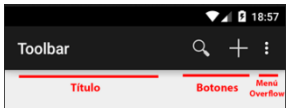
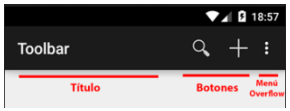
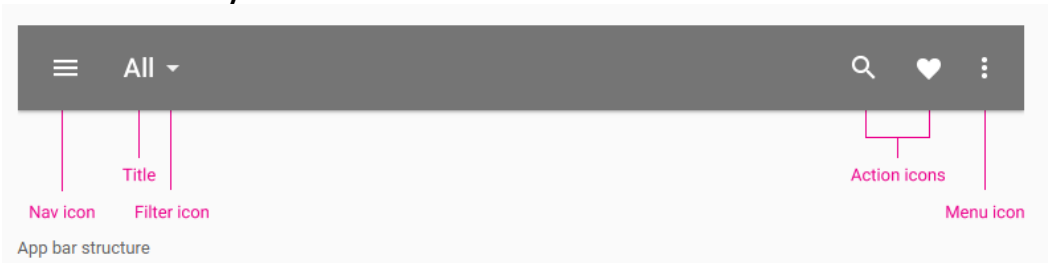


App Bar

App Bar

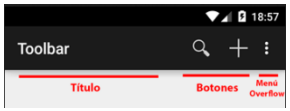


- La Action Bar o App Bar como se la ha rebautizado con la llegada de Material Design es la barra de título y herramientas que aparece en la parte superior de la gran mayoría de aplicaciones actuales de la plataforma Android.
- La App Bar normalmente muestra el **título** (puede ser el nombre de la aplicación o la actividad en la que nos encontramos o un filtro de página), una serie de **botones** de acción y un **menú** desplegable (menú de overflow) donde se incluyen más acciones que no tienen espacio para mostrarse como botón o simplemente no se quieren mostrar como tal.
- A la izquierda del título **también puede aparecer si es necesario** el icono indicativo de la existencia de menú lateral deslizante (**navigation drawer**) o el **botón Up** (navegación hacia arriba).



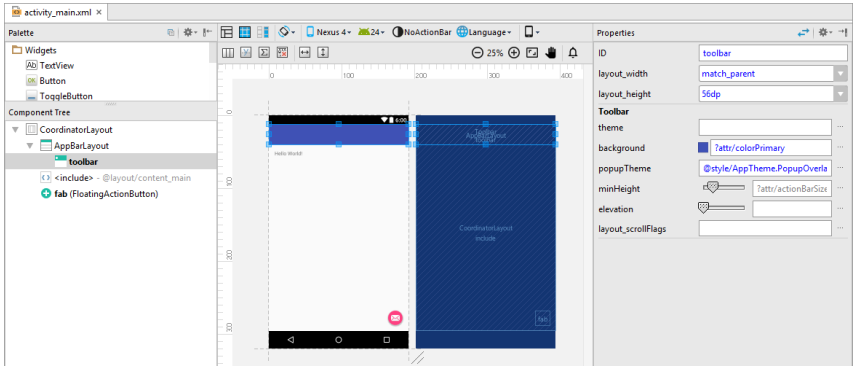
- Si se quiere destacar una acción principal para la pantalla Material Design recomienda utilizar el elemento “Floating Action Button”.

Plantillas



Basic Activity

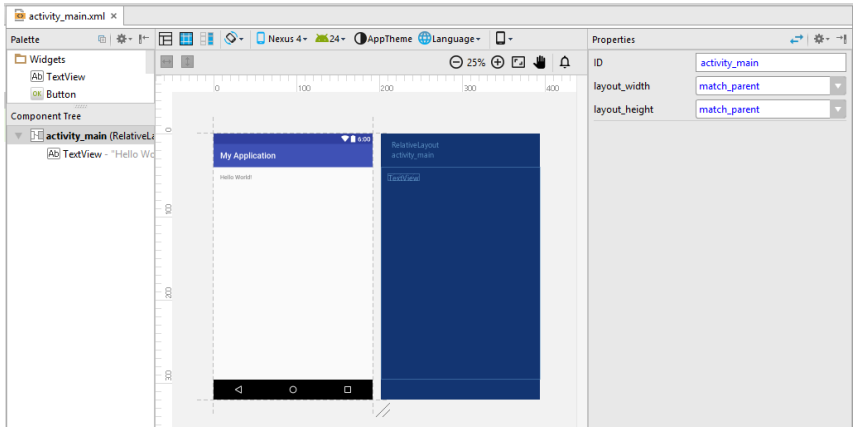
- El asistente de creación de AS ya nos ha hecho gran parte del trabajo.



