

Action Bar y Menú Contextual

.....

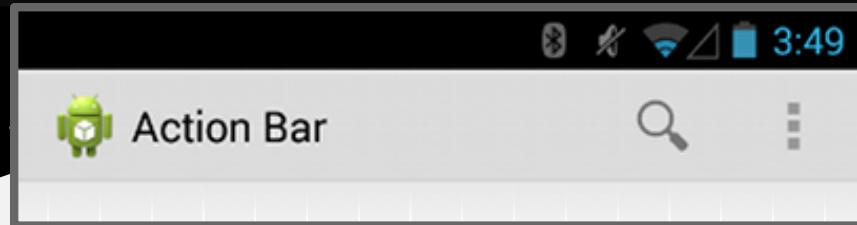
Aplicación Android

Temas

.....

1. Action Bar
2. Menú de Opciones
3. Menú Contextual

Action Bar



Es uno **elementos de diseño más importantes** que se pueden implementar en las Activity de nuestra aplicación.

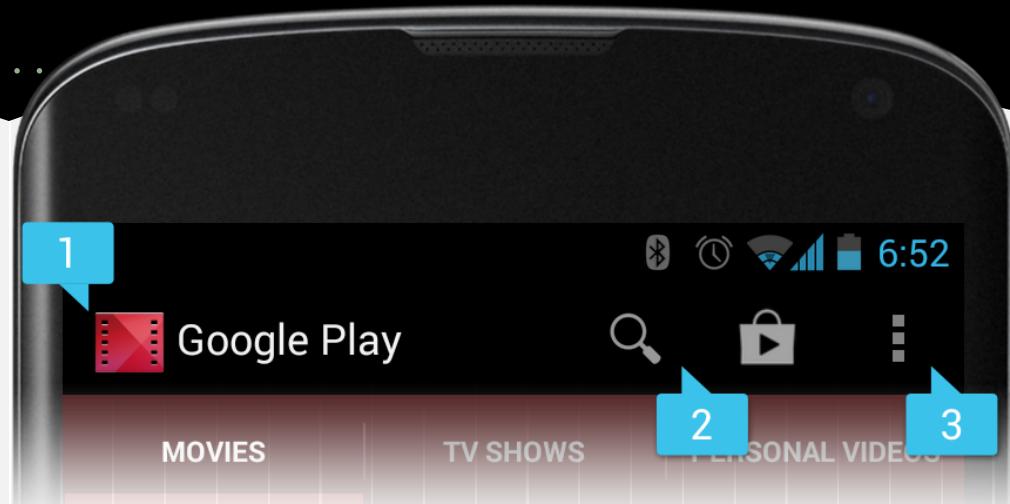
Ofrece una interfaz para que al usuario le sea familiar usar la aplicación Android, porque brinda coherencia entre otras Aplicaciones de Android.

Las funciones clave incluyen:

- Un espacio dedicado a dar una identidad a la aplicación y donde se indica la ubicación del usuario en la aplicación.
- El acceso a las acciones importantes en una forma predecible (como por ejemplo: Buscar).
- Apoyo a la navegación y a la vista de conmutación (con fichas o listas desplegables).

Action Bar

Ejemplo en una aplicación:



1. Un espacio dedicado a dar una **identidad a la aplicación** y donde se indica la ubicación del usuario en la aplicación.
2. El **acceso a las acciones importantes** en una forma predecible (como por ejemplo: **Buscar**).
3. **Apoyo a la navegación** y a la vista de conmutación (con fichas o listas desplegables).

Action Bar

.....

Las APIs del [ActionBar](#) se agregaron por primera vez en Android 3.0 (API nivel 11) pero están disponibles para versiones anteriores en Support Library desde Android 2.1 (API nivel 7).

Precaución: Asegurarse importar la clase ActionBar (y API relacionadas) del paquete apropiado:

- Si los niveles de soporte de API inferiores a 11: importación `android.support.v7.app.ActionBar`
- Si el apoyo a nivel de API sólo 11 y superior: importación `android.app.ActionBar`

Más información del Menú y Action Bar, debe consultarse en la guía de elemento [Menu](#).

<https://developer.android.com/guide/topics/ui/menus.html#context-menu>

<https://developer.android.com/reference/android/app/ActionBar.html>

Android 3.0 o superior

Si usamos Android 3.0 (API nivel 11) os superior, Action Bar está incluido en todas las Activities que usen el tema `Theme.Holo` (o uno de sus descendientes), que es el tema por defecto que otorga el atributo `targetSdkVersion` or `minSdkVersion` cuando es seteado a "11" o superior.

Para agregar un Action bar, sólo hay que agregar el atributo al proyecto en `AndroidManifest`, como se muestra en el ejemplo: **res/AndroidManifest**.

xml



```
<manifest ... >  
  <uses-sdk android:minSdkVersion="11" ... />  
  ...  
</manifest>
```

Soportar Android inferiores a 3.0

Agregar la Action Bar para ser ejecutada en versiones anteriores de Android 3.0 (desde 2.1 a 3.0) requiere la inclusión de la Support Library de Android en la aplicación.

Se debe leer el documento de configuración de la Support Library y configurar la librería appcompat v7 (una vez que haya descargado el Package de la Library, siga las instrucciones para agregar las bibliotecas con recursos).

En el archivo

res/AndroidManifest.xml

debe quedar:



```
<manifest ... >  
    <uses-sdk android:minSdkVersion="7"  
              android:targetSdkVersion="18" />  
    ...  
</manifest>
```

Soportar Android inferiores a 3.0

Una vez que tenga la Support Library de Android hay que actualizar la clase de su Activity para que extienda de la clase Action Bar:

src/(namespace)/MainActivity.java



```
public class MainActivity extends ActionBarActivity { ... }
```

En el archivo **res/AndroidManifest.xml**, actualizar o bien el elemento <application> o <activity> para utilizar uno de los temas Theme.AppCompat.



