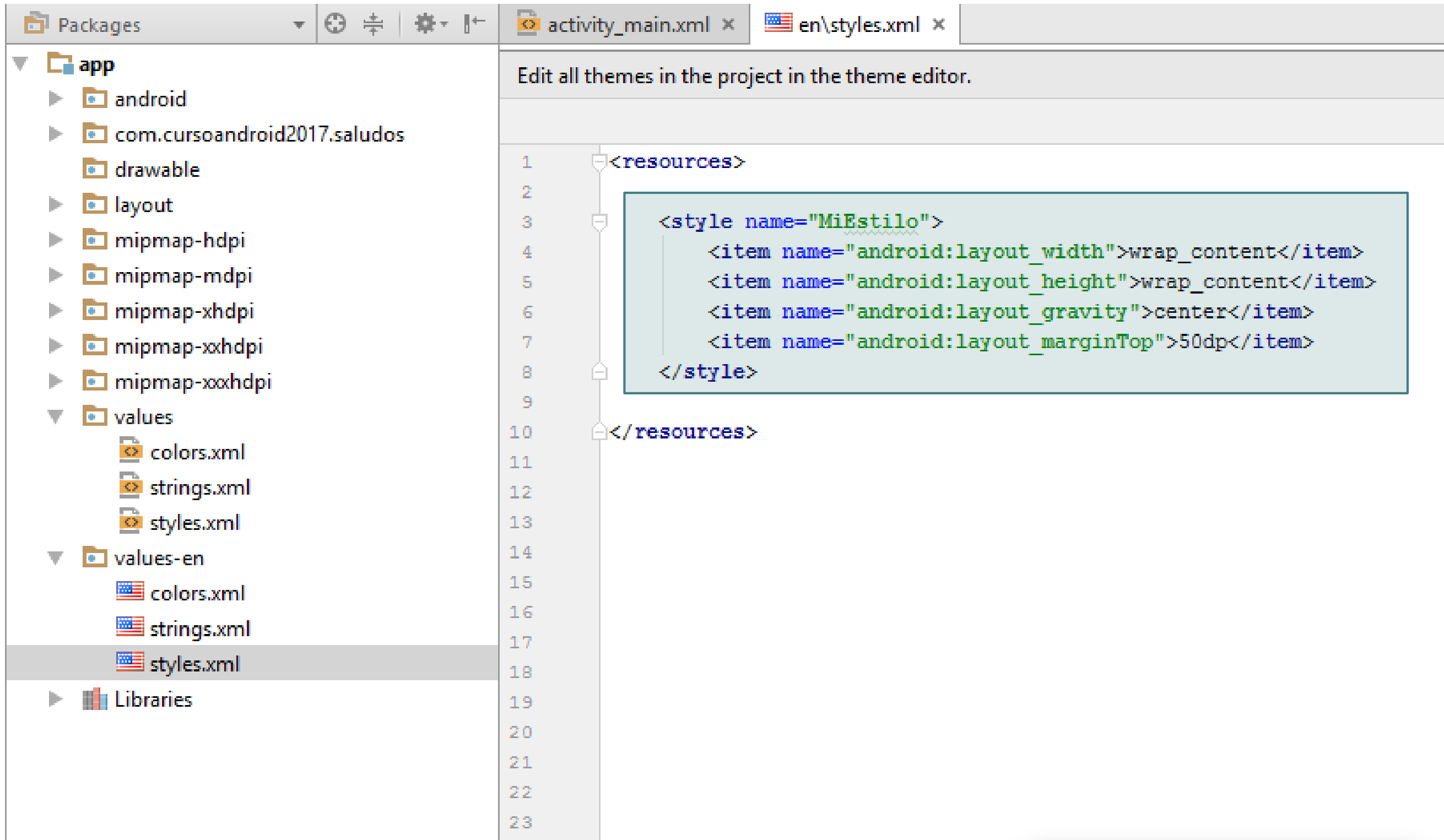
Verificar efecto  
en el panel  
Preview

# 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



The screenshot shows the Android Studio interface with the 'styles.xml' file open in the 'values-en' directory. The file contains the following XML code:

```
1 <resources>
2
3   <style name="MiEstilo">
4       <item name="android:layout_width">wrap_content</item>
5       <item name="android:layout_height">wrap_content</item>
6       <item name="android:layout_gravity">center</item>
7       <item name="android:layout_marginTop">50dp</item>
8   </style>
9
10 </resources>
```

A light blue box highlights the style definition from line 3 to line 8. The interface also shows the 'Packages' pane on the left with the project structure, including the 'app' directory and its subdirectories like 'android', 'com.cursoandroid2017.saludos', 'drawable', 'layout', 'mipmap-hdpi', 'mipmap-mdpi', 'mipmap-xhdpi', 'mipmap-xxhdpi', 'mipmap-xxxhdpi', 'values', 'values-en', and 'Libraries'.

