**1.2. MENIURI CONTEXTUALE**

Meniul contextual este un tip de meniu care conține comenzi legate de obiectul indicat în acel moment de cursor.

**1.2.1 Floating menu**

Printre avantajele oferite de acest tip de meniu se numără:

* Salvarea mișcărilor mouse-ului. Este evitată astfel selectarea fiecărui obiect în parte și apoi “mutarea cursorului” în bara de instrumente pentru a alege o acțiune
* Aflarea mai multor informații despre obiecte. Fiecare obiect poate avea o serie de comenzi specifice.

În vederea realizării meniurilor, este necesară crearea unui folder nou în resources (în acest caz l-am denumit “menu”).

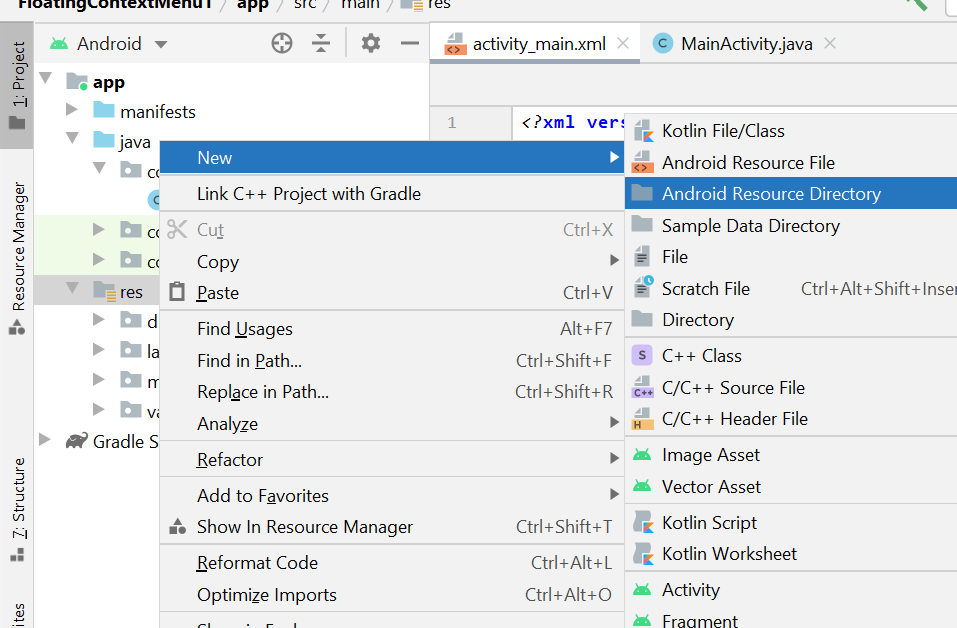


Figura nr. 1 – Crearea unui nou director (menu)