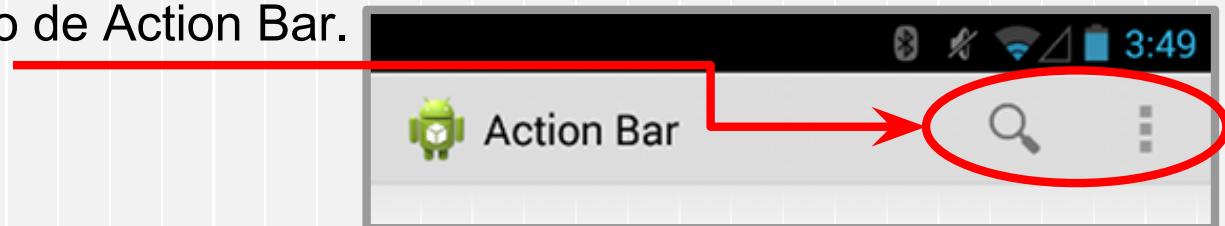
```
<activity android:theme="@style/Theme.AppCompat.Light" ... >
```

Botones de Acción

La Action Bar permite añadir botones para los elementos de acción más importantes relacionados con el contexto actual de la aplicación.

Los que aparecen directamente en la Action Bar con un icono y/o texto se conocen como botones de acción.

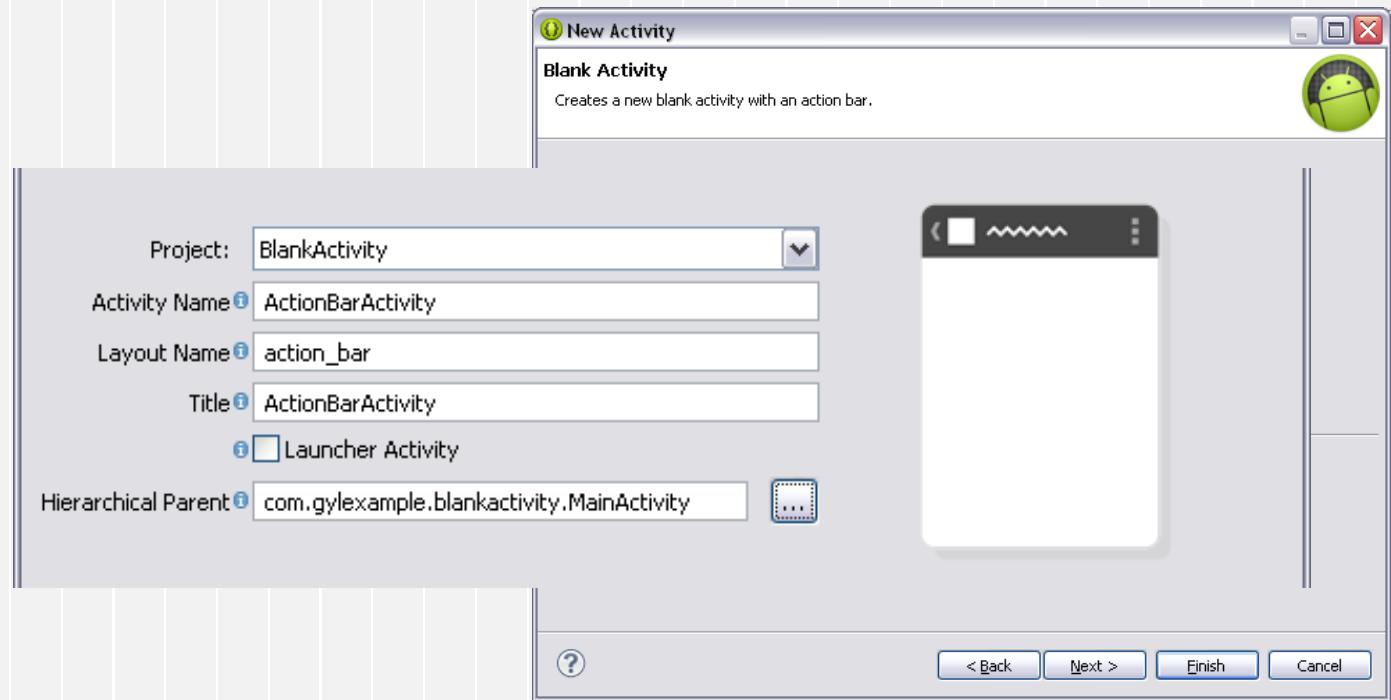
Las Acciones que no caben en la Action Bar o no son importantes están ocultos en el menú de desbordamiento de Action Bar.



La figura muestra un Action Bar con un Botón Acción de búsqueda y el Desbordamiento, que indica que hay más acciones.

