

Nombre: Paulina Ugalde Carreño.

Materia: Desarrolla Aplicaciones Móviles.

Grupo: 4° “ F ”      **UNIDAD      II**

Profesora: Nancy Nieves López.

Escuela: CBTis 118

**PRÁCTICAS 21 - 27**

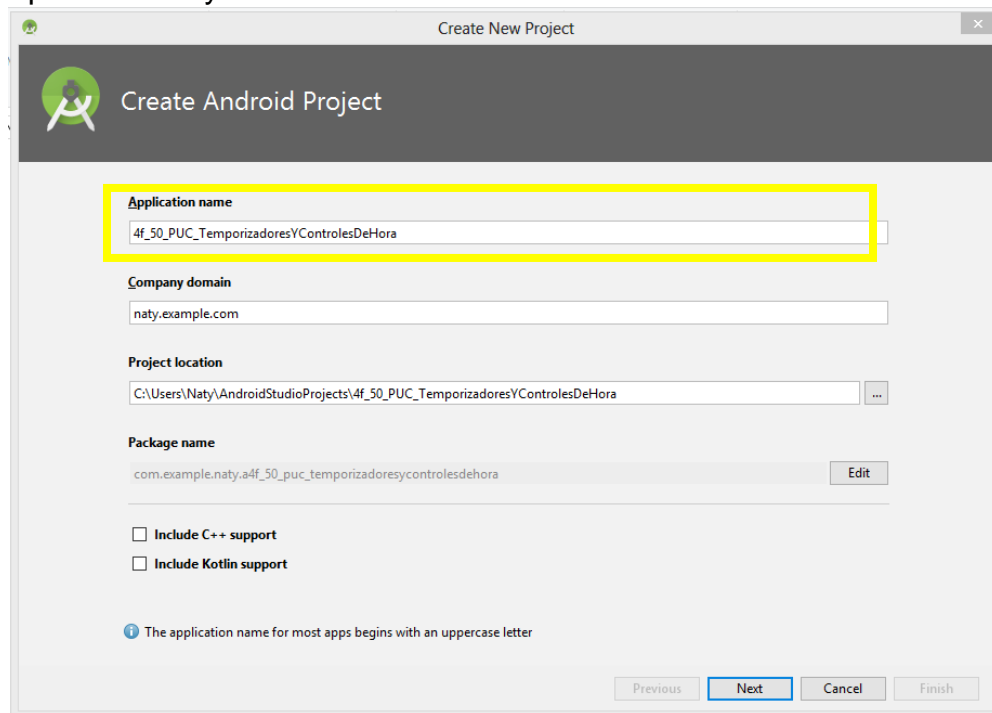
## “Prácticas 21 - 27 Android Studio”.

**Objetivo:** Familiarizarse y practicar con el entorno de Desarrollo Android Studio y Java a base de programas.

### PRÁCTICA 21 - TEMPORIZADORES Y CONTROLES DE HORA

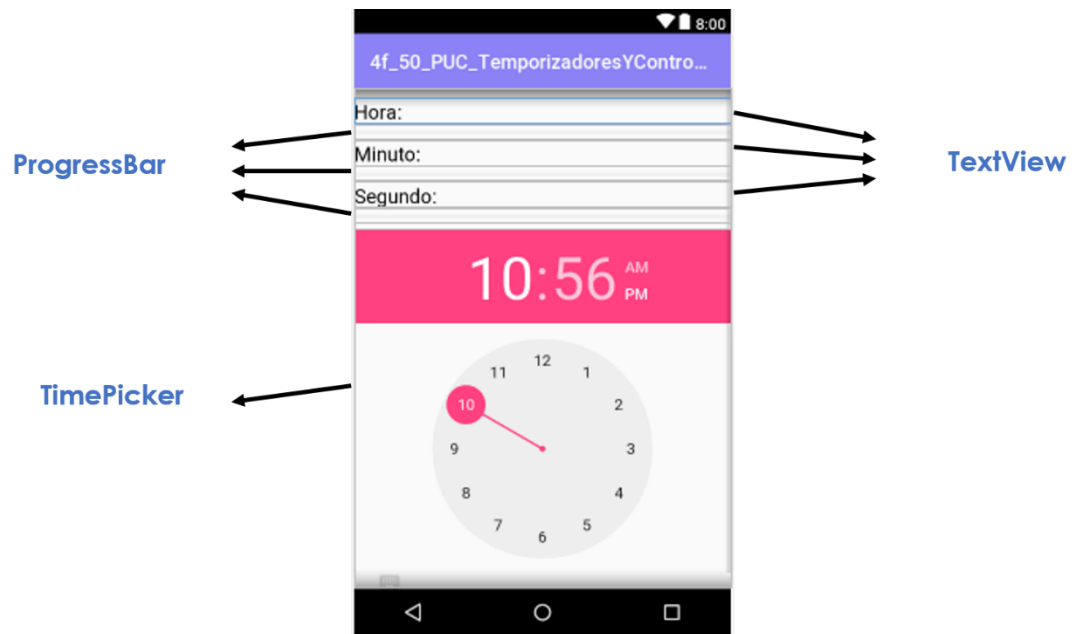
#### DISEÑO

1. **Paso 1:** Abre Android Studio y crea un nuevo proyecto con el nombre de Temporizadores y controles de hora.



**NOTA:** Como podrás notar, hemos agregado antes del nombre mi Grado y Grupo, número de lista e iniciales.

2. **Paso 2:** Agrega los siguientes componentes:



**NOTA:** Recuerda colocar su respectivo id a cada componente.

- 3. Paso 3:** O, en su defecto, agrega manualmente el código xml en la parte de Text.

```
activity_principal.xml x Principal.java x
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:orientation="vertical"
8   tools:context=".Principal">
9
10  <ScrollView
11    android:layout_width="match_parent"
12    android:layout_height="wrap_content"
13    android:orientation="vertical">
14
15    <LinearLayout
16      android:layout_width="match_parent"
17      android:layout_height="wrap_content"
18      android:orientation="vertical">
19
20      <TextView
21        android:id="@+id/txt1"
22        android:layout_height="wrap_content"
23        android:layout_width="match_parent"
24        android:layout_marginTop="10dp"
25        android:text="Hora: "
26        android:textSize="20dp"
27        android:textColor="@color/negro"/>
28
```

activity\_principal.xml x Principal.java x

28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56

```
<ProgressBar
    android:id="@+id/barra_hora"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/txt2"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="Minuto:"
    android:textSize="20dp"
    android:textColor="@color/negro"/>

<ProgressBar
    android:id="@+id/barra_minutos"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/txt3"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="Segundo:"
    android:textSize="20dp"
    android:textColor="@color/negro"/>
```

```

56
57 <ProgressBar
58     android:id="@+id/barra_segundos"
59     style="?android:attr/progressBarStyleHorizontal"
60     android:layout_width="match_parent"
61     android:layout_height="wrap_content" />
62
63 <TimePicker
64     android:layout_gravity="center"
65     android:id="@+id/control_hora"
66     android:layout_width="wrap_content"
67     android:layout_height="wrap_content"
68     android:layout_marginTop="5dp"
69     android:layout_alignParentBottom="true"
70     android:timePickerMode="clock"
71     android:layout_alignParentStart="true"></TimePicker>
72
73 </LinearLayout>
74 </ScrollView>
75 </LinearLayout>

```

Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.

### PROGRAMACIÓN

1. **Paso 1:** Es hora de ubicarnos en el archivo de nuestra clase main con su extensión .java. Agrega el siguiente código:

```
activity_principal.xml x Principal.java x
1 package com.example.naty.a4f_50_puc_temporizadoresycontrolesdehora;
2
3 import android.app.TimePickerDialog;
4 import android.os.Build;
5 import android.os.Handler;
6 import android.support.annotation.RequiresApi;
7 import android.support.v7.app.AppCompatActivity;
8 import android.os.Bundle;
9 import android.widget.ProgressBar;
10 import android.widget.TimePicker;
11
12 import java.util.Calendar;
13 import java.util.Timer;
14 import java.util.TimerTask;
15
16 public class Principal extends AppCompatActivity {
17     //Declaramos las variables globales
18     public ProgressBar barra_hora;
19     public ProgressBar barra_minutos;
20     public ProgressBar barra_segundos;
21     Timer contador;
22     public TimePicker reloj;
23     Handler handler;
24     int hora, minuto, segundo;
25     double hour, minu, seg;
26
```

```
activity_principal.xml x Principal.java x
28 @RequiresApi(api = Build.VERSION_CODES.M)
29 @Override
30 protected void onCreate(Bundle savedInstanceState) {
31     super.onCreate(savedInstanceState);
32     setContentView(R.layout.activity_principal);
33     //Referenciamos los controles gráficos
34     barra_hora = (ProgressBar) findViewById(R.id.barra_hora);
35     barra_minutos = (ProgressBar) findViewById(R.id.barra_minutos);
36     barra_segundos = (ProgressBar) findViewById(R.id.barra_segundos);
37     reloj = (TimePicker) findViewById(R.id.control_hora);
38
39     //Les asignamos los valores a nuestras variables
40     segundo = 0;
41     minuto = reloj.getMinute();
42     hora = reloj.getHour();
43
44     /*
45     Este código crea un contador a partir del objeto Timer. El temporizador se ejecutará cada segundo.
46     Cuando se produce el intervalo, ejecuta lo que haya en el método "Runnable", concretamente dentro de "run".
47     Ahí es donde obtendremos la fecha y hora del sistema y actualizaremos los valores de las barras de desplazamiento
48     y del control de hora.
49     */
50     final Runnable actualiza = () -> {
51         //Aquí el segundo comenzará a ir de uno en uno
52         segundo = segundo + 1;
53
54         //Para que cuando se llene la progressbar de los segundos, vaya aumentando un minuto
55         if(segundo==60){
56             segundo = 0;
57             minuto = minuto + 1;
58         }
59     }
```

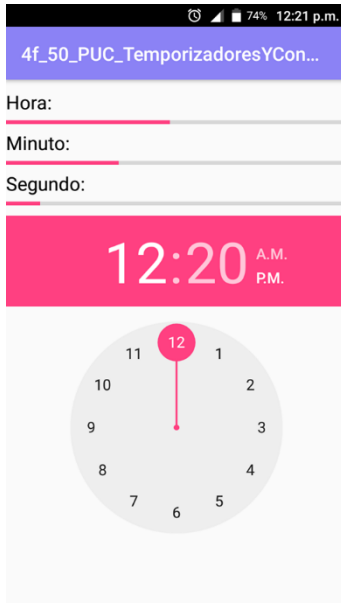
```

59     }
60
61     //Cuando se haya completado 60 minutos (es decir, una hora), se aumentará una hora en su progress bar correspondiente
62     if(minuto==60){
63         minuto = 0;
64         hora = hora +1;
65     }
66     barra_hora.setProgress(100/24*hora);
67     barra_minutos.setProgress((int) (1.6666666666666667*minuto));
68     barra_segundos.setProgress((int) (1.6666666666666667*segundo));
69     reloj.setHour(hora);
70     reloj.setMinute(minuto);
71 };
72
73
74     contador = new Timer( name: "DigitalClock");
75     contador.scheduleAtFixedRate(new TimerTask() {
76         @Override
77         public void run() {
78             runOnUiThread(actualiza);
79         }
80     }, delay: 1, period: 1000);
81

```

**Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.**

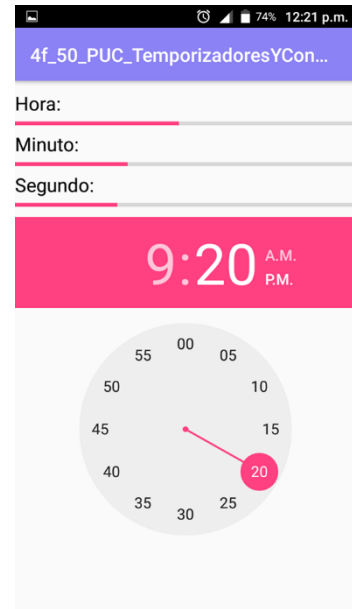
Finalmente, procederemos a ejecutar el programa:



Al iniciar la aplicación, automáticamente tomará la hora de nuestro teléfono y esta última, se verá reflejada en los progressbar



Si cambiamos la hora, **las progressbar no cambiarán.**



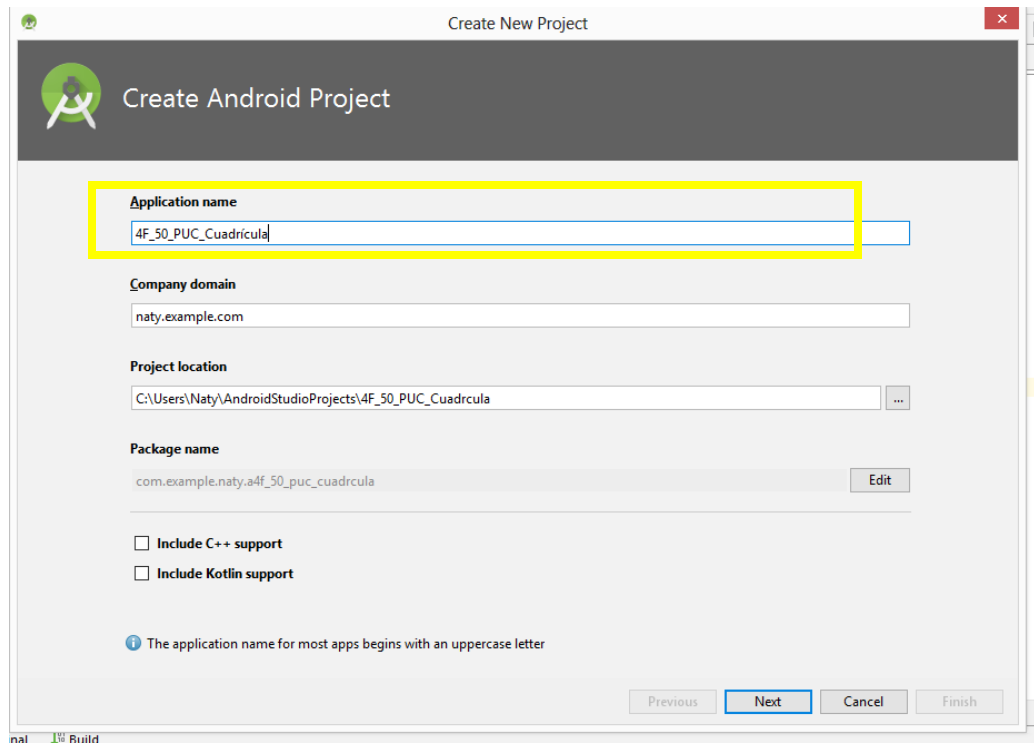
¡TÚ PUEDES MEJORAR ESTE CÓDIGO!

LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## PRÁCTICA 22 - CUADRÍCULA

### DISEÑO

1. **Paso 1:** Abre Android Studio y crea un nuevo proyecto con el nombre de Cuadrícula.