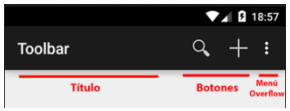
- RECOMENDADA!
- Se usa el control Toolbar (añadido en la librería de soporte de compatibilidad).

Empty Activity

- El desarrollador debe preparar tanto la interfaz gráfica como el código java (métodos) y xml (menu).



Pasos para añadir la App Bar a una actividad basada en plantilla Empty



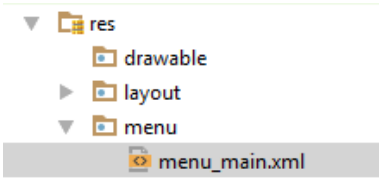
- Añadir la librería de soporte de compatibilidad al proyecto: implementation 'com.google.android.material:material:1.0.0'
- En el manifiesto, establecer el tema de la actividad utilizando uno de los AppCompat NoActionBar. Así se impide el uso de la nativa (y antigua) ActionBar: android:theme="@style/AppTheme.NoActionBar"
- Añadir Toolbar en el layout dentro del contenedor AppBarLayout

```
<android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />

</android.support.design.widget.AppBarLayout>
```

- En el método onCreate() de la actividad, establecer que el control Toolbar sea la ActionBar llamando al método setSupportActionBar() e importar la librería correspondiente (import androidx.appcompat.widget.Toolbar):

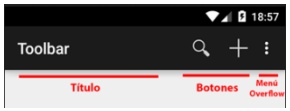
```
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
```

- Crear un recurso menú de opciones básico

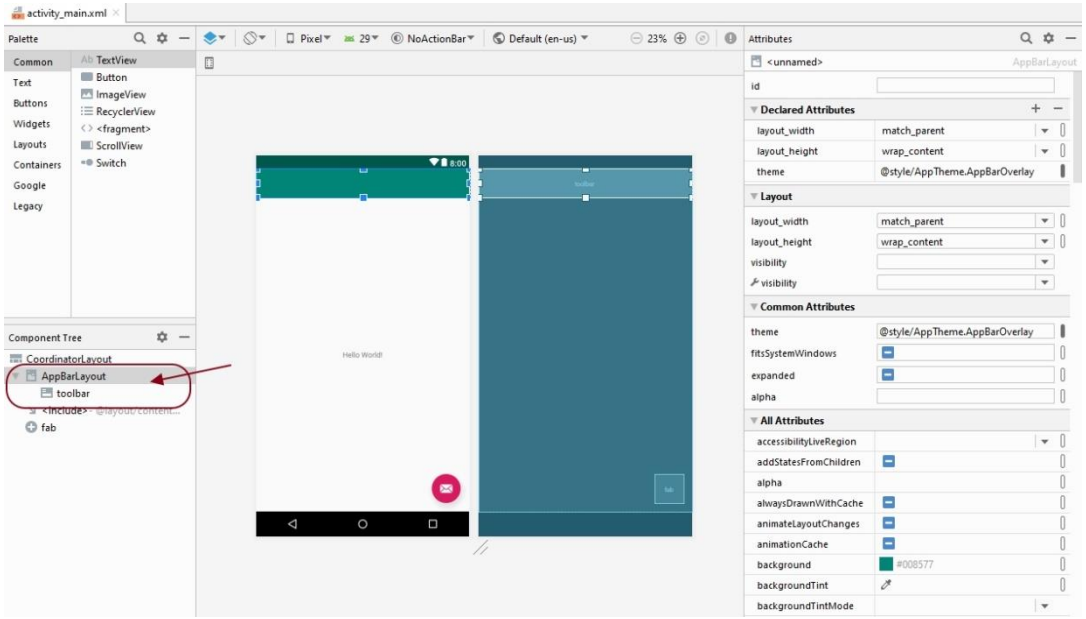


- **Trabajo que el asistente ya ha hecho si se escoge la plantilla Basic Activity y con el que se consigue que la actividad tenga una App Bar básica (con el nombre de la aplicación y un menú)**

