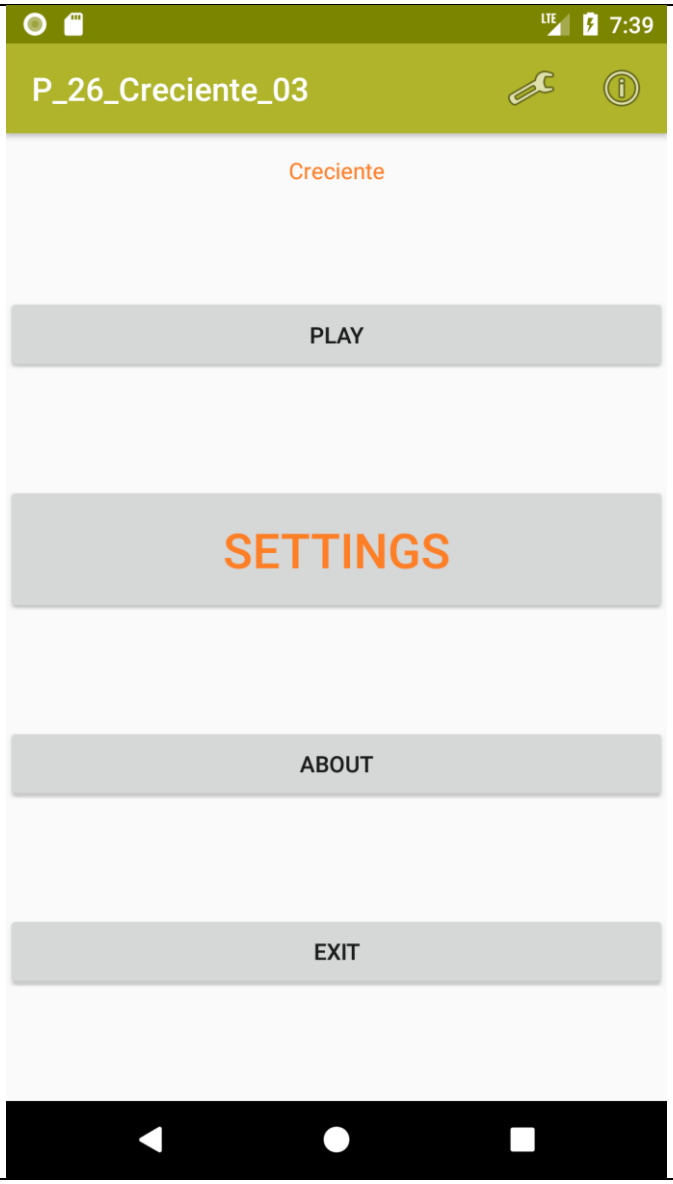
Ejercicio 1: App Bar

1. Copia tu proyecto P_26_Creciente_03 para que pase a ser P_35_Creciente_04 y consiga que la actividad principal cuente con una App Bar como muestra la siguiente imagen (os vais a arrepentir de no haber escogido plantilla Basic cuando la creamos!):



Iconos:

Icono del sistema Android llamado

ic_menu_info_details

Icono del sistema Android llamado

ic_menu_preferences

Al seleccionar el ítem “Información” debe abrirse la misma actividad que cuando se pulsa el botón.

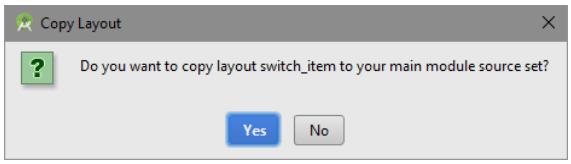
Al seleccionar el ítem “Configuración” debe visualizarse un “Toast” con el mensaje “En preparación”.

Ejercicio 2: App Bar – Switch Item

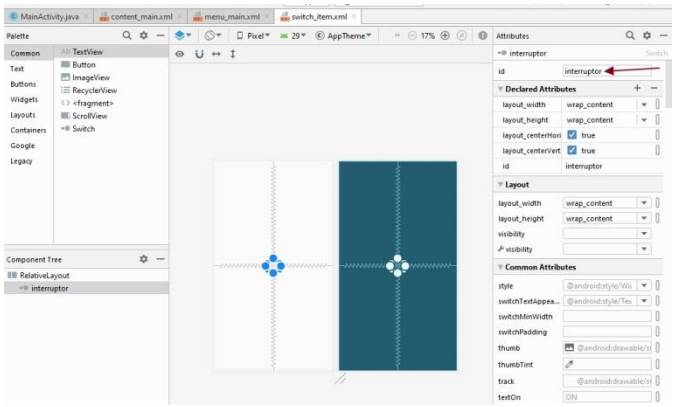
El proyecto P_36_AppBar_Switch tiene una MainActivity que parte de una plantilla Basic (y así nos quita trabajo!).

En el menú de la AppBar nos interesa añadir un switch ítem (recuerda, switch es una vista que nos permite escoger entre dos estados, un interruptor).

Cuando añadimos desde el asistente de AS, nos aparece la ventana



en la que se nos pregunta si deseamos que el asistente nos cree el layout para dicho "interruptor". Obviamente, le diremos que haga ese trabajo por nosotros, lo único que deberemos hacer es darle un identificador al switch del layout creado (switch_item.xml):

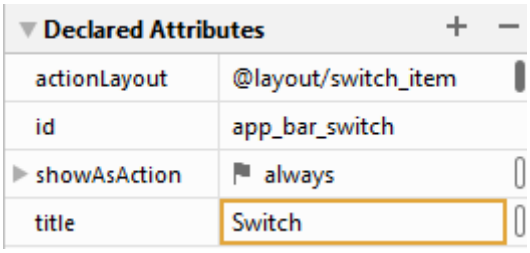


Dejaremos los atributos por defecto, pero podríamos cambiarlos (ver enlace):

XML attributes	
android:showText	Whether to draw on/off text.
android:splitTrack	Whether to split the track and leave a gap for the thumb drawable.
android:switchMinWidth	Minimum width for the switch component.
android:switchPadding	Minimum space between the switch and caption text.
android:switchTextAppearance	TextAppearance style for text displayed on the switch thumb.
android:textOff	Text to use when the switch is in the unchecked/"off" state.
android:textOn	Text to use when the switch is in the checked/"on" state.
android:textStyle	Style (bold, italic, bolditalic) for the text.
android:thumb	Drawable to use as the "thumb" that switches back and forth.
android:thumbTextPadding	Amount of padding on either side of text within the switch thumb.
android:thumbTint	Tint to apply to the thumb.
android:thumbTintMode	Blending mode used to apply the thumb tint.
android:track	Drawable to use as the "track" that the switch thumb slides within.
android:trackTint	Tint to apply to the track.
android:trackTintMode	Blending mode used to apply the track tint.
android:typeface	Typeface (normal, sans, serif, monospace) for the text.

Inherited XML attributes
From class android.widget.CompoundButton
From class android.widget.TextView
From class android.view.View

Vuelve al menú y comprueba que el switch ítem está configurado para que siempre se muestre como acción en la AppBar. Observa que tiene un atributo llamado actionLayout mediante el cual podremos acceder al layout:



Cambiemos el código del método onCreateOptionsMenu():