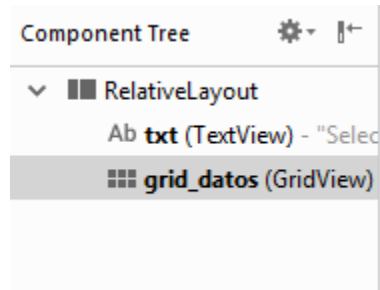


**NOTA:** Como podrás notar, hemos agregado antes del nombre mi Grado y Grupo, número de lista e iniciales.

**2. Paso 2:** Agrega los siguientes componentes:



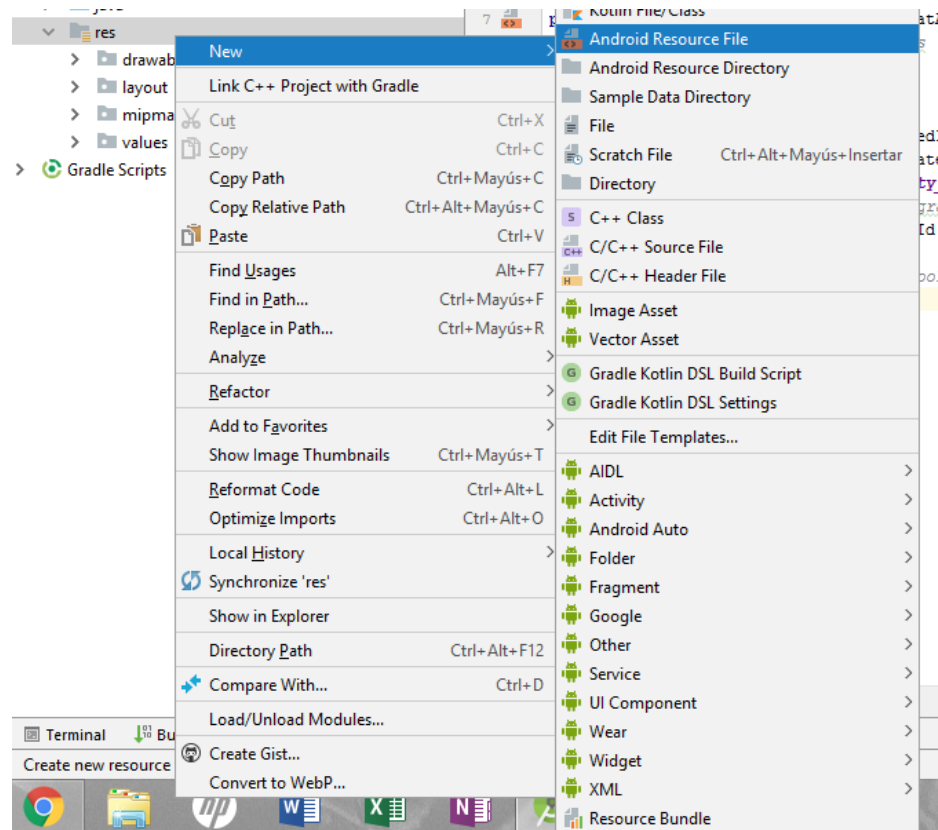
3. **Paso 3:** Coloca su respectiva id.



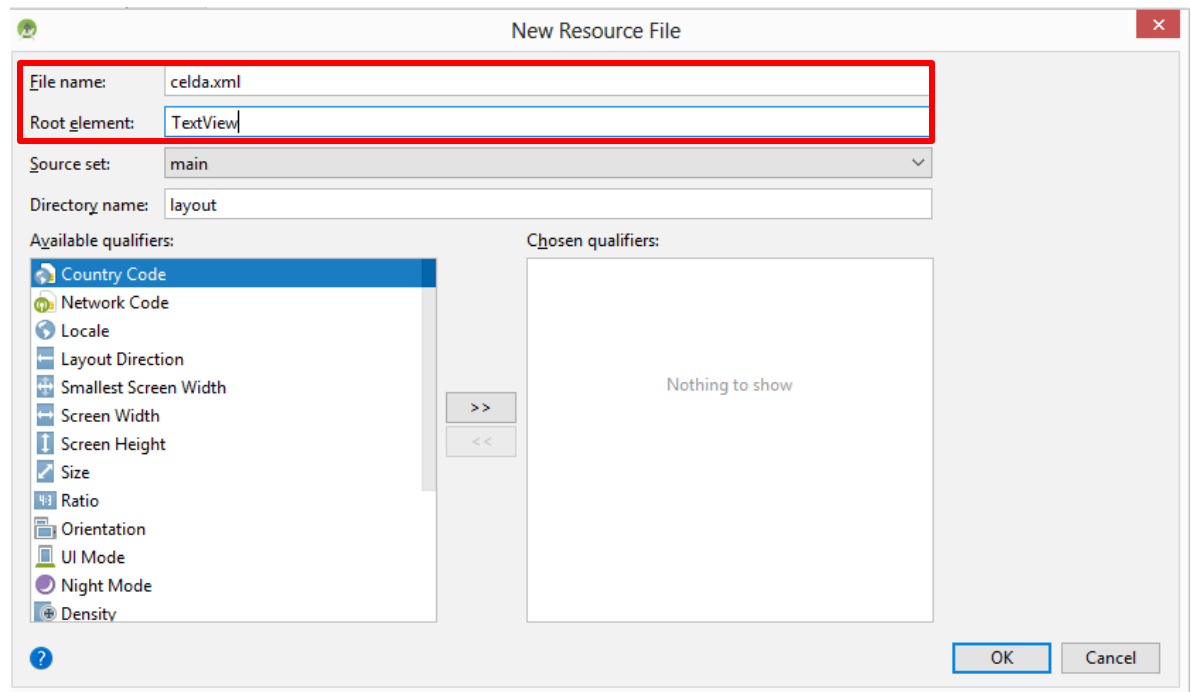
4. **Paso 4:** Colocaré el código xml de cada uno de nuestros componentes.

```
activity_principal.xml x colors.xml x Principal.java x
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".Principal">
8
9     <TextView
10         android:id="@+id/txt"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Selecciona una celda:"
14         android:textColor="@color/negro"
15         android:textSize="20dp"
16         app:layout_constraintBottom_toBottomOf="parent"
17         app:layout_constraintLeft_toLeftOf="parent"
18         app:layout_constraintRight_toRightOf="parent"
19         app:layout_constraintTop_toTopOf="parent" />
20
21     <GridView
22         android:id="@+id/grid_datos"
23         android:layout_width="match_parent"
24         android:layout_height="match_parent"
25         android:layout_alignParentStart="true"
26         android:layout_alignParentTop="true"
27         android:layout_centerHorizontal="true"
28         android:layout_marginTop="30dp"
29         android:numColumns="auto_fit" />
30
31 </RelativeLayout>
```

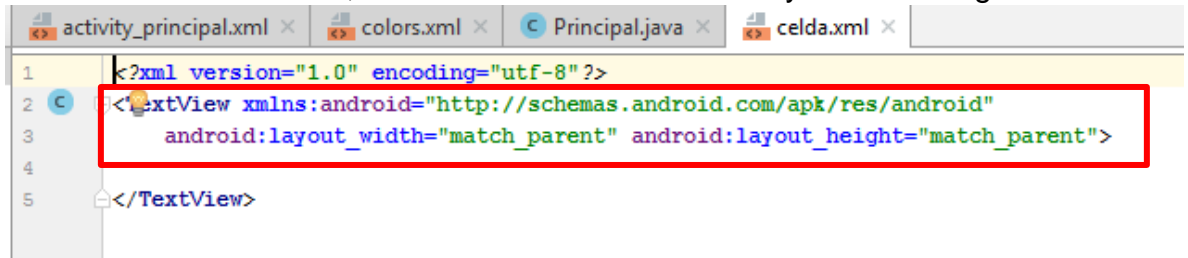
5. **Paso 5:** Ahora crearemos un nuevo archivo xml, para ello ubícate en la carpeta *res* y da clic derecho, después en *New/Android Resource File*.



6. **Paso 6:** Se te abrirá una ventana en la cual cambiaremos por la siguiente información.



7. **Paso 7:** Una vez creada, ubícate en la sección de texto y escribe lo siguiente:

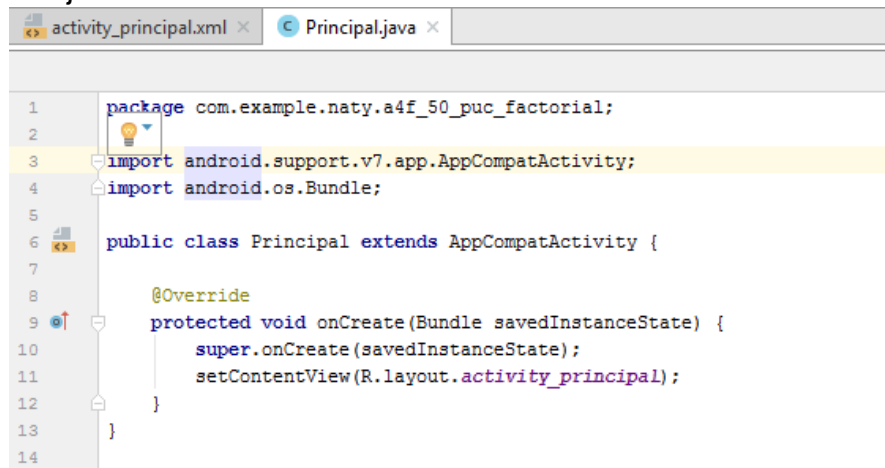


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <TextView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent" android:layout_height="match_parent">
4
5 </TextView>
```

Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.

## PROGRAMACIÓN

2. **Paso 1:** Es hora de ubicarnos en el archivo de nuestra clase main con su extensión .java.



```
1 package com.example.naty.a4f_50_puc_factorial;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class Principal extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_principal);
12     }
13 }
14
```

3. **Paso 2:** Escribe el siguiente código:

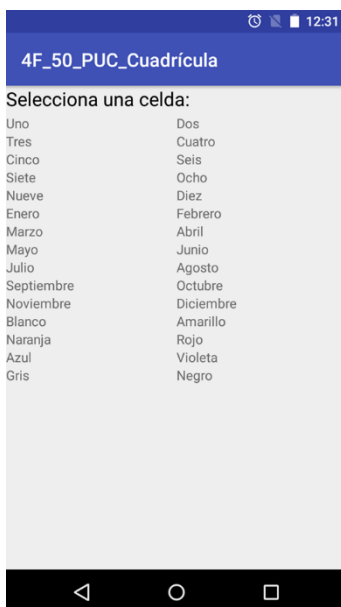
```

