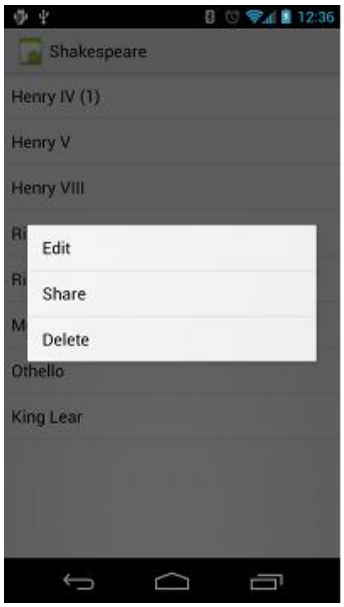
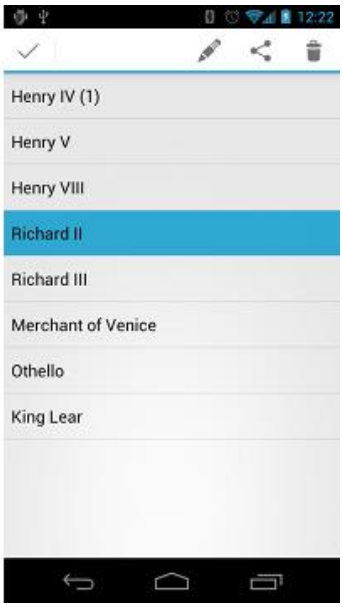
Menús Contextuales

Los menús contextuales. Este tipo de menú siempre va asociado a un control concreto de la pantalla y se muestra al realizar una pulsación larga sobre éste. Suele mostrar opciones específicas disponibles únicamente para el elemento pulsado. Por ejemplo, en un control de tipo lista podríamos tener un menú contextual que apareciera al pulsar sobre un elemento concreto de la lista y que permitiera editar su texto o eliminarlo de la colección. `android:id`. El ID identificativo del elemento, con el que podremos hacer referencia dicha opción.

Existen dos formas de proporcionar menús contextuales:





El primero es un menú flotante contextual: el menú aparece como un menú flotando encima de la pantalla (similar a un dialogo) cuando el usuario realiza un click largo en una vista que declaramos que soporta el menú contextual.

El segundo es una acción en modo contextual: Esta forma es una implementación de Android de un ActionMode que muestra una barra de acción contextual en la parte de arriba de la pantalla con las acciones que pueden aplicarse sobre el/los elementos seleccionados. A diferencia del primer

caso, con esta forma el usuario puede realizar una acción sobre multiples elementos (si tu aplicación lo permite).

Menu flotante Contextual

La creación y utilización de este tipo de menús es muy parecida a lo que ya vimos para los menús y submenús básicos, pero presentan algunas particularidades que hacen interesante tratarlos al margen del resto en este nuevo artículo.

Empecemos por un caso sencillo. Vamos a partir de un proyecto nuevo, que ya debe contener por defecto una etiqueta de texto con un *Hello World*).

Vamos a añadir en primer lugar un menú contextual que aparezca al pulsar sobre la etiqueta de texto. Para ello, lo primero que vamos a hacer es indicar en el método `onCreate()` de nuestra actividad principal que la etiqueta tendrá asociado un menú contextual. Esto lo conseguimos con una llamada a `registerForContextMenu()`:

```
1
2       public void onCreate(Bundle savedInstanceState) {
3           super.onCreate(savedInstanceState);
4           setContentView(R.layout.main);
5
6           //Obtenemos las referencias a los controles
7           lblMensaje = (TextView)findViewById(R.id.LblMensaje);
8
9           //Asociamos los menús contextuales a los controles
10          registerForContextMenu(lblMensaje);
        }
```

A continuación, igual que hacíamos con `onCreateOptionsMenu()` para los menús básicos, vamos a sobreescribir en nuestra 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 actuaremos igual que para los menús básicos, *inflando* el menú XML que hayamos creado con las distintas opciones, o creando a mano el menú mediante el método `add()` [para más información leer el artículo anterior]. En nuestro ejemplo hemos definido un 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="OpcEtiqueta1"></item>
<item android:id="@+id/CtxLblOpc2"
    android:title="OpcEtiqueta2"></item>

