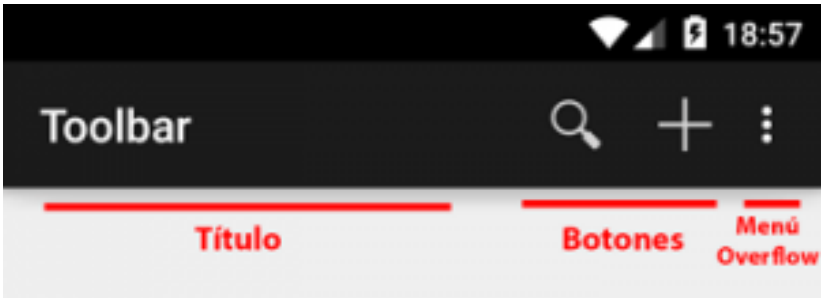
GUIA PRÁCTICA 6: Manejo de Menús

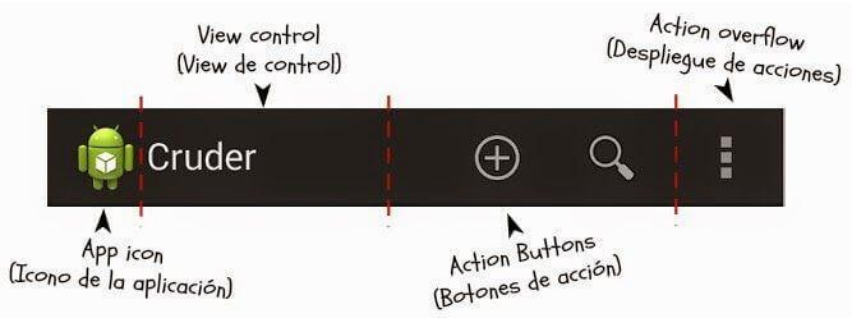
Aplicación: Menús Tradicionales, Sub Menus y ActionBar

La ActionBar, o App bar como se la ha rebautizado con la llegada de Material Design y Android 5.0, 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 de la aplicación o la actividad en la que nos encontramos, 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.



Antes de la llegada de Material Design, a la izquierda del título también podía (y solía) aparecer el icono de la aplicación, aunque esto último ya no se recomienda en las últimas guías de diseño de la plataforma. Lo que sí puede aparecer a la izquierda del título son los iconos indicativos de la existencia de menú lateral deslizante (navigation drawer) o el botón de navegación hacia atrás/arriba.

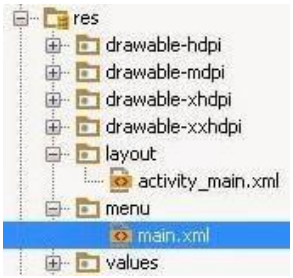


Para practicar la barra de acción vamos a crear una sencilla aplicación que nos permita entender cómo funciona este elemento. Se realizará una ActionBar que provea las operaciones CRUD para los elementos de una base de datos.

Mostrará 5 acciones: Añadir, Editar, Eliminar, Buscar y Ajustes.

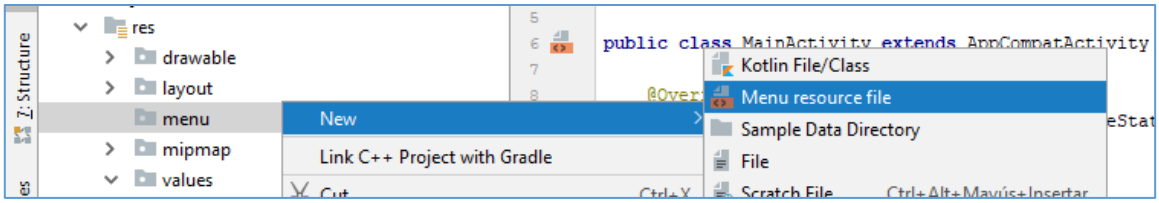
Lo primero que haremos será crear un nuevo proyecto con una actividad principal en blanco llamado: **MenuActionBar**

Para añadir el diseño a la ActionBar necesitamos usar un archivo de diseño que contenga los nodos necesarios para generar las opciones.