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it is present.
   getMenuInflater().inflate(R.menu.menu_main, menu);
   //Obtenemos el ítem del menú
   MenuItem menuItem=menu.findItem(R.id.app_bar_switch);
   //Obtenemos la vista a la que hemos asociado la acción
   View view=menuItem.getActionView();
   //En esa vista obtenemos el switch
   Switch miSwitch = view.findViewById(R.id.interruptor);
   //Ponemos listener de cambios
   miSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(CompoundButton buttonView,boolean isChecked) {
         if (isChecked) {
            Toast.makeText(getApplication(), "ON", Toast.LENGTH_SHORT).show();
         } else {
            Toast.makeText(getApplication(), "OFF", Toast.LENGTH_SHORT).show();
         }
      }
   });
   return true;
}
```

Obviamente, el switch ítem tiene "mejores" aplicaciones que mostrar un mensaje!

Ejercicio 3: App Bar – Search Item (Optativo)

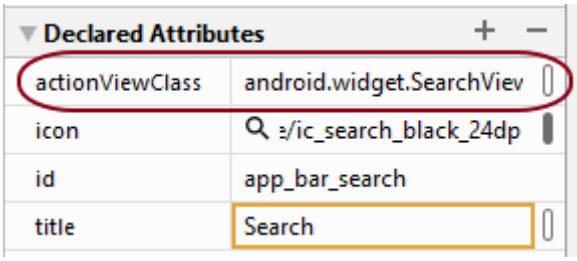
El proyecto P_37_AppBar_Search tiene una MainActivity que parte de una plantilla Basic.

El widget de búsqueda será útil cuando el usuario quiera buscar algo en la aplicación o en el sistema. Agregar widget de búsqueda implica estos pasos:

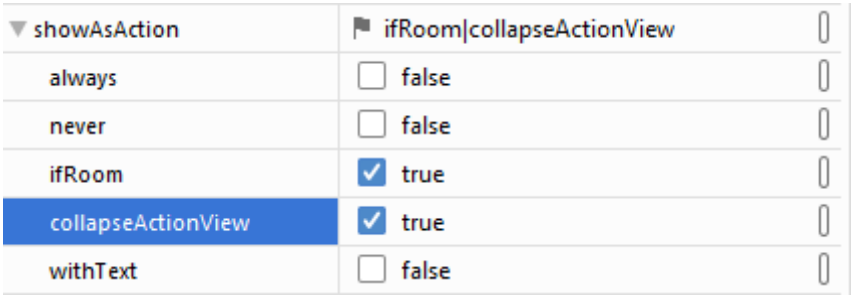
1. **Agregar el ítem de búsqueda en el menú de la barra de acción y tratarlo en código java**:

   Añade en el menú proporcionado por la plantilla un Search ítem.

   Observa que tiene un atributo actionViewClass con valor android.widget.SearchView:

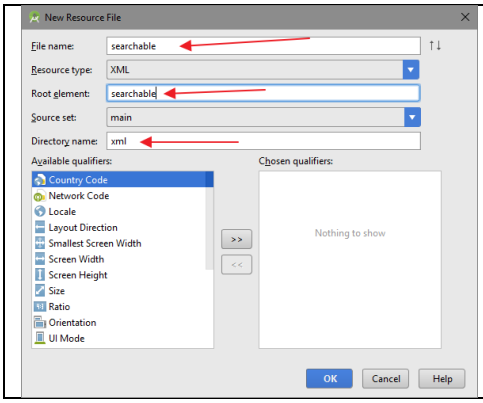


   El valor collapseActionView en el atributo showAsAction permite a SearchView expandirse para ocupar toda la AppBar y volver a colapsarse cuando no esté en uso.



2. **Definir la configuración de búsqueda en fichero xml**

   Una configuración de búsqueda define cómo se comporta SearchView y se guarda en un archivo res/xml/searchable.xml. Como mínimo, debe contener un atributo android:label que tenga el mismo valor que el atributo android:label del elemento <application> o <activity> del fichero AndroidManifest.xml. Sin embargo, también es recomendable agregar un atributo android:hint para darle al usuario una idea de lo que debe teclear en el cuadro de búsqueda:




```
<?xml version="1.0" encoding="utf-8"?>
<searchable
xmlns:android="http://schemas.android.com/apk/res/android"
   android:label="@string/app_name"
   android:hint="@string/consejo" >
</searchable>
```

3. Asociar la configuración de búsqueda con SearchView

Necesitamos escribir un pequeño fragmento de código en el método onCreateOptionsMenu() de la actividad que tiene el widget de búsqueda:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it is present.
   getMenuInflater().inflate(R.menu.menu_main, menu);
   SearchManager searchManager = (SearchManager) getSystemService(Context.SEARCH_SERVICE);
   SearchView searchView = (SearchView) menu.findItem(R.id.app_bar_search).getActionView();
   searchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentName()) );
   return true;
}
```

La llamada a getSearchableInfo() de la clase SearchManager devuelve un objeto SearchableInfo que representa la configuración de "búsqueda a medida" que se crea desde el archivo XML de configuración. Dicho objeto se asocia con la vista del search ítem que lanzará la actividad.

4. **Crear la actividad que recibe el texto tecleado**, realiza la búsqueda (normalmente en una BD) y muestra los resultados.