Acesta va avea precizat la **resource type: menu.**

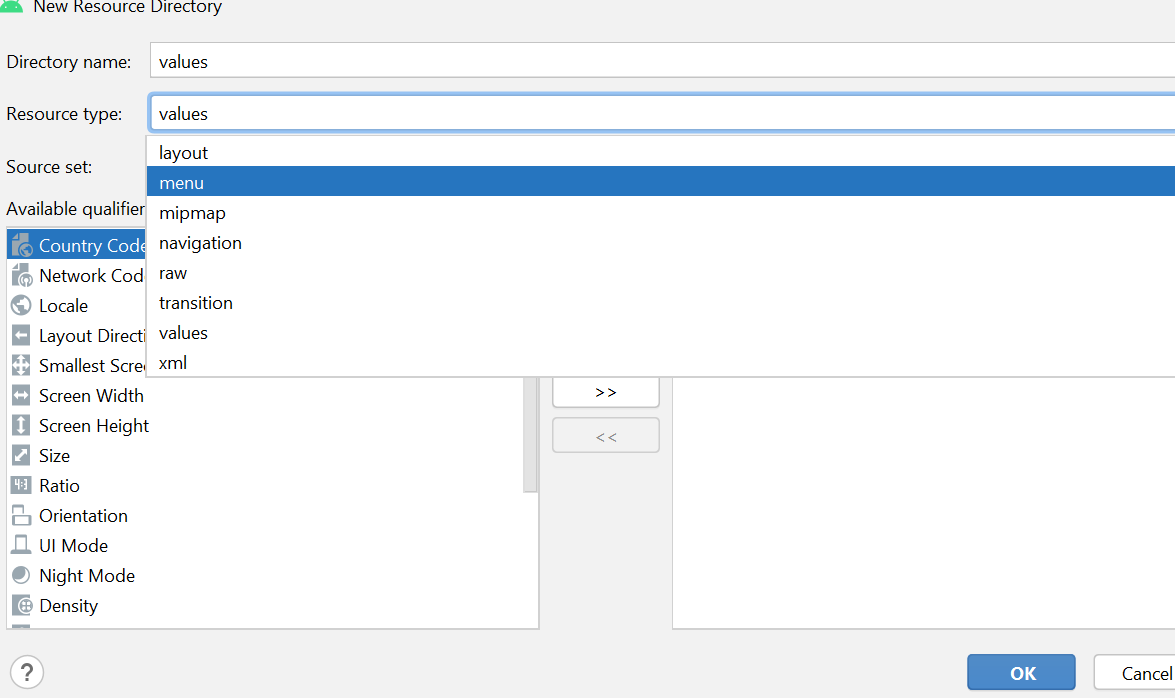


Figura nr. 2 – Modificare Resource type în menu

În acest folder este adăugat un **Menu resource file** pe care l-am denumit “meniu1”.

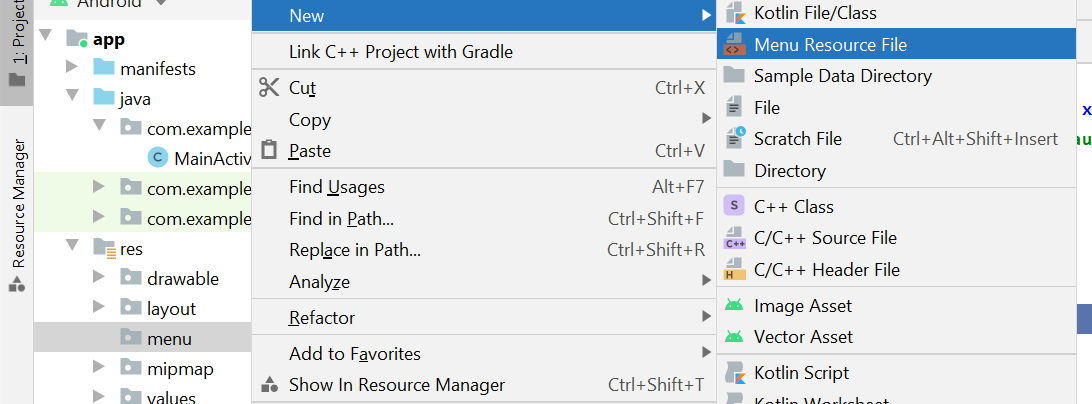


Figura nr. 3 – Crearea unui nou Fișier XML aferent meniului

În acest caz, implementarea este realizată asupra unui text (TextView). Așadar, în **activity-main.xml**, vom avea următorul cod:

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout 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"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**TextView  
 android:id="@+id/txtView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Optiuni"  
 android:textSize="30dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />  
</**androidx.constraintlayout.widget.ConstraintLayout**>

În **meniu1.xml** (fișierul XML al meniului), sunt create de fapt opțiunile pe care le va conține meniul.

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"**>  
  
 //crearea optiunilor din meniu  
 <**item  
 android:id="@+id/option1"  
 android:title="Option 1"**></**item**>  
  
 <**item  
 android:id="@+id/option2"  
 android:title="Option 2"**></**item**>  
</**menu**>

În **MainActivity.java** în metoda onCreate va fi apelată metoda **registerForContextMenu()**, care practic „reține” obiectul asupra căruia va fi implementat Floating menu. În acest caz, metoda va avea ca parametru chiar textul „Optiuni”.

Metoda **onContextItemSelected()** este apelată atunci când utilizatorul decide să aleagă o opțiune din meniu. Este determinată clar care opțiune a fost aleasă (în acest caz, alegerea unei opțiuni determină apariția unui mesaj de informare pe ecran).

**getItemId()** preia elementele meniului din **meniu1.xml** prin intermediul **android:id.** În cazul în care sunt preluate elementele din meniu, este retunat **true**. În cazul in care aplicația are mai multe fragmente, apelul către superclasa determină secvența de cod să “intre” în fiecare fragment în parte în vederea returnării de **true** sau **false.**

În continuare, vom suprascrie metoda **onCreateContextMeniu()**, în cadrul căreia vor fi adăugate în meniu opțiunile din **meniu1.xml.**

**package** com.example.floatingcontextmenu1;  
**import** androidx.annotation.NonNull;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.ContextMenu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//definire si initializare text*  
 TextView textView;  
 textView = findViewById(R.id.***txtView***);  
 *//memorarea obiectului asupra caruia se va activa meniul*   
 registerForContextMenu(textView);  
 }  
  
 @Override  
 **public void** onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  
 **super**.onCreateContextMenu(menu, v, menuInfo);  
 *//”popularea” meniului cu elementele din fisierul XML*  
 getMenuInflater().inflate(R.menu.***menu1***, menu);  
 }

