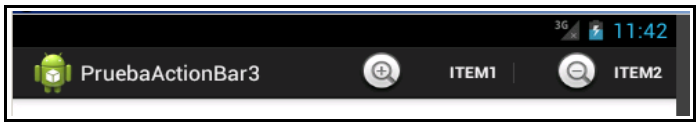
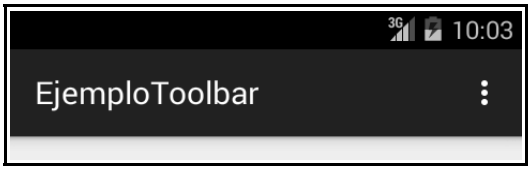
Toolbar

La Toolbar es un Widget en la parte superior de la pantalla que identifica a la aplicación y permite al usuario un acceso rápido y standard a los controles de navegación. Antes de la versión de Android 5.0, El widget que cumplía esta función era llamado ActionBar :



ActionBar utilizada antes de Android 5.0

A partir de Android 5.0 apareció la ToolBar, la cual nos da una mayor flexibilidad, permitiéndonos modificar su tamaño, color, agregarle widgets, etc.



Nueva Toobar de Android 5.0

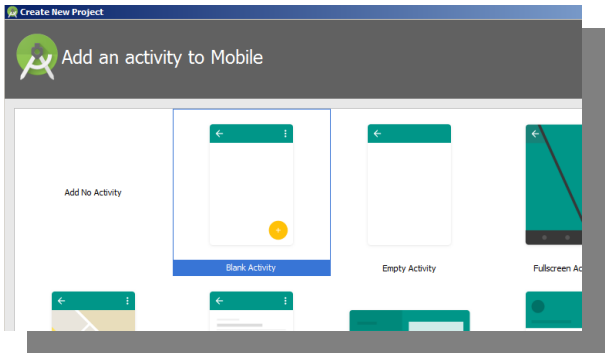
Compatibilidad

La ToolBar esta disponible a partir de la API 21 (Android 5.0 Lolipop). Para utilizarla en versiones anteriores, es necesario incluir la biblioteca de compatibilidad "appcompat" la cual se agrega automáticamente a nuestro proyecto cuando lo creamos a partir de Android 4.0. Para verificar esta condición, podemos chequear el archivo build.gradle de nuestra app, en donde deberá aparecer la dependencia:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.2.1'
}
```

Usando la Toolbar en nuestra Activity

Al crear una Activity del tipo "Blank Activity" se generara un layout que incluira a la toolbar como un widget mas:



Al layout que se genera tiene el siguiente contenido:

```
<android.support.design.widget.CoordinatorLayout
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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.lslutnfra.primeraapp.MainActivity">

    <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>

    <include layout="@layout/content_main" />

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

Contenido

El widget llamado Toolbar se encuentra contenido en otro widget que no tiene visibilidad para el usuario, llamado AppBarLayout, el cual nos permitira hacer animaciones en la Toolbar cuando el usuario hacer scroll, junto con el Layout CoordinatorLayout el cual engloba todo el contenido de la pantalla.

Lo que nosotros queramos poner como contenido de la pantalla (debajo de la toolbar) se encuentra en el archivo content_main.xml, el cual se incluye en este archivo mediante el tag de include.

Por último, debemos obtener la referencia de la Toolbar en nuestra Activity, y asignarla como ActionBar, este codigo ya aparece en el onCreate:

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

Se utiliza el método setSupportActionBar ya que en estos ejemplos se utiliza la biblioteca de compatibilidad, de otro modo, puede utilizarse el método setActionBar.
Al ejecutar la aplicación, veremos la ToolBar utilizada como ActionBar.

NOTA: Pueden agregarse otras Toolbars en el layout, pero solo una será la que funcione como ActionBar.

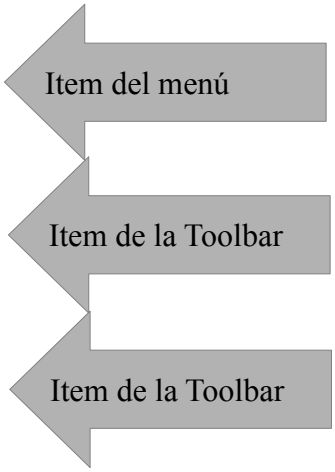
Agregando "Action Items"

Para agregar ítems a la derecha de la Toolbar (botones de compartir, buscar, etc) se deben definir en el mismo archivo xml en donde se definen los ítems del menú.

Ejemplo:

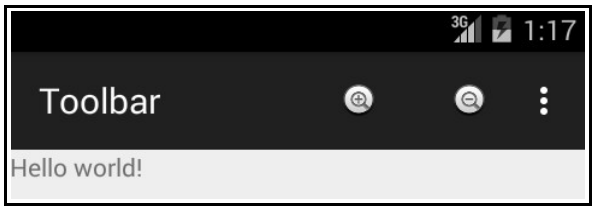
En el archivo menu_main.xml definiremos el ítems que queremos ver en la Toolbar, junto con sus íconos:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never"
        android:title="@string/action_settings"/>

    <item
        android:id="@+id/menu_item1"
        android:orderInCategory="100"
        app:showAsAction="always"
        android:icon="@android:drawable/btn_plus"
        android:title="item1"/>
    <item
        android:id="@+id/menu_item2"
        android:orderInCategory="100"
        app:showAsAction="always"
        android:icon="@android:drawable/btn_minus"
        android:title="item2"/>