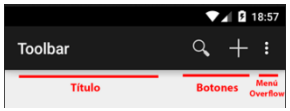
AppBarLayout



- La Toolbar representa el pequeño segmento rectangular, la App Bar es un contenedor con más posibilidades de personalización:
  - Expandir o colapsar la Toolbar para dejar espacio a la parte principal de la actividad
  - Scrolling



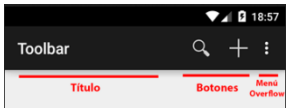
Atributos básicos Toolbar



- android:layout_height. Asignando a esta propiedad el alto recomendado por Material Design de una action bar (?attr/actionBarSize).
- android:background. Se asigna 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.
- 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. Para conseguir que la barra sea oscura pero el menú claro podemos asignar un tema específico sólo a este menú, en nuestro caso el tema ThemeOverlay.AppCompat.Light.

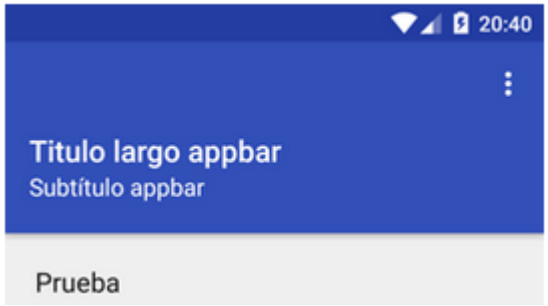
```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />

</com.google.android.material.appbar.AppBarLayout>
```

Ventajas de Toolbar

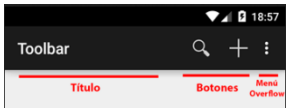


- Se puede seguir personalizando la action bar muy fácilmente, por ejemplo modificando su tamaño o añadiendo controles en su interior.
- Por ejemplo:
  - Hemos asignado dos nuevas propiedades en el control Toolbar. En primer lugar hemos incrementado el alto del control a 128dp, y además hemos establecido la propiedad gravity al valor "bottom", de forma que los componentes que añadamos en su interior se alineen sobre el borde inferior.
  - Adicionalmente, dentro del elemento Toolbar hemos añadido dos etiquetas de texto para mostrar el título y el subtítulo, como si se tratara de cualquier otro contenedor. Lo único a destacar es que he utilizado dos estilos predefinidos para estas etiquetas (AppCompat.Widget.ActionBar.Title y AppCompat.Widget.ActionBar.Subtitle) aunque podría utilizarse cualquier formato para estos elementos.
- Al disponer del control Toolbar como componente independiente podemos utilizarlo en otros lugares de nuestra interfaz y no siempre como barra de acciones superior.

```
<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_height="128dp"
    android:layout_width="match_parent"
    android:minHeight="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:gravity="bottom"
    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/txtAbTitulo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/Título"
            style="@style/TextAppearance.AppCompat.Widget.ActionBar.Title" />

        <TextView android:id="@+id/txtAbSubTitulo"
            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
```



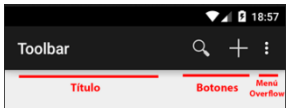
Acciones



- En la parte derecha de la App Bar aparecen las acciones más importantes y/o usuales de la actividad.
- Las acciones de la App bar pueden seleccionarse mediante botones de acción o desde el menú overflow

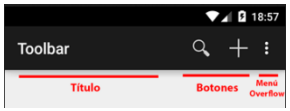
- Ya sea por botones o por menú overflow, constituyen lo que llamamos menú de acciones (opciones) de la aplicación.

Creación de un menú de opciones



- Como casi siempre, vamos a tener dos alternativas a la hora de mostrar un menú en nuestra aplicación Android:
  - mediante la **definición del menú en un fichero XML** y su posterior uso como un **recurso** más de la aplicación al que después se puede referenciar desde el código Java. Esta forma es una **buena práctica** porque nos permite separar el contenido del menú del código de nuestra aplicación. Además, resulta más sencillo visualizar la estructura y el contenido del menú en un formato XML.
  - directamente mediante **código** (no recomendable!).