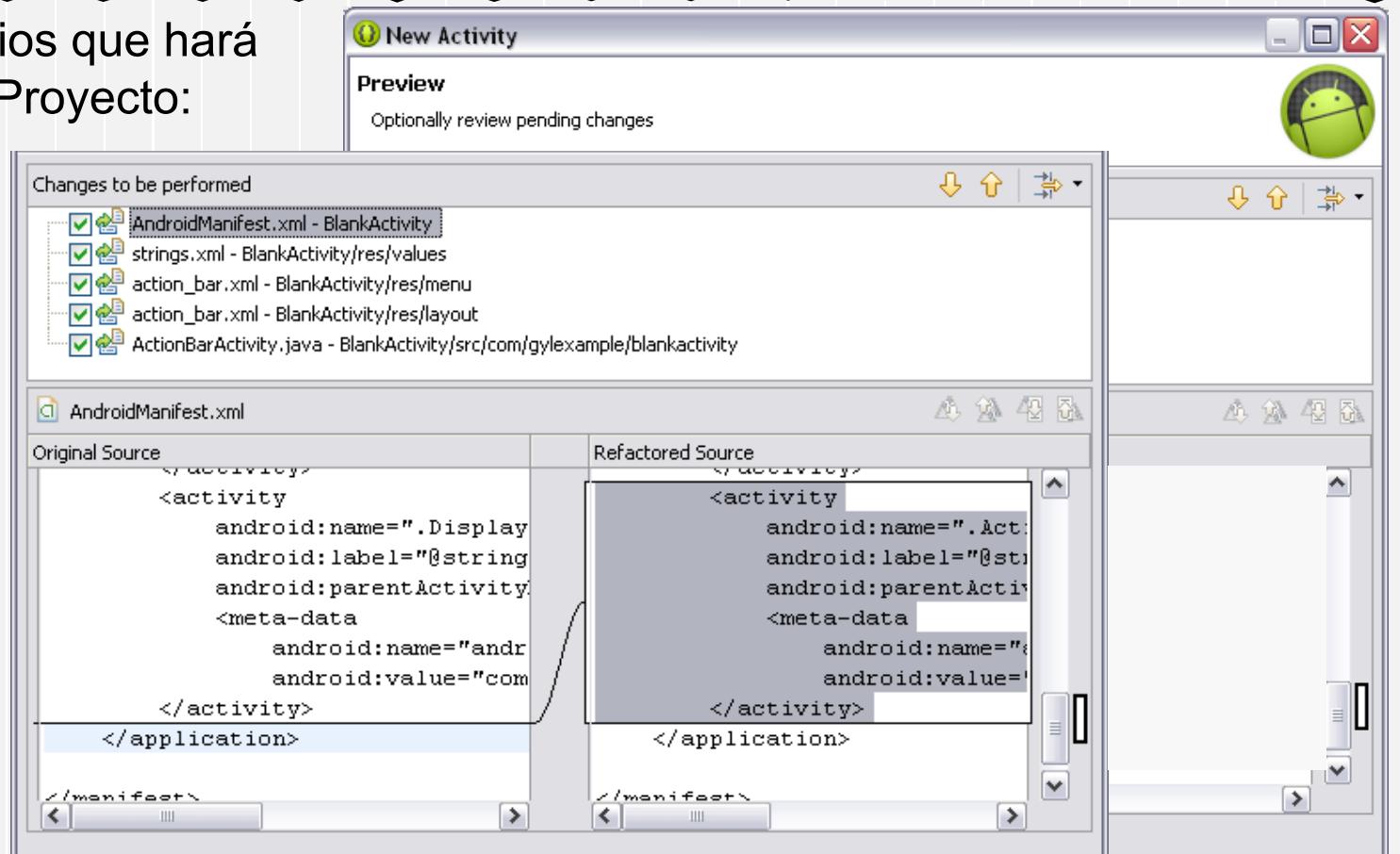
Ejercitación

Crear una actividad nueva que muestre un **ActionBar**.



Agregar Activity con ActionBar

Informa los cambios que hará el IDE Eclipse al Proyecto:



Modificar XML Menu

Al crear una Activity se crea un menu en la carpeta res\menu

Actualizarlo para que tenga los siguientes items:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

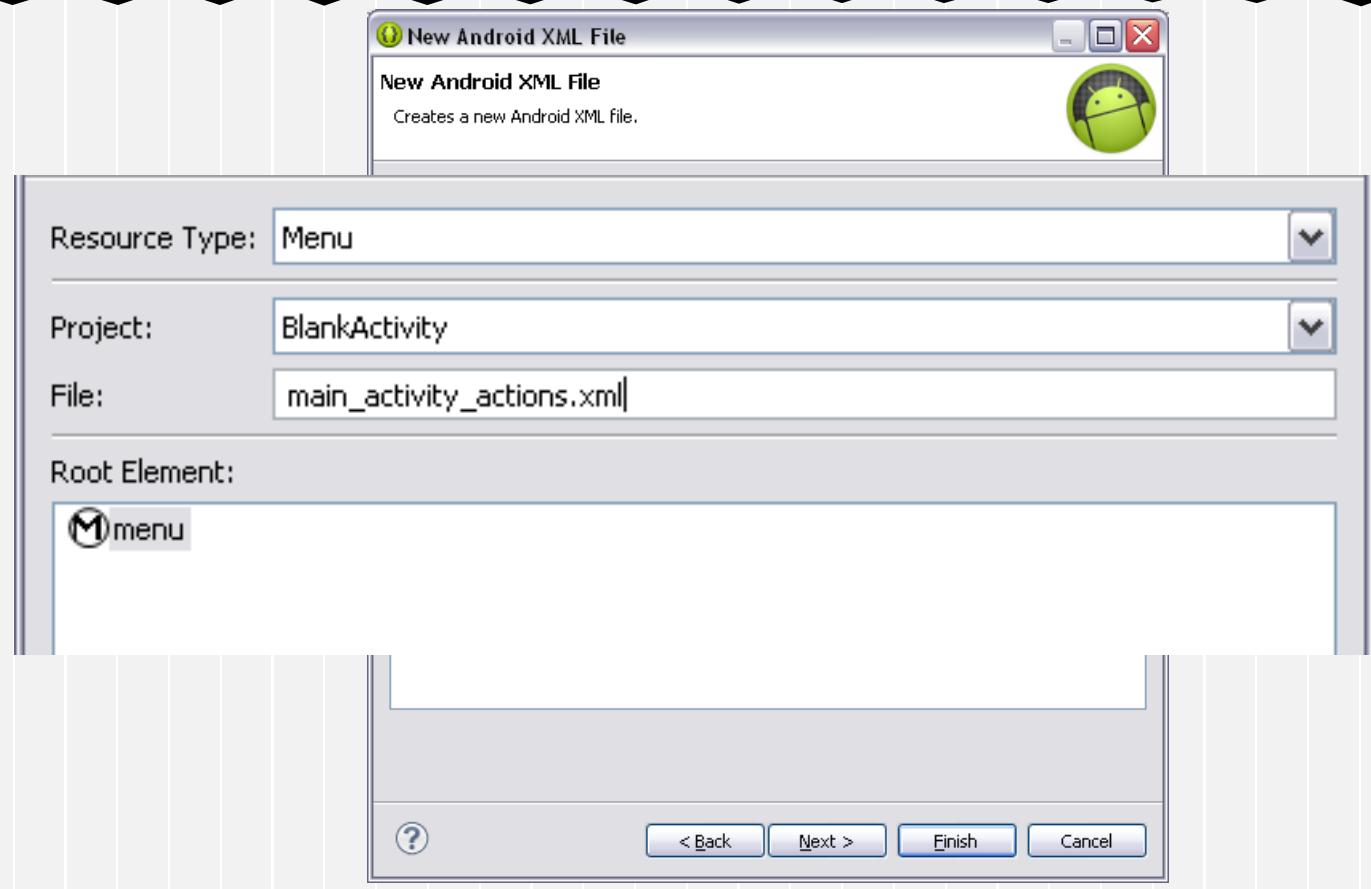
    <item android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"/>

    <item android:id="@+id/action_call"
        android:icon="@drawable/ic_action_call"
        android:title="@string/action_call" />
</menu>
```

Agregar XML Menu

Ahora vamos a crear
nuevos elementos
menu:

1. Sobre carpeta
res → botón der
→ New → Other
→ Android XML
File.
2. Next → Finish



Agregar XML Menu

Se crea un archivo xml en la carpeta menu con el siguiente formato:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    "Aqui se debe agregar los ítems del Action Bar"
</menu>
```

Agregar XML Menu

Se deben agregar los ítems de menu.