</menu>
```



Si el atributo "showAsAction" toma el valor "never", el ítem no es parte de la Toolbar, en cambio si toma el valor "always" aparecerá siempre, si toma el valor "ifRoom" el ítem se mostrará si hay lugar para hacerlo. El ícono se define en el atributo "icon"

Ahora los ítems se verán en la Toolbar:



Capturando los eventos

Los eventos de los ítems de la Action Bar se capturan igual que los ítems del menú; sobreescribiendo el método "onOptionsItemSelected", y preguntando por los ids de los ítems:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId())
        {
                case R.id.menu_item1:{
                        Log.d("main","Click en action item 1");
                        break;
                }
                case R.id.menu_item2:{
                        Log.d("main","Click en action item 2");
                        break;
                }
        }
        return true;
}
```

Botón share

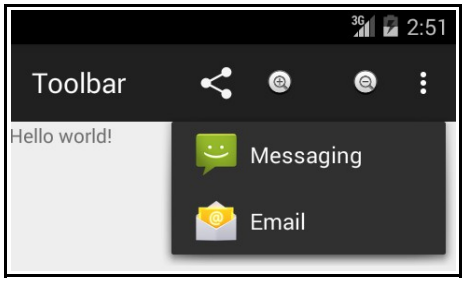
Es posible definir un ítem de menú que se encargue de la típica acción de "Compartir". Para ello debemos agregar el siguiente ítem en el archivo de menú:

```
<item
    android:id="@+id/menu_item_share"
    app:showAsAction="always"
    android:title="Share"
    app:actionProviderClass="android.support.v7.widget.ShareActionProvider" />
```

Luego modificamos el código de nuestra Activity, y definimos un atributo ShareActionProvider:

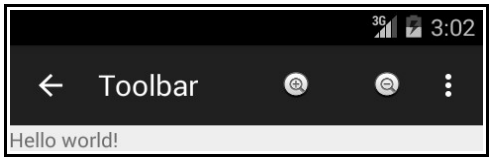
```
private ShareActionProvider mShareActionProvider;
//...

@Override
public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.menu_main, menu);

    MenuItem item = menu.findItem(R.id.menu_item_share);

    mShareActionProvider = (ShareActionProvider) MenuItemCompat.getActionProvider(item);

    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("text/plain");
    intent.putExtra(Intent.EXTRA_TEXT,"http://www.lslutnfra.com");
    mShareActionProvider.setShareIntent(intent);

    return true;
}
```

Obtenemos el objeto ShareActionProvider desde el MenuItem mediante el método estático getActionProvider de la clase MenuItemCompat (recordar que todos los ejemplos utilizan la biblioteca de compatibilidad) luego definimos un Intent y se lo asignamos al objeto ShareActionProvider.



Botón de back

Al configurar el botón como "Back", aparecerá a la izquierda del mismo una flecha indicando que al presionarse se volverá a la pantalla anterior.



Para que esta flecha aparezca, debemos ejecutar el método "setDisplayHomeAsUpEnabled" en el onCreate de la Activity:

```
ActionBar ab = getSupportActionBar();
ab.setDisplayHomeAsUpEnabled(true);
```

Para capturar el evento sobre este botón, se debe preguntar por el id definido en la clase R de Android llamado "home":

```
switch(item.getItemId()){
        case R.id.menu_item1:
        {
                Log.d("main","Click en action item 1");
                break;
        }
        case R.id.menu_item2:
        {
                Log.d("main","Click en action item 2");
                break;
        }
        case android.R.id.home:{
                finish(); // Simulo back
                break;
        }
}
```

Agregando un SearchView

Es posible agregar en la Toolbar un Widget que tenga un botón de búsqueda y un campo para escribir un texto. Para agregar este elemento, debemos definir un item de menú con las siguientes características:

```
<item android:id = "@+id/campo_buscar"
        android:orderInCategory = "100"
        app:showAsAction = "ifRoom"
        android:title = "Buscar"
        app:actionViewClass = "android.support.v7.widget.SearchView" />
```

Figure 5. An action bar with a collapsible `SearchView`.

Para escuchar los eventos de este widget (tendremos un evento cada vez que el usuario modifica un caracter de lo que va escribiendo, y uno cuando realiza la búsqueda) deberemos implementar la interface SearchView.OnQueryTextListener, en el siguiente ejemplo, implementamos la interface en una Activity:

```
public class MainActivity extends ActionBarActivity implements
                                                        SearchView.OnQueryTextListener {
       @ Override
       public boolean onQueryTextSubmit(String s) {
              Log.d( "activity" , "Hago una busqueda con:" + s);
              return false ;
       }

       @ Override
       public boolean onQueryTextChange(String s) {
              Log.d( "activity" , "cambio texto:" + s);
              return false ;
       }
```

Para setear el listener (es decir la Activity en nuestro caso) al objeto SearchView que se encuentra en la Toolbar, deberemos hacerlo en el método onCreateOptionsMenu (donde creamos los ítems de menú) obteniendo primero el objeto MenuItem que corresponde a este widget definido en el archivo XML de menú, y luego a partir del *MenuItem* , mediante el método estático *getActionView* de la clase *MenuItemCompat* , podemos obtener la referencia del SearchView.

Una vez obtenido el objeto SearchView, le seteamos el listener mediante el método *setOnQueryTextListener* .

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
       getMenuInflater().inflate(R.menu.menu_main, menu);
       MenuItem searchItem = menu.findItem(R.id.campo_buscar);

       SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem);
       // seteamos listener con “this”
       searchView.setOnQueryTextListener( this );
       return true ;
}
```