Creando un menú de opciones como recurso

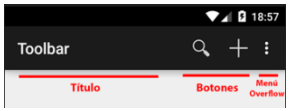


- Los **ficheros XML de menú se deben colocar en la carpeta "res/menu"** del proyecto.
- La estructura básica de estos ficheros es muy sencilla: tiene **un elemento principal <menu> que contendrá una serie de elementos <item>** que se corresponderán con las distintas opciones a mostrar en el menú.
- Estos elementos <item> tendrán a su vez varias propiedades básicas, como su ID (android:id), su texto (android:title) o su icono (android:icon). Los iconos utilizados deberán estar por supuesto en las carpetas "res/drawable-…" del proyecto o utilizar los de la plataforma. Pueden descargarse los recomendados desde aquí.
- Si deseamos "sub-menús" basta con que dentro del elemento <item> se introduzca otro menú

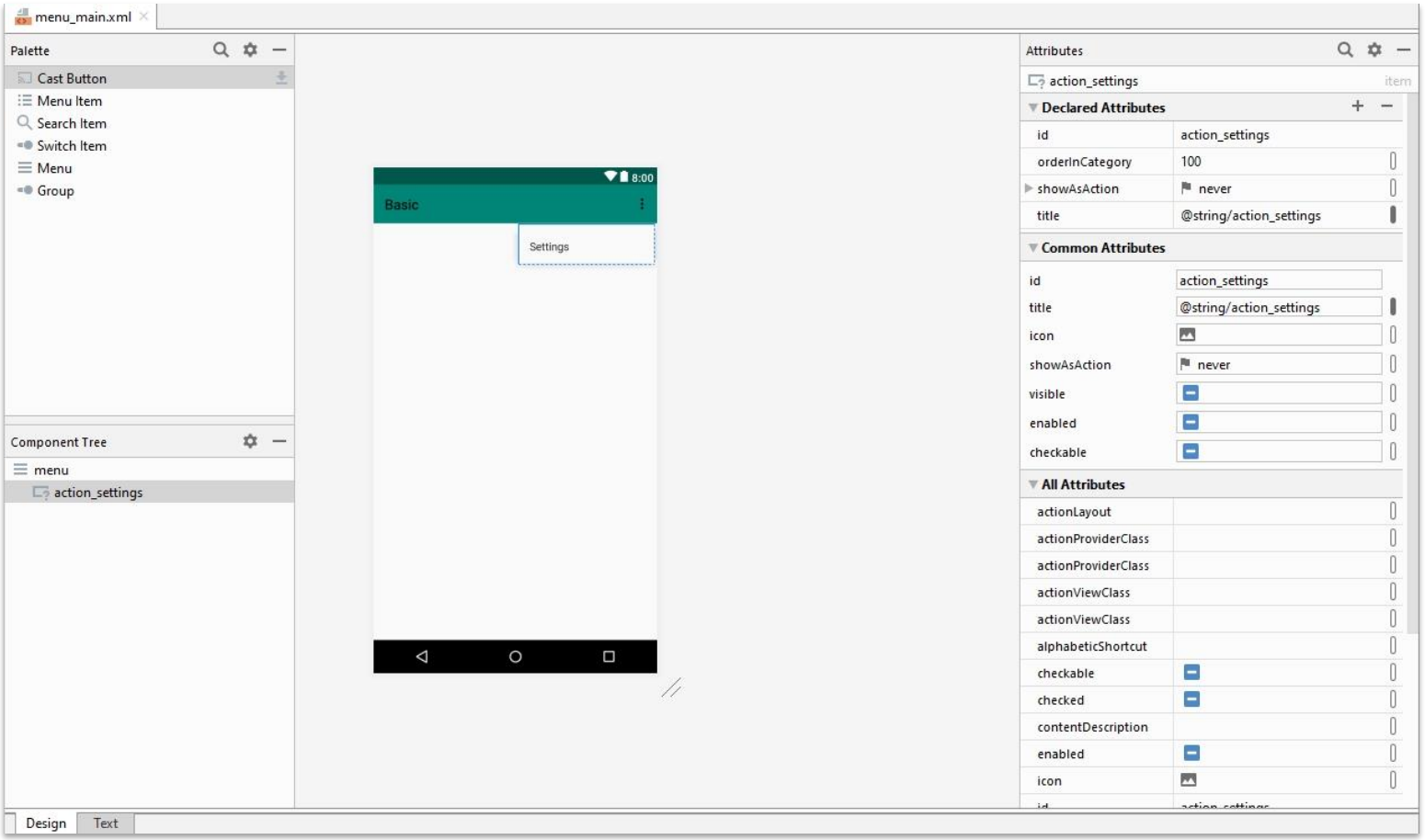
menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.pdm.p_37_appbar.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="Settings"
        app:showAsAction="never" />
</menu>
```

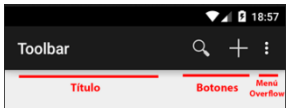
Editor de menús en AS



