



Programación multimedia y dispositivos móviles

UT-5.2 - AppBar y ToolBar

<https://developer.android.com/guide/>
<https://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/>



Actionbar / Appbar y Toolbar

Los menús son un componente común de la interfaz de usuario en muchos tipos de aplicaciones a fin de presentar al usuario acciones y otras opciones en las actividades.

A partir de Android 3.0 (nivel de API 11), los dispositivos con Android ya no tienen que proporcionar un botón Menú exclusivo. Con este cambio, las apps para Android dejarán de depender de los paneles de menú tradicionales y, en su lugar, proporcionarán una barra de la app para mostrar las acciones más comunes del usuario.

Aunque haya cambiado el diseño de la experiencia del usuario para algunos elementos de menú, la semántica para definir un conjunto de acciones y opciones sigue basándose en las API de Menu.



Actionbar / Appbar

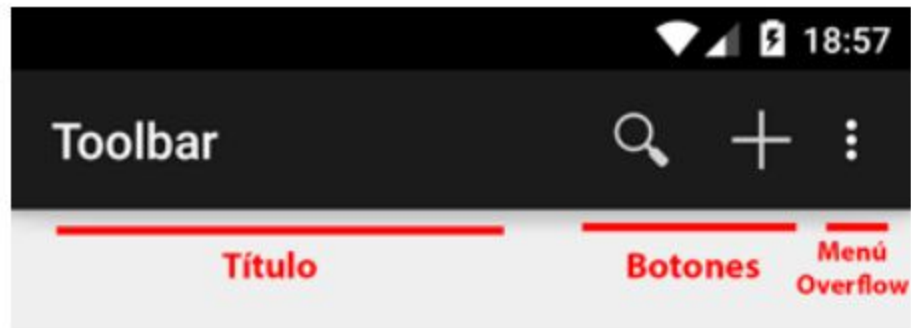
La app bar, o también llamada action bar, es la barra de título y herramientas que aparece en la parte superior de la gran mayoría de aplicaciones actuales de la plataforma Android.

Esta pequeña barra tiene enormes utilidades, como por ejemplo:

- ❑ Proveernos acceso rápido a las acciones más comunes y solicitadas por los usuarios
- ❑ Organizar la navegación entre actividades
- ❑ Proporcionarnos un espacio donde diferenciar nuestra aplicación de otras aplicaciones (a través del título, iconos particulares y demás)

Actionbar / Appbar

La barra de acción se divide en tres partes fundamentales que debemos reconocer antes de empezar a programar sobre ella. Por lo que veremos la siguiente ilustración sobre su estructura:





Actionbar / Appbar

- ❏ **Título:** Es muy habitual ver un texto estático que visualiza el nombre de la aplicación, aunque se puede quitar o ver lo que se quiera.
- ❏ **Botones de acción:** Representan las acciones más populares dentro de la aplicación, las cuales podemos ejecutar rápidamente al presionarlos.
- ❏ **Menú overflow:** Este segmento contiene una lista de acciones que no son tan populares, pero pueden ser necesitadas en algún momento por el usuario para tener acceso de forma sencilla.



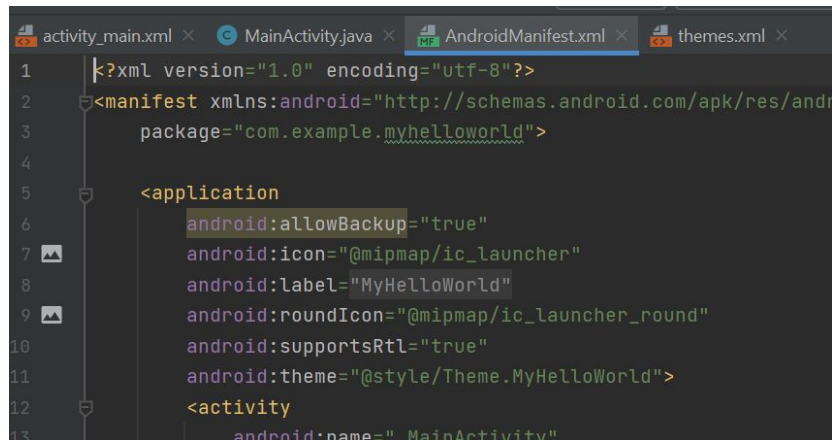
Actionbar / Appbar

En la actualidad, Android proporciona este componente a través de la librería appcompat, que suele venir ya incluida por defecto en todos nuestros proyectos, la podemos encontrar en la sección dependencies del fichero build.gradle.