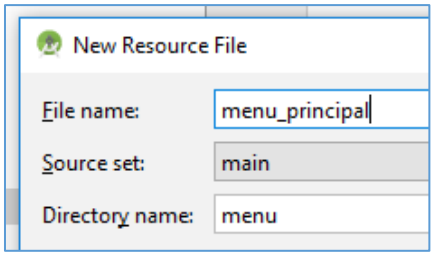


En algunas versiones de Android Studio se autogenera un archivo de recursos en la dirección **main/res/menu**

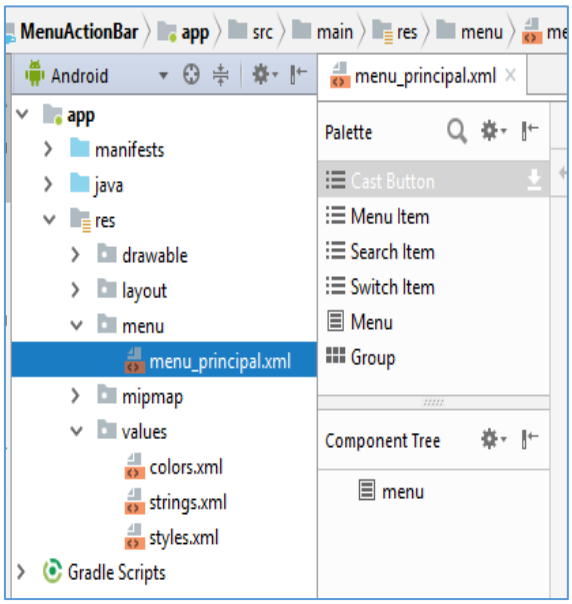
Si este archivo main.xml no se generó automáticamente con el proyecto, entonces se deberá crear manualmente, creando primero la carpeta (clic derecho en **res**, luego en **new** y por último en **Directory**) luego se crea el archivo de diseño del menu de la siguiente manera:

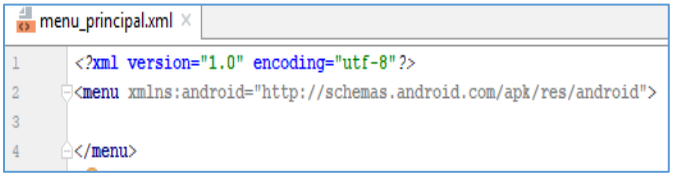


Colocarle el nombre deseado para el menu que tendrá las opciones del ActionBar



Listo tendremos el archivo de diseño para configurar las opciones de menu, ya sea del modo grafico o del modo texto:





Observe que muestra un elemento contenedor llamado ***<menu>*** que representa el ActionBar y cada elemento **<item>** que se colocara dentro se refiere a los botones de acción que representan las opciones del menu.

Se dijo que el menu mostraría 5 acciones de las cuales se debe elegir las que son más populares para los usuarios de la aplicación. Podrían ser la búsqueda y añadir un nuevo elemento a la base de datos por ser más relevantes, el resto de opciones no es tan frecuente usarlas.
Una vez elegidos los que deseamos ver persistentemente en la actividad, procederemos a escribir el archivo de diseño. El código debería quedar así:

menu_principal.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto" >
    <item android:id="@+id/add"
        android:icon="@android:drawable/ic_menu_add"
        android:title="@string/add"
        android:orderInCategory="1"
        app:showAsAction="ifRoom" />
    <item android:id="@+id/search"
        android:icon="@android:drawable/ic_menu_search"
        android:title="@string/search"
        android:orderInCategory="2"
        app:showAsAction="ifRoom" />
    <item android:id="@+id/edit"
        android:title="@string/edit"
        android:orderInCategory="3"
        app:showAsAction="never" />
    <item android:id="@+id/delete"
        android:title="@string/delete"
        android:orderInCategory="4"
        app:showAsAction="never" />
    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```

Recuerde que debe tener todos los textos definidos declarativamente en el archivo **strings.xml**

strings.xml

Edit translations for all locales in the translations editor.

```xml
<resources>
    <string name="app_name">MenuActionBar</string>
    <string name="add">Agregar</string>
    <string name="search">Buscar</string>
    <string name="edit">Editar</string>
    <string name="delete">Eliminar</string>
    <string name="action_settings">Ajustes</string>
    <string name="info_ini">Selecciona una opcion del Menu</string>
</resources>
```