- AS proporciona un editor de menús para simplificar la tarea:

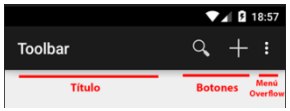


Atributo showAsAction

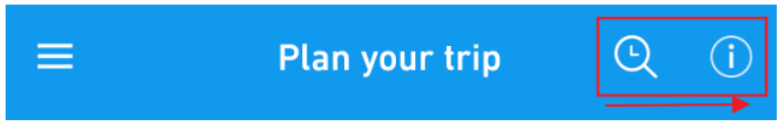


- Indica si la acción se mostrará como botón de acción o como parte del menú de overflow.
- Puede tomar varios valores:
  - **always**: La opción siempre se mostrará como botón de acción (este valor puede provocar que los elementos se solapen si no hay espacio suficiente para ellos)
  - **ifRoom**: Se 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.
  - **collapseActionView**. La opción es "retráctil" (útil en búsquedas)
  - Pueden combinarse si son compatibles: ifRoom|withText

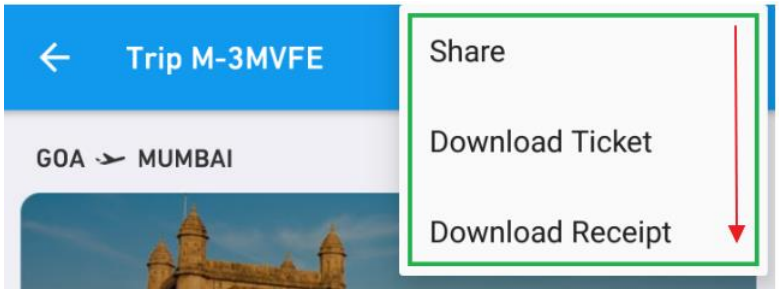
Atributo orderInCategory



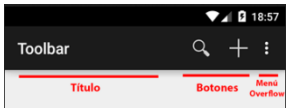
- Es un valor entero que dicta el orden en que aparecerán los elementos del menú dentro del menú cuando se muestre.
  - Para los elementos del menú en ActionBar: los elementos aparecerán de izquierda a derecha en ActionBar según el orden ascendente.