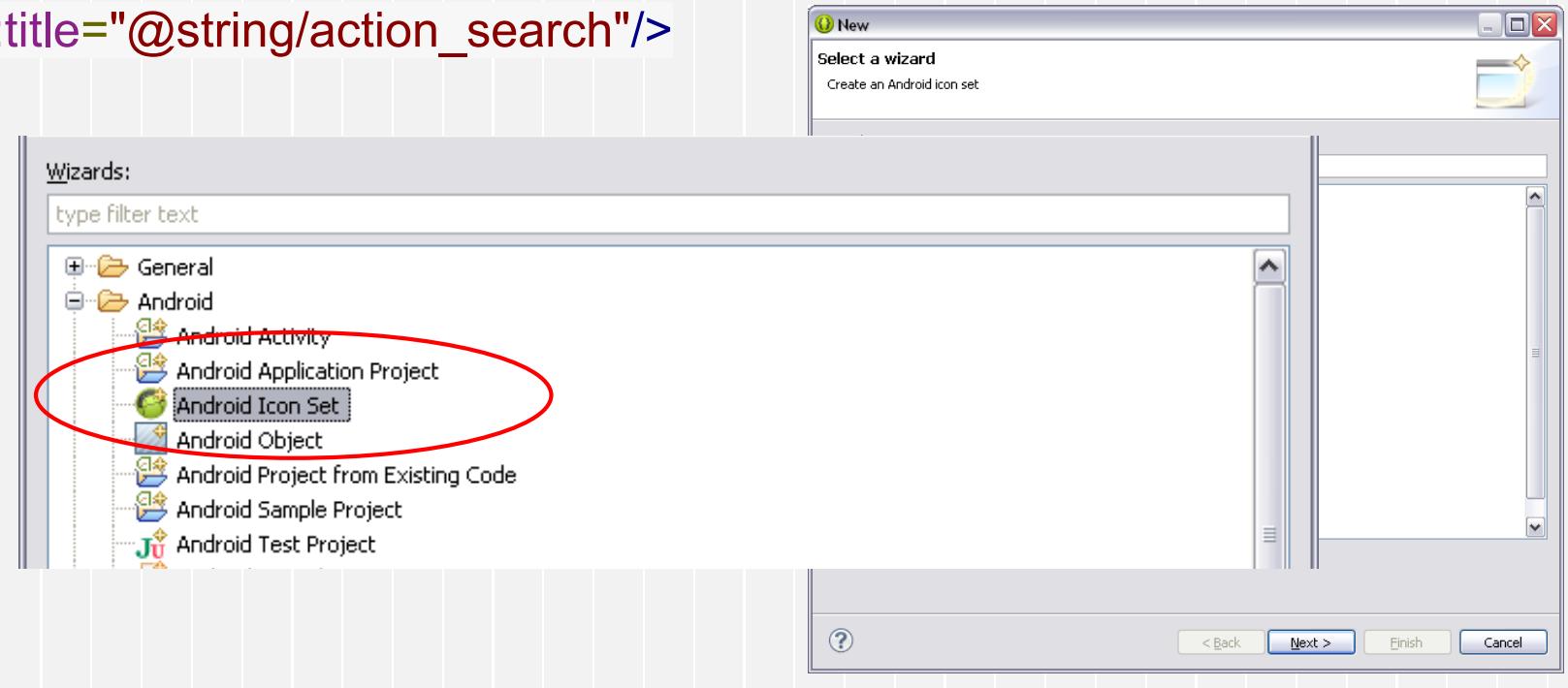
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item android:id="@+id/action_search"
          android:icon="@drawable/ic_action_search"
          android:title="@string/action_search"/>