1 package com.example.naty.a4f_50_puc_cuadrucula;
2
3 import ...
4
10
11 public class Principal extends AppCompatActivity {
12     //Declaramos las variables globales
13     public TextView seleccion;
14     public String[] matriz = {"Uno", "Dos", "Tres", "Cuatro", "Cinco", "Seis",
15         "Siete", "Ocho", "Nueve", "Diez", "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
16         "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre", "Blanco", "Amarillo",
17         "Naranja", "Rojo", "Azul", "Violeta", "Gris", "Negro"};
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.activity_principal);
23         //Referenciamos los controles gráficos
24         seleccion = (TextView) findViewById(R.id.txt);
25         GridView g = (GridView) findViewById(R.id.grid_datos);
26
27         //Asignamos como origen los datos de la matriz
28         g.setAdapter(new ArrayAdapter<String>(context: this, R.layout.celda, matriz));
29
30         //Definimos un listener
31         g.setOnItemClickListener(new AdapterView.OnItemClickListener() {
32             @Override
33             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
34                 seleccion.setText(matriz[position]);
35             }
36         });
37     }
38 }

```

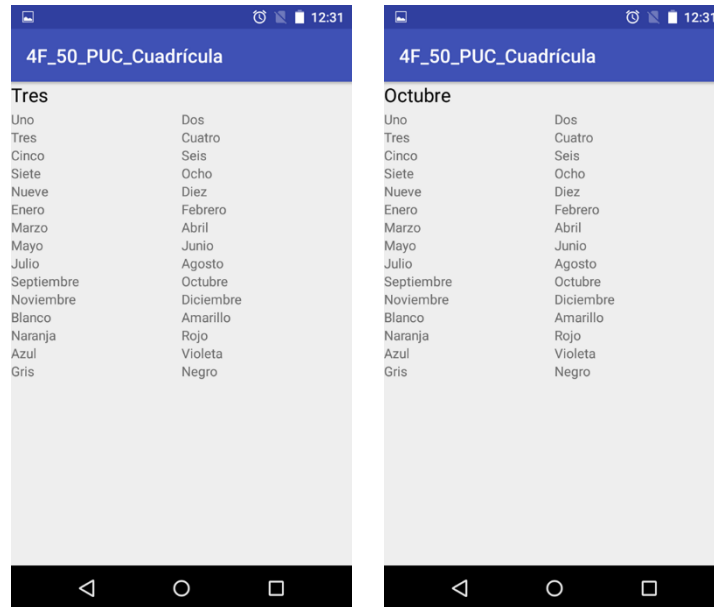
Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.

Finalmente, procederemos a ejecutar el programa:



Al seleccionar una celda (cualquier dato de la tabla), se mostrará este dato en nuestro cuadro de texto de arriba.

## EJEMPLOS:

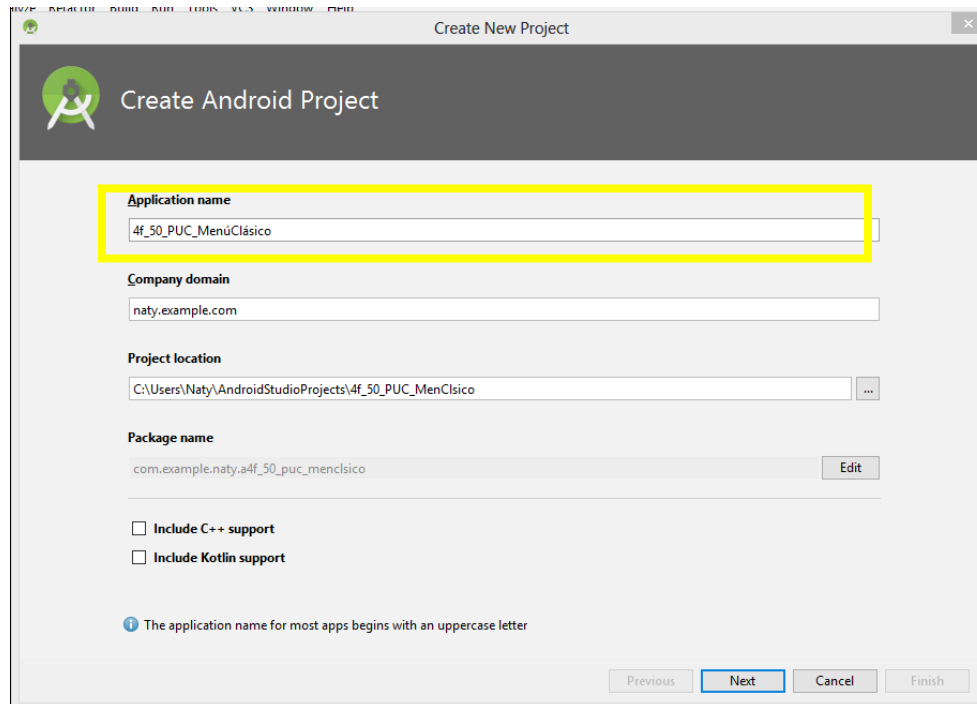


LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## PRÁCTICA 23 - SEEKBAR

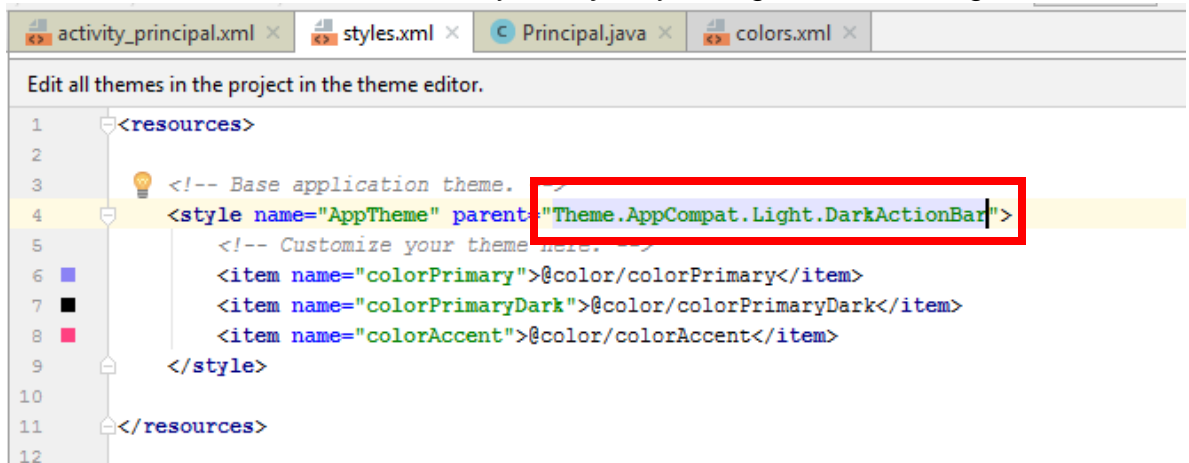
### DISEÑO

- Paso 1:** Abre Android Studio y crea un nuevo proyecto con el nombre de SeekBar.



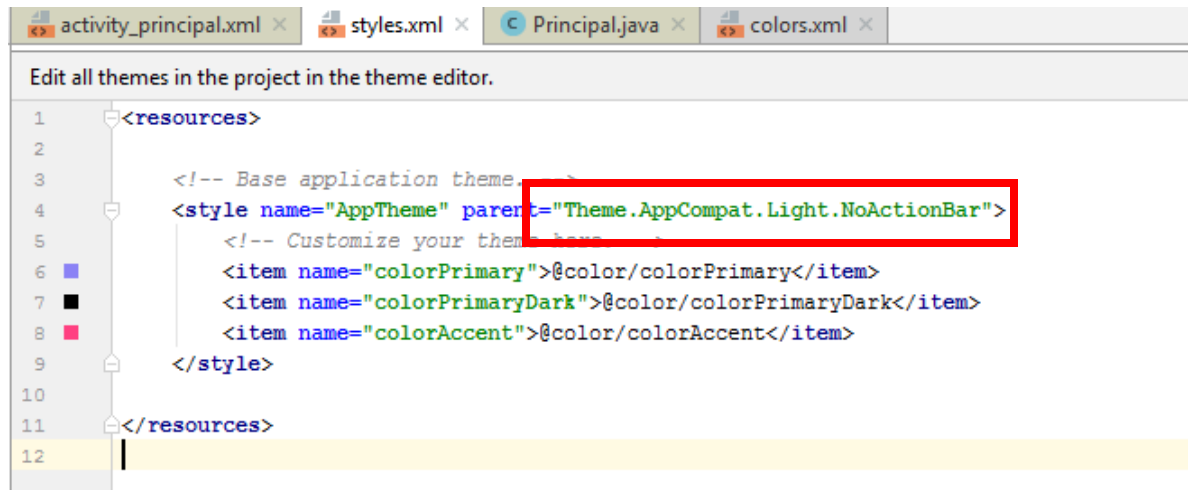
**NOTA:** Como podrás notar, hemos agregado antes del nombre mi Grado y Grupo, número de lista e iniciales.

**2. Paso 2:** Ahora iremos a nuestro layout *styles* y configuraremos lo siguiente:



Y cambiaremos por:

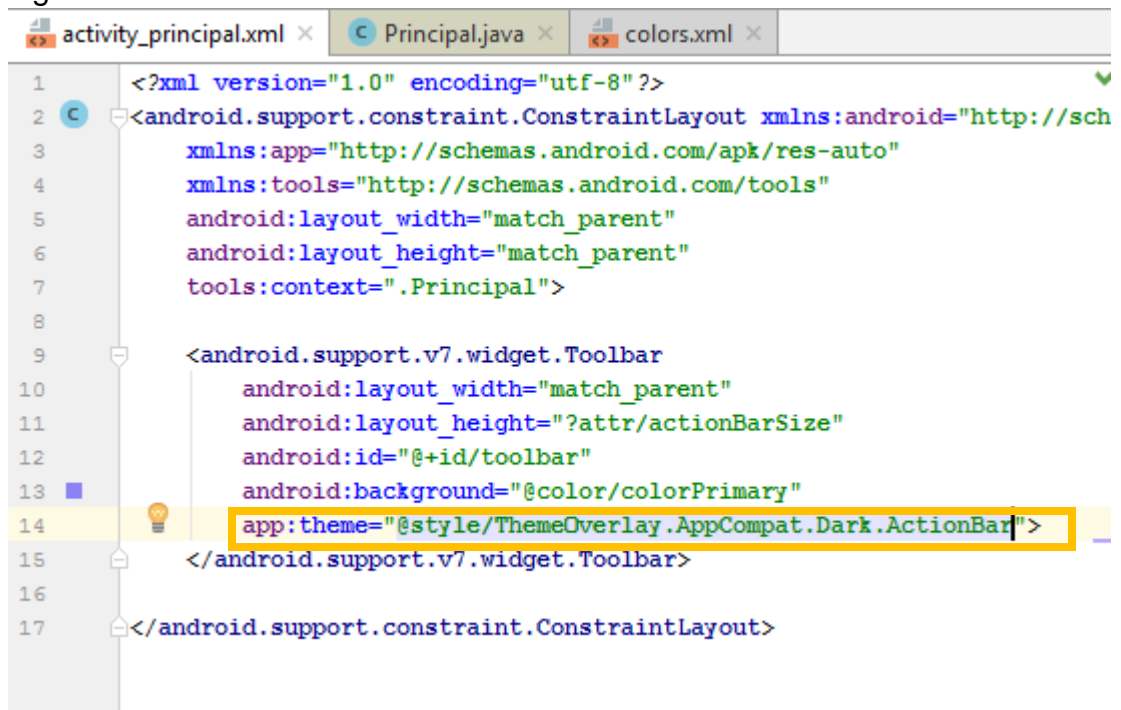




The screenshot shows the Android Studio interface with the 'activity\_principal.xml' file open. The 'styles.xml' tab is also visible. The code in 'activity\_principal.xml' defines a theme named 'AppTheme' that inherits from 'Theme.AppCompat.Light.NoActionBar'. The theme is customized with three items: 'colorPrimary' (referencing '@color/colorPrimary'), 'colorPrimaryDark' (referencing '@color/colorPrimaryDark'), and 'colorAccent' (referencing '@color/colorAccent'). A red box highlights the parent theme attribute.

```
1 <resources>
2
3 <!-- Base application theme. -->
4 <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
5     <!-- Customize your theme here. -->
6     <item name="colorPrimary">@color/colorPrimary</item>
7     <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
8     <item name="colorAccent">@color/colorAccent</item>
9 </style>
10
11 </resources>
12
```

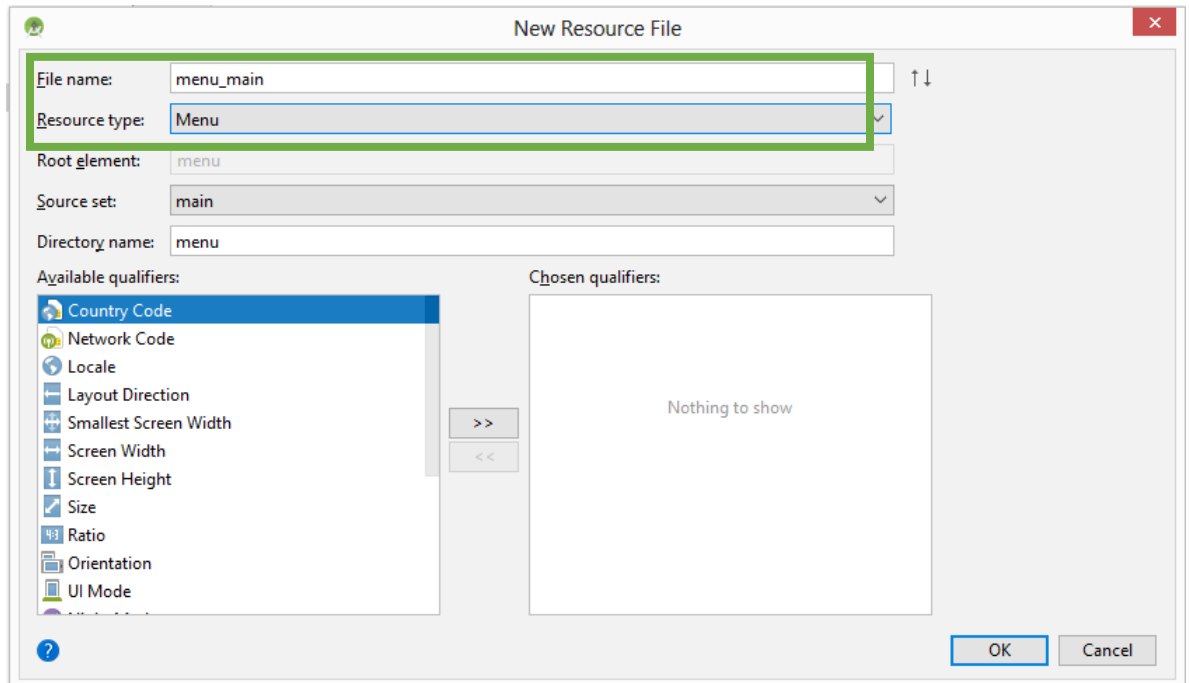
- Paso 3:** He iremos al xml de nuestra actividad principal y agregaremos lo siguiente:



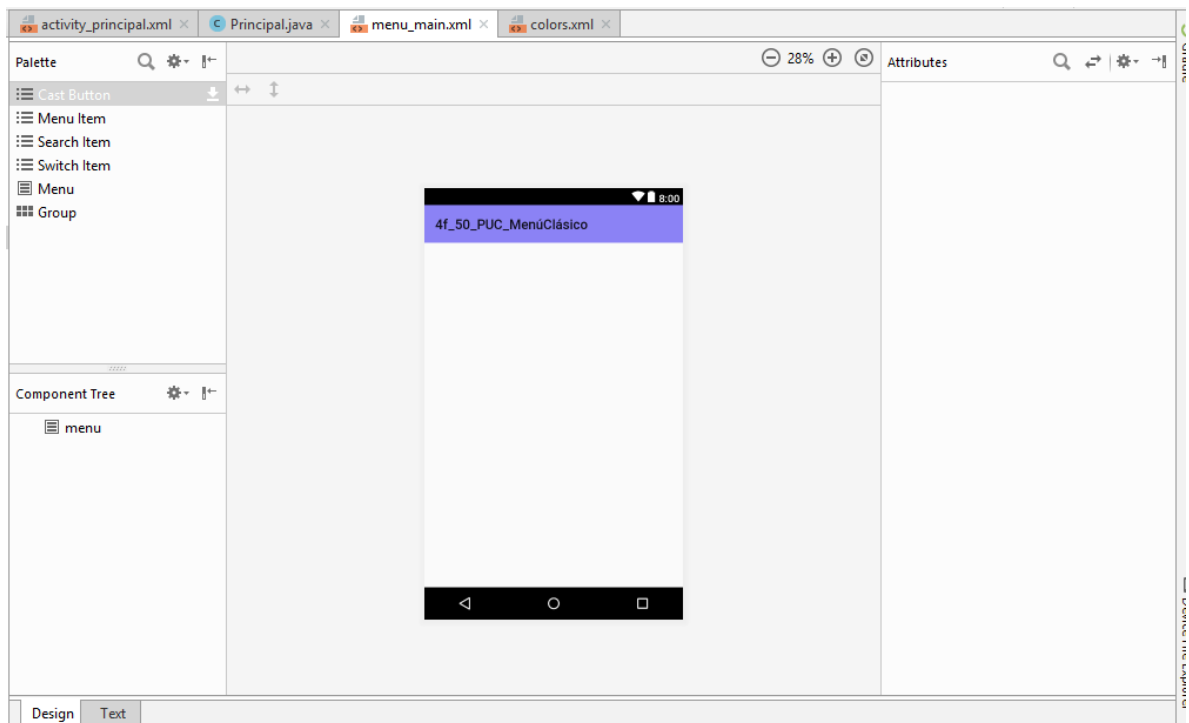
The screenshot shows the 'activity\_principal.xml' file in the Android Studio XML editor. The code defines a 'ConstraintLayout' containing a 'Toolbar' widget. The 'Toolbar' widget is configured with attributes for layout width and height, id, background color, and theme. A yellow box highlights the 'app:theme' attribute, which is set to '@style/ThemeOverlay.AppCompat.Dark.ActionBar'. A lightbulb icon is visible next to the 'app:theme' attribute, indicating a suggestion or warning.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".Principal">
8
9     <android.support.v7.widget.Toolbar
10         android:layout_width="match_parent"
11         android:layout_height="?attr/actionBarSize"
12         android:id="@+id/toolbar"
13         android:background="@color/colorPrimary"
14         app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
15 </android.support.v7.widget.Toolbar>
16
17 </android.support.constraint.ConstraintLayout>
```

- Paso 4:** Ahora nos ubicaremos en la carpeta *res/Android resource file* y nos abrirá una ventana, de la cual cambiaremos lo siguiente:



5. **Paso 5:** Una vez que hayamos dado clic en ok, se nos abrirá la siguiente ventana:

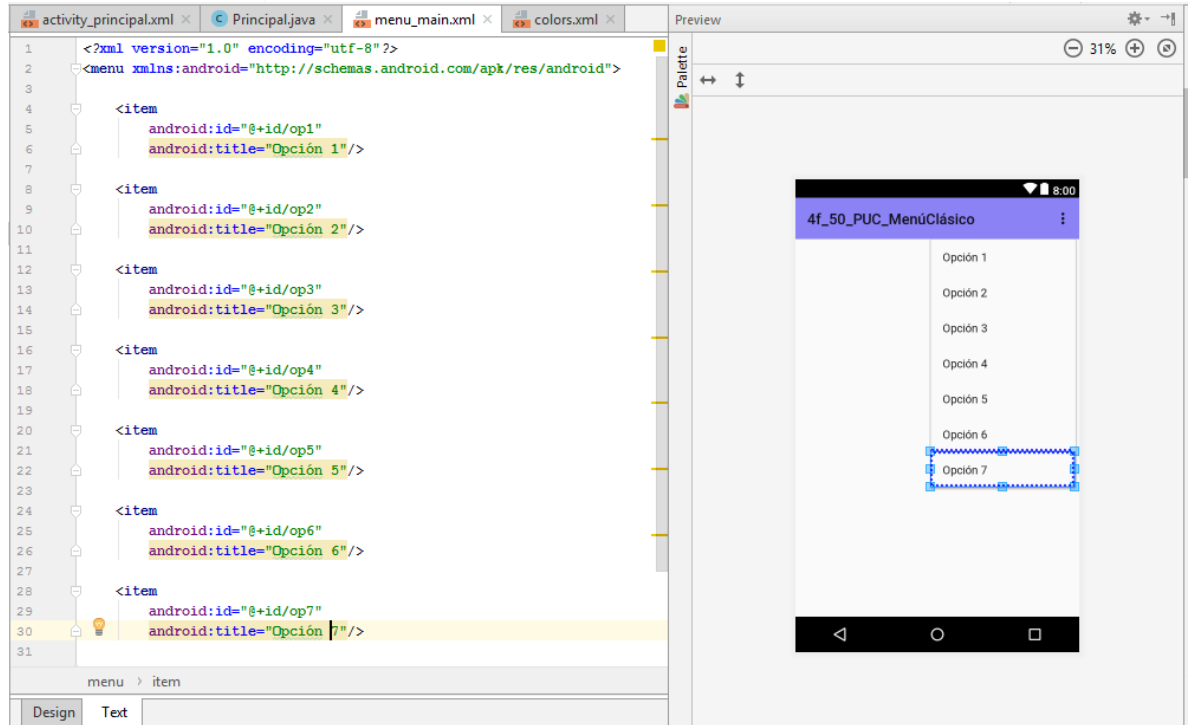


Y nos ubicaremos en la parte de Text (ubicada en la esquina inferior izquierda de la imagen).

Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.

## PROGRAMACIÓN

1. **Paso 1:** Una vez ubicados en la sección de Text del menú anterior, escribiremos lo siguiente:



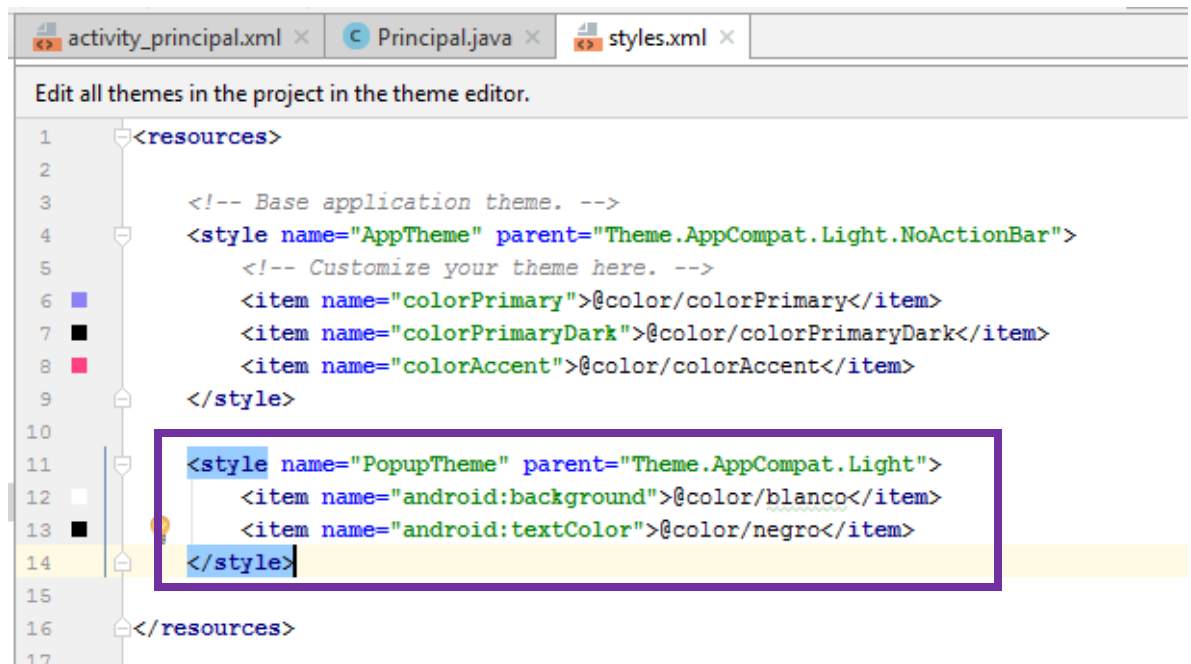
2. **Paso 2:** Es hora de ubicarnos en el archivo de nuestra clase main con su extensión .java; escribiremos lo siguiente:

```
activity_principal.xml x Principal.java x menu_main.xml x
1 package com.example.naty.a4f_50_puc_mencsico;
2
3 +import ...
9
10 public class Principal extends AppCompatActivity {
11     //Declaramos las variables globales
12     public Toolbar toolbar;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_principal);
18         //Referenciamos los controles gráficos
19         toolbar = (Toolbar) findViewById(R.id.toolbar);
20
21         //Asignamos el soporte de la toolbar
22         setSupportActionBar(toolbar);
23     }
24
25     @Override
26     public boolean onCreateOptionsMenu(Menu menu) {
27         //Obtenemos los items previamente declarados en el menu_main
28         getMenuInflater().inflate(R.menu.menu_main, menu);
29         return true;
30     }
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    //Aquí verificamos cuál item del menu_main está seleccionado y agregamos un Toast que diga que opción has seleccionado
    if(item.getItemId() == R.id.op1){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 1", Toast.LENGTH_SHORT).show();
    }else if(item.getItemId() == R.id.op2){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 2", Toast.LENGTH_SHORT).show();
    }else if(item.getItemId() == R.id.op3){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 3", Toast.LENGTH_SHORT).show();
    }else if(item.getItemId() == R.id.op4){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 4", Toast.LENGTH_SHORT).show();
    }else if(item.getItemId() == R.id.op5){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 5", Toast.LENGTH_SHORT).show();
    }else if(item.getItemId() == R.id.op6){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 6", Toast.LENGTH_SHORT).show();
    }else if(item.getItemId() == R.id.op7){
        Toast.makeText( context: this, text: "Ha hecho clic en la Opción 7", Toast.LENGTH_SHORT).show();
    }
    return super.onOptionsItemSelected(item);
}
```

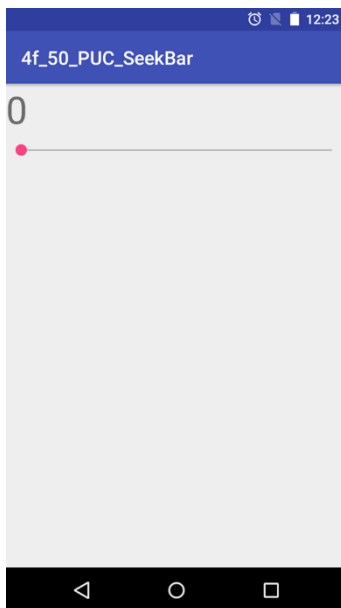
3. Paso 3: Y volveremos a *styles* y añadiremos un nuevo estilo:



```
1 <resources>
2
3 <!-- Base application theme. -->
4 <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
5 <!-- Customize your theme here. -->
6 <item name="colorPrimary">@color/colorPrimary</item>
7 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
8 <item name="colorAccent">@color/colorAccent</item>
9 </style>
10
11 <style name="PopupTheme" parent="Theme.AppCompat.Light">
12 <item name="android:background">@color/blanco</item>
13 <item name="android:textColor">@color/negro</item>
14 </style>
15
16 </resources>
17
```

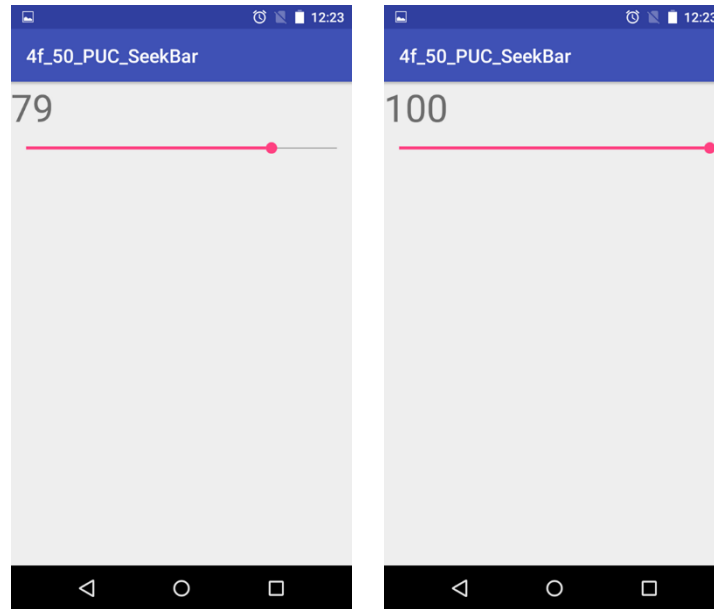
Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.

Finalmente, procederemos a ejecutar el programa:



Al iniciar la aplicación, nuestra barra y nuestro cuadro de texto (automáticamente) inicializarán en cero. Al mover la barra, el cuadro de texto cambiará.

 EJEMPLOS:

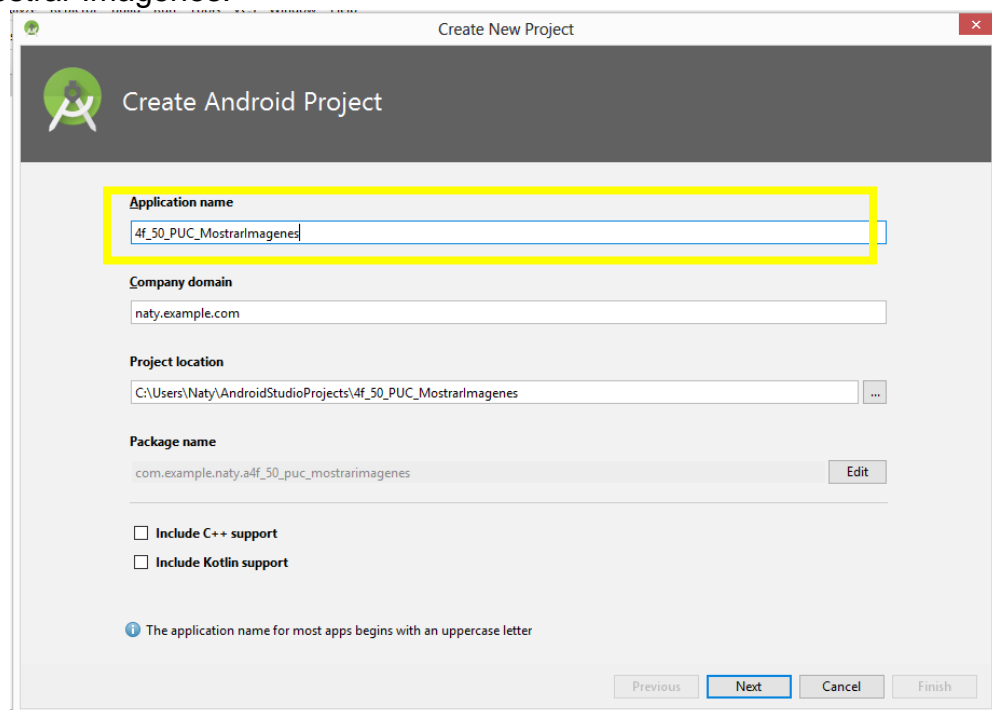


LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## PRÁCTICA 24 – MOSTRAR IMAGENES

### DISEÑO

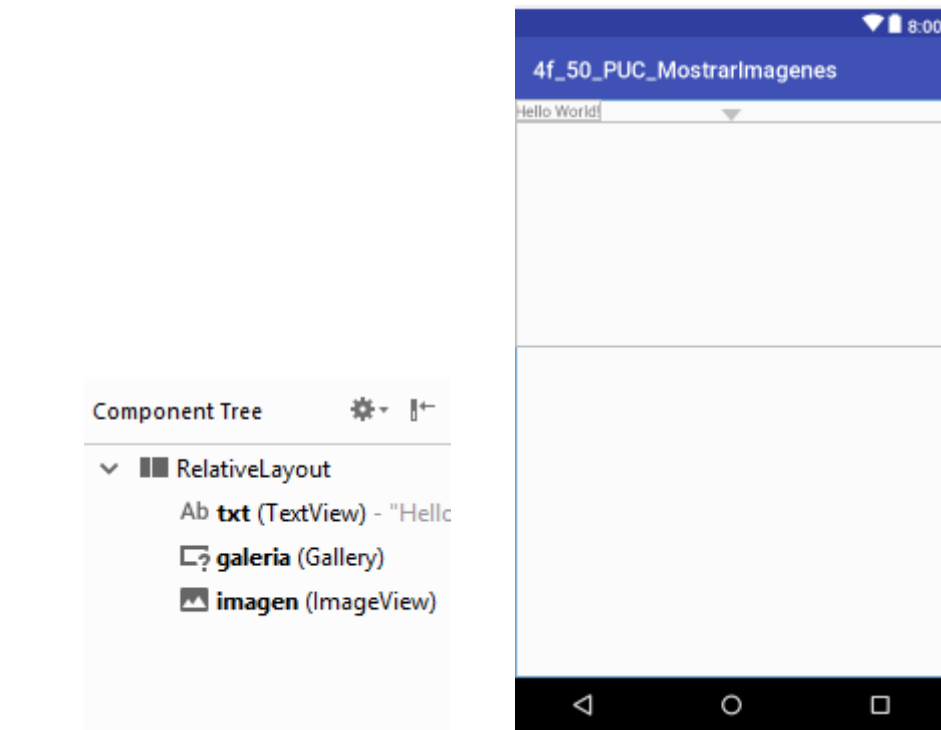
1. **Paso 1:** Abre Android Studio y crea un nuevo proyecto con el nombre de Mostrar Imágenes.