  - Para los elementos del overflowmenú: los elementos se mostrarán de arriba a abajo según el orden ascendente que haya especificado.



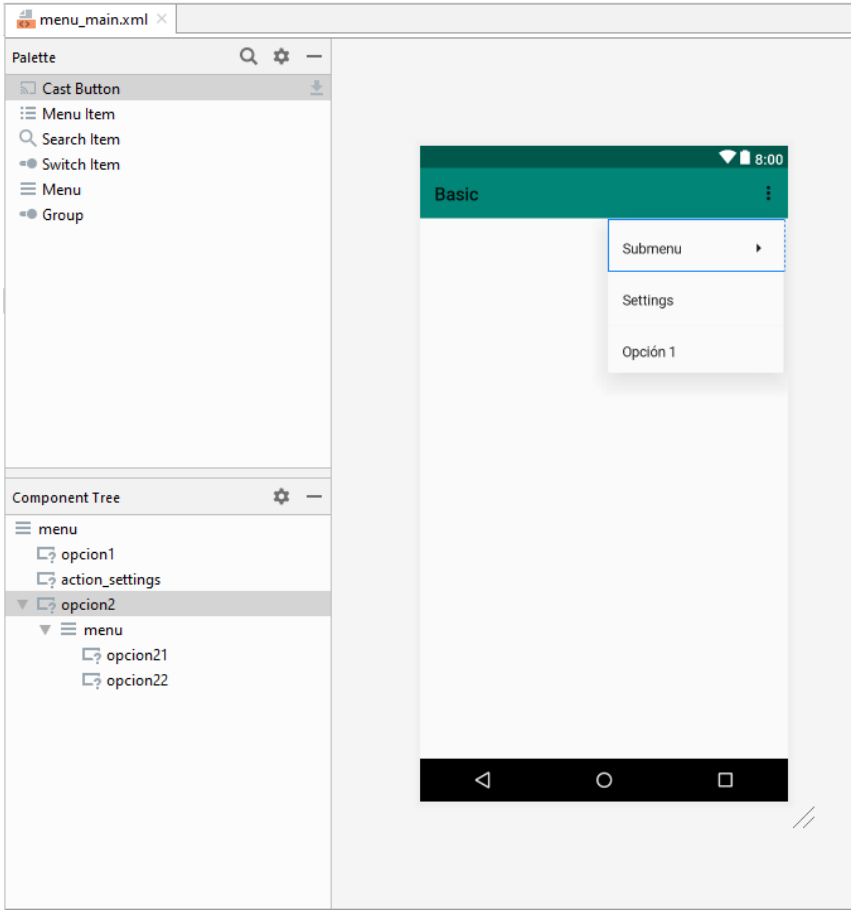
Submenú



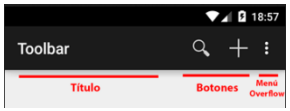
- No es más que un **menú secundario que se muestra al pulsar una acción determinada de un menú principal**.
- Se muestran en forma de lista emergente, cuyo título contiene el texto de la opción elegida en el menú principal.

menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.pdm.basic.MainActivity">
    <item
        android:id="@+id/opcion1"
        android:orderInCategory="500"
        android:title="Opción 1"
        app:showAsAction="never" />
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="Settings"
        app:showAsAction="never" />
    <item
        android:id="@+id/opcion2"
        android:title="Submenu">
        <menu>
            <item
                android:id="@+id/opcion21"
                android:title="Item" />
            <item
                android:id="@+id/opcion22"
                android:title="Item" />
        </menu>
    </item>
</menu>
```