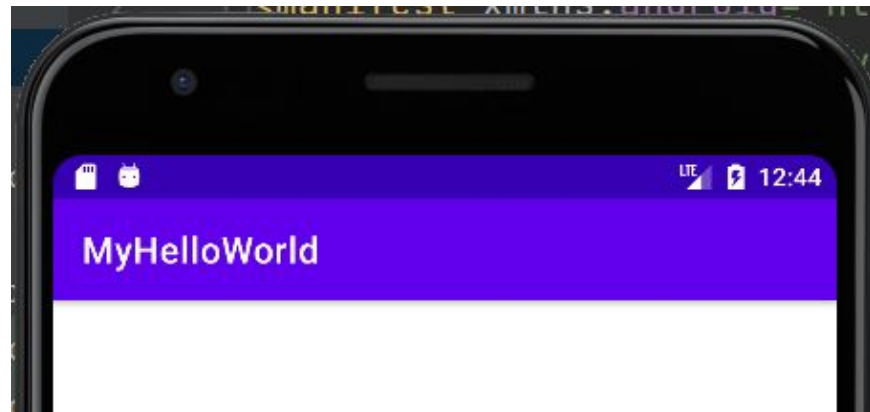
Podemos hacer uso de la action bar de dos formas diferentes. La primera de ellas, más sencilla aunque menos flexible, consiste en utilizar la action bar por defecto que se añade a nuestras actividades por tan sólo utilizar un tema (theme) determinado en la aplicación que incluya una ActionBar. La segunda, más flexible y personalizable aunque requiere algo código, consiste en añadir de forma explícita un componente Toolbar o Material Toolbar a nuestra interfaz de usuario (a nuestro nivel de conocimiento, ambos controles parecen equivalentes)

Actionbar / Appbar - Título

La action bar por defecto tan sólo contiene el título. El título mostrado por defecto corresponde al título de la aplicación, definido en el fichero AndroidManifest.xml, en el atributo label del elemento application correspondiente a dicha actividad



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.myhelloworld">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="MyHelloWorld"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportsRtl="true"
11        android:theme="@style/Theme.MyHelloWorld">
12        <activity
13            android:name=".MainActivity">
```

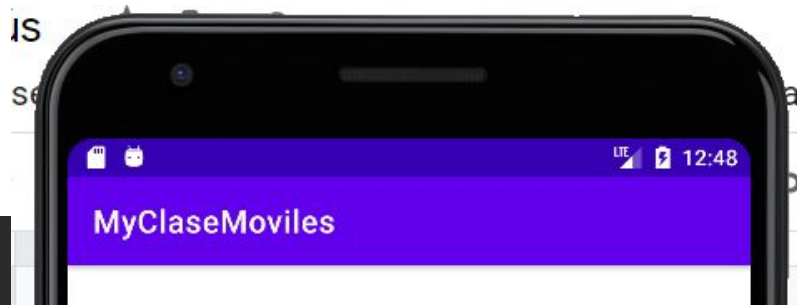


Actionbar / Appbar - Título

Para cambiar este título entramos en el fichero strings.xml y definimos una nueva variable con lo que queramos ver. Esta nueva variable la referenciamos dentro del archivo de manifiesto.

```
activity_main.xml x MainActivity.java x AndroidManifest.xml x strings.xml x
translations for all locales in the translations editor.
<resources>
  <string name="app_name">MyHelloWorld</string>
  <string name="titulo">MyClaseMoviles</string>
</resources>
```