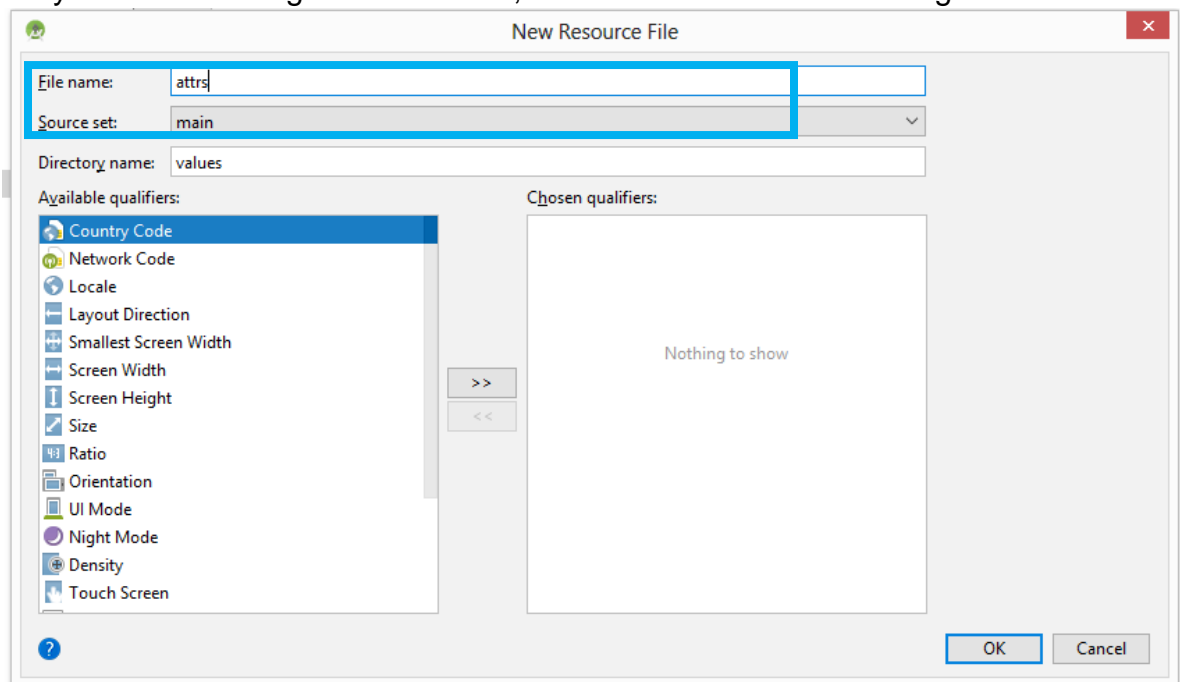
**NOTA:** Como podrás notar, hemos agregado antes del nombre mi Grado y Grupo, número de lista e iniciales.

- Paso 2:** Ve al archivo con extensión .xml principal en la parte de *Text* y agrega los siguientes componentes.

- Paso 3:** Cuando termines, tus componentes deberían verse de la siguiente manera en la sección de Design.



4. **Paso 4:** Nos ubicaremos en la carpeta *res/clic derecho/new layout resource file* y nos abrirá la siguiente ventana, en la cual cambiaremos lo siguiente:



5. **Paso 5:** Una vez que hayamos pulsado sobre el botón OK, nos abrirá una nueva ventana, en la cual, nos posicionaremos en la parte de *Text* y escribiremos la siguiente línea:

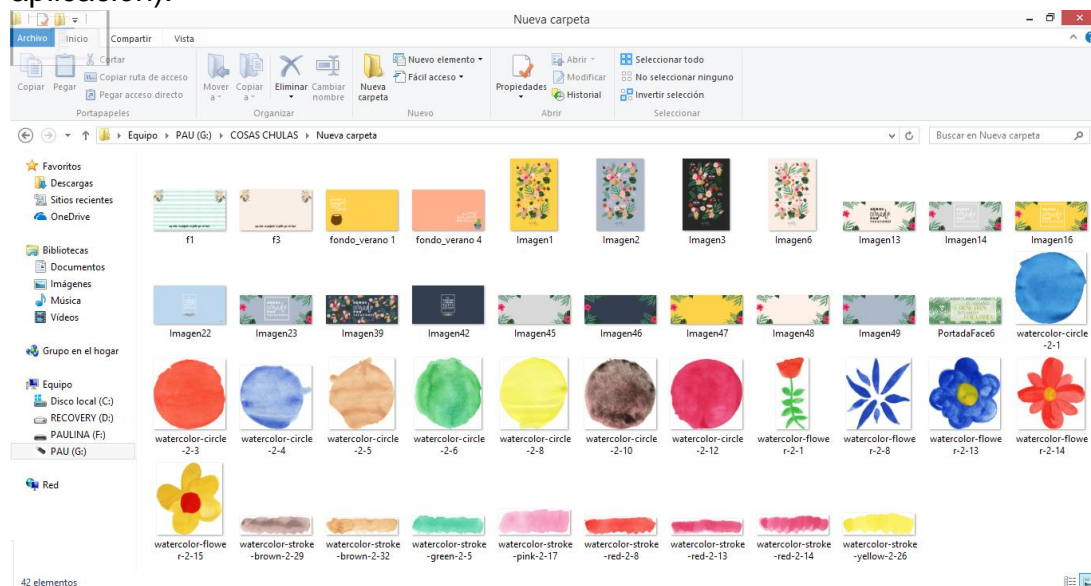


```

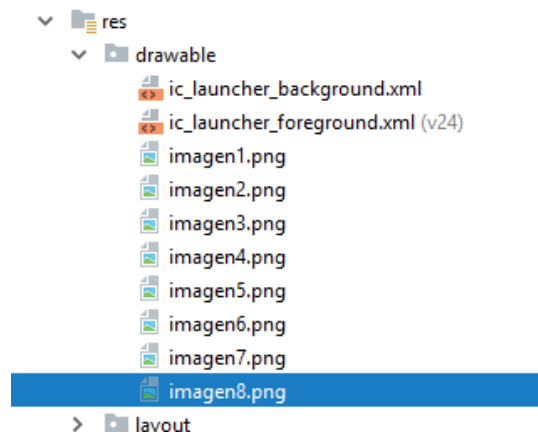
1      <?xml version="1.0" encoding="utf-8"?>
2      <resources>
3          <declare-styleable name="galeria">
4              <attr name="android:galleryItemBackground" />
5          </declare-styleable>
6      </resources>

```

6. **Paso 6:** Como penúltimo paso de la parte de diseño, buscaremos imágenes en nuestro computador (esto con el fin de poder mostrarlas en nuestra aplicación).



7. **Paso 7:** Y como último paso, las agregaremos en la carpeta drawable (recuerda no usar mayúsculas y/o empezar con algún número y/o signo).



Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.

## PROGRAMACIÓN

1. **Paso 1:** Es hora de escribir el siguiente código en nuestro archivo principal con extensión .java

```
Principal.java x BitmapUtils.java x ImageAdapter.java x
1 package com.example.naty.a4f_50_puc_mostrarimagenes;
2
3 import ...
4
5 public class Principal extends AppCompatActivity {
6     //Declaramos las variables globales
7     ImageView imagen;
8     Gallery galeria;
9     final Integer[] ids_imagenes={R.drawable.imagen1, R.drawable.imagen2, R.drawable.imagen3, R.drawable.imagen4,
10         R.drawable.imagen5, R.drawable.imagen6, R.drawable.imagen7, R.drawable.imagen8};
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_principal);
16         //Referenciamos los controles gráficos
17         galeria = (Gallery) findViewById(R.id.galeria);
18         imagen = (ImageView) findViewById(R.id.imagen);
19
20         //Adaptador con imagenes
21         galeria.setAdapter(new ImageAdapter(this, ids_imagenes));
22
23         /*al seleccionar una imagen, la mostramos en el centro de la pantalla a mayor tamaño
24         con este listener, sólo se mostrarían las imágenes sobre las que se pulsa
25         */
26         galeria.setOnItemClickListener(new AdapterView.OnItemClickListener() {
27             @Override
28             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
29                 //Creamos un Toast que nos diga la posición de la imagen que hemos seleccionado
30                 Toast.makeText(getApplicationContext(), "Imagen " + (position+1) + " seleccionada", Toast.LENGTH_SHORT).show();
31
32                 //Ahora mostramos la imagen seleccionada con el control de imagen: (Cargamos la matriz)
33                 imagen.setImageBitmap(BitmapUtils.decodeSampledBitmapFromResource(getResources(), ids_imagenes[position], reqWidth: 300, reqHeight: 0));
34             }
35         });
36
37         //con este otro listener se mostraría la imagen seleccionada en la galería, esto es, la que se encuentre en el centro en cada momento
38         galeria.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
39             @Override
40             public void onItemSelected(AdapterView<?> parent, View v, int position, long id) {
41                 imagen.setImageBitmap(BitmapUtils.decodeSampledBitmapFromResource(getResources(), ids_imagenes[position], reqWidth: 400, reqHeight: 0));
42             }
43
44             @Override
45             public void onNothingSelected(AdapterView<?> arg0) {
46                 // TODO Auto-generated method stub
47             }
48         });
49     }
50 }
```

2. **Paso 2:** Crea otro java.class y nómbrala como BitmapUtils, en ella escribirás lo siguiente:

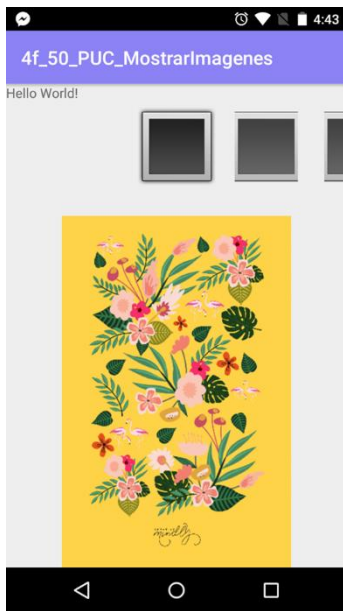
```
Principal.java x BitmapUtils.java x ImageAdapter.java x
1 package com.example.naty.a4f_50_puc_mostrarimagenes;
2
3 import android.content.res.Resources;
4 import android.graphics.Bitmap;
5 import android.graphics.BitmapFactory;
6
7 //Métodos para reescalar imágenes
8 public abstract class BitmapUtils {
9
10     public static int calculateInSampleSize(
11         BitmapFactory.Options options, int reqWidth, int reqHeight) {
12
13         // Asignamos una altura y un ancho a cada imagen
14         final int height = options.outHeight;
15         final int width = options.outWidth;
16         int inSampleSize = 1;
17
18         if (height > reqHeight || width > reqWidth) {
19
20             //Calcular relaciones de altura y ancho a la altura y ancho solicitados
21             final int heightRatio = Math.round((float) height / (float) reqHeight);
22             final int widthRatio = Math.round((float) width / (float) reqWidth);
23
24             /*Elegimos la relación más pequeña del valor (con ayuda del inSampleSize) esto garantizará
25             una imagen final con la dimensión igual a lo solicitado anteriormente
26             */
27             inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
28         }
29
30         return inSampleSize;
31     }
32
33     public static Bitmap decodeSampledBitmapFromResource(Resources res, int resId,
34         int reqWidth, int reqHeight) {
35
36         // Primero usamos el inJustDecodeBounds=true para checar las dimensiones
37         final BitmapFactory.Options options = new BitmapFactory.Options();
38         options.inJustDecodeBounds = true;
39         BitmapFactory.decodeResource(res, resId, options);
40
41         // Calculamos inSampleSize
42         options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);
43
44         // Configuramos el Bitmap con el tamaño previamente declarado en el inSampleSize
45         options.inJustDecodeBounds = false;
46         return BitmapFactory.decodeResource(res, resId, options);
47     }
48 }
```

3. **Paso 3:** Como paso final, crea un último java.class con el nombre de ImageAdapter, escribe lo siguiente:

```
Principal.java x BitmapUtils.java x ImageAdapter.java x
1 package com.example.naty.a4f_50_puc_mostrarimagenes;
2
3 import ...
4
12
13 public class ImageAdapter extends BaseAdapter {
14     //Declaramos las variables globales
15     Context contexto;
16     Integer[] ids_imagenes;
17     int img_fondo;
18
19     /*guardamos las imágenes reescaladas para mejorar el rendimiento
20     usando un SparseArray */
21     SparseArray<Bitmap> imagenesEscaladas = new SparseArray<> ( initialCapacity: 8 );
22
23
24     public ImageAdapter(Context c, Integer[] ids_imagenes) {
25         super();
26         this.contexto = c;
27         this.ids_imagenes = ids_imagenes;
28
29         //Adaptamos un estilo
30         TypedArray a = contexto.obtainStyledAttributes(R.styleable.galeria);
31         img_fondo = a.getResourceId(R.styleable.galeria_android_galleryItemBackground, defValue: 1);
32         a.recycle();
33     }
34
35
36 //Número de imágenes
37 @Override
38 public int getCount() {
39     return ids_imagenes.length;
40 }
41
42 //El ID de la imagen
43 @Override
44 public Object getItem(int position) {
45     return position;
46 }
47
48 @Override
49 public long getItemId(int position) {
50     return position;
51 }
52
53 //Crea una imagen por cada elemento del adaptador
54 @Override
55 public View getView(int position, View convertView, ViewGroup parent) {
56     ImageView imageView = new ImageView(contexto);
57     //Reescalamos la imagen
58     if (imagenesEscaladas.get(position) == null)
59     {
60         Bitmap bitmap = BitmapUtils.decodeSampledBitmapFromResource(contexto.getResources(), ids_imagenes[position], reqWidth: 120, reqHeight: 0);
61         imagenesEscaladas.put(position, bitmap);
62     }
63     //Se aplica el estilo
64     imageView.setBackgroundResource(img_fondo);
65     return imageView;
66 }
67 }
```

Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.

Finalmente, procederemos a ejecutar el programa:



Al iniciar la aplicación, se mostrarán nuestras imágenes en pequeño (**No se visualizan en este ejemplo por un problema de API menor**).

Se muestra la imagen.

Al cambiar de imagen, se muestra un mensaje que dice el número de imagen que fue seleccionada.