@Override  
 **public boolean** onContextItemSelected(@NonNull MenuItem item) {  
 **switch** (item.getItemId())  
 {

*//in functie de ce optiune alege utilizatorul, apare un mesaj text pe ecran*  
 **case** R.id.***option1***:  
 Toast.*makeText*(**this**, **"Optiunea 1 este selectata"**, Toast.***LENGTH\_LONG***).show();  
 **return true**;  
 **case** R.id.***option2***:  
 Toast.*makeText*(**this**, **"Optiunea 2 este selectata"**, Toast.***LENGTH\_LONG***).show();  
 **return true**;  
 **default**:  
 **return super**.onContextItemSelected(item);  
  
 }  
 }  
}

Vom porni un emulator pe care să ruleze mini-aplicația. Rezultatul este vizibil doar atunci când utilizatorul ține apăsat puțin mai mult pe click în dreptul textului “Optiuni”.

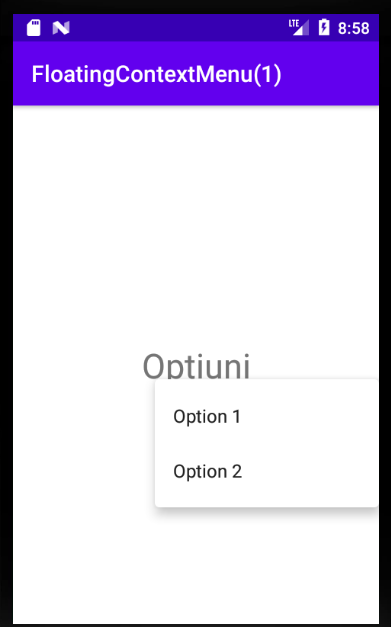
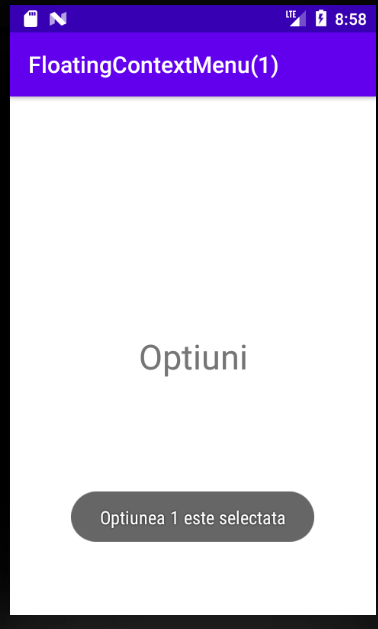
 

Figura nr. 5 – Selecția unei opțiuni din meniu

Figura nr. 4 – Apariția meniului

**1.2.2.1. Contextual action mode**

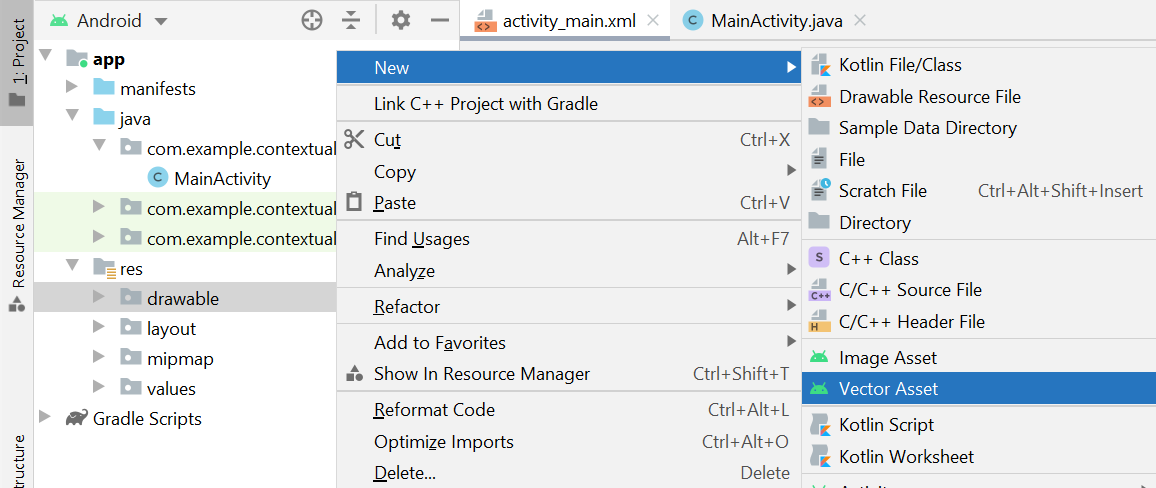
Acest tip de meniu este destul de similar cu meniul anterior, în sensul în care este activat tot în momentul unui click prelungit asupra obiectului dorit. Singura diferență o constituie însă faptul că apare în partea de sus a ecranului.

