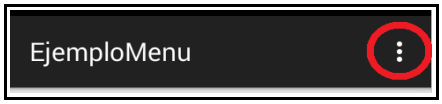
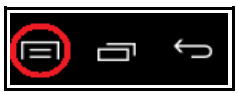
Menúes

El sistema operativo provee un botón llamado "Menú", al presionarse, la Activity que esté activa en ese momento, tendrá la posibilidad de capturar ese evento y generar un menú en pantalla.
Cuando el dispositivo no posee botón menú, aparecerá un botón en la ToolBar para desplegar las opciones, sin embargo, el mecanismo explicado a continuación es el mismo.



Botón de menú en ToolBar



Botón de menú Físico

Para activar el menú, la Activity posee dos métodos a los que debe hacerse Override:

- **onCreateOptionsMenu():** Se llamará al presionar el botón menú, aquí se debe construir el menú en pantalla.
- **onOptionsItemSelected():** Se llamará al presionar una opción en el menú.

Creando el menú

Existen dos formas de crear el menú, por código, o mediante un archivo xml de resource, explicaremos esta última opción por ser la recomendada.

Dentro de la carpeta menú que se encuentra dentro de los recursos (carpeta res) se encontrarán los archivos xml que definen los ítems de menú.

Luego, se debe utilizar la clase MenuInflater para generar el objeto "Menu" a partir del archivo xml.

Ejemplo de archivo **menu_main.xml**:

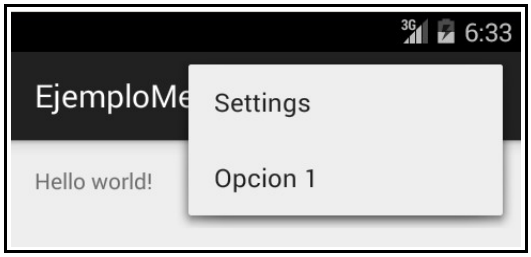
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">

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

    <item android:id="@+id/action_opcion1"
        android:title="Opcion 1"
        android:orderInCategory="101"
        app:showAsAction="never" />

</menu>
```

Para crear el menú a partir de este archivo, dentro del método onCreateOptionsMenu():

```
@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;
}
```

El método "getMenuInflater()" es propio de la clase Activity, y nos devuelve un objeto MenuInflater, el cual a partir del método "inflate()" podrá cargar al objeto "menu" con los ítems definidos en el archivo xml.

Al ejecutar la aplicación, veremos que ahora el icono de menú, que se encuentra en la Toolbar, tendrá nuestras dos opciones:



Recibiendo eventos del menú

Para recibir los eventos del menú, solo basta con hacer Override del método "onOptionsItemSelected()" el cual tendrá como parámetro un objeto del tipo MenuItem, este objeto, corresponderá con el ítem seleccionado en el menú.

El objeto MenuItem posee el método "getItemId()" el cual devolverá el Id del ítem, (correspondiente al id pasado como parámetro en el método "add()" o el definido en el xml al crear el menú.)

Ejemplo:

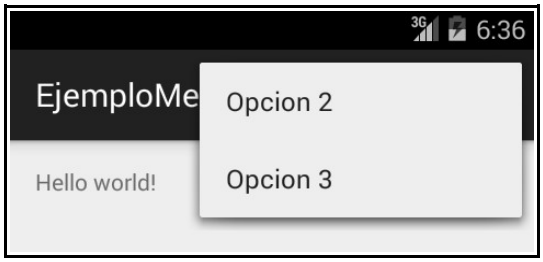
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_settings) {
         Log.d("menu","Click en settings");
        return true;
    }
    if (id == R.id.action_opcion1) {
        Log.d("menu","Click en opcion 1");
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

En el caso de que el ítem seleccionado no concuerde con ninguno de los creados, probablemente sea porque se eligió una de las opciones de sistema, por lo que en ese caso, llamamos a el método "onOptionsItemSelected()" de la clase padre, mediante "super".

Anidamiento de ítems

Para tener ítems anidados, se debe colocar un tag de "menu" dentro de un ítem, y colocar los ítems dentro. Ahora el presionar "Opcion 1" aparecerán los otros dos ítems:

```
<item android:id="@+id/action_opcion1"
        android:title="Opcion 1"
        android:orderInCategory="101"
        app:showAsAction="never" >
        <menu>
            <item android:id="@+id/action_opcion2"
                android:title="Opcion 2"
                android:orderInCategory="101"
                app:showAsAction="never" />
            <item android:id="@+id/action_opcion3"
                android:title="Opcion 3"
                android:orderInCategory="101"
                app:showAsAction="never" />
        </menu>
    </item>
```



Diálogos

Los diálogos son ventanas que le aparecen al usuario sobre la pantalla activa, (igual que los menúes contextuales) pero que permiten mostrar cualquier tipo de contenido. La analogía con un sistema operativo de escritorio, sería las ventanas de alertas o información, donde el usuario puede elegir "Aceptar" ,"Cancelar" , "Si", "No", etc.

Un aspecto importante a tener en cuenta con los diálogos en Android, es que son asincrónicos, es decir, cuando se dispara el diálogo, el código no queda bloqueado, como pasa en la mayoría de los lenguajes con programas para escritorio, sino que el programa que dispara la ventana sigue corriendo, y éste debe encargarse de capturar el evento cuando el diálogo es cerrado.

Los pasos para construir un diálogo son los siguientes:

- Crear un objeto Builder
- Setear cantidad de botones, y el contenido de la ventana.
- Setear los listeners para los botones.
- Decirle al Builder que cree el diálogo
- Crear una clase que herede de FragmentDialog
- Sobreescribir el método onCreateDialog y devolver el objeto AlertDialog creado con el builder.

Ejemplo:

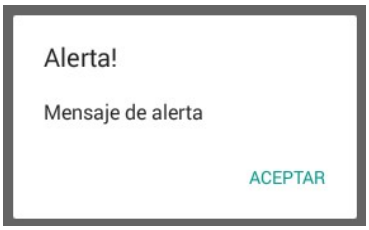
Primero, definimos una clase que funcionará como listener de los botones que aparezcan en el diálogo. Para ello, creamos una clase llamada "ListenerAlert" e implementamos la interfaz "onClickListener" correspondiente al Package "android.content.DialogInterface"