</menu>
```

Por su parte el evento `onCreateContextMenu()` quedaría de la siguiente forma:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                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);
        }
    }
```

Ahora vamos con algunas particularidades. Los menús contextuales se utilizan a menudo con controles de tipo lista, lo que añade algunos detalles que conviene mencionar. Para ello vamos a añadir a nuestro ejemplo una lista con varios datos de muestra y vamos a asociarle un nuevo menú contextual. Modificaremos el layout XML de la ventana principal para añadir el control `ListView` y modificaremos el método `onCreate()` para obtener la referencia al control, insertar varios datos de ejemplo, y asociarle un menú contextual:

NOTA: Esto se puede hacer tanto para ListView como para GridView, pero google está abandonado poco a poco este tipo de Menús, y el elemento ReciclerView ya no es compatible con las siguientes indicaciones.

```

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

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

        //Rellenamos la lista con datos de ejemplo
        String[] datos =
            new String[]{"Elem1","Elem2","Elem3","Elem4","Elem5"};

        ArrayAdapter<String> adaptador =
            new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, datos);

        lstLista.setAdapter(adaptador);

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

```

Como en el caso anterior, vamos a definir en XML otro menú para asociarlo a los elementos de la lista, lo llamaremos "*menu_ctx_lista*":

```
<?xml version="1.0" encoding="utf-8"?>
<menu
  xmlns:android="http://schemas.android.com/apk/res/android">

<item android:id="@+id/CtxLstOpc1"
    android:title="OpcLista1"></item>
<item android:id="@+id/CtxLstOpc2"
    android:title="OpcLista2"></item>

</menu>
```

Como siguiente paso, y dado que vamos a tener varios menús contextuales en la misma actividad, necesitaremos modificar el evento `onCreateContextMenu()` para que se construya un menú distinto dependiendo del control asociado. Esto lo haremos 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()`. Utilizaremos para ello una llamada al método `getId()` de dicho parámetro:

```

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                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.LstLista)
    {
        AdapterView.AdapterContextMenuInfo info =
            (AdapterView.AdapterContextMenuInfo)menuInfo;

        menu.setHeaderTitle(
            lstLista.getAdapter().getItem(info.position).toString());

        inflater.inflate(R.menu.menu_ctx_lista, menu);
    }
}

```

Vemos cómo en el caso del menú para el control lista hemos ido además un poco más allá, y hemos personalizado el título del menú contextual [mediante `setHeaderTitle()`] para que muestre el texto del elemento seleccionado en la lista. Para hacer esto nos hace falta saber la posición en la lista del elemento seleccionado, algo que podemos conseguir haciendo uso del último parámetro recibido en el evento `onCreateContextMenu()`, llamado `menuInfo`. Este parámetro contiene información adicional del control que se ha pulsado para mostrar el menú contextual, y en el caso particular del control `ListView`contiene la posición del elemento concreto de la lista que se ha pulsado. Para obtenerlo, convertimos el parámetro `menuInfo` a un objeto de tipo `AdapterContextMenuInfo` y accedemos a su atributo `position` tal como vemos en el código anterior.

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:

```
1
2       @Override
3       public boolean onContextItemSelected(MenuItem item) {
4
5           AdapterContextMenuInfo info =
6               (AdapterContextMenuInfo) item.getMenuInfo();
7
8          switch (item.getItemId()) {
9              case R.id.CtxLblOpc1:
10                 lblMensaje.setText("Etiqueta: Opcion 1 pulsada!");
11                 return true;
12             case R.id.CtxLblOpc2:
13                 lblMensaje.setText("Etiqueta: Opcion 2 pulsada!");
14                 return true;
15             case R.id.CtxLstOpc1:
16               lblMensaje.setText("Lista[" + info.position + "]: Opcion 1 pulsada!");
17                  return true;
18             case R.id.CtxLstOpc2:
19               lblMensaje.setText("Lista[" + info.position + "]: Opcion 2 pulsada!");
20                  return true;
21             default:
22                  return super.onContextItemSelected(item);
       }
   }
```

Como vemos, aquí también utilizamos la información del objeto `AdapterContextMenuInfo` para saber qué elemento de la lista se ha pulsado, con la única diferencia de que en esta ocasión lo obtenemos mediante una llamada al método `getMenuInfo()` de la opción de menú (`MenuItem`) recibida como parámetro.