Implementarea este realizată tot asupra unui textView, însă în loc de acesta, pot exista diverse elemente (buton, imagine etc.). Așadar , în **activity-main.xml** avem codul:

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout 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"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>

<**TextView  
 android:id="@+id/txtView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello world!"  
 android:textSize="30dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />  
</**androidx.constraintlayout.widget.ConstraintLayout**>

De data aceasta, ca și opțiuni vom crea niște elemente de tip grafic. Pentru a realiza aceasta operațiune, este nevoie să realizăm în meniul **drawable** două elemente noi (**Vector asset**). În **Clip art** pot fi alese diverse iconițe sugestive. Noi am creat un buton “de tip share” și unul “de tip delete”.

****

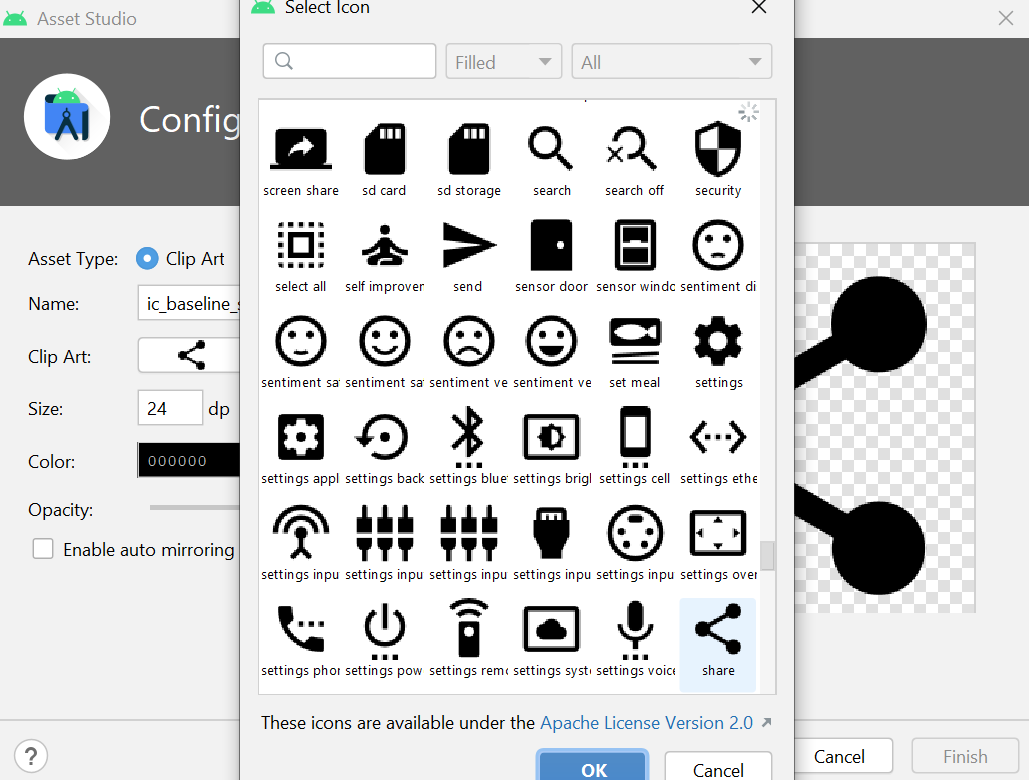


Figura nr. 6 – Crearea unui Vector Asset

Figura nr. 7 – Crearea unei iconițe (icon)

In **meniu1.xml**  sunt create astfel cele 2 opțiuni. Pe lângă **id** și **title**, de această dată trebuie adăugate și iconițele aferente prin **android:icon.**

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"**>  
  
 <**item  
 android:id="@+id/optionShare"  
 android:title="Optiunea share"  
 android:icon="@drawable/share\_icon"**></**item**>  
 <**item  
 android:id="@+id/optionDelete"  
 android:title="Optiunea delete"  
 android:icon="@drawable/delete\_icon"**></**item**>  
</**menu**>

În clasa **MainActivity.java**, setăm în primul rând “ce face aplicația” atunci când utilizatorul dă un click mai lung asupra textului “Hello world!”. De această dată trebuie implementată interfața **ActionMode.Callback** care conține acțiunile pentru bara de meniu. Acest tip de meniu are un “ciclu de viață”, în sensul că metodele din interfață sunt apelate pe rând.