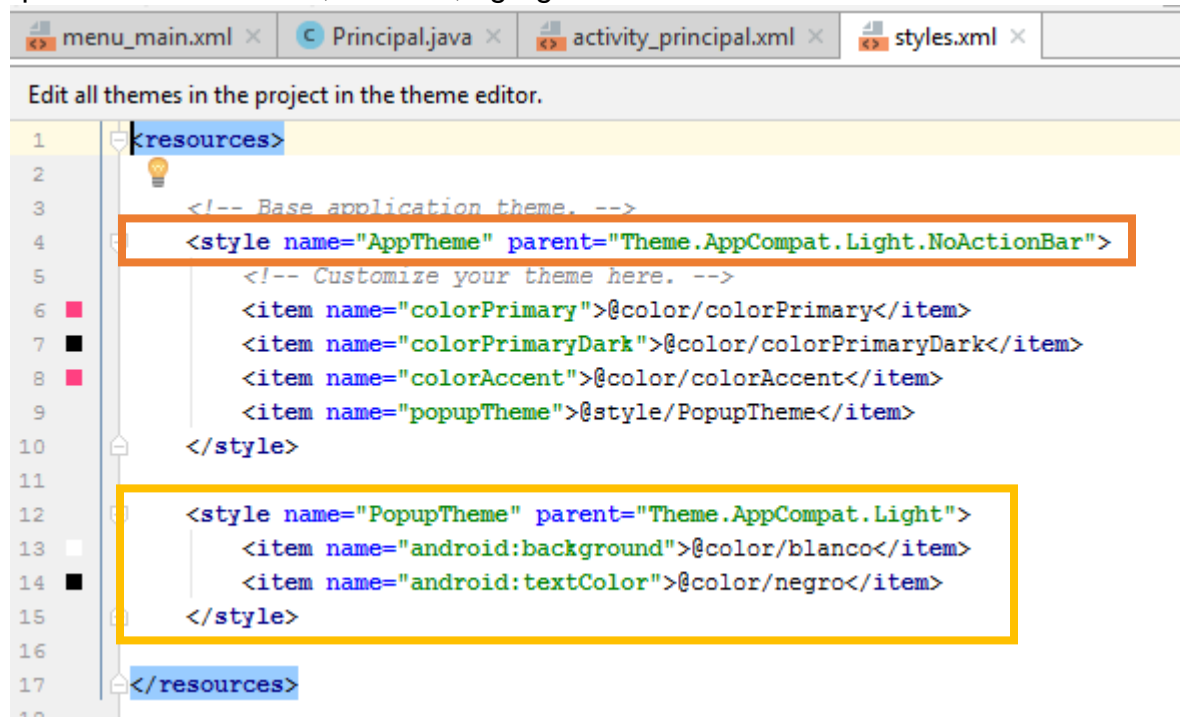
LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## PRÁCTICA 25 – MENU CLÁSICO

### DISEÑO

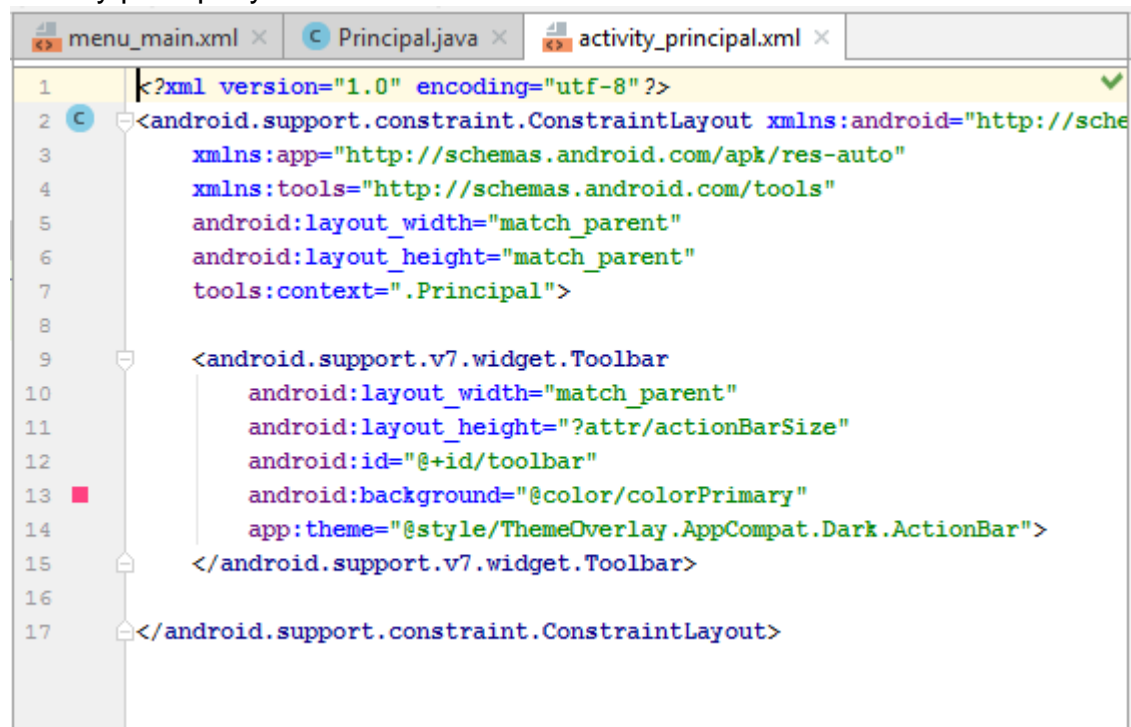
1. **Paso 1:** Abre Android Studio y crea un nuevo proyecto con el nombre de Menú Clásico.

2. **Paso 2:** Es hora de ir a styles.xml (ubicado en *res/values/styles.xml*) y quitamos el ActionBar; además, agregaremos un nuevo estilo.



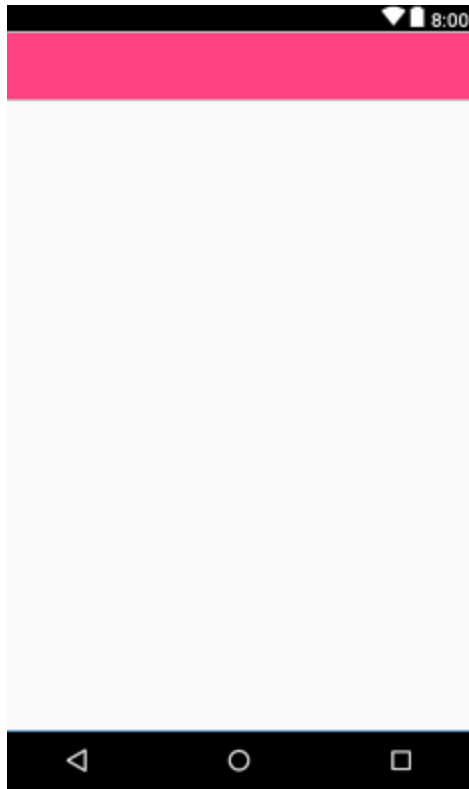
```
1 <resources>
2
3 <!-- Base application theme. -->
4 <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
5     <!-- Customize your theme here. -->
6     <item name="colorPrimary">@color/colorPrimary</item>
7     <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
8     <item name="colorAccent">@color/colorAccent</item>
9     <item name="popupTheme">@style/PopupTheme</item>
10 </style>
11
12 <style name="PopupTheme" parent="Theme.AppCompat.Light">
13     <item name="android:background">@color/blanco</item>
14     <item name="android:textColor">@color/negro</item>
15 </style>
16
17 </resources>
```

3. **Paso 3:** Ahora nos ubicaremos en el archivo con extensión .xml de nuestra Activity principal y añadiremos una toolbar.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".Principal">
8
9     <android.support.v7.widget.Toolbar
10         android:layout_width="match_parent"
11         android:layout_height="?attr/actionBarSize"
12         android:id="@+id/toolbar"
13         android:background="@color/colorPrimary"
14         app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
15     </android.support.v7.widget.Toolbar>
16
17 </android.support.constraint.ConstraintLayout>
```

Tu diseño deberá verse así:



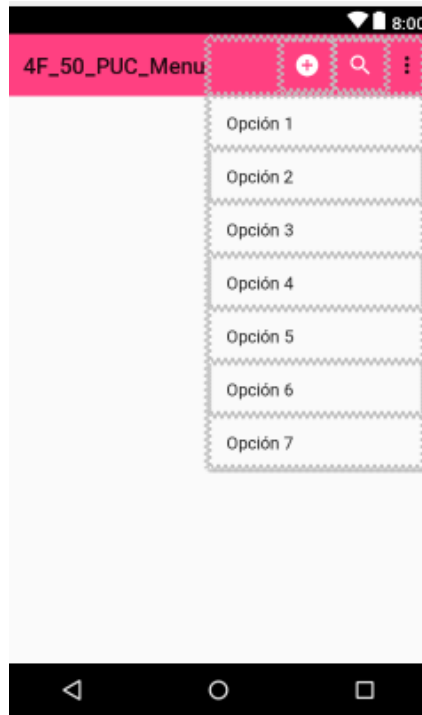
4. **Paso 4:** Una vez que hayamos modificado el diseño, crearemos un nuevo menú llamado `menu_main`, y en él, agregaremos nuestros ítems (u opciones).

```
menu_main.xml x Principal.java x activity_principal.xml x styles.xml x
1      <?xml version="1.0" encoding="utf-8" ?>
2      <menu xmlns:android="http://schemas.android.com/apk/res/android"
3          xmlns:app="http://schemas.android.com/apk/res-auto">
4
5          <item
6              android:id="@+id/op1"
7              android:title="Opción 1"/>
8
9          <item
10             android:id="@+id/op2"
11             android:title="Opción 2"/>
12
13         <item
14             android:id="@+id/op3"
15             android:title="Opción 3"/>
16
17         <item
18             android:id="@+id/op4"
19             android:title="Opción 4"/>
20
21         <item
22             android:id="@+id/op5"
23             android:title="Opción 5"/>
24
25         <item
26             android:id="@+id/op6"
27             android:title="Opción 6"/>
28
29         <item
30             android:id="@+id/op7"
31             android:title="Opción 7"/>
32
33         <item
34             android:title=""
35             android:id="@+id/configura"
36             android:icon="@drawable/ic_add_circle_black_24dp"
37             app:showAsAction="always"/>
38
39         <item
40             android:title=""
41             android:id="@+id/bus"
42             android:icon="@drawable/ic_search_black_24dp"
43             app:showAsAction="always"/>
44
45     </menu>
```



La razón de porqué los dos últimos tienen el `app:showAsAction="always"` es para que siempre sean visibles, no como nuestras opciones.

No estoy segura de haberme explicado del todo, por lo que te dejaré la visualización del diseño del menú:



**Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.**

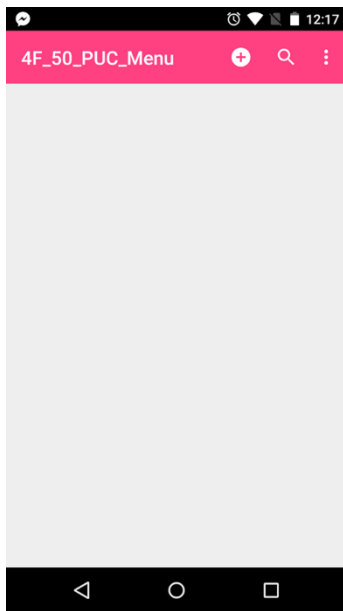
## **PROGRAMACIÓN**

- 1. Paso 1:** Es hora de escribir el siguiente código en nuestro archivo principal con extensión `.java`

