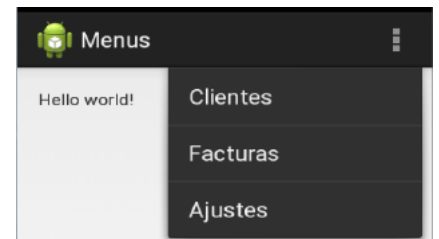


Menús

Crear elementos de menús en Android es muy sencillo. Tan sólo tienes que definir un archivo de recursos XML dentro de la carpeta “Menu” del proyecto. Con sólo dos etiquetas (*menu* e *item*) puedes hacer cualquier organización jerárquica de menús. La etiqueta `<menu>` describe un grupo de elementos (ítem) que será cada entrada del menú. Pero ojo, tenemos que advertirte: Los menús y submenús han caído en desuso desde Android 3.0+, principalmente por la existencia de la ActionBar que explicamos en el siguiente apartado.




Hay varios tipos de menús:

- *Menús de Opciones o Ajustes (settings)*: Son los menús que salen en la barra de acción de tu App y se controlan a través de un método callback que genera automática Android Studio y que se llama `onOptionsItemSelected`. Este método es invocado cuando el usuario selecciona uno de los elementos de este menú. Android studio también genera el código para crear el menú `onCreateOptionsMenu`, que “infla” un determinado menú diseñado en un fichero de recursos xml.
- *Menús contextuales*: Son los menús flotantes que aparecen cuando un usuario hace un click de manera prolongada en un elemento de la interfaz.
- *Menús de Pop-up*: Visualizan elementos de menú en una lista vertical. Estos menús aparecen anclados al elemento de la IU que provocó su aparición.
- *Submenús*: Aparecen al seleccionar una opción de menú, reemplazando el menú principal por las opciones del submenú.

Si utilizas el recurso `my.xml` estás definiendo el menú por defecto de la aplicación. La función `onCreateOptionsMenu` “infla” este recurso por defecto. Por ejemplo, imagina que quieres hacer una App típica de gestión comercial. Puedes tener como menú de opciones de tu App el siguiente menú:



Los menús en la barra de acción, a partir de android 3.0, aparecen también en la parte superior derecha al presionar el botón de *Overflow* (icono  en la barra de acción), en versiones anteriores aparecen en la esquina inferior izquierda al pulsar la tecla de menú del dispositivo.

El código en XML para este menú es:

```

<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MyActivity" >
    <item android:id="@+id/ajustes"
        android:title="Ajustes"
        android:icon="@android:drawable/ic_menu_more"
        android:orderInCategory="100"
        android:showAsAction="ifRoom"
    >
    <menu> <!-- submenú -->
        <item android:id="@+id/confS"
            android:title="Configuración de Starks"
            android:orderInCategory="1"/>
        <item android:id="@+id/confL"
            android:title="Configuración de Lannisters"
            android:orderInCategory="2"/>
        <item android:id="@+id/confB"
            android:title="Configuración de Baratheons"
            android:orderInCategory="3"/>
    </menu>
</item>
<item android:id="@+id/Arbol"
    android:title="Árbol Genialógico"
    android:orderInCategory="1">
    <menu> <!-- submenú -->
        <item android:id="@+id/nuevaEntrada"
            android:title="Nueva Entrada"
            android:orderInCategory="1" />
        <item android:id="@+id/verEstructura"
            android:title="Ver Estructura"
            android:orderInCategory="2" />
    </menu>
</item>
<item android:id="@+id/Alarmas"
    android:title="Alarmas"
    android:orderInCategory="2">
</item>
<item android:id="@+id/verMiembros"
    android:title="Ver Miembros"
    android:orderInCategory="3">
    <menu>
        <group android:checkableBehavior="single">
            <item android:id="@+id/asesinados"
                android:title="Asesinados"
                android:orderInCategory="1" />
            <item android:id="@+id/vivos"
                android:title="Vivos"
                android:orderInCategory="2" />
            <item android:id="@+id/heridos"
                android:title="Heridos"
                android:orderInCategory="3" />
            <item android:id="@+id/aparecer"
                android:title="Por aparecer"
                android:orderInCategory="4" />
        </group>
    </menu>
</item>
</menu>

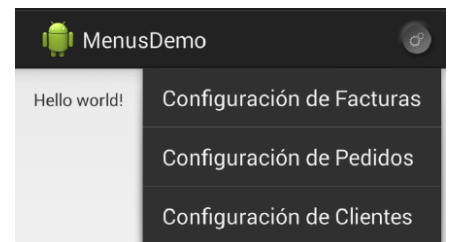
```

Observa cómo cuando dentro de una pareja de etiquetas <menu> </menu> incluyes un <item> se crea una opción y si anidas de nuevo un pareja <menu> </menu> se crea un submenú.

Las opciones de cada <item> son las siguientes:

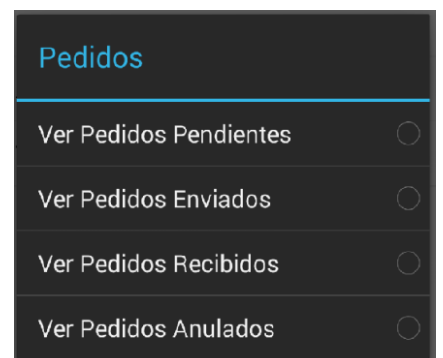
- android:id Identificador para el elemento de menú. Gracias al *id* puedes luego diferenciar en el código en qué elemento se hizo “Click”
- android:title Texto que aparece en el elemento de menú.
- android:orderInCategory Orden en el que aparece el elemento de menú dentro de su agrupación.
- android:icon Icono para el elemento de menú. Los iconos no aparecerán si tu aplicación se ejecuta en Android 3.0 o mayor. Tan sólo se mostrarán si forman parte de tu ActionBar.
- android:showAsAction Indica si el elemento aparecerá como un elemento en la barra de Acción. El valor *never*, hará el elemento de menú no se muestre en la barra de acción. Si se especifica *always*, siempre aparecerá y si se especifica *ifRoom* solo aparecerá si hay espacio suficiente en la barra de acción. Por ejemplo, si quisiéramos cierta parte de nuestro menú (por ejemplo, el submenú de configuración) apareciera vinculado a la barra de acción, por ejemplo, mediante un icono, tendríamos que usar las propiedades:

```
<item android:id="@+id/ajustes"
      android:title="Ajustes"
      android:icon="@android:drawable/ic_menu_more"
      android:orderInCategory="100"
      android:showAsAction="ifRoom"
  >
  <menu> <!-- submenú -->
    <item android:id="@+id/confFacturas"
          android:title="Configuración de Facturas"
          android:orderInCategory="1" />
    <item android:id="@+id/confPedidos"
          android:title="Configuración de Pedidos"
          android:orderInCategory="2" />
    <item android:id="@+id/confClientes"
          android:title="Configuración de CLientes"
          android:orderInCategory="3" />
  </menu>
</item>
```



También puedes diseñar menús con opciones seleccionables utilizando la opción <group android:checkableBehavior="modo"> siendo modo *single* (sólo un elemento es seleccionable, tipo RadioButton), *all* (todos los elementos son seleccionables, tipo CheckBox) y *none* (ningún elemento seleccionable). Por ejemplo:

```
<menu>
  <group android:checkableBehavior="single">
    <item android:id="@+id/PedPend"
          android:title="Ver Pedidos Pendientes"
          android:orderInCategory="1" />
    <item android:id="@+id/PedEnviados"
          android:title="Ver Pedidos Enviados"
          android:orderInCategory="2" />
    <item android:id="@+id/PedRecibidos"
          android:title="Ver Pedidos Recibidos"
          android:orderInCategory="3" />
    <item android:id="@+id/PedAnulados"
          android:title="Ver Pedidos Anulados"
          android:orderInCategory="4" />
  </group>
</menu>
```



1.1.- Acción a los elementos del menú

Para responder al click en un elemento de menú, hay que modificar el método `onOptionsItemSelected`:

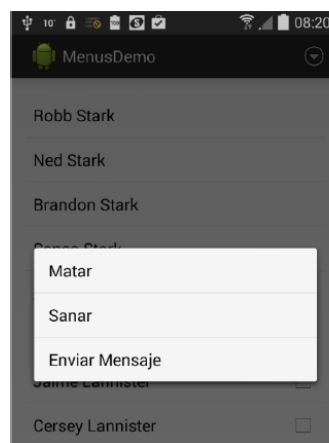
```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    switch(id) {
        case R.id.BuscarCliente:
            Toast.makeText(getApplicationContext(), "Se ha pulsado Buscar
            Cliente", Toast.LENGTH_LONG).show();
            return true;
        case R.id.Clientes:
            Toast.makeText(getApplicationContext(), "Se ha pulsado Clientes",
            Toast.LENGTH_LONG).show();
            return true;
        case R.id.Facturas:
            Toast.makeText(getApplicationContext(), "Se ha pulsado Facturas",
            Toast.LENGTH_LONG).show();
            return true;
        case R.id.ajustes:
        case R.id.confClientes:
        case R.id.confFacturas:
        case R.id.confPedidos:
        case R.id.NuevaFactura:
        case R.id.NuevoCliente:
            Toast.makeText(getApplicationContext(), "Se ha pulsado otro elemento de
            menu (" + item.getTitle() + ")", Toast.LENGTH_LONG).show();
            return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Este método recibe como parámetro el elemento seleccionado *MenuItem Item*, y a partir de ahí podemos discriminar qué elemento se seleccionó.

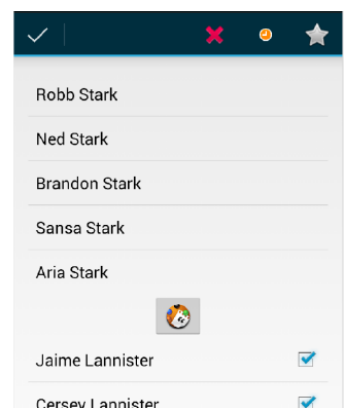
Por ejemplo, con el método `getItemId()` del objeto ítem nos quedamos con el identificador y después podemos comparar con un evaluador múltiple switch qué elemento de menú se pulso. El método exige que se retorne el valor *true* si se procesó correctamente el click del menú.

1.2.- Menús contextuales

Un menú contextual ofrece opciones que afectan a un elemento de la interfaz, por ejemplo un elemento de una lista o una imagen. Hay dos formas de hacer menús contextuales:



Menú contextual

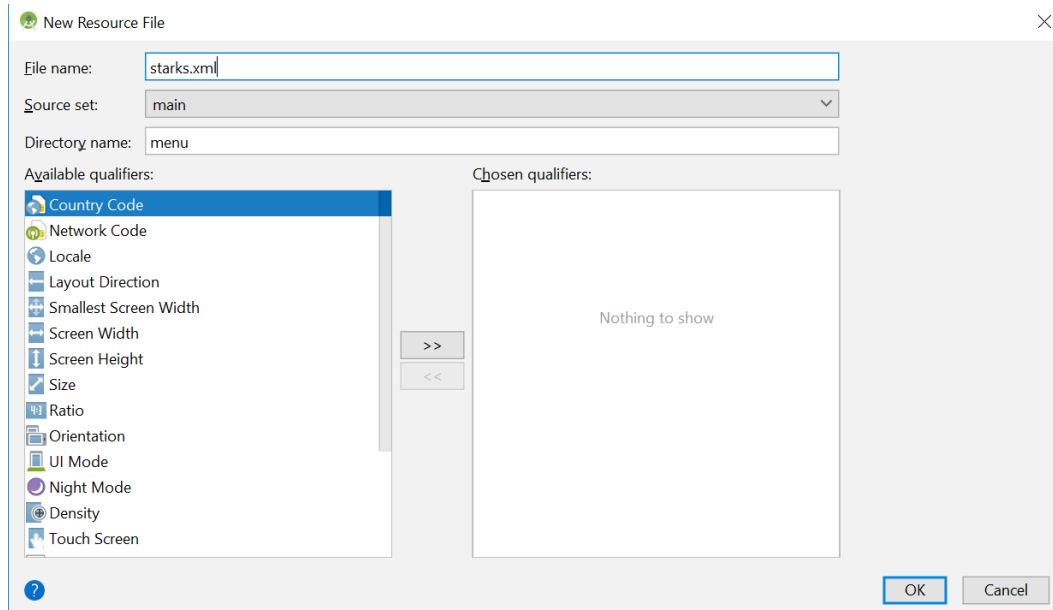


Menú en Contextual Action Bar (CAB)

El primero se utiliza cuando el usuario realiza una pulsación larga (long click) en un elemento de la interfaz y el segundo visualiza una barra de acción contextual, en inglés Contextual ActionBar (CAB). Esta CAB se puede programar para que aparezca también cuando se produce un click largo en un elemento de la IU o, por ejemplo, cuando se selecciona uno o varios elementos de una lista.

A continuación mostramos en un caso práctico los dos tipos de menús. Primero el menú flotante:

Inflar consistía en, a partir de la definición de un fichero de recursos xml crear una vista. Pues eso mismo es lo que hay que hacer, definir un recurso de menú, para luego “inflarlo”.



Después hay que registrar el componente al que se le va a asociar el menú contextual, mediante el método `registerForContextMenu()`. Después hay que crear (override) el método `onCreateContextMenu()` de Actividad e inflar el menú que hayas creado en xml.

Después, para responder a la selección del elemento del menú, se programa el método `onContextItemSelected()`.

La práctica realizada es una App para jugar con los personajes de una famosa serie de televisión. En esta aplicación tenemos dos listas de personajes. A la primera lista (con los personajes de la familia Stark) le vamos a vincular un menú contextual flotante y a la segunda lista (con los personajes de la familia Lannister) le vamos a asociar un menú contextual de Acción, tal y como puedes ver en la primera figura de esta sección.

1.2.1.- Menú contextual

Este es el fichero de menú xml que vamos a crear:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:icon="@android:drawable/ic_delete"
    android:title="Matar"
    android:id="@+id/matar"
    ></item>
<item android:icon="@android:drawable/ic_menu_edit"
    android:title="Sanar"
    android:id="@+id/sanar">
    ></item>
<item android:icon="@android:drawable/sym_call_incoming"
    android:title="Enviar Mensaje"
```

```

        android:id="@+id/enviarmensjae">
    </item>
</menu>

```

Primero hay que registrar la lista como la asociada al menú contextual, esto lo puedes hacer con la función `registerForContextMenu(lista)` en el método `onCreate` de la actividad:

```

//Creamos lista de starks para el menú contextual
starks=(ListView)findViewById(R.id.listaStarks);
registerForContextMenu(starks);

```

Después, tienes que crear el menú `onCreateContextMenu()` para inflar el menú desde el archivo de recursos xml:

```

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenu.ContextMenuInfo menuInfo) {
    MenuInflater m=getMenuInflater();
    m.inflate(R.menu.starks,menu);
    super.onCreateContextMenu(menu, v, menuInfo);
}

```

Finalmente, tienes que crear el método `onContextItemSelected()` para responder a los eventos del menú contextual. Este método recibirá como parámetro el elemento de menú pulsado *item*. A través de este parámetro puedes obtener, con el método `getMenuInfo()` de *item* puedes obtener un objeto `AdapterContextMenuInfo` con el que poder conocer sobre qué elemento de la `listView` fue pulsado el menú contextual.

```

@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
    item.getContextMenuInfo();

    switch (item.getItemId()) {
        case R.id.matar:
            Toast.makeText(getApplicationContext(), "Hemos matado a " +
                starks.getItemAtPosition(info.position),
                Toast.LENGTH_LONG).show();
            return true;
        case R.id.sanar:
            Toast.makeText(getApplicationContext(), "Hemos sanado a " +
                starks.getItemAtPosition(info.position),
                Toast.LENGTH_LONG).show();
            return true;
        case R.id.enviarmensjae:
            Toast.makeText(getApplicationContext(), "Le hemos enviado un mensaje
                a " + starks.getItemAtPosition(info.position),
                Toast.LENGTH_LONG).show();
            return true;
        default:
            Toast.makeText(getApplicationContext(), "Le hemos hecho otra cosa
                a " + starks.getItemAtPosition(info.position),
                Toast.LENGTH_LONG).show();
            return true;
    }
}

```

1.2.2.- Menú Action Bar Contextual (anteriores a versión 5.0)

Crear el menú contextual en la barra de acción, exige un poco más de código, pero el fundamento es el mismo. Primero, crear el menú. Ten en cuenta que en la barra de acción si se

muestran los iconos, al contrario que en un menú normal, por tanto, la adición de la propiedad `android:icon` toma esencial importancia:

Primero creamos el menú en el fichero `lannisters.xml`

```
<?xml version="1.0" encoding="utf-8"?>

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

    <item android:icon="@android:drawable/ic_delete"
        android:title="Aniquilar"
        android:id="@+id/aniquilar"
    ></item>

    <item android:icon="@android:drawable/presence_away"
        android:title="Encerrar"
        android:id="@+id/encerrar"
    ></item>

    <item android:icon="@android:drawable/btn_star"
        android:title="Salvar"
        android:id="@+id/salvar"
    ></item>

</menu>
```

Después, dentro del método `onCreate` de la actividad, creamos la segunda lista de elementos a través de un `ArrayAdapter<String>` de tal manera que sean todos seleccionables. Observa la última instrucción que registra el listener de Click en los elementos de la lista (hay que implementar en la actividad la interfaz `ListView.OnItemClickListener`, para que cuando se seleccione en una de las opciones de la lista, aparezca la Action Bar Contextual).

```
//creamos lista de lannisters (seleccionable múltiple) para el
//Action Mode Context Menu
ListView listaLannisters=(ListView)findViewById(R.id.listaLannisters);

ArrayAdapter<String> adaptador=new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_multiple_choice,
    getResources().getStringArray(R.array.lannisters));

listaLannisters.setAdapter(adaptador);
listaLannisters.setOnItemClickListener(this);
```

El gran truco para mostrar la barra de acción contextual es la creación de dos objetos, uno de tipo `ActionMode`, que representa el modo contextual en la barra de acción y otro de tipo `ActionMode.Callback`, que es la interfaz que se debe programar para responder a sus eventos y que tiene su propio ciclo de vida también:

```
//Menú ActionMode
private ActionMode.Callback mActionModeCallback = new ActionMode.Callback() {

    // Called when the action mode is created; startActionMode() was called
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        // Inflate a menu resource providing context menu items
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.lannisters, menu);
        return true;
    }

    // Called each time the action mode is shown. Always called after
    onCreateActionMode, but
```

```

// may be called multiple times if the mode is invalidated.
@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
    return false; // Return false if nothing is done
}

// Se llama a este método cuando se ha pulsado en la lista de los lannisters
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {

    switch (item.getItemId()) {
        case R.id.aniquilar:
            //hay que crear un Aniquilar() para recorrer todos los elementos
            //seleccionado (checked) en la listView
            Toast.makeText(getApplicationContext(), "Hemos aniquilado a algún
Lannister", Toast.LENGTH_LONG).show();
            return true;
        case R.id.encerrar:
            Toast.makeText(getApplicationContext(), "Hemos encerrado a algún
Lannister", Toast.LENGTH_LONG).show();
            return true;
        case R.id.salvar:
            Toast.makeText(getApplicationContext(), "Hemos salvado a algún
Lannister", Toast.LENGTH_LONG).show();
            return true;

        default:
            return false;
    }
}

// Called when the user exits the action mode
@Override
public void onDestroyActionMode(ActionMode mode) {
    mActionMode = null;
}
};

```

En este código tienes que fijarte en la creación de la interfaz: El método *onCreateActionMode()* infla el menú *lannisters.xml* adjuntándose a la barra de acción. El método *onActionItemClicked()* discrimina el elemento del menú que se seleccionó y se actúa en consecuencia.