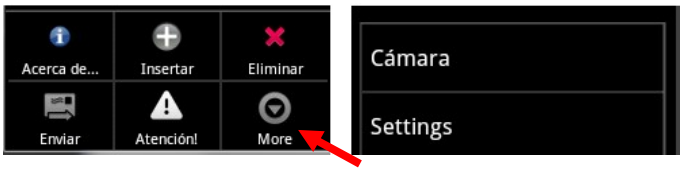
MENUS

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

Pág. 127 y sgts. del Manual de SGOliver

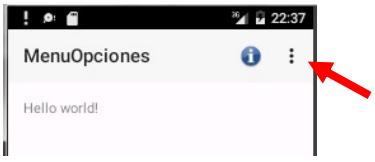
1. INTRODUCCION

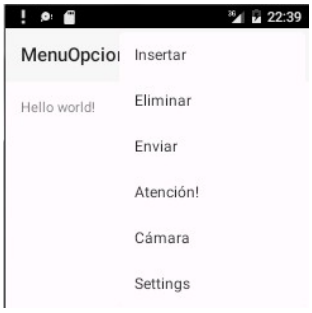
- Los menús son útiles para mostrar opciones adicionales que no están directamente visibles en la UI principal de una aplicación.
- Android dispone de varios tipos de menús:
  - Los **menús de opciones**: son los más habituales. Su posición y forma de visualizarse en la pantalla del dispositivo depende de la versión del sistema que se esté utilizando:
    - Para la **API 10 e inferiores**, aparecen en la parte inferior de la pantalla al pulsar el botón "**menú**" del teléfono. Se visualizan como máximo 6 items. Si el menú contiene más de 6 elementos, Android engloba el sexto item y los siguientes dentro de la última opción ("**more**").



    - Para la **API 11 y superiores**, los items de un menú de opciones se visualizan en la barra de acción. Por defecto, el sistema coloca todos los items en el botón de overflow, situado a la derecha de la barra de acción. Pero es posible cambiar esta posición defectiva y hacer que aparezcan visibles sobre la barra.

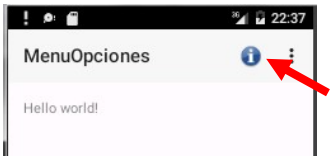
Más documentación: http://developer.android.com/intl/es/training/appbar/index.html

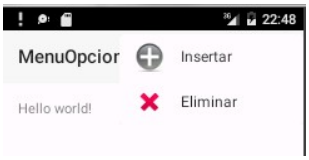




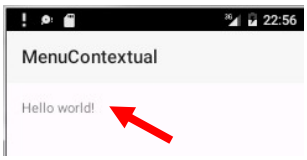
Las opciones del menú también son accesibles si pulsamos el botón de "menú" del teléfono. En este caso, se muestran en la parte inferior de la pantalla del terminal:

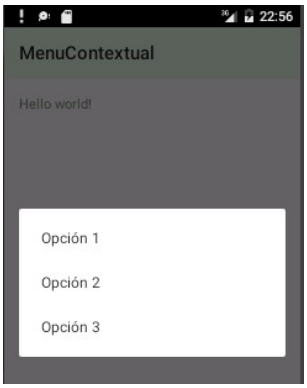
- Los **submenús**, o menús secundarios, que se pueden mostrar al pulsar sobre una opción de un menú principal.





- 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.

- El archivo consta de un elemento principal **<menu>**, y una serie de elementos **<item>** que se corresponden con las distintas opciones a mostrar en el menú**.** Por ejemplo:

```
<menu xmlns:android=http://schemas.android.com/apk/res/android…>
    <item android:id="@+id/item1" android:icon="@drawable/acercade"
        android:title="Acerca de..." android:showAsAction="ifRoom"/>
</menu>
```

- Un elemento <ítem> 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 ActionBar. Por ejemplo:

    - ***"ifRoom",*** si hay espacio.
    - *"**never**"*, "***always***": autoexplicativos…

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

```
<item android:id="@+id/item1" android:icon="@drawable/acercade"
    android:title="Acerca de..." android:showAsAction="ifRoom">
    <menu>
        <item android:id="@+id/item2"android:icon="@drawable/insertar"
              android:title="Insertar" />
        <item android:id="@+id/item3"android:icon="@android:drawable/ic_delete"
              android:title="Eliminar" />
    </menu>
</item>
```

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

- Esta operación se realiza 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;
}
```

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.

En versiones inferiores a la API 11, el sistema llama a este método cuando el usuario abre el menú por primera vez. En la API 11 y superiores, el sistema llama a este método cuando arranca la actividad, para poder mostrar los items en la barra de acción.

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

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

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        switch (id) {
            case R.id.action_settings:
                Toast.makeText(this, "Opción settings pulsada",
                Toast.LENGTH_SHORT).show();
                return true;
            case R.id.item1:
                Toast.makeText(this, "Opción 1 pulsada",
                Toast.LENGTH_SHORT).show();
                return true;
            (…)
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```

- Podemos emplear el método ***getTitle()*** de la clase ***MenuItem*** para recuperar el texto de una opción del menú, y no escribirla como hemos hecho en la sentencia Toast (como una cadena de texto explícita):

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id) {
        case R.id.action_settings:
            Toast.makeText(this, "Opción "+item.getTitle()+" pulsada",
            Toast.LENGTH_SHORT).show();
            return true;
        case R.id.item1:
            Toast.makeText(this, "Opción "+item.getTitle()+" pulsada",
            Toast.LENGTH_SHORT).show();
            return true;
        (…)
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

- Podemos probar todo esto en los **ejercicios 1 y 2**.

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 un elemento View.
- Los pasos a seguir son:
  - 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.