Primul pas prin care trece **ActionMode.Callback** este **onCreateActionMode()**. Primul parametru al acestei metode este de tip **ActionMode** și reprezintă de fapt chiar bara care apare deasupra. În acest moment, opțiunile create în **meniu1.xml** sunt introduse în partea de sus a ecranului, iar lângă acestea este setat și un titlu.

În metoda **onActionItemClicked()** este determinată care opțiune a fost aleasă de utilizator (alegerea unei opțiuni determină apariția unui mesaj de informare pe ecran). **getItemId()** preia elementele meniului din **meniu1.xml** prin intermediul **android:id.**

Metoda **onDestroyActionMode()** semnifică de fapt dispariția meniului și este apelată odată cu acționarea butonului “back (🡨)”.

**package** com.example.contextualactionmode1;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** androidx.appcompat.view.ActionMode;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **private** ActionMode **actionMode**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//definire si initializare TextView*  
 TextView textView;  
 textView = findViewById(R.id.***txtView***);  
 *//setam ce se intampla la un click mai lung pe textView* textView.setOnLongClickListener(**new** View.OnLongClickListener() {  
 @Override  
 **public boolean** onLongClick(View v) {  
 *//daca actionMode (meniul) este deja activat, nu mai trebuie activat inca odata* **if**(**actionMode** != **null**)  
 {  
 **return false**;  
 }

*//este "chemata" logica din spatele ActionMode.Callback*  
 **actionMode** = startSupportActionMode(**actionModeCallBack**);  
 **return true**;  
 }  
 });  
 }

**private** ActionMode.Callback **actionModeCallBack** = **new** ActionMode.Callback() {

*//se "activeaza" in urma click-ului*  
 @Override  
 **public boolean** onCreateActionMode(ActionMode mode, Menu menu) {

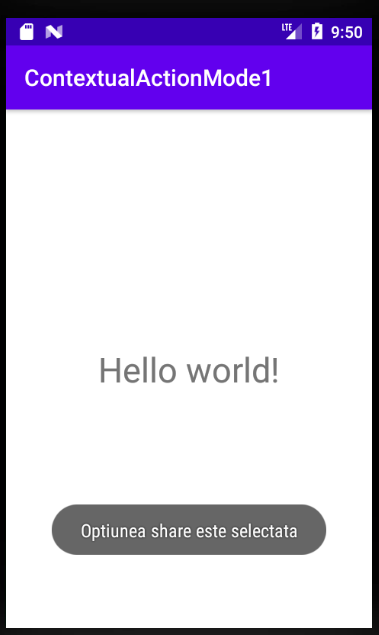
*//meniul este „populat” cu elementele din XML si cu un titlu*  
 mode.getMenuInflater().inflate(R.menu.***meniu1***,menu);  
 mode.setTitle(**"Alege o optiune"**);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onPrepareActionMode(ActionMode mode, Menu menu) {  
 **return false**;  
 }  
  
 @Override  
 **public boolean** onActionItemClicked(ActionMode mode, MenuItem item) {  
 **switch** (item.getItemId())  
 {

*//in functie de ce optiune alege utilizatorul, apare un mesaj text*   
 **case** R.id.***optionShare***:  
 Toast.*makeText*(MainActivity.**this**, **"Optiunea share este selectata"**, Toast.***LENGTH\_LONG***).show();  
 mode.finish();  
 **return true**;  
 **case** R.id.***optionDelete***:  
 Toast.*makeText*(MainActivity.**this**, **"Optiunea delete este selectata"**, Toast.***LENGTH\_LONG***).show();  
 **return true**;  
 **default**:  
 **return false**;  
 }  
 }  
  
 @Override  
 **public void** onDestroyActionMode(ActionMode mode) {

*//atunci cand dispare meniul, dispare si focusul pe acesta*

**actionMode** = **null**;  
 }  
 };  
}

În continuare, vom porni un emulator pe care să ruleze mini-aplicația. După un click prelungit asupra textului “Hello world!”, apare meniul în partea de sus a ecranului. După selecția unui buton (share sau delete), apare un mesaj pe ecran.



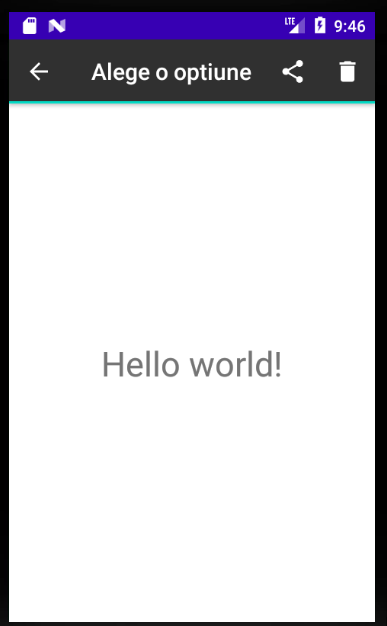


Figura nr. 9 – Selectarea unei opțiuni

Figura nr. 8 – Apariția meniului în partea de sus a ecranului

**1.2.2.1. List View Contextual action mode cu CheckBox**

Acest tip de meniu se activează de asemenea la acționarea unui click prelungit asupra unui obiect. Diferența față de meniul anterior o constituie faptul că pot fi întreprinse cu ajutorul meniului acțiuni pentru mai multe elemente selectate în același timp.