En nuestro ejemplo, simplemente vamos a mostrar el texto teclado en un TextView. Añade el código al método onCreate():

```
public class SegundaActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      …
      getSupportActionBar().setDisplayHomeAsUpEnabled(true);

      TextView textView=findViewById(R.id.textView);
      Intent intent=getIntent();
      if (Intent.ACTION_SEARCH.equals(intent.getAction())) {
         String query = intent.getStringExtra(SearchManager.QUERY);
         textView.setText(query);
      }
   }

}
```

5. Definir la actividad de búsqueda por defecto y fijar el filtro de intención en el fichero AndroidManifest.xml

```
<activity
   android:name=".MainActivity"
   android:label="@string/app_name"
   android:theme="@style/AppTheme.NoActionBar">
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />

      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
   <meta-data
      android:name="android.app.default_searchable"
      android:value=".SegundaActivity" />
</activity>
<activity
   android:name=".SegundaActivity"
   android:label="@string/title_activity_segunda"
   android:parentActivityName=".MainActivity"
   android:theme="@style/AppTheme.NoActionBar">
   <meta-data
      android:name="android.support.PARENT_ACTIVITY"
      android:value="com.pdm.p_37_appbar_search.MainActivity" />
   <intent-filter>
      <action android:name="android.intent.action.SEARCH" />
   </intent-filter>
   <meta-data
      android:name="android.app.searchable"
      android:resource="@xml/searchable" />
</activity>
```

Ejercicio 4: App Bar – Search Item (Optativo)

El proyecto P_38_AppBar_Search_2 tiene una MainActivity que parte de una plantilla Basic.

La MainActivity tiene en su menú un Search ítem igual que el ejercicio anterior, la diferencia es que es la propia MainActivity la que maneja la búsqueda (en nuestro caso, visualizar en TextView). Los pasos 1, 2 y 3 del ejercicio anterior son iguales.

No hay una segunda actividad, por tanto en AndroidManifest hay que definir todas las particularidades de MainActivity, la más importante es fijar el atributo launchMode con valor singleTop (recuerda unidad 26_Ciclo_de_vida):

```
<activity
  android:name="com.pdm.p_38_appbar_search_2.MainActivity"
  android:label="@string/app_name"
  android:theme="@style/AppTheme.NoActionBar"
  android:launchMode="singleTop">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
  <meta-data
    android:name="android.app.default_searchable"
    android:value=".MainActivity" />
  <intent-filter>
    <action android:name="android.intent.action.SEARCH" />
  </intent-filter>
  <meta-data
    android:name="android.app.searchable"
    android:resource="@xml/searchable" />
</activity>
```

Hay un pequeño cambio en el código para tratar el modo de lanzamiento. Cuando se establece en singleTop, la instancia existente de la clase de actividad sigue recibiendo el nuevo intento que tiene que ser manejado en el método onNewIntent() de la actividad:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  …
  textView=findViewById(R.id.textView);
  Intent intent=getIntent();
  if (intent!=null)
    manejaIntent(intent);
}

private void manejaIntent(Intent intent) {
  if( Intent.ACTION_SEARCH.equals(intent.getAction())) {
    String query = intent.getStringExtra(SearchManager.QUERY);
    textView.setText(query);
  }
}

@Override
protected void onNewIntent(Intent intent) {
  super.onNewIntent(intent);
  manejaIntent(intent);
}
```

Cada vez que el usuario envía una consulta, se llama al método onNewIntent() que puede llamar al mismo método que onCreate() para mostrar el resultado.

Ejercicio 5: App Bar – ShareProvider (Optativo)

El proyecto P_39_AppBar_Share tiene una MainActivity que parte de una plantilla Basic.

Añade al menú, un ítem (en principio normal), identifícalo como action_share y desde el modo texto (para usar la librería de soporte convenientemente, añade el proveedor de acciones:

```
<item
   android:id="@+id/action_share"
   android:orderInCategory="101"
   android:title="Item"
   app:showAsAction="always"
   app:actionProviderClass="androidx.appcompat.widget.ShareActionProvider" />
```

Realiza la siguiente modificación en la clase:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it is present.
   getMenuInflater().inflate(R.menu.menu_main, menu);
   MenuItem shareItem = menu.findItem(R.id.action_share);
   ShareActionProvider shareActionProvider= (ShareActionProvider) MenuItemCompat.getActionProvider(shareItem);
   Intent intent = new Intent(Intent.ACTION_SEND);
   intent.setType("text/plain");
   intent.putExtra(Intent.EXTRA_TEXT, "Texto a enviar");
   shareActionProvider.setShareIntent(intent);
   return true;
}
```

Más adelante, cuando sepamos trabajar con ficheros y/o imágenes ya veremos la manera de compartirlos en vez de compartir el mensaje "Texto a enviar".