```
<application
  android:allowBackup="true"
  android:icon="@mipmap/ic_launcher"
  android:label="@string/titulo"
  android:roundIcon="@mipmap/ic_launcher_round"
```





Menús

Un menú se define mediante un fichero XML que se coloca en la carpeta /res/menu. El menú se definirá mediante un elemento raíz <menu> que contendrá una serie de elementos <item> que representarán cada una de las opciones. Los elementos <item> por su parte podrán incluir varios atributos que lo definan, entre los que destacan los siguientes:

- ❑ **android:id**. El ID identificativo del elemento, con el que podremos hacer referencia dicha opción.
- ❑ **android:title**. El texto que se visualizará para la opción.
- ❑ **android:icon**. El icono asociado a la acción. Tiene que estar como recurso drawables.
- ❑ **android:showAsAction**. Si se está mostrando una action bar, este atributo indica si la opción de menú se mostrará como botón de acción o como parte del menú de overflow. Puede tomar varios valores:



Actionbar / Appbar - Menús

- **ifRoom**. Se mostrará como botón de acción sólo si hay espacio disponible.
- **withText**. Se mostrará el texto de la opción junto al icono en el caso de que éste se esté mostrando como botón de acción.
- **never**. La opción siempre se mostrará como parte del menú de overflow.
- **always**. La opción siempre se mostrará como botón de acción. Este valor puede provocar que los elementos se solapen si no hay espacio suficiente para ellos.

Hacemos un menú

Vamos a crear un menú. En primer lugar creamos un proyecto nuevo:

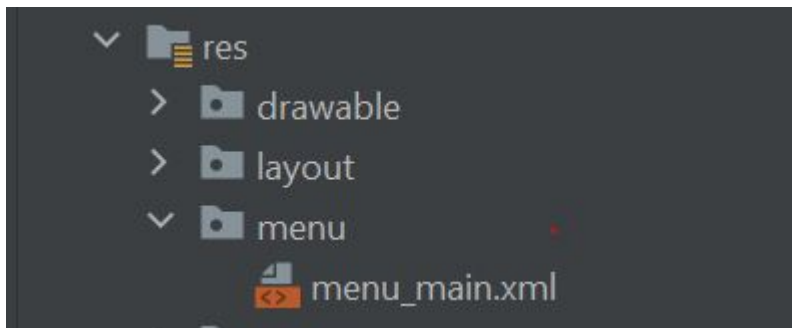
- ❑ Importamos al proyecto las 4 imágenes de la carpeta de recursos (matrix, morfeo, roja y azul)
- ❑ En la actividad ponemos una ImageView donde ponemos la imagen de Morfeo
- ❑ Incluimos un TextView (yo he puesto este texto: “Esta es tu última oportunidad. Después de esto, no hay vuelta atrás....Recuerda: todo lo que ofrezco es la verdad. Nada más.”)
- ❑ Ponemos además el título que queramos (recuerda, define una variable en el fichero string.xml y referenciala en el AndroidManifest.xml)
- ❑ **Ponemos en nuestra actividad un control de tipo MaterialToolbar**



**Esta es tu última oportunidad.
Después de esto, no hay vuelta
atrás....Recuerda: todo lo que ofrezco
es la verdad. Nada más.**

Hacemos un menú

Ahora creamos la carpeta “menu” dentro de res. En ella creamos un menú (yo lo he llamado menu_main.xml)





Hacemos un menú

Nuestro menú tendrá tres opciones: “roja”, “azúl” y «salir corriendo», la primera para que se muestre en la app bar, si hay espacio, como botón con su icono correspondiente, la segunda igual pero añadiendo su texto junto al icono, y la tercera para que siempre se muestre en el menú de overflow.

En primer lugar vamos a la carpeta de string.xml y creamos los literales que queremos que se muestren:

```
<string name="roja">Roja</string>
```

```
<string name="azul">Azúl</string>
```

```
<string name="indeciso">Sal corriendo</string>
```

Hacemos un menú

Ahora crearemos el fichero /res/menu/menu_main.xml con el siguiente contenido:

- ❑ “roja” se verá, si hay espacio, como botón con su icono correspondiente.
- ❑ “azul” la segunda igual pero añadiendo su texto junto al icono. Como podéis ver además en la segunda opción se pueden combinar varios valores de showAsAction utilizando el caracter “|”.
- ❑ “indeciso” se verá en el menú de overflow.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/action_roja"
          android:title="@string/roja"
          android:icon="@drawable/roja"
          android:orderInCategory="100"
          app:showAsAction="ifRoom" />
    <item android:id="@+id/action_azul"
          android:title="@string/azul"
          android:icon="@drawable/azul"
          android:orderInCategory="150"
          app:showAsAction="ifRoom|withText" />
    <item android:id="@+id/action_opciones"
          android:title="@string/indeciso"
          android:orderInCategory="100"
          app:showAsAction="never" />
</menu>
```



MaterialToolbar - Codificamos la actividad

Para acabar tenemos que modificar nuestra clase principal añadiendo la siguiente lógica:

Dentro del onCreate

Instanciamos en una variable un objeto de la clase Toolbar. Casteamos nuestra Toolbar y la asignamos como actionBar en la actividad. También definimos una variable de tipo TextView y el imageView.

```
lateinit var tv1: TextView
lateinit var myimage: ImageView
lateinit var myToolbar: Toolbar
```

```
myToolbar = findViewById(R.id.myToolbar)
tv1 = findViewById(R.id.tv2);
myimage = findViewById(R.id.imageView);
setSupportActionBar(myToolbar)
```



Hacemos un menú

Una vez definido el menú en su fichero XML correspondiente tan sólo queda asociarlo a nuestra actividad principal. Esto se realiza sobrescribiendo el método `onCreateOptionsMenu()` de la actividad, dentro del cual lo único que tenemos que hacer normalmente es inflar el menú llamando al método `inflate()` pasándole como parámetro el ID del fichero XML donde se ha definido.