Avem nevoie tot de două butoane (share și delete) realizate în meniul **drawable**. Este necesară și crearea unui XML pentru meniu, care este similar meniului prezentat anterior.

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"**>  
  
 <**item  
 android:id="@+id/optiuneShare"  
 android:title="Optiune share"  
 android:icon="@drawable/icon\_share"**></**item**>  
 <**item  
 android:id="@+id/optiuneDelete"  
 android:title="Optiune delete"  
 android:icon="@drawable/icon\_delete"**></**item**>  
</**menu**>

După cum am menționat, de această dată avem nevoie de o listă de elemente, nu doar de un textView. Așadar, în folderul **Values,** în **Strings.xml** adaugăm o listă cu optiuni:

<**resources**>  
 <**string name="app\_name"**>ContextualActionMode2</**string**>  
  
 <**string-array name="arrayOptiuni"**>  
 <**item**>Optiunea 1</**item**>  
 <**item**>Optiunea 2</**item**>  
 <**item**>Optiunea 3</**item**>  
 <**item**>Optiunea 4</**item**>  
 <**item**>Optiunea 5</**item**>  
 <**item**>Optiunea 6</**item**>  
 </**string-array**>  
</**resources**>

c

Din punct de vedere al layout-ului, vom avea nevoie atât de layout-ul pentru listă, cât și de cel principal, pentru opțiuni. Așadar, în **activity-main.xml,** va fi definit un **ListView:**

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout  
 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"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**ListView  
 android:id="@+id/listOptiuni"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**/>  
  
</**RelativeLayout**>

Definim în folderul **layout** un nou fișier XML (**layout-optiuni.xml**), care va avea un **TextView,** dar și un element **CheckBox**, deoarece ne dorim ca din punct de vedere vizual, să fie mai ușor de observat ce opțiuni am ales. De asemenea, ne dorim ca atunci când se deschide mini-aplicația, check box-ul să nu fie vizibil, ci doar după acționarea unui click lung pe o anumita opțiune.

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"**>  
  
 <**TextView  
 android:id="@+id/txtOptiune"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:gravity="center"  
 android:layout\_marginLeft="25dp"  
 android:textSize="18dp"  
 android:textStyle="bold"** />  
  
 <**CheckBox  
 android:id="@+id/checkBox"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentRight="true"  
 android:layout\_centerInParent="true"  
 android:layout\_marginRight="25dp"  
 android:visibility="gone"** />  
  
</**RelativeLayout**>

Definim o clasă nouă (**ListViewAdapter.java**), în care sunt inițializate lista de opțiuni, TextBox-ul și CheckBox-ul. Este definit un constructor pentru lista de opțiuni, dar și pentru context.

De asemenea, definim un **View** în cadrul căruia sunt adăugate liniile cu opțiuni, elementele de tip CheckBox (împreună cu logica din spatele clasei **MainActivity.java**).

**setOnCheckedChangeListener()** definește ce se înâtmplă în meniu în momentul în care sunt selectate sau de-selectate optiuni.

**stergeOptiuni()** este o funcție care după cum ii spune si numele, șterge opțiunile pe care le selectează utilizatorul.

**package** com.example.contextualactionmode2;  
**import** android.app.Activity;  
**import** android.content.Context;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.CheckBox;  
**import** android.widget.CompoundButton;  
**import** android.widget.TextView;  
**import** androidx.annotation.NonNull;  
**import** androidx.annotation.Nullable;  
**import** java.util.ArrayList;  
**import** java.util.List;

**public class** ListViewAdapter **extends** ArrayAdapter<String> {  
 *//declararea unei liste de optiuni* **private** List<String> **optiuni** = **new** ArrayList<>();  
 **private** Context **context**;  
 *//definire constructor* **public** ListViewAdapter(List <String> optiuni, Context context)  
 {  
 **super**(context, R.layout.***layout\_optiuni***,optiuni);  
 **this**.**context** = context;  
 **this**.**optiuni** = optiuni;  
 }

@NonNull  
 @Override  
 **public** View getView(**int** position, @Nullable View convertView, @NonNull ViewGroup parent) {  
 *//definire layout* LayoutInflater inflater = ((Activity)**context**).getLayoutInflater();  
 View row = inflater.inflate(R.layout.***layout\_optiuni***,parent,**false**);  
 *//declarare si instantiere text optiune* TextView txtOptiune;  
 txtOptiune = row.findViewById(R.id.***txtOptiune***);  
 txtOptiune.setText(**optiuni**.get(position));  
 *//declarare si instantiere checkBox* CheckBox checkBox = row.findViewById(R.id.***checkBox***);  
 checkBox.setTag(position);  
 *//verificare daca utilizatorul a dat click pe o optiune* **if**(MainActivity.*isSelected*)  
 {  
 *//apar checkBox-urile* checkBox.setVisibility(View.***VISIBLE***);  
 }  
 **else** {  
 *//checkBox-urile nu sunt vizibile* checkBox.setVisibility(View.***GONE***);  
 }  
 checkBox.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 *//definirea unei variabile care "retine" pozitia elementului selectat* **int** position = (**int**)buttonView.getTag();  
 *//verificare daca exista elemente selectate* **if**(MainActivity.*selectedItems*.contains(**optiuni**.get(position)))  
 {  
 MainActivity.*selectedItems*.remove(**optiuni**.get(position));  
 }  
 **else** {  
 MainActivity.*selectedItems*.add(**optiuni**.get(position));  
 }  
 *//afisare in dreptul meniului mesaj cu cate elemente sunt selectate* MainActivity.*actionMode*.setTitle(MainActivity.*selectedItems*.size()+ **" elemente selectate"**);  
 }  
 });  
 **return** row;  
 }

*//functie care parcurge elementele si le sterge* **public void** stergeOptiuni(List<String>items)  
 {  
 **for** (String item : items)  
 {  
 **optiuni**.remove(item);  
 }  
 *//actualizarea listei de optiuni dupa stergere* notifyDataSetChanged();  
 }  
}

În **MainActivity.java** este implementată toată logica din spatele meniului. Sunt definite o listă cu opțiuni, una cu elementele selectate, un ListView, dar și clasa anterioară, adică ListViewAdapter.

Am definit o funcție **getOptiuni()** în care practic populăm lista cu opțiuni. La crearea meniului (**onCreate()**) vor fi vizibile opțiunile. Reunim logica din **ListViewAdapter.java** cu cea din clasa curentă și permitem utilizatorului să aleagă mai multe obiecte prin apelul către **setChoiceMode()** cu argumentul **CHOICE\_MODE\_MULTIPLE\_MODAL.** În **setMultiChoiceModeListener()** sunt specificate acțiunile pentru meniul contextual prin logica implementată în interfața ActionMode.Callback.

Este de asemenea implementată interfața **ActionMode.Callback,** care este de fapt un “ciclu de viață” al meniului.

În **onCreateActionMode()** este “populat” meniul cu elementele din **meniu1.xml.** Acțiunea “se concentrează” asupra meniului (**actionMode**), iar **isSelected** este setat pe true pentru că după click devin vizibile ckeckBox-urile.

În **onActionItemClicked()** este definită logica din spatele acițonării butonului “delete” din meniu.

În **onDestroyActionMode()** meniul dispare (nu mai este în prim plan), checkbox-urile dispar, iar în **selectedItems.clear()** numărul de elemente selectate este practic șters din memorie și este astfel “resetat”, pornind de la 0 următoarea dată când va fi creat meniul.

**package** com.example.contextualactionmode2;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.ActionMode;  
**import** android.view.Menu;  
**import** android.view.MenuInflater;  
**import** android.view.MenuItem;  
**import** android.widget.AbsListView;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** ListView **listView**;  
 **private** ListViewAdapter **listViewAdapter**;  
 *//declararea listei de optiuni* **private** List<String> **options** = **new** ArrayList<>();  
 *//declararea listei de elemente selectate* **public static** List<String> *selectedItems* = **new** ArrayList<>();  
 *//declararea unei variabile care verifica daca tilizatorul a dat click pe optiunea dorita* **public static boolean** *isSelected* = **false**;  
 **public static** ActionMode *actionMode* = **null**;

@Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//se afiseaza optiunile* getOptiuni();  
 *//instantierea listView cu lista de optiuni* **listView** = findViewById(R.id.***listOptiuni***);  
 *//permiterea selectarii mai multor obiecte* **listView**.setChoiceMode(AbsListView.***CHOICE\_MODE\_MULTIPLE\_MODAL***);  
 *//asignarea logicii din interfata* **listView**.setMultiChoiceModeListener(**modeListener**);  
 *//instantierea clasei ListViewAdapter* **listViewAdapter** = **new** ListViewAdapter(**options**,**this**);  
 **listView**.setAdapter(**listViewAdapter**);  
 }  
 *//implementarea interfetei corespunzatoare meniului* AbsListView.MultiChoiceModeListener **modeListener** = **new** AbsListView.MultiChoiceModeListener() {  
 @Override  
 **public void** onItemCheckedStateChanged(ActionMode mode, **int** position, **long** id, **boolean** checked) {  
  
 }  
 *//logica din spatele crearii meniului* @Override  
 **public boolean** onCreateActionMode(ActionMode mode, Menu menu) {  
 *//adaugarea in meniu a butoanelor definite anterior* MenuInflater menuInflater = mode.getMenuInflater();  
 menuInflater.inflate(R.menu.***meniu1***,menu);  
 *//aparitia checkBox-urilor  
 isSelected* = **true**;  
 *actionMode* = mode;  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onPrepareActionMode(ActionMode mode, Menu menu) {  
 **return false**;  
 }  
  
 @Override  
 **public boolean** onActionItemClicked(ActionMode mode, MenuItem item) {  
 **switch** (item.getItemId()) {  
 *//ce se inatmpla atunci cand utilizatorul actioneaza butonul "delete"* **case** R.id.***optiuneDelete***:  
 **listViewAdapter**.stergeOptiuni(*selectedItems*);  
 *//inchiderea meniului* mode.finish();  
 **return true**;  
 **default**:  
 **return false**;  
 }  
 }  
  
 @Override  
 **public void** onDestroyActionMode(ActionMode mode) {  
 *//dispar checkBox-urile, iar numarul optiunilor selectate se "reseteaza"  
 isSelected* = **false**;  
 *actionMode* = **null**;  
 *selectedItems*.clear();  
  
 }  
 };  
  
 **private void** getOptiuni()  
 {  
 *//instantierea array-ului de optiuni* String[] items = getResources().getStringArray(R.array.***arrayOptiuni***);  
 *//parcurgerea tuturor elementelor si adaugarea lor in array-ul cu optiuni* **for**(String item : items)  
 {  
 **options**.add(item);  
 }  
 }  
}

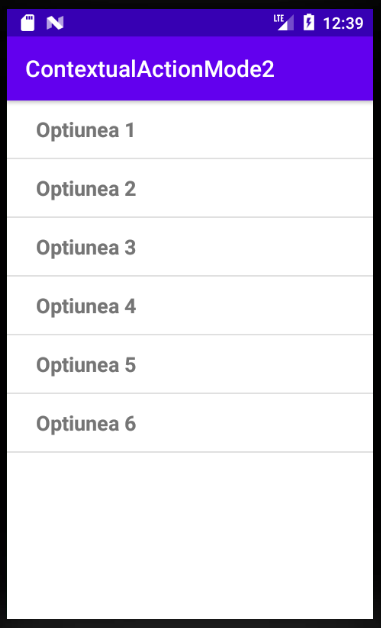
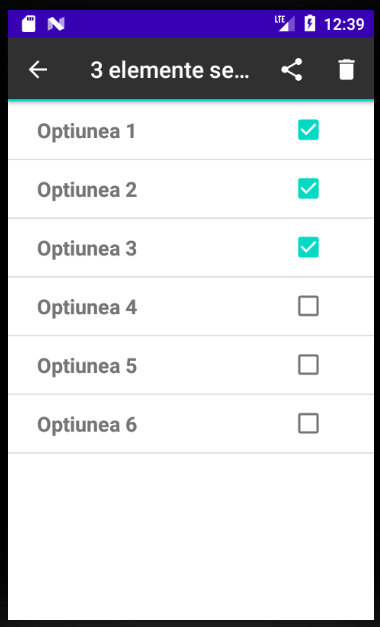
** **

Figura nr. 11 – Selectarea mai multor opțiuni și apariția meniului

Figura nr. 10 – Lista de opțiuni

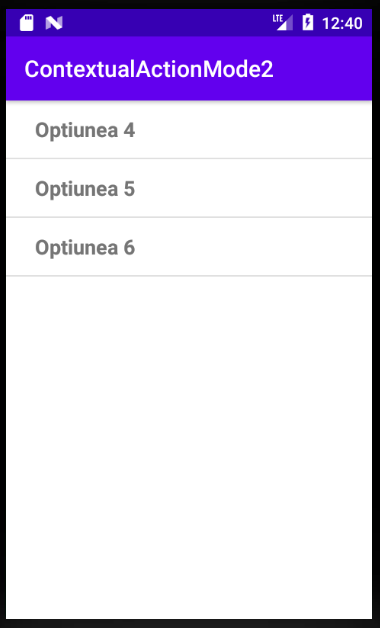
****

Figura nr. 12 – Ștergerea opțiunilor selectate