

Interfaz de usuario:

“Action Bar/AppBar/ToolBar”

Parte 2

Toolbar (I)

- ▶ **Toolbar** es un **nuevo componente** proporcionado por la librería **appcompat**.
- ▶ Es una forma **más flexible y personalizable** de añadir una action bar a una app.
- ▶ Se **añade en nuestros layouts XML como cualquier otro control**, y **no sólo en la parte superior de la pantalla** a modo de app bar.
- ▶ Es decir, **se puede añadir en cualquier parte donde queramos poner una barra de acciones**.
- ▶ De momento, veremos cómo usar la **Toolbar** a modo de **ActionBar**.

¿Qué necesito para usar Toolbar?


- ▶ **Toolbar** está incluida en la **librería de soporte appcompat-v7**.
- ▶ En versiones actuales de Android Studio, esta librería viene **incluida por defecto**. *Debemos asegurarnos de tener la última versión.*
- ▶ Si por cualquier razón no viene incluida, lo haremos **añadiendo su referencia en la sección *dependencies*** del fichero ***build.gradle***:

```
dependencies {  
    ...  
    compile 'com.android.support:appcompat-v7:22.1.1'  
}
```

Pasos a seguir (I)

1. Creamos un nuevo proyecto y nos **aseguramos de que incluye la librería *appcompat-v7***, como se indicó antes.
2. Nos aseguramos de que **nuestra Activity extiende de *AppCompatActivity***:

```
import android.support.v7.app.AppCompatActivity;  
//...  
  
public class MainActivity extends AppCompatActivity {  
    //...  
}
```

3. Debemos “**desactivar**” la funcionalidad por defecto que vimos en el anterior documento (**ActionBar por defecto**). Para ello:
 -  Configuramos en nuestra app un tema que no incluya ActionBar. ***Esto lo podemos hacer de 2 formas:***
 - a) **Cambiando el tema** de `<application>` en el **manifest**
 - b) **Cambiando el tema del que hereda nuestro tema principal** en el fichero `/res/values/styles.xml`

Pasos a seguir (II)

4. En nuestro caso, vamos a modificar `/res/values/styles.xml` para que **nuestro tema extienda de alguno de los siguientes** (dependiendo de si queremos partir de un tema claro u oscuro):
- `Theme.AppCompat.NoActionBar`
 - `Theme.AppCompat.Light.NoActionBar`

Vamos a escoger, por ejemplo, el segundo. Nos quedará:

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">@color/color_primary</item>
        <item name="colorPrimaryDark">@color/color_primary_dark</item>
        <item name="colorAccent">@color/color_accent</item>
    </style>

</resources>
```

5. Podemos cambiar los colores de la barra de acción como hicimos en el anterior documento (se hace igual)

Pasos a seguir (III)

6. **Añadimos un componente Toolbar a nuestro XML del layout** en la parte de **arriba del todo**, ya que estamos simulando una AppBar.

La etiqueta XML adecuada es:

`<android.support.v7.widget.Toolbar...>`

5. Las *especificaciones de Material Design* aconsejan que las barras de acción (**AppBar**) tengan una **elevación de 4dp**.
6. Por eso, este código ejemplo, añade una ToolBar y le da el aspecto de flotando sobre la activity:

```
<android.support.v7.widget.Toolbar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/my_toolbar"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
/>
</android.support.v7.widget.Toolbar>
```

Pasos a seguir (IV)

Mi **layout completo** será:

<LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:showIn="@layout/activity_main" tools:context=".MainActivity"
android:orientation="vertical">
```

<android.support.v7.widget.Toolbar

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/my_toolbar"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
    >
```

</android.support.v7.widget.Toolbar>

<!-- Resto de la interfaz de usuario -->

</LinearLayout>

Pasos a seguir (V)

Explicación propiedades Toolbar:

```
<android.support.v7.widget.Toolbar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/my_toolbar"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
/>
```

- 1) **layout_width** y **layout_height** → como todos los componentes
- 2) **minHeight** → si asignamos esta **propiedad al alto estándar de una action bar** (*?attr/actionBarSize*) conseguimos que los botones de acción y menú de overflow queden siempre en la parte superior de la barra de herramientas aunque incrementemos su tamaño mediante **layout_height**
- 3) **background** → asignaremos a esta propiedad el valor *?attr/colorPrimary* de forma que se utilice como **color de la barra de herramientas** el que hemos definido como **colorPrimary** en el tema de la aplicación (archivo */res/values/styles.xml*).

Pasos a seguir (V)

Explicación propiedades Toolbar:

```
<android.support.v7.widget.Toolbar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/my_toolbar"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
/>
```

- 4) **elevation** → define la **elevación del componente**, lo que determina la sombra que proyectará sobre el elemento inferior.
 - La **elevación estándar** de la action bar definida en las **guías de diseño** es de **4dp**.
 - Esta propiedad sólo tiene efecto cuando la app se ejecuta sobre Android 5.0 o superior, aunque más adelante veremos cómo solucionarlo.
- 5) **theme** → Mediante esta propiedad definimos el **tema a utilizar por la Toolbar** (y que heredarán sus controles hijos). **No debemos confundir esto con el tema definido a nivel de aplicación en el fichero *styles.xml*.**
- 6) **app:popupTheme** → Sólo se usa cuando sea necesario. Permite definir el **tema a aplicar sobre el menú overflow** (*para distinguirlo de la Toolbar*).

- Por ejemplo, si queremos usar un tema claro a nivel de aplicación y la barra de acción oscura, (*eso hicimos en el ejemplo de ActionBar antigua*) haremos:
- Tema de la aplicación → elegimos un tema claro: *Theme.AppCompat.Light* o similar.
 - Tema de la toolbar → utilizaremos un tema oscuro asignando la propiedad *android:theme de la Toolbar*, en este caso con el tema *ThemeOverlay.AppCompat.Dark.ActionBar*.
 - app:popupTheme → Esta propiedad la utilizaremos sólo si es necesario.
 - ❑ En nuestro caso particular lo es, para “arreglar” un efecto colateral de utilizar el tema oscuro para la Toolbar.
 - ❑ Y es que utilizar este tema también tiene como efecto que el menú de overflow aparezca de color oscuro, con lo cual, no lo veríamos.
 - ❑ Para conseguir que la barra sea oscura pero el menú claro podemos asignar un tema específico claro sólo a este menú utilizando la propiedad *app:popupTheme*, en nuestro caso el tema *ThemeOverlay.AppCompat.Light*.

Continuación Pasos a seguir (VI)

Ya tenemos el XML configurado. He incluido un cuadro de texto y un CheckBox para ver que queda como en la ActionBar vieja (documento anterior).

7) **El menú overflow** se define igual que antes.

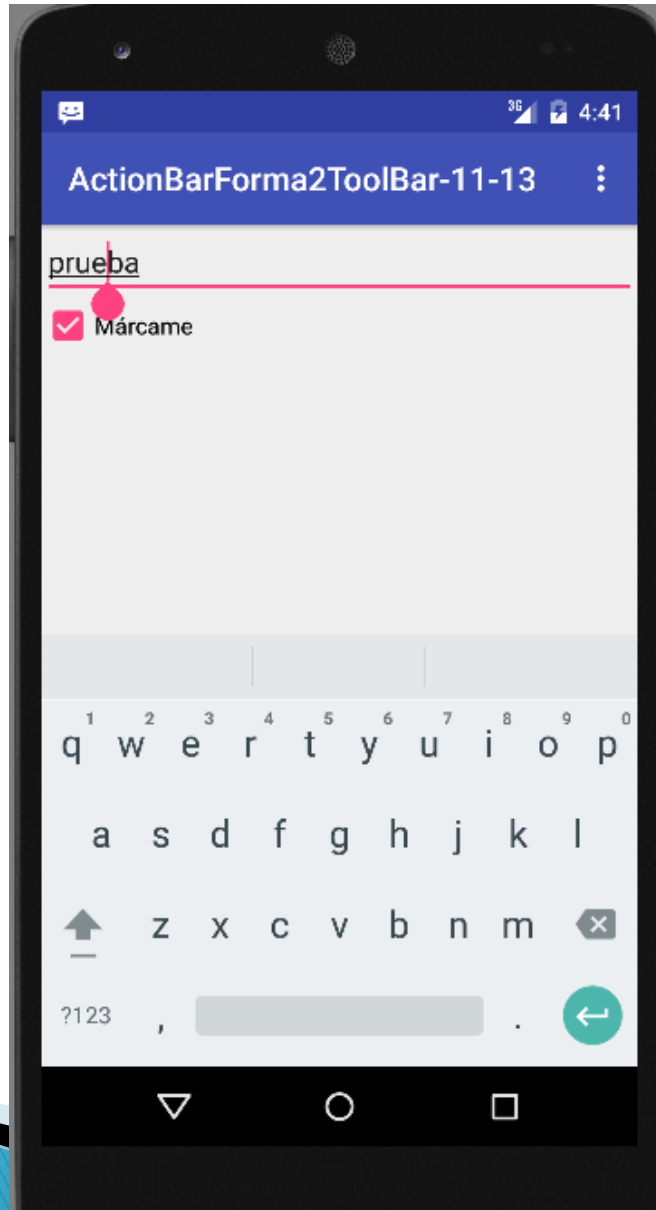
8) **PASO MÁS IMPORTANTE:**

- ❑ en **JAVA** debemos indicar que nuestra **Toolbar** actuará como **ActionBar** (app bar) de la activity.
- ❑ Para ello, en el método `onCreate()` llamamos al método “**setSupportActionBar()**” pasándole la referencia a la **Toolbar**:

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);  
        setSupportActionBar(toolbar);  
    }  
}
```

Pasos a seguir (VII)

Si lo ejecutamos el resultado será igual al ejemplo hecho con la ActionBar:



OCULTAR LA BARRA DE ACCIÓN (ToolBar)

- ▶ Este método que explico aquí sólo funciona con Toolbar.
- ▶ Con ActionBar usada de la forma antigua no funciona, aunque no da error.
- ▶ Para ocultar la barra de acción, tecleais:

`getSupportActionBar().hide();`

Devuelve objeto
ActionBar

Método de **ActionBar** que
la oculta.
Existe show() también

PROBLEMAS QUE SUPONE USAR ASÍ Toolbar y SOLUCIONES

- **PROBLEMAS:**

1. Como **Toolbar se añade al XML del layout de una Activity**, habría que hacerlo para cada Activity de la app.

- **SOLUCIÓN:**

1. **Definir la Toolbar en un fichero XML propio**
llamado **/res/layout/toolbar.xml**
2. **Incluirla** en nuestros XML de las activities con la **cláusula <include...>**

PROBLEMAS QUE SUPONE USAR ASÍ Toolbar y SOLUCIONES

- Veamos el fichero /res/layout/toolbar.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
>
</android.support.v7.widget.Toolbar>
```


PROBLEMAS QUE SUPONE USAR ASÍ Toolbar y SOLUCIONES

- Veamos ahora cómo quedaría cualquier layout de una Activity donde queramos incluir esa Toolbar:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

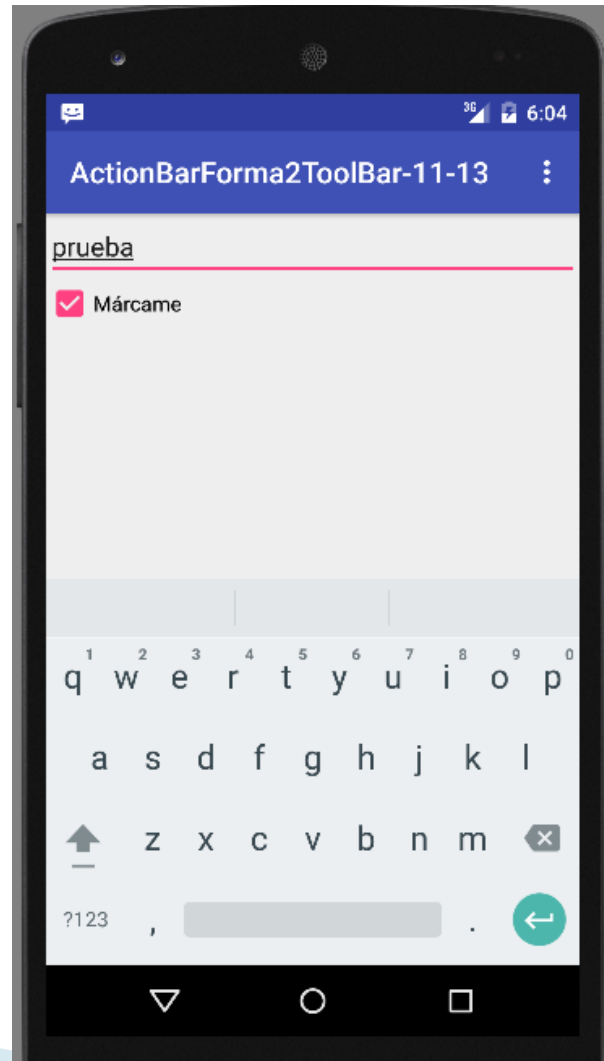
    <include android:id="@+id/appbar"
        layout="@layout/toolbar" />

    <!-- Resto de la interfaz de usuario -->

</LinearLayout>
```