También el diseño de la pantalla principal (**activity_main.xml**) se debe agregar dos atributos al control **TextView** que pone automáticamente Android Studio.

```
android:id="@+id/info"
android:text="@string/info_ini"
```

Esto servirá para identificarlo en la programación y mostrar el mensaje inicial del strings.xml y posteriormente se codificará para mostrar un mensaje indicando la opción de menu que ha sido seleccionada.

```
public class MainActivity extends AppCompatActivity {
    private TextView info;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Obteniendo la intancia del textview
        info =(TextView)findViewById(R.id.info);
    }
```

Programar los Eventos en la ActionBar

Una vez diseñado el cuerpo de App bar, procedemos a programar los eventos que puedan producirse sobre ella. Para ello disponemos de métodos callback especiales similares a los que poseen las actividades.

Inflar el menú de la ActionBar

En el archivo **MainActivity.java** se inserta el método **onCreateOptionsMenu()**. Este método es autoinvocado para inflar todo el código XML que se tiene en el archivo de diseño.

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

Lo primero que se hace es obtener una referencia del **MenuInflater** al cual está relacionada la actividad. Este objeto es el encargado de inflar el código de los menús para combinarlo con la actividad. En la segunda instrucción se invoca al método **inflate()** para inflar el archivo **R.menu.*menu_principal***. Finalmente se retorna en true para indicar que todo salió como se esperaba.

Programar los eventos de los Action Buttons

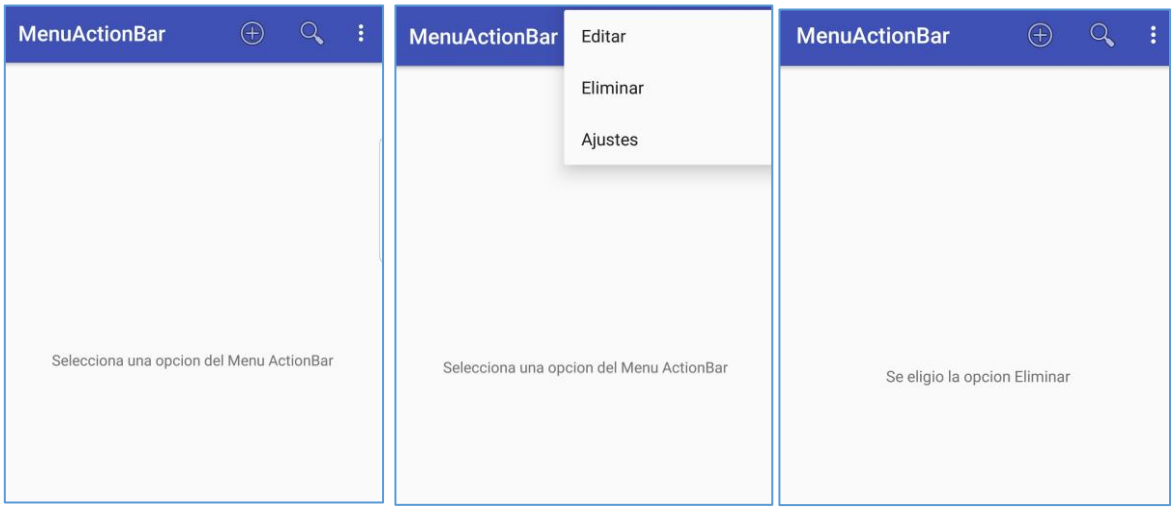
Ahora usaremos el método **onOptionsItemSelected()** para asignar las funciones a cada botón. Este método es autoinvocado cuando el usuario presiona un botón.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    switch (item.getItemId()) {
        case R.id.add:
            info.setText("Se eligio la opcion Agregar");
            return true;
        case R.id.search:
            info.setText("Se eligio la opcion Buscar");
            return true;
        case R.id.edit:
            info.setText("Se eligio la opcion Editar");
            return true;
        case R.id.delete:
            info.setText("Se eligio la opcion Eliminar");
            return true;
        case R.id.action_settings:
            info.setText("Se eligio la opcion Ajustes");
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }

}
```

Realizar las pruebas de la Aplicación.



Ejemplo 9. Menús Contextuales

Se crea el proyecto: **MenuContext**

Seleccionar la plantilla: **Empty Activity,** por defecto trae etiqueta de texto con un mensaje de ***Hello World***. Solo se le agrega el atributo id

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

Se añadirá en primer lugar un menú contextual que aparezca al pulsar sobre la etiqueta de texto. Para ello, lo primero que se hace es indicar en el método **onCreate()** de la actividad principal que la etiqueta tendrá asociado un menú contextual. Esto se consigue con una llamada a **registerForContextMenu()**:

```
public class MainActivity extends AppCompatActivity {

    private TextView lblMensaje;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Obtenemos las referencias a los controles
        lblMensaje = (TextView)findViewById(R.id.LblMensaje);

        //Asociamos los menús contextuales a los controles
        registerForContextMenu(lblMensaje);
```

Se procede a crear la carpeta de menú y dentro de ella el menú en XML llamado **menu_ctx_etiqueta.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/CtxLblOpc1"
        android:title="@string/opc_etiqueta1">
    </item>
    <item android:id="@+id/CtxLblOpc2"
        android:title="@string/opc_etiqueta2">
    </item>
</menu>
```

Recuerde configurar los textos en el archivo **strings.xml**

```
<resources>
    <string name="app_name">MenuContext</string>
    <string name="msg">Hello World!</string>
    <string name="opc_etiqueta1">Opción 1</string>
    <string name="opc_etiqueta2">Opción 2</string>
</resources>
```

A continuación, igual que se hacía con **onCreateOptionsMenu**() para los menús básicos, se va sobrescribir en la actividad el evento encargado de construir los menús contextuales asociados a los diferentes controles de la aplicación. En este caso el evento se llama **onCreateContextMenu**(), y a diferencia de **onCreateOptionsMenu**() éste se llama cada vez que se necesita mostrar un menú contextual, y no una sola vez al inicio de la aplicación. En este evento se actuará igual que para los menús básicos, inflando el menú XML.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);

    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_ctx_etiqueta, menu);
}
```

Por último, para implementar las acciones a realizar tras pulsar una opción determinada del menú contextual vamos a implementar el evento **onContextItemSelected**() de forma análoga a cómo hacíamos con **onOptionsItemSelected**() para los menús básicos:

```
@Override
public boolean onContextItemSelected(MenuItem item) {

    switch (item.getItemId()) {
        case R.id.CtxLblOpc1:
            lblMensaje.setText("Etiqueta: Opcion 1 pulsada!");
            return true;
        case R.id.CtxLblOpc2:
            lblMensaje.setText("Etiqueta: Opcion 2 pulsada!");
            return true;
        default:
            return super.onContextItemSelected(item);
    }

}
```

Con esto ya está lista la aplicación para probar el funcionamiento del menú contextual, solo tendría que mantener presionado por unos segundos la etiqueta y debería mostrar el nuevo mensaje.

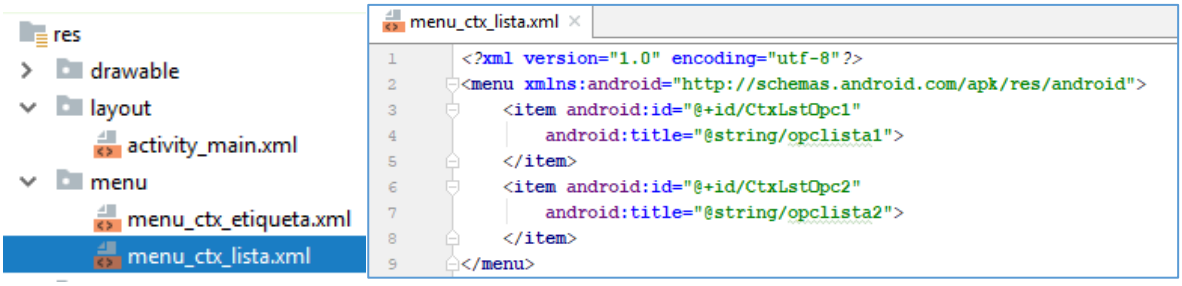
Menús Contextuales Parte II:

Se procede a modificar el proyecto anterior para agregar otro menú contextual a otro elemento de la aplicación (en este caso a un **ListView**) y se pueden tener tantos menús contextuales como sea necesario.

Se modifica el layout XML de la ventana principal para añadir el control ListView