```
menu_main.xml x Principal.java x
1 package com.example.naty.a4f_50_puc_menclsico;
2
3 +import ...
9
10 public class Principal extends AppCompatActivity {
11     //Declaramos las variables globales
12     public Toolbar toolbar;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_principal);
18         //Referenciamos los controles gráficos
19         toolbar = (Toolbar) findViewById(R.id.toolbar);
20
21         //Asignamos el soporte de la toolbar
22         setSupportActionBar(toolbar);
23     }
24
25     @Override
26     public boolean onCreateOptionsMenu(Menu menu) {
27         //Obtenemos los items previamente declarados en el menu_main
28         getMenuInflater().inflate(R.menu.menu_main, menu);
29         return true;
30     }
31
32     @Override
33     public boolean onOptionsItemSelected(MenuItem item) {
34
35         //Aquí verificamos cuál item del menu_main está seleccionado y agregamos un Toast que diga que opción has seleccionado
36         if(item.getItemId() == R.id.op1){
37             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 1", Toast.LENGTH_SHORT).show();
38         }else if(item.getItemId() == R.id.op2){
39             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 2", Toast.LENGTH_SHORT).show();
40         }else if(item.getItemId() == R.id.op3){
41             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 3", Toast.LENGTH_SHORT).show();
42         }else if(item.getItemId() == R.id.op4){
43             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 4", Toast.LENGTH_SHORT).show();
44         }else if(item.getItemId() == R.id.op5){
45             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 5", Toast.LENGTH_SHORT).show();
46         }else if(item.getItemId() == R.id.op6){
47             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 6", Toast.LENGTH_SHORT).show();
48         }else if(item.getItemId() == R.id.op7){
49             Toast.makeText(context: this, text: "Ha hecho clic en la Opción 7", Toast.LENGTH_SHORT).show();
50         }else if(item.getItemId() == R.id.configura){
51             Toast.makeText(context: this, text: "Configuración hecha", Toast.LENGTH_SHORT).show();
52         }else if(item.getItemId() == R.id.bus){
53             Toast.makeText(context: this, text: "Busqueda hecha", Toast.LENGTH_SHORT).show();
54         }
55
56         return super.onOptionsItemSelected(item);
57     }
58 }
59 }
```

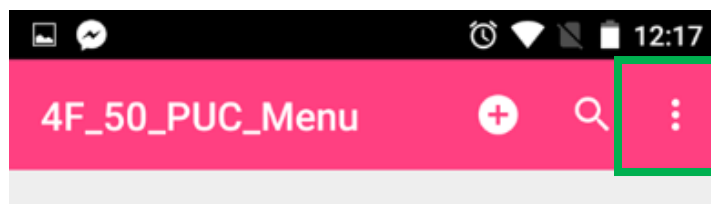
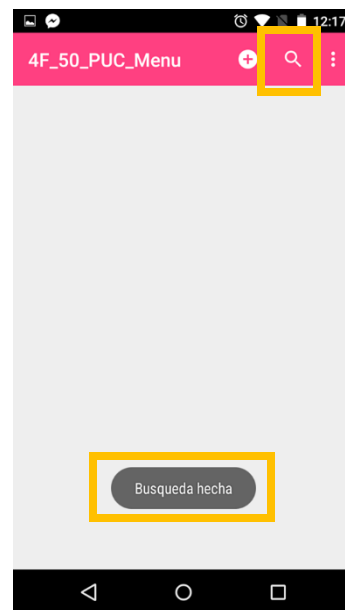
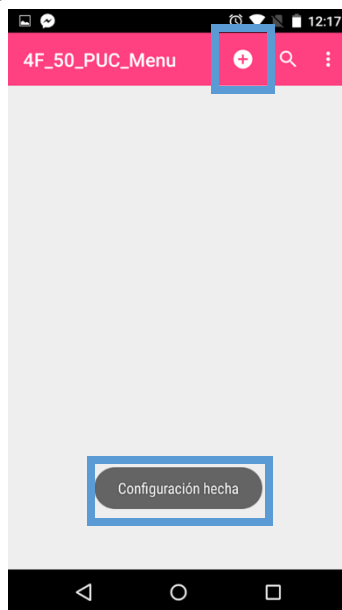
Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.

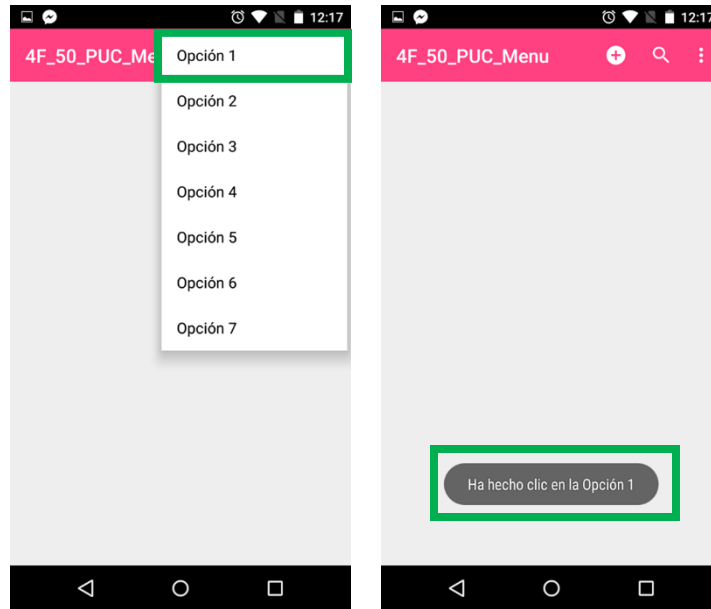
Finalmente, procederemos a ejecutar el programa:



Al iniciar la aplicación, se mostrará la siguiente ventana. Al pulsar en cada una de las distintas opciones, se hará algo distinto.

#### EJEMPLOS:



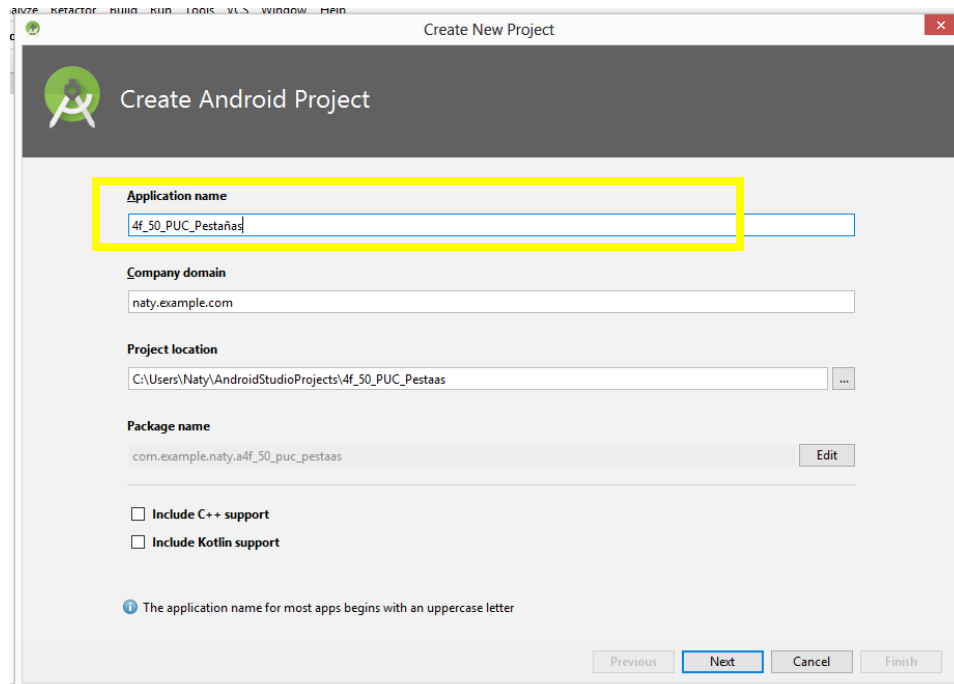


LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## PRÁCTICA 26 - PESTAÑAS

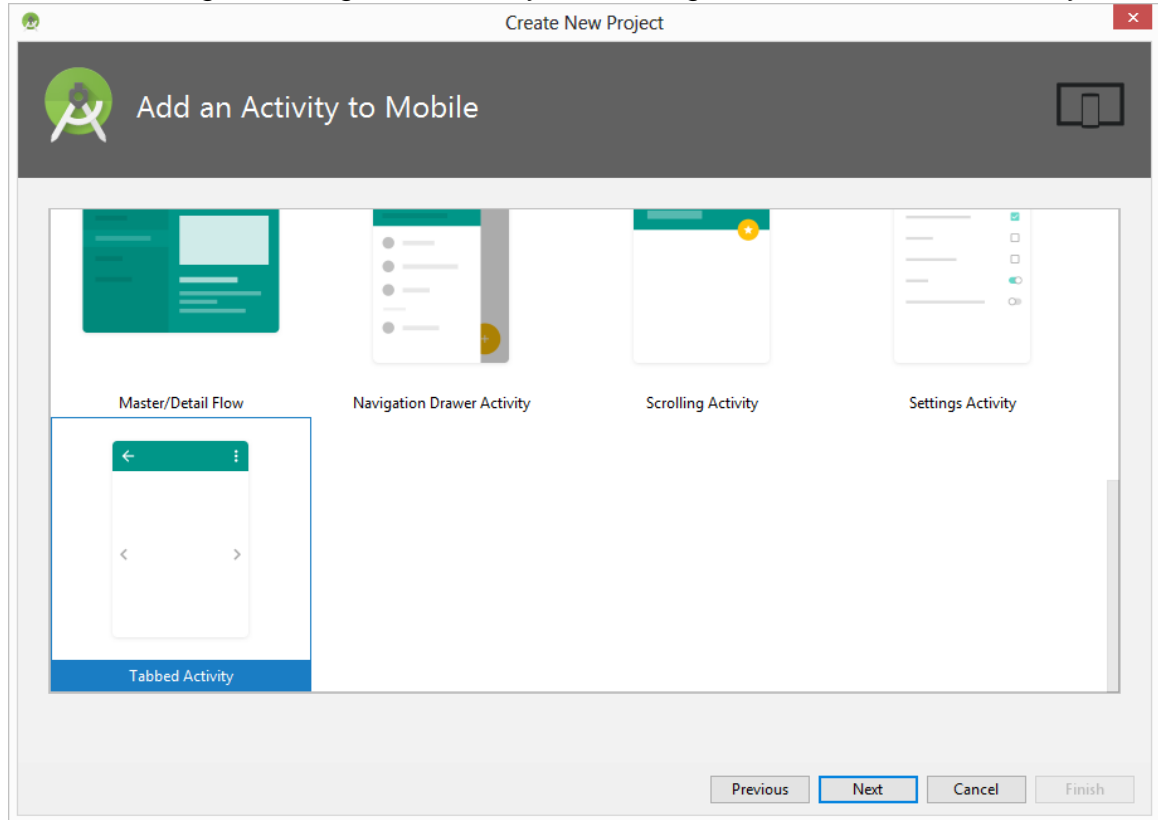
### DISEÑO

- Paso 1:** Crea un nuevo proyecto en AS (Android Studio) y nómbralo como Pestañas.



**NOTA:** Como podrás notar, hemos agregado antes del nombre mi Grado y Grupo, número de lista e iniciales.

2. **Paso 2:** En lugar de elegir una Activity vacía, elegiremos una Tabbed Activity.

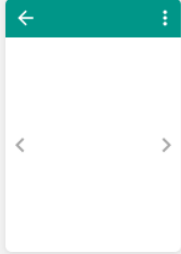


3. **Paso 3:** Una vez que hayamos dado clic en siguiente, nos aparecerá la siguiente ventana:

Create New Project

### Configure Activity

Creates a new blank activity, with an action bar and navigational elements such as tabs or horizontal swipe.



Activity Name: MainActivity

Layout Name: activity\_main

Fragment Layout Name: fragment\_main

Title: MainActivity

Menu Resource Name: menu\_main

Navigation Style: Swipe Views (ViewPager)

The name of the activity class to create

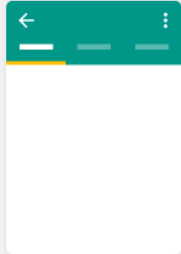
Previous Next Cancel Finish

De la cual cambiaremos lo siguiente:

Create New Project

### Configure Activity

Creates a new blank activity, with an action bar and navigational elements such as tabs or horizontal swipe.



Activity Name: Principal

Layout Name: activity\_principal

Fragment Layout Name: fragment\_principal

Title: Principal

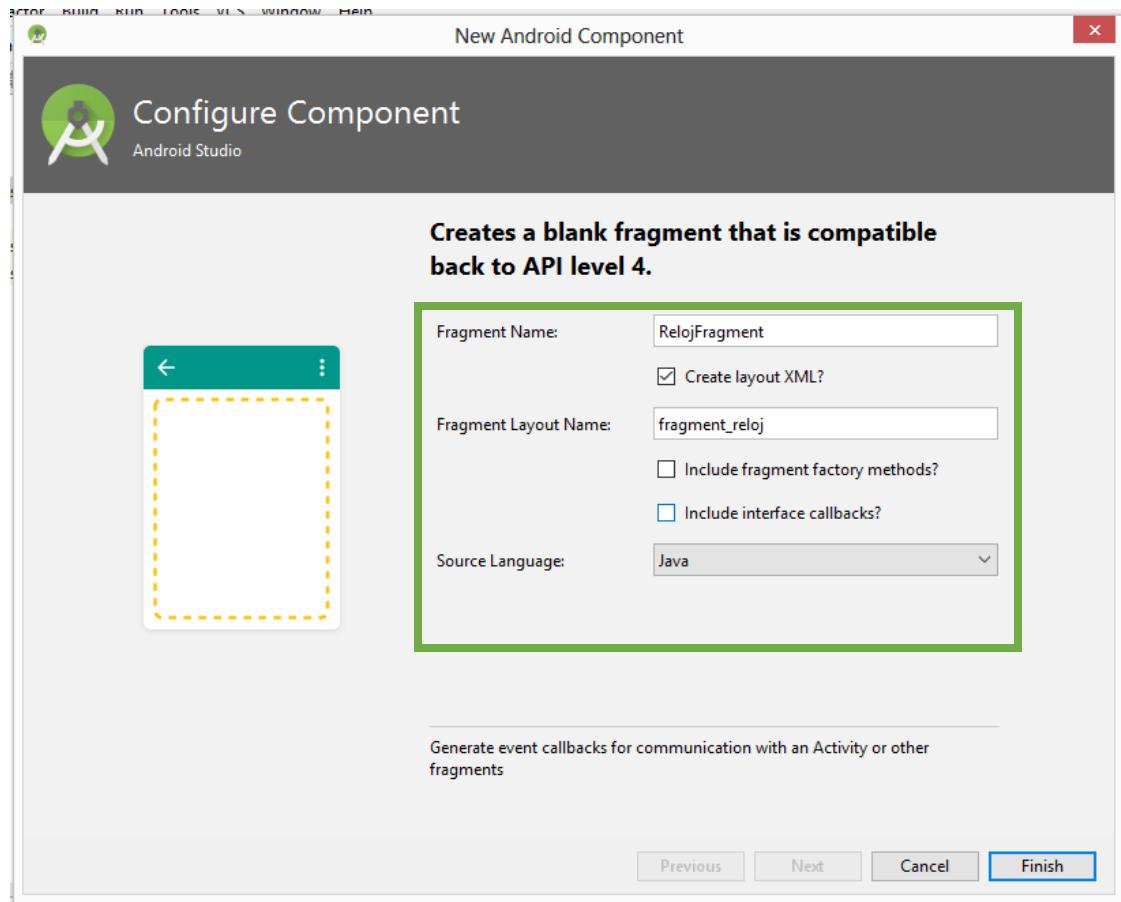
Menu Resource Name: menu\_principal

Navigation Style: Action Bar Tabs (with ViewPager)

Additional features to include, such as a fragment, swipe views, or a navigation drawer

Previous Next Cancel Finish

4. **Paso 4:** Da clic en finalizar de la pantalla anterior y espera a que tu proyecto se cargue. No te apures, puede tardar un poco.
5. **Paso 5:** Una vez cargado tu proyecto, es hora de crear un Fragment, es prácticamente el mismo proceso que crear una Activity. Para crearlos ubícate en *res/clic derecho/New/Fragment/Empty Fragment*.
6. **Paso 6:** Ya que hayas realizado el proceso anterior, se te mostrará una pantalla de la cual deseccionaremos algunas cosas y cambiaremos su nombre por RelojFragment.

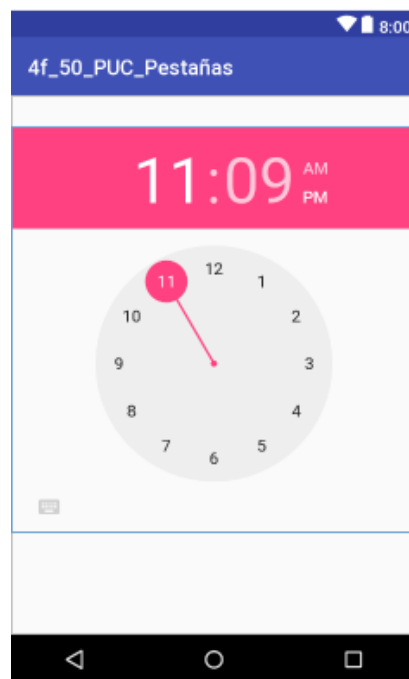


Repite el proceso anterior con un nuevo Fragment. Nómbralo como RelojFragment.

**NOTA:** Es de suma importancia que, al trabajar con Fragment, siempre después de su nombre añadas la palabra Fragment.

7. **Paso 7:** Es hora de ubicarnos en nuestro archivo con extensión .xml de nuestro Reloj Frament. Agrega un TimePicker.

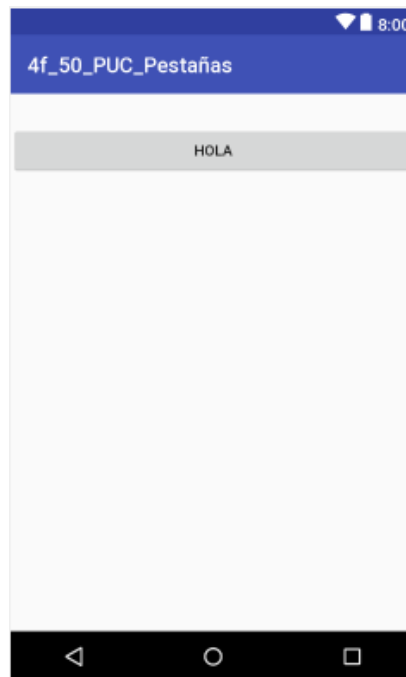
```
fragment_boton.xml x activity_principal.xml x fragment_principal.xml x fragment_reloj.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     tools:context=".RelojFragment">
7
8     <!-- TODO: Update blank fragment layout -->
9
10    <TimePicker
11        android:layout_width="match_parent"
12        android:layout_height="wrap_content"
13        android:layout_marginTop="30dp"
14        android:backgroundTint="#ffffff">
15    </TimePicker>
16
17 </FrameLayout>
```



8. **Paso 8:** Ahora ve al archivo con extensión .xml pero de tu BotonFragment. Agrega un botón:



```
fragment_boton.xml × activity_principal.xml × fragment_principal.xml ×
1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="match_parent"
6   tools:context=".BotonFragment">
7
8   <!-- TODO: Update blank fragment layout -->
9   <Button
10     android:layout_width="match_parent"
11     android:layout_height="wrap_content"
12     android:text="HOLA"
13     android:layout_marginTop="30dp"/>
14
15 </FrameLayout>
```



Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.

## PROGRAMACIÓN

1. **Paso 1:** Es hora de ubicarnos en el archivo de nuestra clase main con su extensión .java.
2. **Paso 2:** Como podrás darte cuenta, se te mostrará automáticamente todo un código. Es de suma importancia que **NO LO BORRES**.