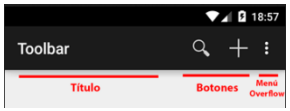


Código de la Activity



- Hay que usar los métodos:
  - **onCreateOptionsMenu()**: Permite crear un Menú de acciones que se podrá usar en la actividad
  - **onOptionsItemSelected()**: Cuando el usuario selecciona una opción del menú, el sistema llama al método onOptionItemSelected() de la actividad para responder al evento generado.

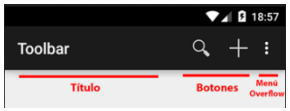
Método onCreateOptionsMenu()



- El asistente de AS ya ha escrito el código básico necesario.
- Se obtiene una referencia al *inflater* mediante el método **getMenuInflater()**
- El método **inflate()** vincula el identificador del archivo de recursos y el objeto de la clase Menu que llega como parámetro
- Por último se devuelve el valor true para confirmar que debe mostrarse el menú.

```
…
 @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);
      return true;
   }
…
```

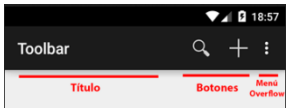
Método onOptionsItemSelected()



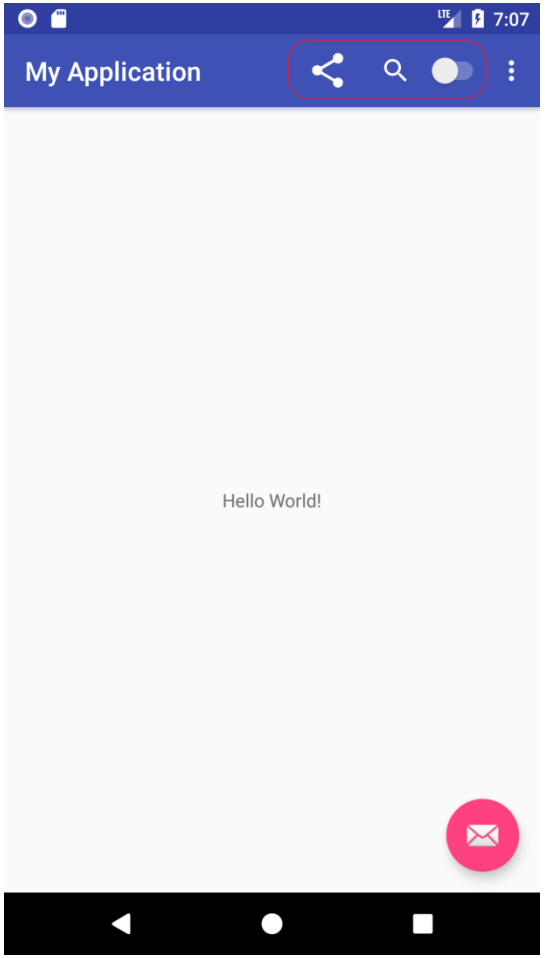
- Este método pasa el objeto MenuItem que corresponde a la opción seleccionada.
- Para conocer el ID de la opción utilizamos el método getItemId() que retorna el identificador único del ítem y que se definió desde el atributo android:id en XML.

```
…
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id. action_settings:
         …. (/*Lógica  al seleccionar el primer item 1; */
         break;
      case R.id.item2:
         …. (/*Lógica  al seleccionar el primer item 2; */
         break;
      default:
         return super.onOptionsItemSelected(item);
   }
   return true;
}
…
```

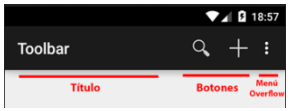
Más funcionalidades



- Ya hemos visto como cómo definir una *acción* simple, que puede ser un botón o un elemento del menú. Pero hay componentes más versátiles.
- Una ***vista de acción*** es una acción que proporciona una gran funcionalidad dentro de la AppBar. Por ejemplo, una vista de acción de búsqueda permite al usuario escribir su texto de búsqueda en la barra de la aplicación o una vista de acción de switch permite escoger entre dos estados.
- Un ***proveedor de acciones*** es una acción con su propio diseño personalizado. La acción aparece inicialmente como un botón o elemento de menú, pero cuando el usuario hace clic en la acción, el proveedor de acciones controla el comportamiento de la acción de la forma que quiera definir. Por ejemplo, el proveedor de acciones puede responder a un clic mostrando un menú.
- Los widgets especializados de vista de acción y proveedor de acciones son:
  - Switch item
  - Search item
  - ShareActionProvider (para el intercambio de información con otras aplicaciones). Necesita usar las librerías de soporte de compatibilidad.
  - Cast Button (para conectar, controlar y desconectarse de los receptores de transmisión como puedan ser auriculares bluetooth, Google Cast, etc.)



Icono "Up" en App Bar



- Todas las pantallas de una aplicación que no son la entrada principal (la "main") deben ofrecer al usuario una forma para ir a dicha pantalla principal pulsando el botón "Up" de la barra de acción (no es igual que botón "Atrás" que vuelve a la actividad anterior!).
- Para la actividad que se desee que cuente con la posibilidad de navegar a la actividad principal (MainActivity) hay que:
  - Hacer la declaración de esa jerarquía en el fichero Manifest.xml.
  - En el método OnCreate fijar que deseamos el botón "Up"
  - Si tiene AppBar, fijar el comportamiento del botón:
    - Comportamiento normal del botón (volver a actividad principal: No se necesita recoger la acción en el método onOptionsItemSelected() de la actividad basta con que el método llame a su superclase:

      return super.onOptionsItemSelected(item)
    - Otro comportamiento del botón: Hay que recoger en el método onOptionsItemSelected, la acción que se desea para el identificador android.R.id.home. Más información


```
<activity
    android:name=".SegundaActivity"
    android:label="SegundaActivity"
    android:parentActivityName=".MainActivity"
    android:theme="@style/AppTheme.NoActionBar">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.pdm.p_37_appbar.MainActivity" />
</activity>
```

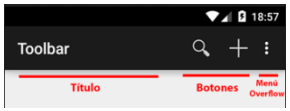
```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    //noinspection SimplifiableIfStatement
    if (id == android.R.id.home) {
        // ACCIONES NO NORMALES A REALIZAR
        return true;
    }

    return super.onOptionsItemSelected(item);
}
```