    <item android:id="@+id/action_compose"
          android:icon="@drawable/ic_action_compose"
          android:title="@string/action_compose" />
</menu>
```

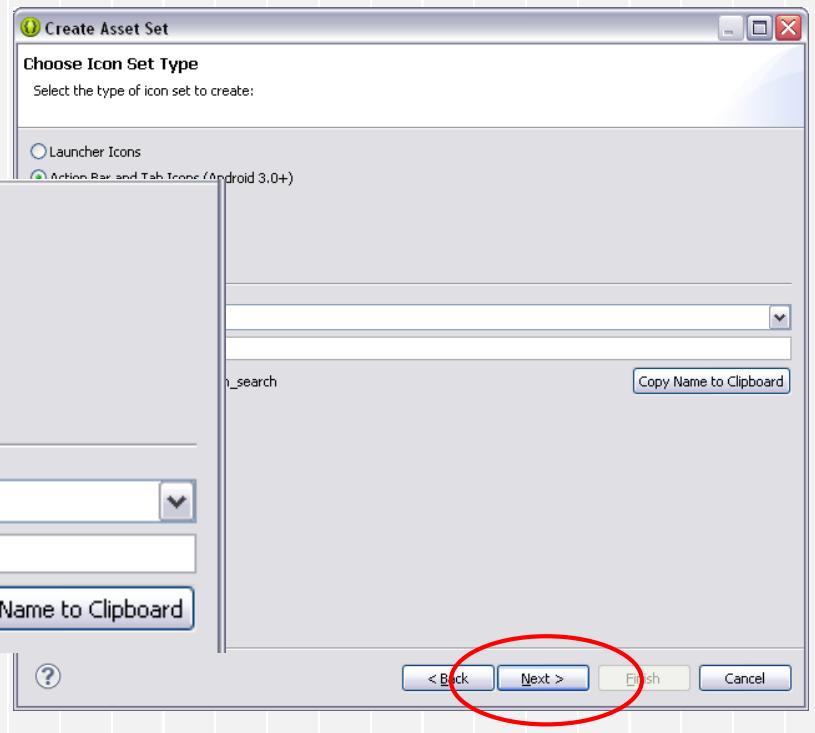
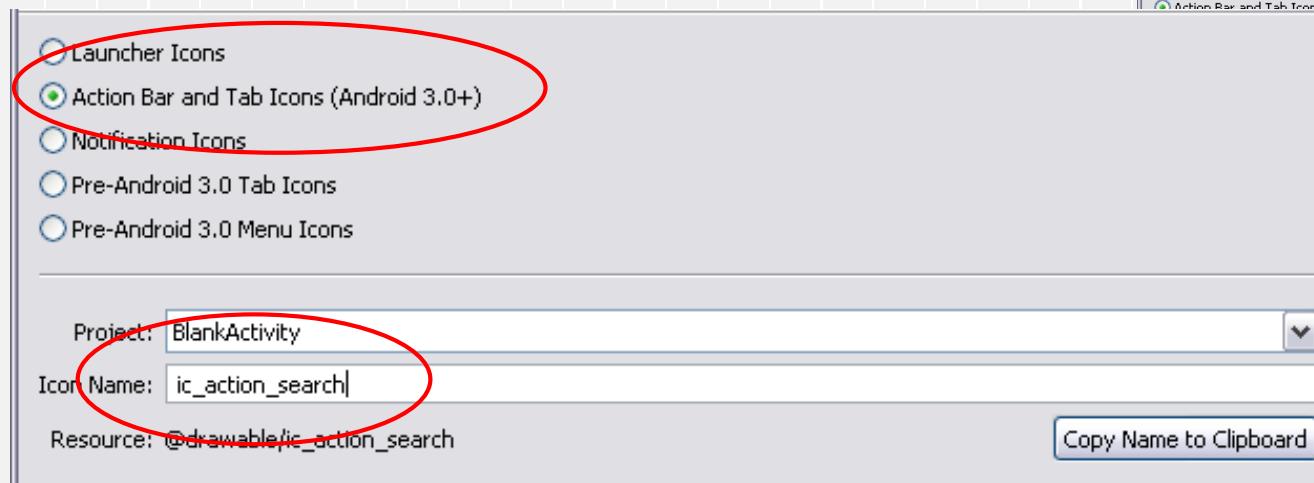
Agregar Icono

```
<item android:id="@+id/action_search"
      android:icon="@drawable/ic_action_search"
      android:title="@string/action_search"/>
```



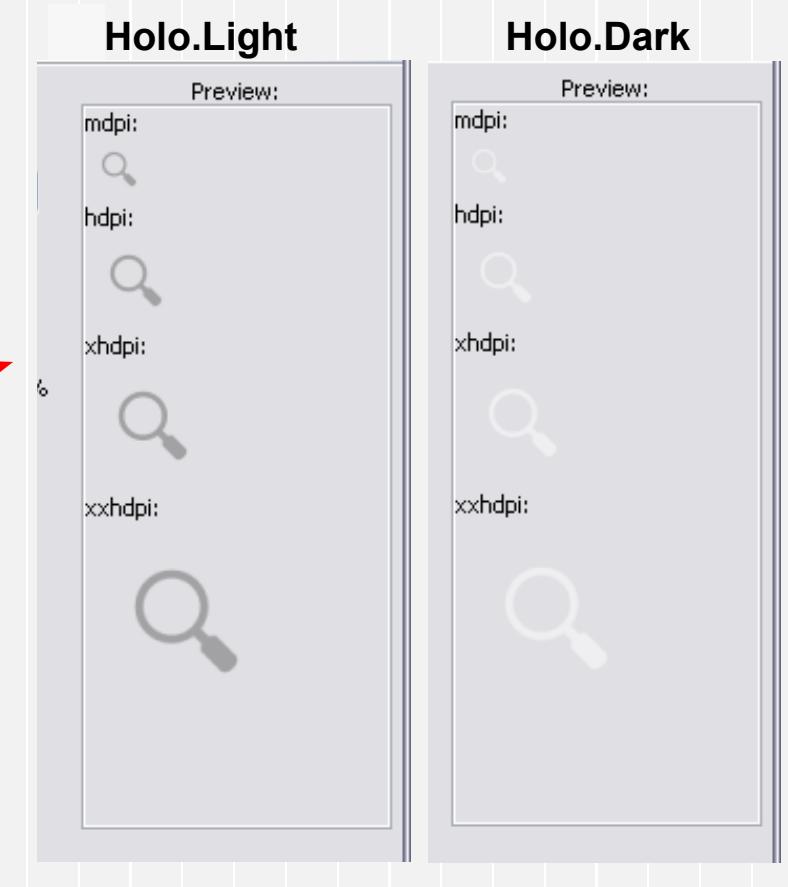
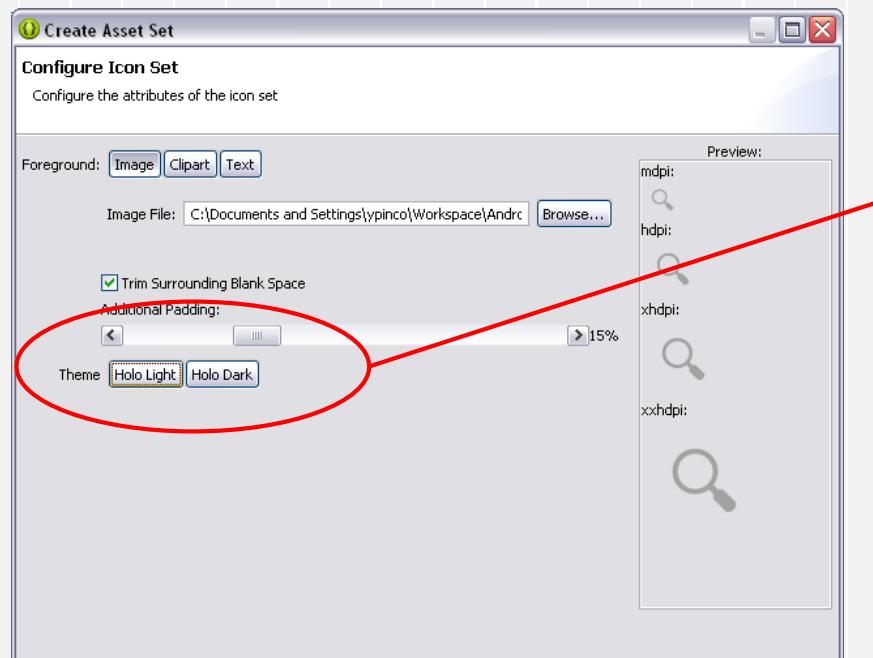
Agregar Icono