```
Principal.java x fragment_boton.xml x fragment_principal.xml x activity_principal.xml x
1 package com.example.naty.a4f_50_puc_pestaa;
2
3 import ...
22
23 public class Principal extends AppCompatActivity {
24
25     /**
26      * The {@link android.support.v4.view.PagerAdapter} that will provide
27      * fragments for each of the sections. We use a
28      * {@link FragmentPagerAdapter} derivative, which will keep every
29      * loaded fragment in memory. If this becomes too memory intensive, it
30      * may be best to switch to a
31      * {@link android.support.v4.app.FragmentStatePagerAdapter}.
32     */
33     private SectionsPagerAdapter mSectionsPagerAdapter;
34
35     /**
36      * The {@link ViewPager} that will host the section contents.
37     */
38     private ViewPager mViewPager;
39
40     @Override
41     protected void onCreate(Bundle savedInstanceState) {
42         super.onCreate(savedInstanceState);
43         setContentView(R.layout.activity_principal);
44
45         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
46         setSupportActionBar(toolbar);
47         // Create the adapter that will return a fragment for each of the three
48         // primary sections of the activity.
49         mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());
50
51         // Set up the ViewPager with the sections adapter.
```

3. **Paso 3:** Ubícate en esta parte del código, aquí agregaremos nuestros Fragments para poder visualizarlos dentro de nuestra aplicación.

```

32  */
33  public class SectionsPagerAdapter extends FragmentPagerAdapter {
34
35      public SectionsPagerAdapter(FragmentManager fm) { super(fm); }
36
37
38
39      @Override
40      public Fragment getItem(int position) {
41          // getItem is called to instantiate the fragment for the given page.
42          // Return a PlaceholderFragment (defined as a static inner class below).
43          //return PlaceholderFragment.newInstance(position + 1);
44          switch (position){
45              case 0:
46                  //Retornamos nuestros fragment
47                  RelojFragment relojFragment = new RelojFragment();
48                  return relojFragment;
49              case 1:
50                  BotonFragment botonFragment = new BotonFragment();
51                  return botonFragment;
52          }
53          return null;
54      }
55  }

```

4. **Paso 4:** Y, por último, ubícate en esta última parte del código, aquí agregamos un contador, porque en sí sólo queremos que se nos muestren 2 Fragments:

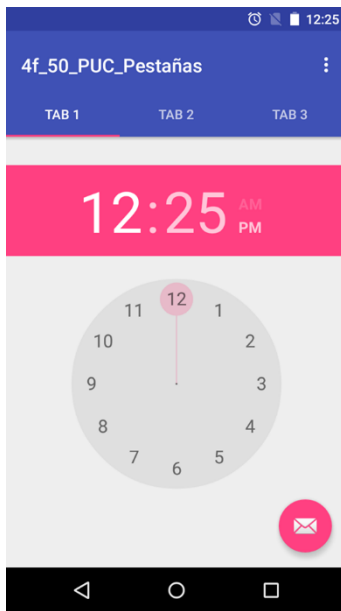
```

5
6      @Override
7      public int getCount() {
8          // Show 2 total pages.
9          return 2;
10     }
11
12     @Nullable
13     @Override
14     public CharSequence getPageTitle(int position) {
15         switch (position){
16             case 0:
17                 return "RELOJ";
18             case 1:
19                 return "BOTON";
20         }
21         return null;
22     }
23 }
24
25

```

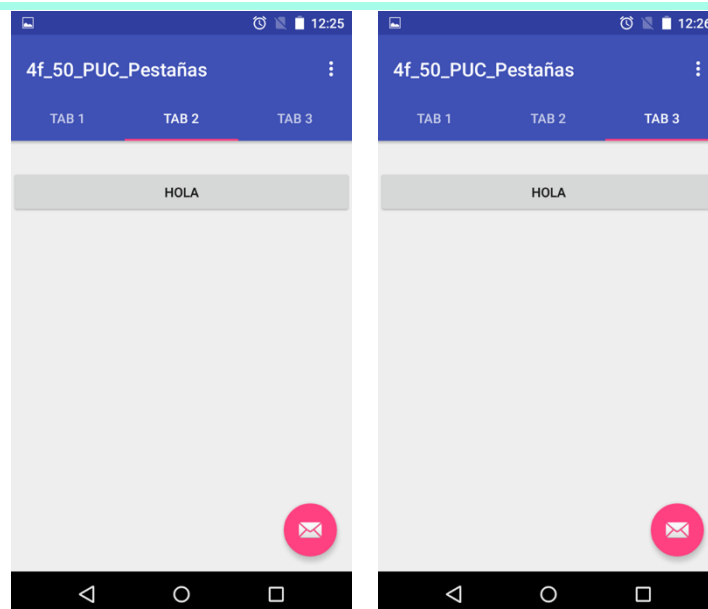
**Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.**

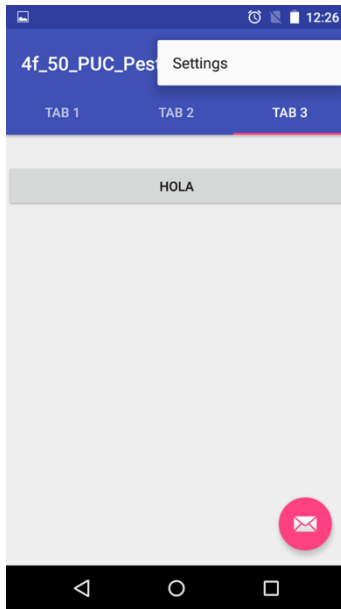
Finalmente, procederemos a ejecutar el programa:



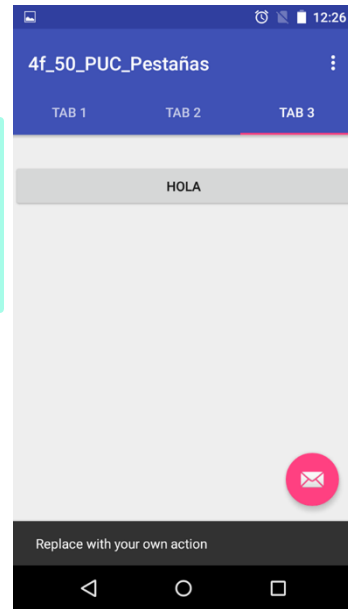
Al iniciar la aplicación, se mostrará nuestro primer Fragment automáticamente en nuestra primera pestaña.

**En nuestra pestaña 2 y 3 se nos mostrará nuestro Fragment que tenía el botón.**





Y estas dos acciones ya vienen predeterminadas en el diseño.



¡TÚ PUEDES MEJORAR ESTE CÓDIGO!

LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## PRÁCTICA 27 - SOLAPAS

### DISEÑO

1. **Paso 1:** Crea un nuevo proyecto en AS (Android Studio) y nómbralo como Solapas.
2. **Paso 2:** Ubícate en el layout de tu Activity principal en la parte de texto. Agrega el siguiente código:

```

activity_principal.xml x Principal.java x list_item.xml x
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   android:id="@+id/layout"
3   android:layout_width="fill_parent"
4   android:layout_height="fill_parent">
5
6   <SlidingDrawer
7     android:id="@+id/simpleSlidingDrawer"
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:content="@+id/content"
11    android:handle="@+id/handle"
12    android:orientation="horizontal"
13    android:rotation="180">
14    <!-- Button for the handle of SlidingDrawer -->
15    <Button
16      android:id="@+id/handle"
17      android:layout_width="wrap_content"
18      android:layout_height="wrap_content"
19      android:background="#A9CCE3"
20      android:rotation="270"
21      android:text="Abrir"
22      android:textColor="#fff" />
23    <!-- layout for the content of the SlidingDrawer -->
24    <LinearLayout
25      android:id="@+id/content"
26      android:layout_width="fill_parent"
27      android:layout_height="fill_parent"
28      android:orientation="horizontal"
29      android:rotation="180">
30      <!-- DEFINE ALL YOUR CONTENT, WIDGETS HERE WHICH YOU WANT TO ADD IN SLIDING DRAWER LAYOUT. -->
31      <ListView
32
33      <!-- DEFINE ALL YOUR CONTENT, WIDGETS HERE WHICH YOU WANT TO ADD IN SLIDING DRAWER LAYOUT. -->
34      <ListView
35        android:id="@+id/simpleListView"
36        android:layout_width="fill_parent"
37        android:layout_height="fill_parent" />
38    </LinearLayout>
39  </SlidingDrawer>
40</LinearLayout>

```

- Paso 3:** Crea un archivo con extensión .xml y nómbralo como list\_item. En la parte de texto debe verse así.

```

activity_principal.xml x Principal.java x list_item.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:orientation="vertical">
6   <!-- TextView for the list item -->
7   <TextView
8     android:id="@+id/name"
9     android:layout_width="fill_parent"
10    android:layout_height="wrap_content"
11    android:padding="10dp"
12    android:textColor="#000000" />
13 </LinearLayout>

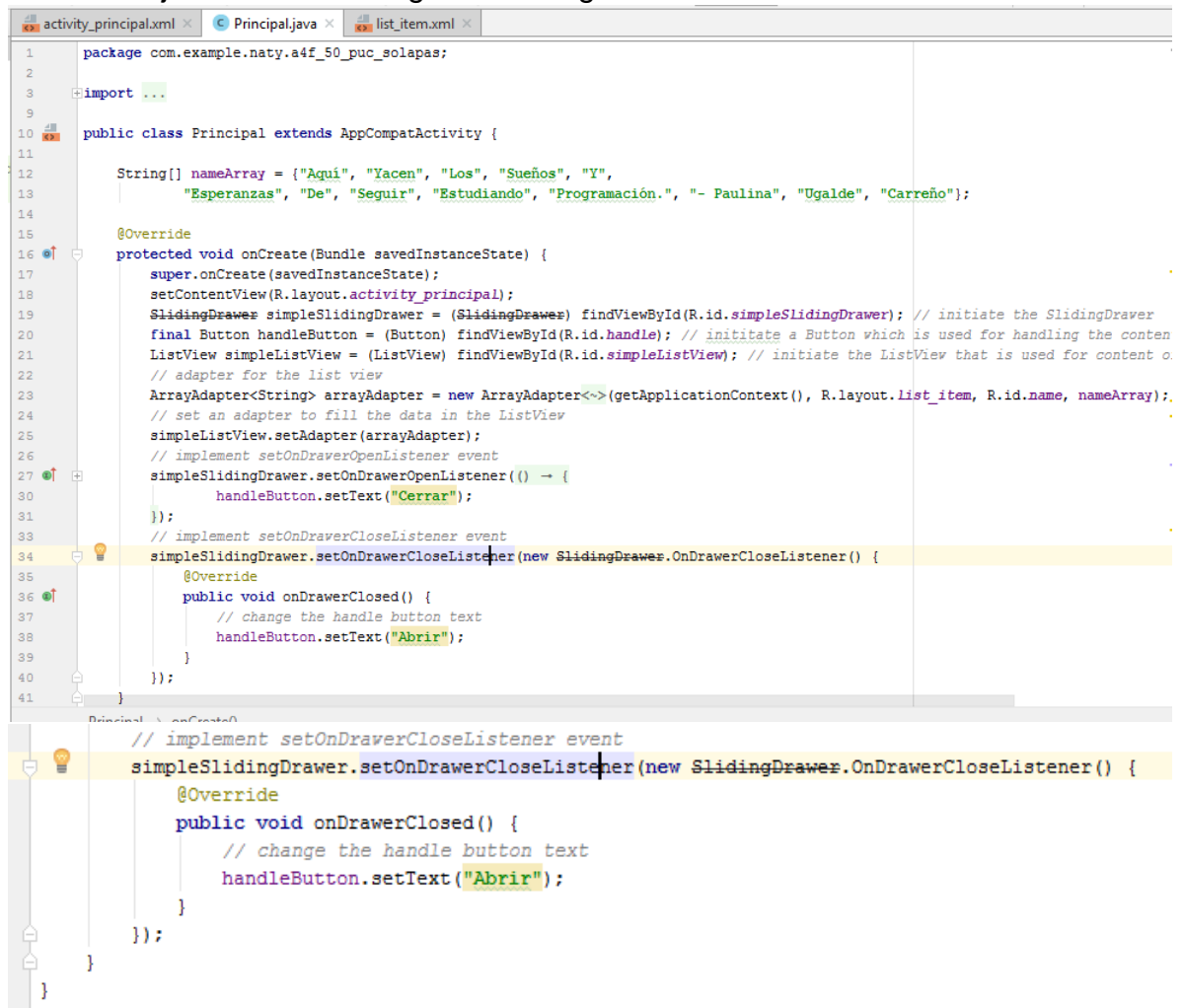
```

Automáticamente ya debería venir el componente, si ese no es el caso, agrégalo.

Y LISTO, HEMOS TERMINADO CON LA PARTE DE DISEÑO; AHORA IREMOS CON LA PARTE DE PROGRAMACIÓN.

## PROGRAMACIÓN

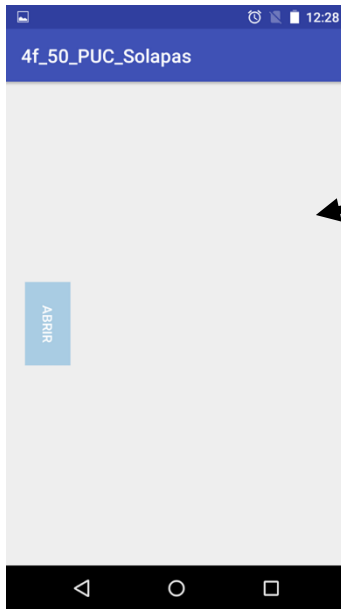
1. **Paso 1:** Es hora de ubicarnos en el archivo de nuestra clase main con su extensión .java. Escribe el siguiente código:



```
1 package com.example.naty.a4f_50_puc_solapas;
2
3 import ...
4
5
6
7
8
9
10 public class Principal extends AppCompatActivity {
11
12     String[] nameArray = {"Aquí", "Yacen", "Los", "Sueños", "Y",
13         "Esperanzas", "De", "Seguir", "Estudiando", "Programación.", "- Paulina", "Ugalde", "Carreño"};
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_principal);
19         SlidingDrawer simpleSlidingDrawer = (SlidingDrawer) findViewById(R.id.simpleSlidingDrawer); // initiate the SlidingDrawer
20         final Button handleButton = (Button) findViewById(R.id.handle); // initiate a Button which is used for handling the content of the list view
21         ListView simpleListView = (ListView) findViewById(R.id.simpleListView); // initiate the ListView that is used for content of the list view
22         // adapter for the list view
23         ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(getApplicationContext(), R.layout.list_item, R.id.name, nameArray);
24         // set an adapter to fill the data in the ListView
25         simpleListView.setAdapter(arrayAdapter);
26         // implement setOnDrawerOpenListener event
27         simpleSlidingDrawer.setOnDrawerOpenListener(() -> {
28             handleButton.setText("Cerrar");
29         });
30         // implement setOnDrawerCloseListener event
31         simpleSlidingDrawer.setOnDrawerCloseListener(new SlidingDrawer.OnDrawerCloseListener() {
32             @Override
33             public void onDrawerClosed() {
34                 // change the handle button text
35                 handleButton.setText("Abrir");
36             }
37         });
38     }
39 }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Y LISTO, HEMOS TERMINADO CON LA PARTE DE PROGRAMACIÓN.

Finalmente, procederemos a ejecutar el programa:



Al iniciar la aplicación,  
automáticamente se muestra un  
botón, que, al pulsarlo, abre una lista.



Al abrir la lista, el botón cambia de  
texto por **"Cerrar"**.

LISTO, HEMOS TERMINADO NUESTRA PRÁCTICA CON ÉXITO.

## Conclusión:

Seis palabras: "Hasta aquí, terminé con la programación".