# 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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cursoandroid2017.saludos">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="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>
</manifest>
```

# Temas - **styles.xml** predeterminado de la aplicación recién construida

**<resources>**

*<!-- Base application theme. -->*

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
  <!-- Customize your theme here. -->  
  <item name="colorPrimary">@color/colorPrimary</item>  
  <item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
  <item name="colorAccent">@color/colorAccent</item>  
</style>
```



```
<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

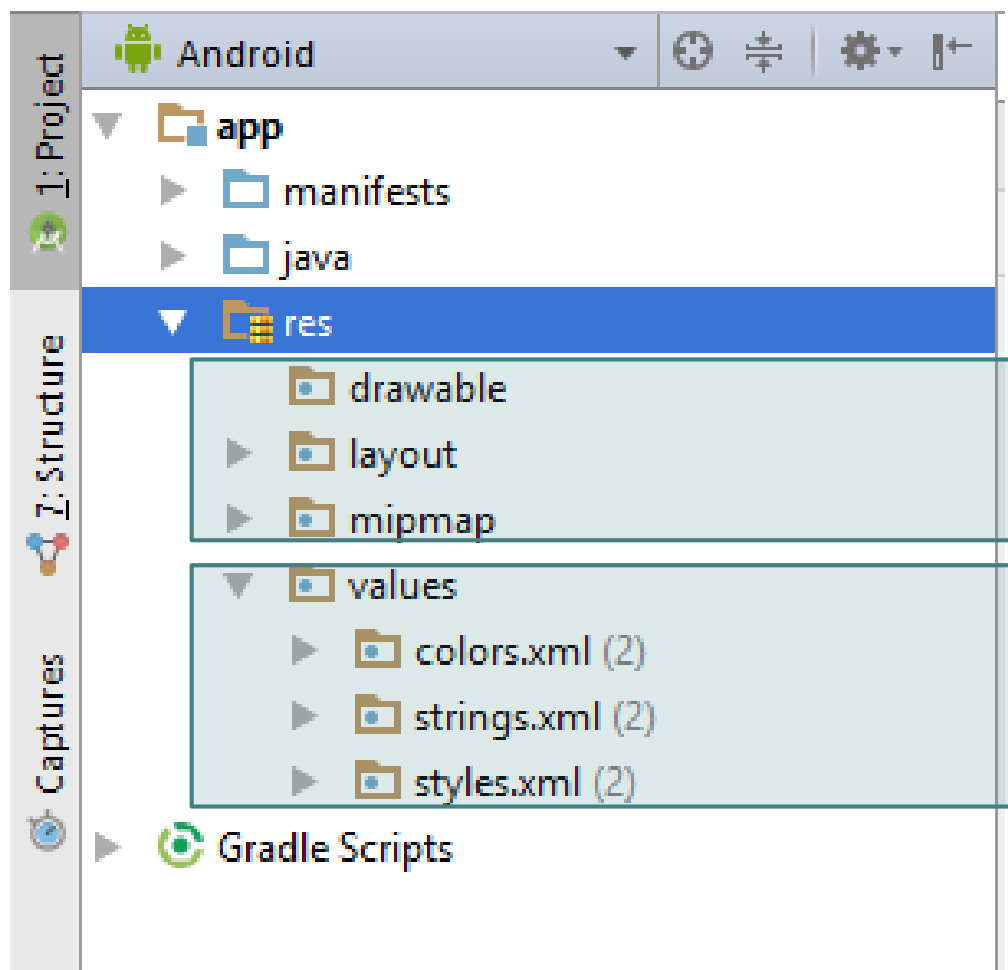
# Recapitulando

- 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** (`styles.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 JAVA

`"@dimen/alto"` en XML

# 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 JAVA

`"@integer/max_cant"` en XML



# 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 JAVA

`"@bool/lunesAbierto"` en XML

# 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 JAVA

`"@id/boton_ok"` en XML

# 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 JAVA de una activity como:

```
int[] v=getResources().getIntArray(R.array.primos);
```