EJECUCIÓN SOBRE Android 5.0 o superior

- ▶ Se ve la sombra bajo la barra de acción y la barra de estado (notificaciones) coloreada igual:

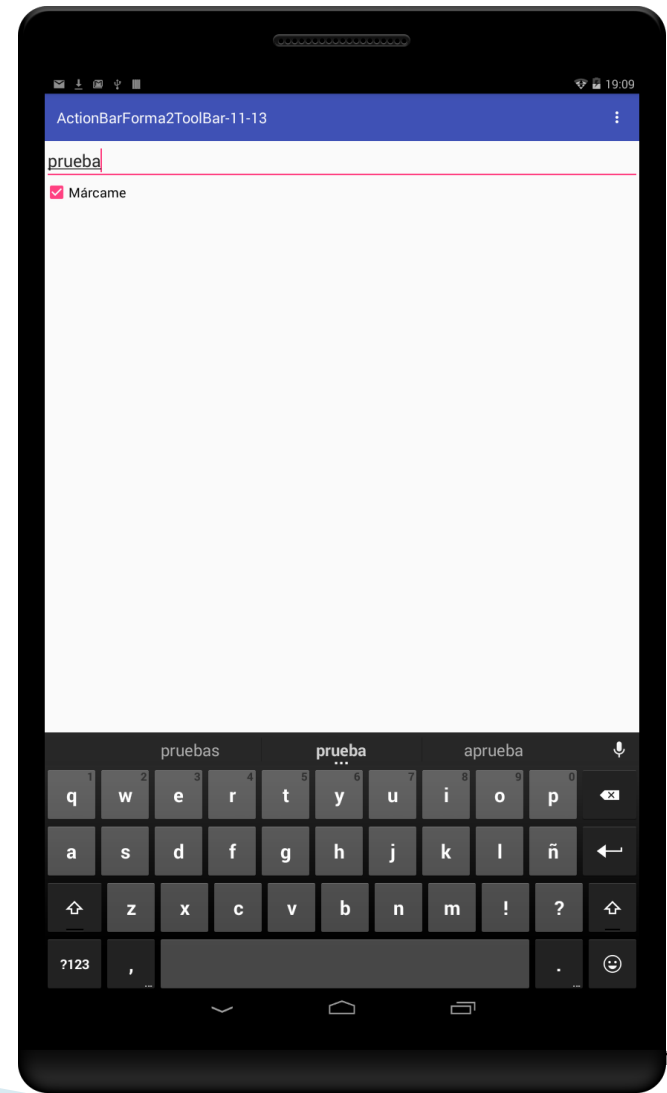


EJECUCIÓN SOBRE Android 4.x o anterior

- ▶ NO SE VE la sombra bajo la barra de acción y la barra de estado (notificaciones) se ve negra:

En este enlace de **SGOLIVER.NET** se muestra un truco que simula la sombra bajo la barra de acción.

Ir a la sección que dice: “*El método que os muestro a continuación está extraído de la aplicación oficial del Google I/O del año 2014...*”



CONCLUSIÓN: Ventajas de usar Toolbar

1. Puedo *personalizar la Action Bar por ejemplo, añadiendo controles dentro de ella, modificando su tamaño, ponerle título y subtítulo,...*
2. *Se puede añadir en cualquier parte de otro layout, por ejemplo, dentro de una CardView.*

EJEMPLO DE UNA Action Bar extendida con 2 líneas de título: *título principal y subtítulo*

- Modificamos el fichero /res/layout/toolbar.xml así:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="128dp"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:gravity="bottom"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="16dp" >

        <TextView android:id="@+id/txtAppBarTitulo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/Título"
```

EJEMPLO DE UNA Action Bar extendida con 2 líneas de título: *título principal y subtítulo*

- Modificamos el fichero `/res/layout/toolbar.xml` así:

```
<TextView android:id="@+id/txtAppBarSubTitulo"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/Subtítulo"
    style="@style/TextAppearance.AppCompat.Widget.ActionBar.Subtitle"
    />
```

```
</LinearLayout>
```

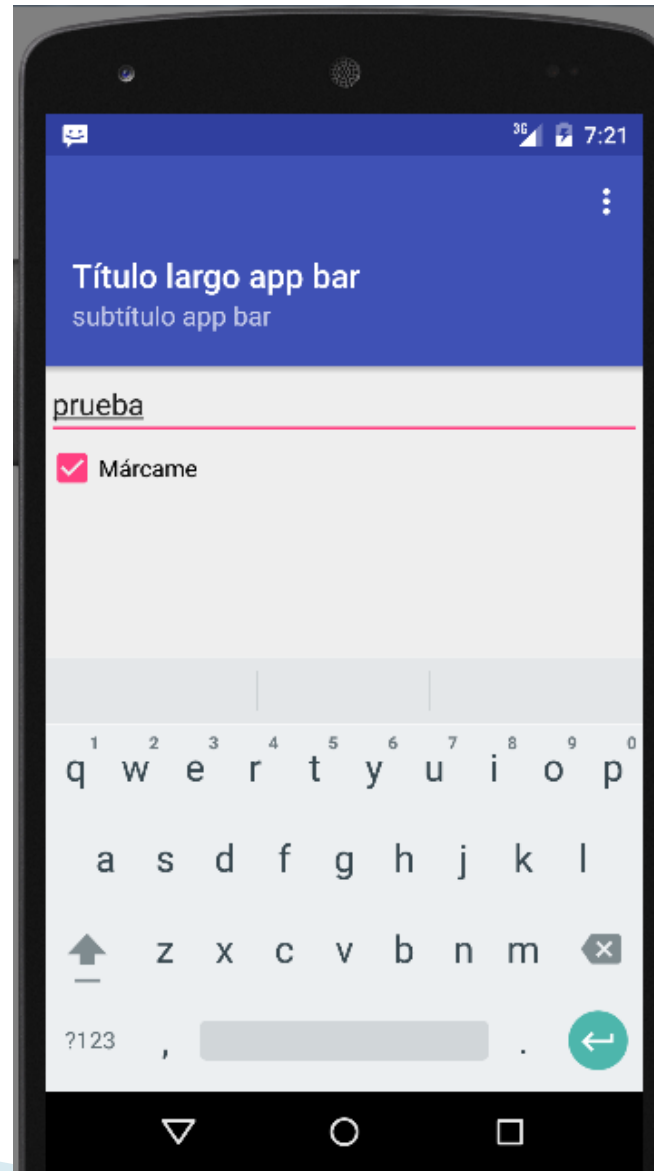
```
</android.support.v7.widget.Toolbar>
```

- Añadimos al `onCreate()` del `MainActivity` esta línea para que no salga el título de la Activity, sino sólo nuestros título y subtítulo:

```
getSupportActionBar().setDisplayShowTitleEnabled(false);
```

- Lo ejecutamos y saldrá:

EJEMPLO DE UNA Action Bar extendida con 2 líneas de título: *título principal y subtítulo*



OTRO EJEMPLO DE UNA Toolbar dentro de una CardView

Cómo se haría lo tenéis en el siguiente [enlace](#) de SGOLIVER, al final del blog.