```
<item android:id="@+id/action_search"
      android:icon="@drawable/ic_action_search"
      android:title="@string/action_search"/>
```



Agregar Icono

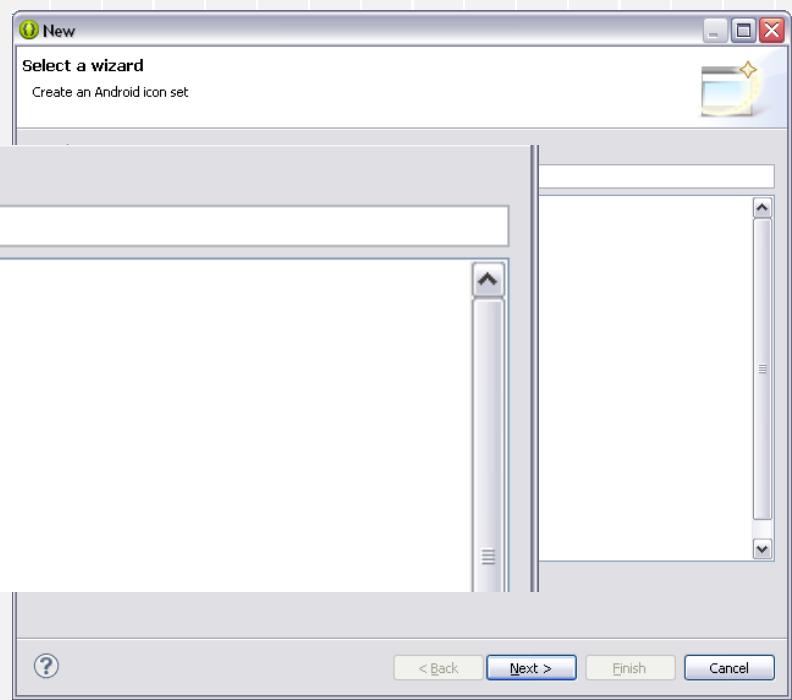
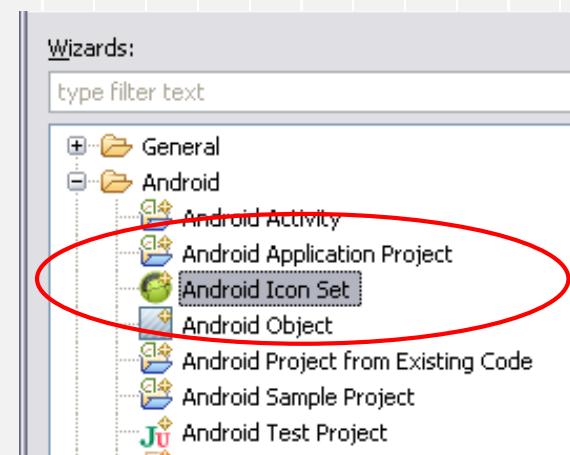
```
<item android:id="@+id/action_search"
    android:icon="@drawable/ic_action_search"
    android:title="@string/action_search"/>
```



Agregar Icono

Hacer lo mismo para el otro ícono

```
<item android:id="@+id/action_compose"  
      android:icon="@drawable/ic_action_call"  
      android:title="@string/action_call" />
```



Agregar XML Menu

Se crea un archivo xml en la carpeta menu con el siguiente formato:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item android:id="@+id/action_search"
      android:icon="@drawable/ic_action_search"
      android:title="@string/action_search"/>

<item android:id="@+id/action_compose"
      android:icon="@drawable/ic_action_call"
      android:title="@string/action_call" />
</menu>
```

→ Agregar en el archivo:
res\values\Strings.xml

Modificar Código Activity

Modificar el método `onCreateOptionsMenu(Menu menu)` para que invoque al menu creado. Para ello modificar el archivo de la carpeta `src\(namespace)\MainActivity.java`:

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu items for use in the action bar  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.main_activity_actions, menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

Modificar Código Activity

Sobreescibir el método onOptionsItemSelected(MenuItem item) para que determine qué menú fue seleccionado. Para ello modificar el archivo de la carpeta `src\(namespace)\MainActivity.java`:

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle presses on the action bar items  
    switch (item.getItemId()) {  
        case R.id.action_search:  
            Toast.makeText(getApplicationContext(), "Click en el ícono Buscar", Toast.LENGTH_LONG).show();  
            return true;  
        case R.id.action_call:  
            Toast.makeText(getApplicationContext(), "Click en el ícono Llamar", Toast.LENGTH_LONG).show();  
            return true;  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

OJO → poner cada uno de los “id_xxx” definidos en menu

Cambiar Estilo Aplicación

AndroidManifest.xml → Modificar la propiedad android:theme de la aplicación.
Compilar para ver las diferencias entre estilos.

```
<application>
    ...
    android:theme="@android:style/Theme.Holo.Light" >
        <!-- Probar los diferentes estilos de Themes Standard Android
            android:theme="@android:style/Theme.Holo.Dialog.NoActionBar.MinWidth"
            android:theme="@style/AppTheme" >
    ...
</application>
```

Botones de Acción



Repasemos.....

Para crear Botones de Acción en ActionBar:

1. Definir los Botones de Acción en el XML
2. Especificar Apariencia de los Botones del Action Bar
3. Definir las acciones que se activarán con cada Botón de Acción
4. Añadir Botón de Actividades de menor nivel

Definir el menu.xml res\menu

main_activity_actions.xml

Todos los botones de acción y otros ítems del Desbordamiento son definidos en el XML del Recurso [menu](#).