```
<ListView
    android:id="@+id/LvLista"
    android:layout_width="match_parent"
    android:layout_height="200dp" />
```

Como en el caso anterior, se define en XML otro menú para asociarlo a los elementos de la lista, lo llamaremos **menu_ctx_lista.xml**



y se modifica el método **onCreate**() para obtener la referencia al control, insertar varios datos de ejemplo, y asociarle un menú contextual:

```
public class MainActivity extends AppCompatActivity {
    private TextView lblMensaje;
    private ListView lvLista;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Obtenemos las referencias a los controles
        lblMensaje = (TextView)findViewById(R.id.LblMensaje);
        lvLista = (ListView)findViewById(R.id.LvLista);
        //Se llena la lista con datos de ejemplo
        String[] datos = new String[]{"Elem1","Elem2","Elem3","Elem4","Elem5"};
        ArrayAdapter<String> adaptador =
                new ArrayAdapter<String>( context: this,
                        android.R.layout.simple_list_item_1, datos);
        lvLista.setAdapter(adaptador);
        //Asociamos los menús contextuales a los controles
        registerForContextMenu(lblMensaje);
        registerForContextMenu(lvLista);
    }
```

Recuerde que las líneas marcadas en amarillo ya habían sido colocadas en la primera parte del ejercicio.

Como siguiente paso, y dado que se va a tener varios menús contextuales en la misma actividad, se necesita modificar el evento **onCreateContextMenu**() para que se construya un menú distinto dependiendo del control asociado. Esto se hace obteniendo el ID del control al que se va a asociar el menú contextual, que se recibe en forma de parámetro (View v) en el evento **onCreateContextMenu**(). Se utiliza para ello una llamada al método **getId**() de dicho parámetro:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);

    MenuInflater inflater = getMenuInflater();

    if(v.getId() == R.id.LblMensaje)
        inflater.inflate(R.menu.menu_ctx_etiqueta, menu);
    else if(v.getId() == R.id.LvLista)
    {
        AdapterView.AdapterContextMenuInfo info =
                (AdapterView.AdapterContextMenuInfo)menuInfo;
        menu.setHeaderTitle(lvLista.getAdapter().getItem(info.position).toString());
        inflater.inflate(R.menu.menu_ctx_lista, menu);
    }
}
```

Nuevamente recuerde que esta modificando y que algunas líneas ya estaban escritas.

La respuesta a este nuevo menú se realizará desde el mismo evento que el anterior, todo dentro de **onContextItemSelected**(). Por tanto, incluyendo las opciones del nuevo menú contextual para la lista el código nos quedaría de la siguiente forma:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info =
            (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.CtxLblOpc1:
            lblMensaje.setText("Etiqueta: Opcion 1 pulsada!");
            return true;
        case R.id.CtxLblOpc2:
            lblMensaje.setText("Etiqueta: Opcion 2 pulsada!");
            return true;
        case R.id.CtxLstOpc1:
            lblMensaje.setText("Lista[Elem" + (info.position+1) + "]: Opcion 1 pulsada!");
            return true;
        case R.id.CtxLstOpc2:
            lblMensaje.setText("Lista[Elem" + (info.position+1) + "]: Opcion 2 pulsada!");
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

Ejercicio

Desarrollar una Aplicación Android con las siguientes características:

- Debe poseer tres menús contextuales en los siguientes Views (o controles)

  a) Un TextView

  Opciones:

  1. ***Cambiar Color***. Cambia el color del EditText.
  2. ***Cambiar Texto***. Cambia el texto del EditText.

  b) Un EditText

  Opciones:

  1. ***Enviar***. Envía el texto del EditText al TextView.
  2. ***Borrar***. Borra el texto del EditText y del TextView.

  c) Un ListView

  Opciones:

  1. ***Enviar a TextView***. Envía el texto del ítem seleccionado al TextView.
  2. ***Enviar a EditText***. Envía el texto del ítem seleccionado al EditText.

- Los elementos (al menos tres) del ListView pueden ser:
  a) Nombres de Equipos de Futbol.
  b) Nombres de Países
  c) Nombres de Frutas