Dentro de nuestra clase MainActivity, una vez cerrado el onCreate:

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
    menuInflater.inflate(R.menu.my_menu, menu)  
    return true  
}
```


Hacemos un menú

Con esto ya deberíamos ver nuestro menú con las opciones que hemos incluido. Al girar la pantalla, en la opción que pusimos texto (si cabía) se pinta el texto





Hacemos un menú

Por último queda saber cómo responder a las pulsaciones que haga el usuario sobre ellos. Para esto sobrescribiremos el método `onOptionsItemSelected()` de la actividad, donde consultaremos la opción de menú que se ha pulsado mediante el método `getItemId()` de la opción de menú recibida como parámetro y actuaremos en consecuencia.

Creamos un método que reciba un texto y que pinte en el TextView el texto recibido:

```
fun mensaje_morfeo(texto: String?) {  
    tv1.text = texto  
}
```

Incluimos la lógica del sobrescribir el `onOptionsItemSelected()` en la clase MainActivity. Este método es autoinvocado cuando el usuario presiona un botón.



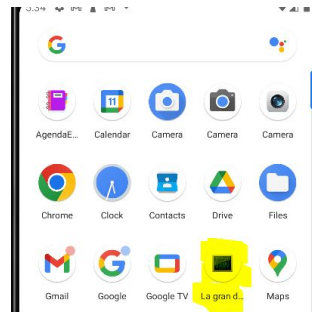
Hacemos un menú

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    val id = item.itemId  
    var cadena = ""  
    if (id == R.id.action_roja) {  
        cadena =  
            "Tomas la píldora roja... te quedas en el País de las Maravillas, y " +  
            "te enseño lo profunda que es la madriguera del conejo"  
    } else if (id == R.id.action_azul) {  
        cadena =  
            "Tomas la píldora azul... la historia termina, te despiertas en tu cama " +  
            "y crees lo que quieras creer"  
    } else if (id == R.id.action_opciones) {  
        this.finish()  
    }  
    mensaje_morfeo(cadena)  
    return super.onOptionsItemSelected(item)  
}
```

Actionbar / Appbar - Hacemos un menú



Acabamos



Finalizamos viendo como se puede ocultar y poner esta appbar. Dentro del onCreate de la MainActivity llamamos al método para ocultarla:

```
getSupportActionBar()?.hide();
```

Luego, he hecho un método para mostrarla. A este método le llamamos desde el onClick del ImageView control donde está la imagen de morfeo.

```
myimage.setOnClickListener{  
    getSupportActionBar()?.show();}
```

Además he cambiado el icono de la aplicación poniendo el de Matrix cambiando la variable icon del manifiesto:

```
android:roundIcon="@drawable/matrix"
```



Toolbar

La ActionBar forma parte del profundo rediseño acometido por Google para Android 3/4. Para poder utilizarla en Android 2 era necesario recurrir a implementaciones propias o de terceros. Con Lollipop llegó Material Design y un nuevo paradigma visual tan importante como el que supuso la llegada de Android 4.

Una de las novedades ha sido la aparición de la Toolbar que pretende ser un reemplazo de la ActionBar más potente y flexible. Mientras que la ActionBar es un elemento del sistema que se muestra en una Activity si se hereda de un tema que la incluya, Toolbar es simplemente un widget que aporta grandes ventajas:



Toolbar

- ❑ Puede ubicarse en cualquier lugar.
- ❑ Podemos tener varias en una misma pantalla.
- ❑ Es un ViewGroup por lo que se pueden incluir dentro de la Toolbar otros widgets.
- ❑ Fácilmente adaptable al tamaño de pantalla.
- ❑ Sigue proporcionando la funcionalidad de la ActionBar.

Hay varios componentes más, con funcionalidades similares, que no vamos a ver en el curso. El más potente parece la App Bar que es una expansión del concepto de la action bar. Aunque la Toolbar representa el pequeño segmento rectangular al que estamos acostumbrados, la App Bar (representada con la clase `AppBarLayout`) es un contenedor con más posibilidades de personalización.

- ❑ Permite dividir los comportamientos de la action bar para independizar las acciones.
- ❑ Incluso permite añadir efectos de scrolling en conjunto con el Coordinator Layout.



Documentación

<https://www.develou.com/android-action-bar/>

<https://www.develou.com/toolbar-en-android-creacion-de-action-bar-en-material-design/>

<https://www.sgoliver.net/blog/actionbar-appbar-toolbars-en-android-i/>

<https://danielme.com/2015/07/14/disenio-android-actionbar-con-toolbar/>