```
TextView etiquetaConMenuContextual = (TextView) findViewById (R.id.tvHello);
registerForContextMenu(etiquetaConMenuContextual);
```

  - Sobreescribir el método ***onCreateContextMenu()*** en la Activity. Este método será llamado de forma automática cada vez que el usuario pulse durante un tiempo 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);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_main, menu);
}
```

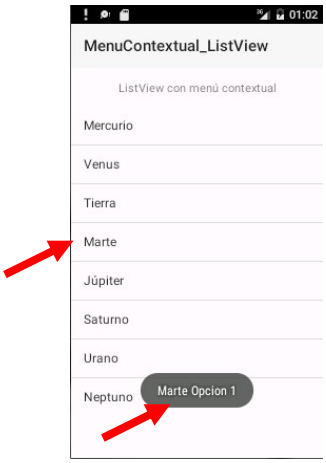
- Sobreescribir el método ***onContextItemSelected()***, que se llama de forma automática cando se selecciona algo dentro del menú contextual. Es decir, este método contendrá las acciones a realizar tras pulsar una opción determinada del menú contextual (similar a *onOptionsItemSelected()* para los menús de opciones).

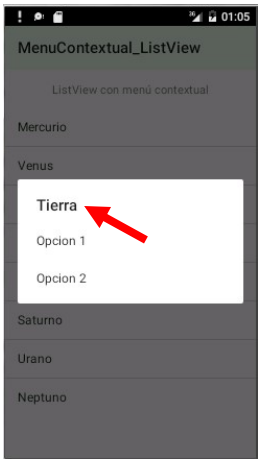
```
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.itemCtx1:
            Toast.makeText(this, "Opcion "+item.getTitle()+" pulsada",
            Toast.LENGTH_SHORT).show();
            return true;
         (…)
        default:
            return super.onContextItemSelected(item);
    }
}
```

- Podemos probar esto en el **ejercicio 3**.

3.1 CREAR UN MENU CONTEXTUAL PARA UNA LISTVIEW

- En principio, aplicamos todo lo visto hasta ahora, asociando el elemento ListView con su menú contextual mediante el método ***registerForContextMenu()***.
- Podemos probar esto en el **ejercicio 5**.
- Pero, lo normal será que debamos tener constancia de **qué elemento de la lista ha sido seleccionado**. Por ejemplo:





- Podemos saber la posición que ocupa en la lista el 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 de la vista que se ha pulsado 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:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo
            menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflador = getMenuInflater();
    // convertimos el parámetro menuInfo en un objeto de tipo AdapterContextMenuInfo
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
         menuInfo;
    // el atributo position nos permite acceder al item que ocupa la posición pulsada
    planeta=lvPlanetas.getAdapter().getItem(info.position).toString();
    // inflamos el menú contextual
    inflador.inflate(R.menu.menu_contextual, menu);
}// end onCreateContextMenu
```

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

```
menu.setHeaderTitle(planeta);
```

- Podemos probar esto en el **ejercicio 6**.

- También podemos conocer qué item de la ListView ha sido pulsado mediante el método ***onContextItemSelected()***.

- En este caso, se utiliza el parámetro ***item***, de la clase **MenuItem**, mediante una llamada al método ***getMenuInfo(),*** que devuelve un objeto de la clase ***ContextMenuInfo.***

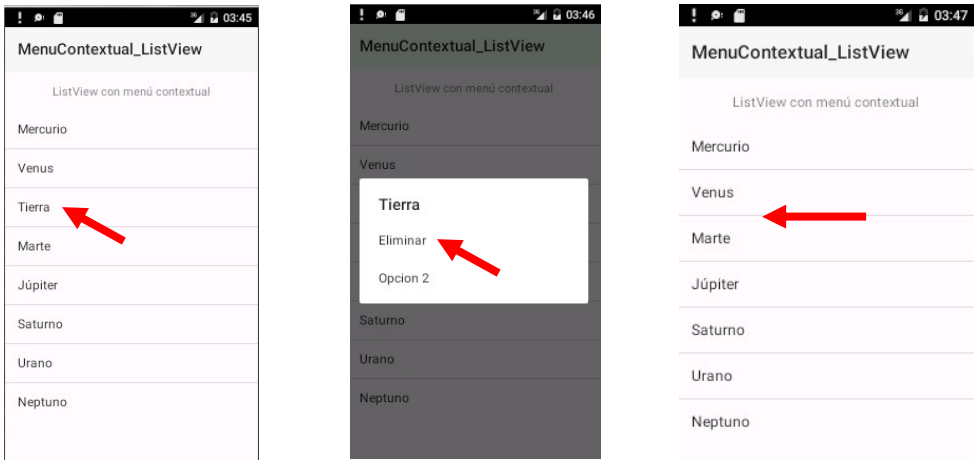
- Este objeto de tipo ***ContextMenuInfo*** se convierte en un objeto de tipo ***AdapterContextMenuInfo*** el cual nos permitirá acceder a su **atributo *position*** tal como hicimos antes, en el método ***onCreateContextMenu()***:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
         item.getMenuInfo();
    planeta=adaptador.getItem(info.position).toString();
    switch (item.getItemId()) {
        case R.id.item1:
            Toast.makeText(this, planeta + " "+item.getTitle(),
            Toast.LENGTH_SHORT).show();
            return true;
            (…)
    }
}
```

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, como su nombre indica, 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(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info =
          (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.item1:
            adaptador.remove(adaptador.getItem(info.position));
            return true;
        (…)
    }
} /
```

El código anterior funciona si **el adaptador se ha configurado a partir de una estructura de datos dinámica**, como un ArrayList.

- Podemos probar esto en el **ejercicio 7**.