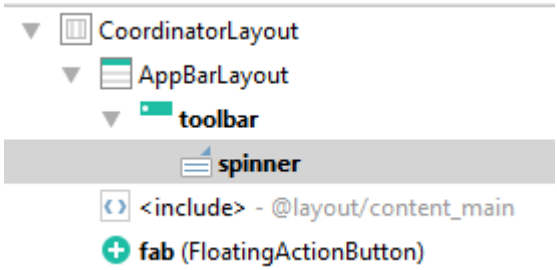
COMPORTAMIENTO NORMAL

SOLO SI NO SE DESEA COMPORTAMIENTO NORMAL

Filtro en App Bar



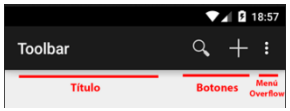
- El uso de filtro (es decir, una lista desplegable) como título integrado en la App Bar es sencillo.
- El control Toolbar se comporta como cualquier otro contenedor, en el sentido de que puede contener a otros controles.



- En el código java se trabaja como con cualquier spinner.

```
//Appbar page filter
Spinner spinner = (Spinner) findViewById(R.id.spinner);
ArrayAdapter<String> adapter = new ArrayAdapter<>(
        getSupportActionBar().getThemedContext(),
        android.R.layout.simple_list_item_activated_1,
        new String[]{"Opción 1 ", "Opción 2 ", "Opción 3 "});
spinner.setAdapter(adapter);
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        //... Acciones al seleccionar una opción de la lista
```

Otros métodos



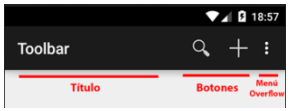
- Obtener la instancia de la App Bar:

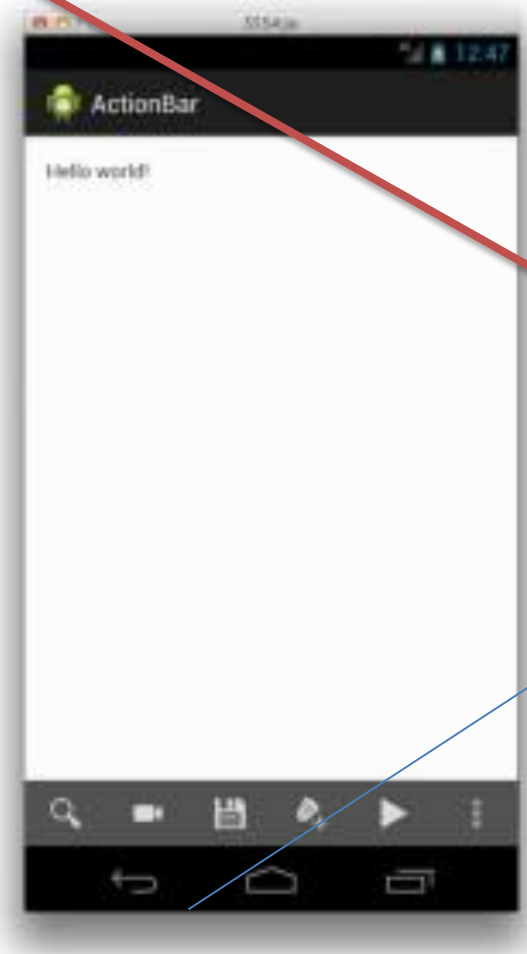
```
ActionBar actionBar = getSupportActionBar();
```

- Ocultar/mostrar la App Bar en tiempo de ejecución:

```
 if(actionBar.isShowing()) {
     actionBar.hide();
}
else {
     actionBar.show();
}
```

Separar el menú de opciones



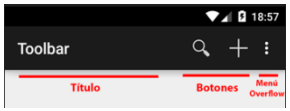


- Es posible mostrar el menú de acciones en la parte inferior de la pantalla.
- Basta con definir en el AndroidManifest dentro de la actividad deseada:

```
<activity
android:name="com.example.actionbar.MainActivity"
  android:label="@string/app_name"
  android:uiOptions="splitActionBarWhenNarrow" >
<intent-filter>
…
```

- Solo aparecerá cuando tenga tantas acciones que el SISTEMA determine que no caben en la App Bar.

Separar el menú de opciones

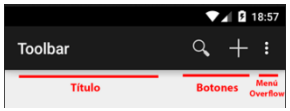


- Usar plantilla

Bottom Navigation Activity

que usa el elemento BottomNavigationView en el layout

Prácticas propuestas



- Realiza la hoja de ejercicios Ejercicios_09_AppBar