```
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.util.Log;

public class ListenerAlert implements OnClickListener{

	@Override
	public void onClick(DialogInterface dialog, int which) {
		Log.d("dialog","Click!");
	}
}
```

Luego, creamos una clase "MiDialogo" y sobreescribimos el método "onCreateDialog"

```
public class MiDialogo extends AppCompatDialogFragment {

	@Override
	public Dialog onCreateDialog(Bundle savedInstanceState) {
		AlertDialog.Builder builder = new
                                    AlertDialog.Builder(getActivity());
		builder.setTitle("Alerta!");
		builder.setMessage("Mensaje de alerta");
		ListenerAlert l = new ListenerAlert();
		builder.setPositiveButton("Aceptar", l);

		// Creamos el dialogo
		AlertDialog ad = builder.create();
		return ad;
	}
```

Dentro del método, creamos un objeto del tipo Builder (del package "android.support.v7.app.AlertDialog") en el cual seteamos las características del diálogo, como el título, el mensaje, y los botones que contendrá. También podemos observar que creamos un objeto de nuestra clase ListenerAlert, y se lo pasamos al botón, para poder recibir el evento de click del mismo.



Además de "setPositiveButton()", existen dos botones más que pueden agregarse:

- setNegativeButton()
- setNeutralButton()

Para identificar qué botón se presionó, en el listener, existe el parámetro "which":

```
public void onClick(DialogInterface dialog, int which)
```

El cual tiene un valor distinto según cual de los tres botones se presionó, los valores están definidos en la clase AlertDialog:

- AlertDialog.BUTTON_NEGATIVE
- AlertDialog.BUTTON_POSITIVE
- AlertDialog.BUTTON_NEUTRAL

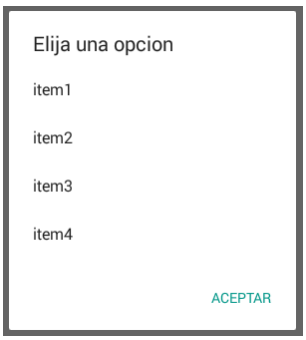
Para lanzar el dialogo desde la Activity, creamos nuestro objeto "MiDialogo" y le ejecutamos el método "show" el cual recibe como parámetros una instancia de "FragmentManager" y un nombre.

```
MiDialogo dialog = new MiDialogo();
dialog.show(getSupportFragmentManager(), "dialogo");
```

Lista de items en dialogo

Es posible mostrar una lista de ítems en el diálogo ejecutando el método "setItems" al objeto builder, en vez de "setMessage".El segundo parámetro de setItems es un listener.

```
builder.setTitle("Elija una opcion");
String[] items = new String[]{"item1","item2","item3","item4"};
builder.setItems(items, this);
```



Diálogos con contenido personalizado

Es posible definir un archivo de layout, al igual que se hace para el contenido de una Activity, y setear dicho archivo, a la ventana de diálogo, de modo que es posible definir cualquier cantidad de objetos en la ventana, con cualquier disposición.

Ejemplo:
Primero, creamos un archivo de layout con el nombre *layout_dialogo*, en este archivo, tendremos una imagen y un campo de texto para ingresar un valor:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="horizontal">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/pesos_icon"
        android:id="@+id/imageView" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="5.0"
        android:id="@+id/editText" />
</LinearLayout>
```

Luego, en la Activity, creamos un objeto del tipo LayoutInflater, el cual nos permitirá crear un objeto View a partir del archivo xml. Una vez obtenido este objeto View, se lo seteamos al objeto builder mediante el método "setView()":

```
// cargamos la vista desde el xml
LayoutInflater li =LayoutInflater.from(getActivity());
View viewAlert = li.inflate(R.layout.layout_dialogo,null);

// cargamos view en el builder
AlertDialog.Builder builder = new
        AlertDialog.Builder(getActivity());
builder.setTitle("Importe");
builder.setView(viewAlert);

// asignamos listeners
ListenerAlert l = new ListenerAlert();
builder.setPositiveButton("Aceptar", l);
builder.setNegativeButton("Cancelar", l);

// Creamos el dialogo
AlertDialog ad = builder.create();
return ad;
```

El resultado del diálogo es el siguiente: