MENUS

Documentación en http://developer.android.com/intl/es/guide/topics/ui/menus.html

1. INTRODUCCION

- Los menús son útiles para mostrar opciones adicionales que no están directamente visibles en la UI principal de una aplicación.

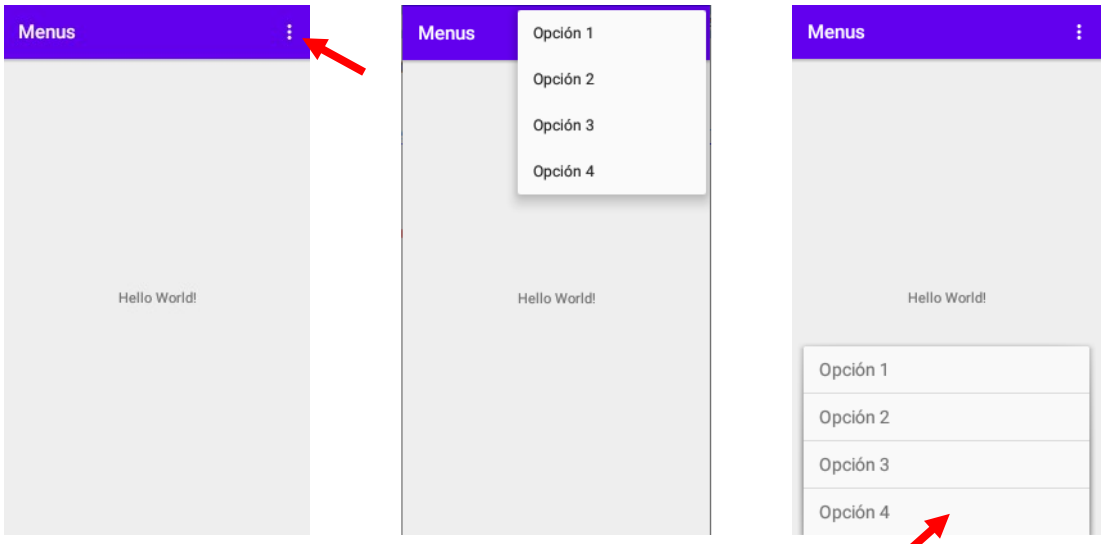
- Android dispone de varios tipos de menús, con variantes que van evolucionando según las nuevas versiones del sistema. Los más básicos son:

  - **Menús de opciones**.
  - **Submenús**.
  - **Menús contextuales**.

- **Menú de opciones:** es el más simple y habitual. Su posición y forma de visualizarse en la pantalla del dispositivo depende de la versión del sistema que se esté utilizando:

  - Desde **API 11**, los items de un menú de opciones están disponibles en la barra de la app. (o presionando el botón *Menú* del dispositivo, si está disponible). Por defecto, el sistema coloca todos los items en el ***botón de overflow***, situado a la derecha de la barra de acción.

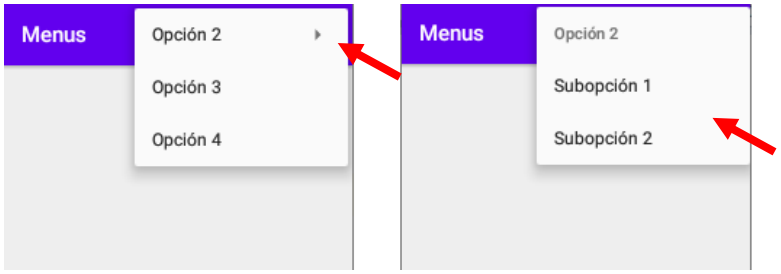
Más documentación: http://developer.android.com/intl/es/training/appbar/index.html



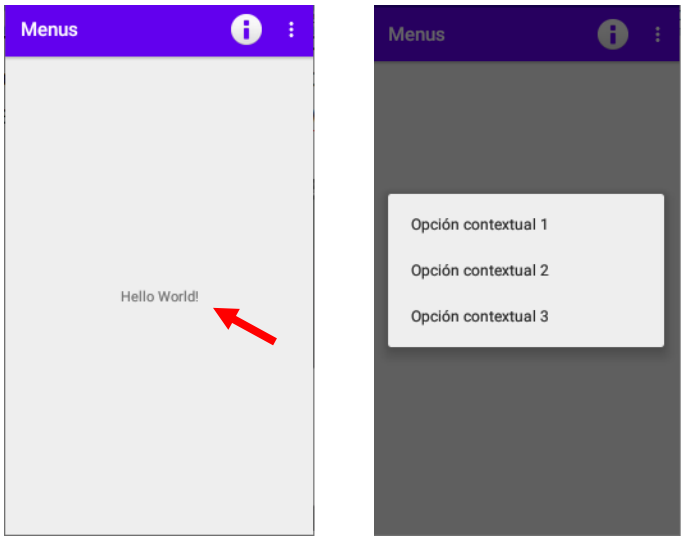
  - En caso de acceder desde el botón de *Menú*, las opciones se muestran en la parte inferior de la pantalla del terminal.

- Para permitir el acceso rápido a determinadas acciones, se puede conseguir que éstas aparezcan en la barra de la app.

- Los **submenús**, o menús secundarios, que se pueden mostrar al pulsar sobre una opción de un menú principal.



- Los **menús contextuales:** se muestran cuando se produce una pulsación larga sobre un elemento que tiene registrado este tipo de menú. Por ejemplo, las siguientes capturas muestran el menú contextual asociado a una TextView:



- Los menús se pueden definir mediante un fichero XML o bien mediante código.

2. CREAR UN MENU DE OPCIONES DESDE UN RECURSO XML

Los pasos a seguir son:

- Si no existe por defecto, **creamos un arhivo XML** con la definición del menú. El archivo se creará dentro de la carpeta **res/menu**, y su nombre puede ser cualquiera.

- La estructura del archivo es como sigue:

  - Un elemento principal **<menu>**
  - Una serie de elementos **<item>** que se corresponden con las distintas opciones a mostrar en el menú**.**

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/opc1" android:title="@string/opc_1"/>
    <item android:id="@+id/opc2" android:title="@string/opc_2"/>
    <item android:id="@+id/opc3" android:title="@string/opc_3"/>
    <item android:id="@+id/opc4" android:title="@string/opc_4"/>
</menu>
```

- Un elemento **<item>** soporta varios **atributos**. Los principales son:

  - ***android:id*** para que dicho item pueda ser identificado desde el código.
  - ***android:title*** y ***android:icon*** se refieren al texto que aparece en esa opción de menú y al icono (recurso drawable), respectivamente. Los iconos utilizados pueden estar en las carpetas "**res\drawable-…"** o bien pueden ser del sistema. En el primer caso se referencian mediante ***@drawable/nombre_del_archivo***, y en el segundo, mediante ***@android:drawable/nombre_del_archivo***.
  - ***android:showAsAction*** permite especificar cuándo y cómo este ítem deberá aparecer en la barra de la app. Por ejemplo:
    - ***"ifRoom",*** si hay espacio.
    - "***never***", "***always***": autoexplicativos…

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/opc1" android:title="@string/opc_1"
          android:icon="@android:drawable/ic_dialog_info"
          app:showAsAction="ifRoom"/>
    //código restante
</menu>
```

- Se puede implementar un submenú dentro de un elemento <item> insertando otro elemento <menu> como hijo de dicho elemento <item>:

```
<item android:id="@+id/opc2" android:title="@string/opc_2">
    <menu>
        <item android:id="@+id/sub_opc1" android:title="@string/sub_opc1"/>
        <item android:id="@+id/sub_opc2" android:title="@string/sub_opc2"/>
    </menu>
</item>
//código restante
```

- Para utilizar el menú en nuestra Activity, necesitamos "inflar" el recurso de tipo "menú" (que es un archivo XML) mediante el método ***inflate()*** de la clase ***MenuInflater***.

```
void        inflate(int menuRes, Menu menu)

            Inflate a menu hierarchy from the specified XML resource.
```

- La clase *MenuInflater,* respecto a los archivos de menú, es equivalente a la clase *LayoutInflater* para los archivos de layout, que hemos visto en la unidad anterior ("Listados y adaptadores")

- Esta operación se realiza dentro del método ***onCreateOptionsMenu().***

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

Primero obtenemos una referencia al objeto "inflador" mediante el método ***getMenuInflater()*** y posteriormente generamos la estructura del menú llamando a su método ***inflate()***, y pasándole como parámetro el ID del menú definido en XML ("*R.menu.menu_main"*). Por último devolvemos el valor *true* para confirmar que debe mostrarse el menú.

De esta forma tenemos creado el menú, pero todavía no tiene funcionalidad. Necesitamos un listener.

- **Evento *onOptionsItemSelected()***

  - Cuando se pulsa un item de un menú, se lanza el evento ***onOptionsItemSelected()*** y para procesar lo que queramos que ejecute nuestra app, hay que sobrescribir este método.
  - Este método recibe como parámetro el item de menú que ha sido pulsado por el usuario, cuyo ID podemos recuperar con el método ***getItemId()***. Según el valor de este ID podremos saber **qué opción ha sido pulsada** y ejecutar la acción correspondiente.
  - Este método devuelve un boolean:
    - ***true***: si procesamos un elemento del menú (*return true*;).
    - ***false***: si no lo procesamos (se llama al método "padre", porque la implementación por defecto retorna false).

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.opc1:
            //código que corresponda
            return true;
        case R.id.opc2:
            //código que corresponda
            return true;
```

```
            //otros casos…
            default:
                return super.onOptionsItemSelected(item);//false
        }
    }
```

- Podemos emplear el método ***getTitle()*** de la clase ***MenuItem*** para recuperar el texto de una opción del menú: ***item.getTitle()***

3. CREAR UN MENU CONTEXTUAL DESDE UN RECURSO XML

- Se pueden crear sobre cualquier View pero, normalmente, se usan con ListView.
- El menú contextual aparece cuando el usuario pulsa durante un tiempo sobre un elemento View.
- Los pasos a seguir son:
  - Crear el archivo con las opciones que necesitemos en el menú contextual.
  - Asociar, en el método onCreate(), el elemento View con su menú contextual mediante el método ***registerForContextMenu()***, al cual le pasamos como parámetro el objeto View.

```
txtHello=findViewById(R.id.txtHello);
registerForContextMenu(txtHello);
```

  - Sobreescribir el método ***onCreateContextMenu()*** en la Activity. Este método será llamado de forma automática cada vez que el usuario realice una pulsación larga sobre el elemento asociado al menú contextual. Y lo que hacemos en este método es **inflar el menú XML** que previamente habremos creado.

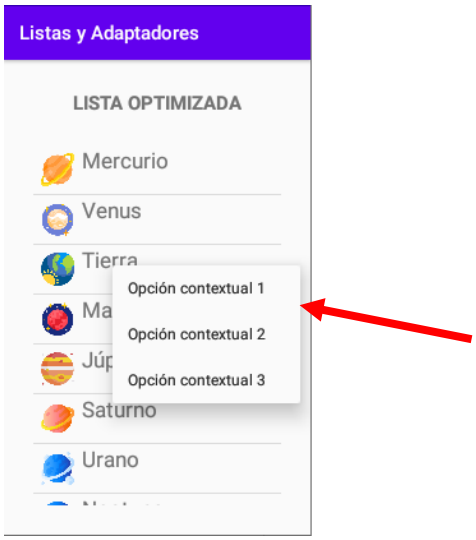
```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    getMenuInflater().inflate(R.menu.menu_contextual, menu);
}
```

  - Implementar el listener. Se hace como en el caso anterior, pero el evento asociado es ***onContextItemSelected()*** (similar a *onOptionsItemSelected()* para los menús de opciones).

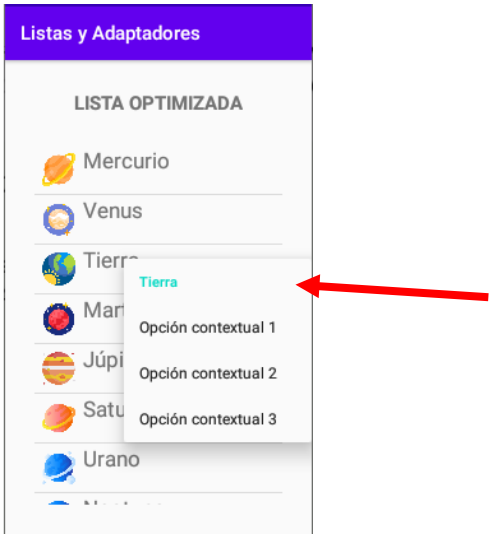
```
@Override
public boolean onContextItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.ctx1:
            //código que corresponda
            return true;
        //otros casos…
        default:
            return super.onContextItemSelected(item);
    }
}
```

3.1 CREAR UN MENU CONTEXTUAL PARA UNA LISTVIEW

- En principio, podemos asociar un menú contextual a una ListView igual que lo hicimos anteriormente sobre una TextView. Por ejemplo:



- Pero, lo normal será que debamos tener constancia de **qué elemento de la lista ha sido seleccionado**. Por ejemplo:



- **El elemento que ha sido pulsado se puede obtener del adaptador** y, en función de cuál sea dicho elemento incluso se podrían inflar diferentes menús contextuales.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo
menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater=getMenuInflater();
    AdapterView.AdapterContextMenuInfo info=(AdapterView.AdapterContextMenuInfo)menuInfo;
    String elemento=lvPlanetas.getAdapter().getItem(info.position).toString();
    menu.setHeaderTitle(elemento);
    inflater.inflate(R.menu.menu_contextual, menu);
}
```

- Podemos saber la posición del elemento seleccionado mediante el último parámetro recibido en el evento ***onCreateContextMenu()***, que es el llamado ***menuInfo***.

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

- El parámetro ***menuInfo*** contiene información adicional sobre la vista que ha sido pulsada para mostrar su menú contextual y, en el caso particular de la vista ListView, contiene la **posición del elemento concreto que se ha pulsado dentro de la lista.**

- Para obtener dicha posición convertimos el parámetro ***menuInfo*** en un objeto de tipo ***AdapterContextMenuInfo*** y después, accedemos a su **atributo** ***position*** tal como vemos en el código siguiente:

```
AdapterView.AdapterContextMenuInfo info=(AdapterView.AdapterContextMenuInfo)menuInfo;
String elemento=lvPlanetas.getAdapter().getItem(info.position).toString();
```

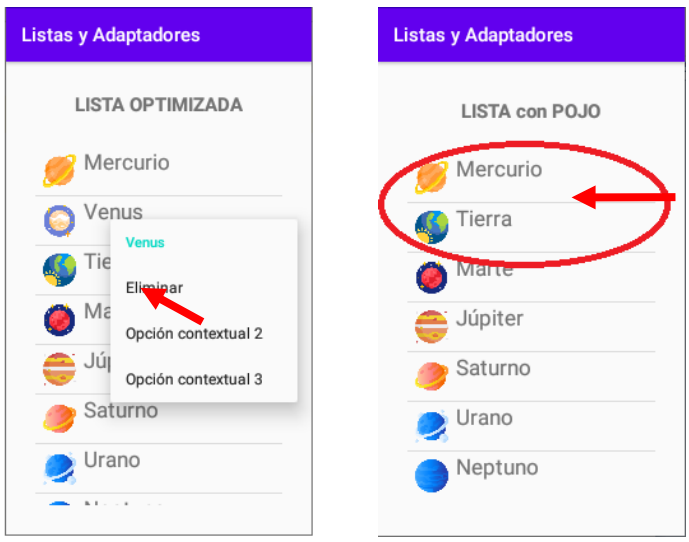
- Ya que disponemos del String con el elemento seleccionado en la ListView, también podemos establecerlo como título del menú contextual, mediante el método ***setHeaderTitle()***

- El **escuchador** del menú contextual se implementa igual que en el apartado anterior, mediante el evento ***onContextItemSelected()***.

3.2 MODIFICAR LA LISTVIEW A CONSECUENCIA DE LAS OPCIONES DEL MENU CONTEXTUAL

- Puede darse el caso de que la ListView deba varíar su contenido durante la ejecución, según la opción seleccionada en el menú contextual.

- Vamos a suponer que ahora nuestro menú contextual consta de dos opciones, una de las cuales es **Eliminar**, que permitirá eliminar el elemento seleccionado:



- Para ello podemos usar los métodos de la clase ***ArrayAdapter:***

  Documentación en http://developer.android.com/reference/android/widget/ArrayAdapter.html

Public Methods	
void	add(T object) Adds the specified object at the end of the array.
T	getItem(int position)
long	getItemId(int position)
void	remove(T object) Removes the specified object from the array.
void	setNotifyOnChange(boolean notifyOnChange) Control whether methods that change the list (`add(T)`, `insert(T, int)`, `remove(T)`, `clear()`) automatically call `notifyDataSetChanged()`.

- Hay que tener en cuenta algo muy importante: si queremos cambiar los datos en nuestro adaptador, **la estructura de datos subyacente debe soportar esta operación**, si no, no podremos hacerlo. Esto es, por ejemplo, el caso de los objetos de clase ArrayList, pero no de la clase Array

```
@Override
public boolean onContextItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.ctx1:
            //Eliminar
            adaptador.remove(adaptador.getItem(posicion));
            return true;
            //resto de código
    }
    return super.onContextItemSelected(item);
}
```

Es importante tener en cuenta que el código anterior funciona sólo si **el adaptador se ha configurado a partir de una estructura de datos dinámica**, como un ArrayList.