Crear un nuevo archivo XML en el directorio del proyecto [res/menu/](#) que se denomine: [main_activity_actions.xml](#)

Agregar un elemento [`<item>`](#) por cada ítem que se quiera incluir del Action Bar.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.
com/apk/res/android" >

    <item
        android:id="@+id/menuitem1"
        android:showAsAction="ifRoom"
        android:title="Prefs">
    </item>
    <item
        android:id="@+id/menuitem2"
        android:showAsAction="ifRoom"
        android:title="Test">
    </item>

</menu>
```

Definir el menu.xml res\menu

En el XML del Menú se declara el comportamiento y apariencia de cada uno de los ITEMS del MENU.

android:showAsAction="ifRoom"

menuitem1 aparecerá como un Botón de Preferencias y estará disponible siempre que haya lugar().

android:showAsAction="never"

menuitem2 va a aparecer siempre en menú de desborde.

Por defecto siempre aparecen en el menú desborde. Las buenas prácticas indican que se debe especificar el atributo aunque no aparezca.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.
com/apk/res/android" >

    <item
        android:id="@+id/menuitem1"
        android:showAsAction="ifRoom"
        android:icon="@drawable/ic_action_prefs"
        android:title="Prefs">
    </item>
    <item
        android:id="@+id/menuitem2"
        android:showAsAction="never"
        android:title="Test">
    </item>

</menu>
```

Definir el menu.xml res\menu

El atributo `android:icon` requiere que se asigne el `ID` del recurso imagen asociada.

El nombre que sigue a `@drawable/` debe ser el nombre de una imagen de mapa de bits que se haya guardado en el directorio del proyecto `res/drawable/`.

(`"@drawable/ic_action_search"` se refiere a `ic_action_search.png`).

El atributo `title` utiliza un recurso de `string` que está definido en `Strings.xml` en carpeta `res/values`.

Importante:

Para asegurarse respetar los lineamientos de la `iconografía` de Android, se deben usar los íconos provistos en el `Paquete de íconos del Action Bar`.

- `Lineamientos de la iconografía`: <https://developer.android.com/design/style/iconography.html#action-bar>
- `Paquete de íconos del Action Bar`: <https://developer.android.com/design/downloads/index.html#action-bar-icon-pack>

Definir el Menu del Action Bar

Hay que implementar el método `onCreateOptionsMenu()` en nuestra `ActivityMain.java` para insertar el recurso menú al objeto `Menu` mencionado.

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.mainmenu, menu);  
    return true;  
}
```

Responder a Botones de Acción

el método `onOptionsItemSelected()` es invocado cuando un usuario presiona un Botón de acción u otro ítem del menú de desborde.

En la implementación del método, la llamada al método `getItemId()` del `MenuItem`, determina qué elemento fue presionado. Y ahí se determina la acción a realizar en función del valor devuelto por `getItemId()`.

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.menuitem1:  
            Toast.makeText(this, "Menu Item 1 selected",  
            Toast.LENGTH_SHORT).show();  
            break;  
        case R.id.menuitem2:  
            Toast.makeText(this, "Menu item 2 selected",  
            Toast.LENGTH_SHORT).show();  
            break;  
        default:  
            break;  
    }  
    return true;  
}
```

Respondiendo al Icono Home

La barra de acciones muestra también el icono de la aplicación.

En el caso de ser pulsado, el **método *onOptionsItemSelected*** es invocado con el valor *android.R.id.home*.

Si quisiéramos hacer algo ante esa selección agregar:

Dentro de *onOptionsItemSelected* agregar en el switch....

```
case android.R.id.home:
```

```
    Intent intent = new Intent(this, OverviewActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    startActivity(intent);
    break;
```

Actividades de Menor Nivel

En el Androidmanifest.xml

Todas las Activity de nuestra aplicación (NO LAUNCHER) deben volver a la pantalla HOME pulsando el botón Arriba en la Acción Bar.

Se debe declarar la actividad padre en el archivo de Android Manifest y de esa manera se habilita el botón Subir

```
<application ... >
...
    <!-- A child of the main activity -->
<activity
    ...
        <!-- Parent activity meta-data to support 4.0 and lower -->
<meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.myfirstapp.MainActivity" />
</activity>
</application>
```

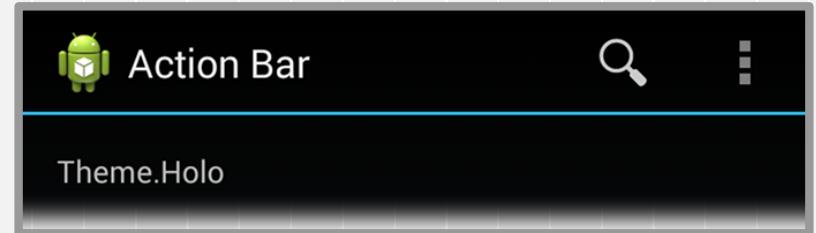
Utilizar Estilos de Android

Android incluye dos temas base que determinan el color de la barra de acción:

```
<application android:theme="@android:style/Theme.Holo.Light" ... />
```

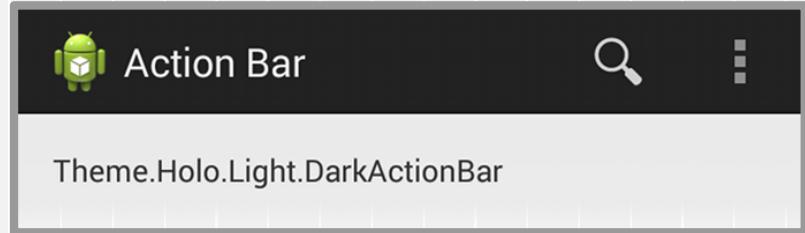
Puede aplicar estos temas a toda su aplicación o para las Activity individuales que declaró en su archivo AndroidManifest con el atributo `android:theme` para el elemento

`<application>` ó `<activity>` ó individual.



Utilizar Estilos de Android

También se puede usar una Action Bar dark con el resto de la Activity en color Blanco declarando el tema como [Theme.Holo.Light.DarkActionBar](#).



Cuando usamos el Support Library, se debe usar el Tema [Theme.AppCompat](#):

- [Theme.AppCompat](#) para el tema "dark".
- [Theme.AppCompat.Light](#) para el tema "light".
- [Theme.AppCompat.Light.DarkActionBar](#) para el tema light con la Action Bar dark.

Hay que asegurarse que los íconos del Action Bar que usemos contrastan con el color de la Action Bar. El [Action Bar Icon Pack](#) nos es de mucha ayuda porque incluye los íconos para las acciones estándar para ambos temas de Action Bar Holo light y Holo dark.

View Custom del ActionBar

Para añadir una vista personalizada a la barra de acciones hay que agregar en `OnCreate()`:

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
  
    ActionBar actionBar = getSupportActionBar();  
  
    actionBar.setCustomView(R.layout.actionbar_view);  
  
    actionBarsetDisplayOptions (ActionBar.DISPLAY_SHOW_CUSTOM | ActionBar.DISPLAY_SHOW_HOME);  
}
```

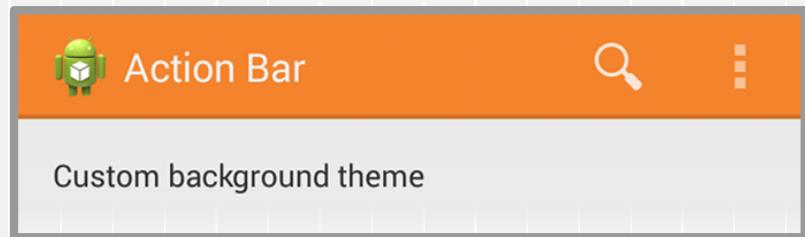
Customizar el ActionBar

Cambiar el color de fondo de la Action Bar requiere crear un tema propio para la Activity que sobreescriba la propiedad `actionBarStyle`. (`res/values/styles.xml`)

Esta propiedad apunta a otro estilo en donde se puede sobreescibir (override) la propiedad `background` (fondo) para especificar un recurso estirable (drawable) para especificar el recurso (drawable) para el fondo del Action Bar.

Si la aplicación usa [Solapas de navegación](#) ó el [split action bar](#), entonces se puede especificar el fondo para esas Barras usando las propiedades: `backgroundStacked` y `backgroundSplit`, respectivamente.

Cuidado: Es importante que se declare apropiadamente el tema padre del cual heredará las propiedades y el tema inherente a ambos estilos. Sin el estilo padre, el Action Bar no contará con varias propiedades hasta que se declaren explícitamente.

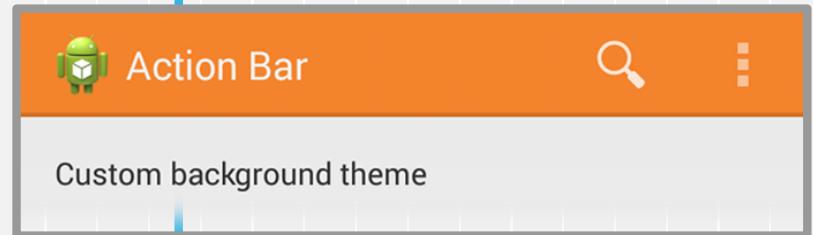


Customizar el Background

Para Android 3.0 o Superior
res/values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
        parent="@android:style/Theme.Holo.Light.DarkActionBar">
        <item name="android: actionBarStyle">@style/MyActionBar</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
        parent="@android:style/Widget.Holo.Light.ActionBar.Solid.Inverse">
        <item
            name="android:background">@drawable/actionbar_background
        </item>
    </style>
</resources>
```



Luego el tema para toda la aplicación o para alguna Activity individual:

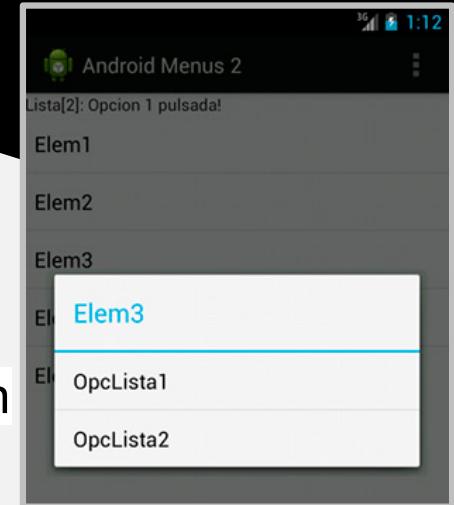
```
<application android:theme="
    @style/CustomActionBarTheme"
    ... />
```

Menú Contextual

Menú contextual para una Vista

Se puede asignar un menú contextual para una View.

También se activa el menú contextual si el usuario presiona por un tiempo "largo" la View.



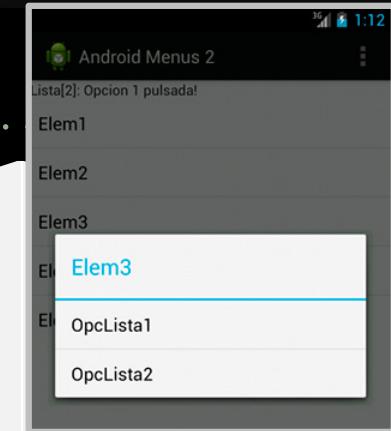
Un menú de contexto para una View se registra a través del método **registerForContextMenu (View)**. El método **onCreateContextMenu()** se llama cada vez que se activa un menú contextual , porque el menú contextual se desecha después de su uso. La plataforma Android también puede añadir opciones a la vista, por ejemplo, EditText ofrece opciones de contexto para seleccionar texto, etc.

Menú Contextual

Para una ActionBar

El modo de acción contextual suele ser activado mediante la selección de un elemento o de larga clic en él.

Este modo de acción contextual, activa un ActionBar temporal que se superpone a la ActionBar de la aplicación.



Para implementarlo, hay que llamar la **startActionMode()** en una vista o en su actividad. Este método obtiene un objeto **ActionMode.Callback** que es responsable de la vida útil de la barra de acciones contextual.