ActionMode Contextual.

Es la evoluación de los menús contextuales, google apuesta por ellos ahora, y ha ganado mucho terreno a los menús contextuales.

Para habilitar este modo en nuestra aplicación, necesitamos

-implementar la interfaz ActionMode.Callback. En estos métodos especificaremos las acciones para la action bar contextual, responderemos a los eventos de click.

- llamar a startActionMode() cuando se haga un click largo en algún elemento que elijamos.

Primero vamos a ver como se implementa la interfaz:

```
public class MiActionModeCallBack implements Callback {


@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        // Inflate a menu resource providing context menu items
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.menu_contextual, menu);
    mode.setTitle("Titulo del Action Mode");
    mode.setSubtitle("Subtitulo");

        return true;
        }


@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false;
        }


@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        switch (item.getItemId()) {
        case R.id.CtxLblOpc1 :
            Log.i("ActionItemClicked","Opcion 1");
        mode.finish();
        return true;
```

```
            case R.id.CtxLblOpc2:
            Log.i("ActionItemClicked","Opcion 1");
            mode.finish();
                return true;
default:
        return false;
        }
        }

@Override
public void onDestroyActionMode(ActionMode mode) {
            }
        };
```

De esta interfaz, solo he completado 2 metodos OnCreatedActionMode() y OnActionItemSelected(). OnCreatedActionMode se le llama cuando se crea la ActionMode, es decir cuando llamamos al método StartActionMode().

Es muy parecido al OnCreateMenuContext o el OnCreateOptionsMenu(), debemos inflar el menú(en este caso he reutilizado el menú del ejemplo anterior) pero a diferencia de ellos tenemos un objeto ActionMode que nos ayudará a personalizar la barra, por ejemplo poniendo título y subtitulo como he hecho.

El método OnActionItemSelected se llamará cuando el usuario clickea una opción de la barra de del ActionMode, en mi caso simplemente muestro en el log la opción clicada.

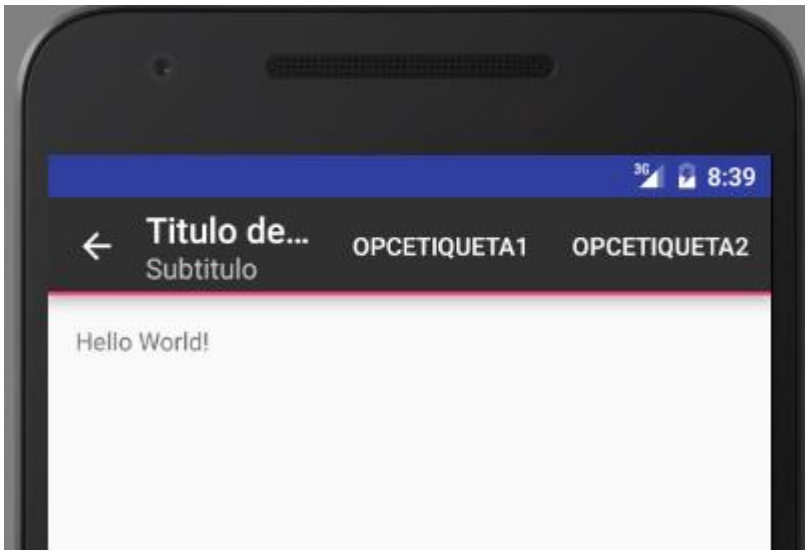
Segundo, llamar a StartActionMode con un click largo.

Para este fin debemos asignar un método que responda al evento OnLongClick del elemento que deseemos y dentro de este iniciar el ActionMode con el ya mencionado StartActionMode:

```
ma= new MiActionModeCallBack();
tv= (TextView) findViewById(R.id.textView);

tv.setOnLongClickListener(new View.OnLongClickListener() {
        public boolean onLongClick(View view) {
        // Start the CAB using the ActionMode.Callback defined above
        MainActivity.this.startActionMode(ma);
        view.setSelected(true);
        return true;
    }
});
```

Cuando hagamos un click largo en la etiqueta(“hello world”) veremos lo siguiente:


8:39
Titulo de...
Subtitulo
OPCETIQUETA1
OPCETIQUETA2
Hello World!