**Aplicații cu activități externe**

O aplicație Android poate să fie compusă din mai multe activități, alături de alte componente cum ar fi componentele layout, drawable etc. Fiecare activitate afișează diferite informații și/sau pune la dispoziția utilizatorului diferite componente grafice care pot permite o interacțiune a acestuia cu aplicația în sine. Multe aplicații standard implementează același principiu, de exemplu aplicații precum Gmail, Google+, Twitter etc., au mai multe activități care pot permite unui utilizator să trimită mesaje în diferite moduri.

O activitate din cadrul unei aplicații Android, poate sa comunice (să deschidă) atât o activitate din cadru său, cât și o activitate care aparține unei alte aplicații (standard), ceea ce poate conduce la dezvoltarea unei aplicații complexe ce conține un lanț de activități de tip **cross device**.

Într-o aplicație, mai multe activități pot comunica între ele cu ajutorul unui **intent** (se anuntă platforma Android de intenția de a se deschide o activitate). De exemplu, o aplicație Android poate să deschidă aplicația Gmail pentru a trimite un mesaj (fig. 1).

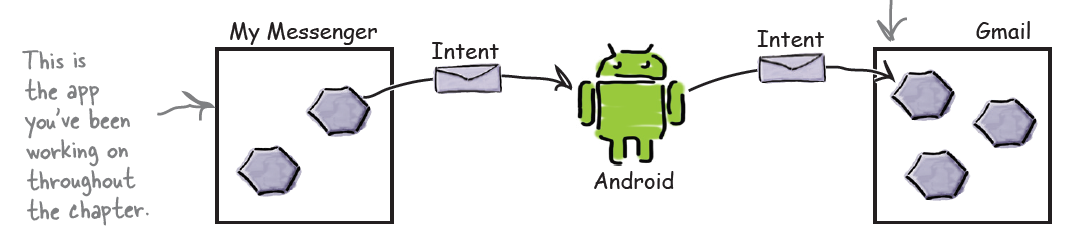


Figura 1 Intent extern

**Acțiuni (Actions) și intent-uri implicite**

În momentul în care o activitate anunță intenția de a deschide o activitate dintr-o alta aplicație de pe un device, mai întâi trebuie studiate următoarele aspecte:

* dacă activitatea (app) este disponibilă pe device-ul respectiv
* care este lista activităților care pot asigura funcționalitatea dorită (de exemplu, prin ce aplicații putem trimite mesaje)
* care este modalitatea de a utiliza activitatea cerută

Conceptul oferit de platforma Android pentru a studia aspectele de mai sus se numește **action.** Printr-o acțiune, platforma Android poate afla care sunt activitățile standard de pe device respectiv ce pot executa o anumită acțiune. De exemplu, platforma Android cunoaște care sunt toate aplicațiile de pe device care pot furniza operația de a trimite un mesaj.

**Aplicație practică:** Se propune dezvoltarea unei aplicații care permite editarea unui mesaj de către utilizator și trimiterea acestuia prin mai multe aplicații standard (mesagerie, Gmail etc).

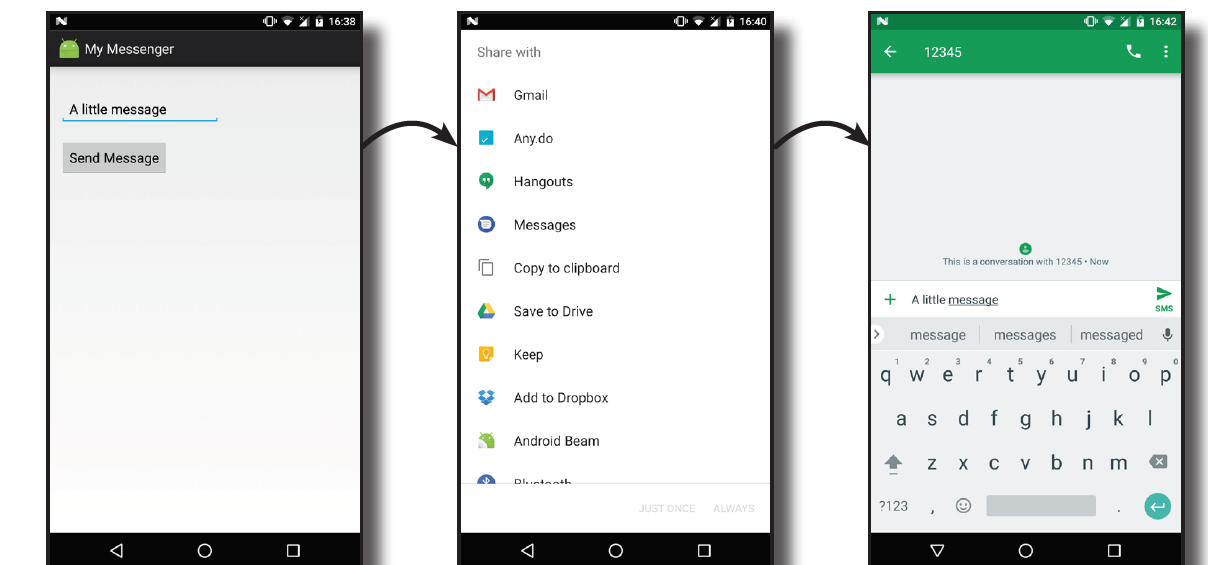


Figura 2 Intent extern

Pentru ca o activitate să deschidă o altă activitate din cadrul aceleași aplicații se instanțiază un obiect de tip Intent folosind un constructor care precizează printr-un argument activitatea vizată. Un astfel de intent se numește **intent explicit**.

Intent intent = new Intent(this, Target.class);

Dacă nu se cunoaște activitatea vizată, atunci intent-ul este unul **implicit**. În acest caz, se specifică platformei Android care este tipul acțiunii dorite (de exemplu trimiterea unui mesaj), iar acesta propune una sau mai multe activități dispuse în cadrul unor aplicațiilor standard de pe device-ul respectiv.

**Sintaxa:** Intent intent = **new Intent(action);**

Argumentul action specifică tipul de activitate dorit. În acest sens, platforma Android pune la dispoziție o serie de acțiuni/operații standard precum:

* ACTION\_DIAL - apel telefonic
* ACTION\_WEB\_SEARCH – căutare o pagină WEB
* ACTION\_SEND – trimiterea unui mesaj

**Exemplu:** Intent intent = new Intent(Intent.ACTION\_SEND);

Odată specificată acțiunea dorită, se pot adăuga intentului diferite informații care să fie transmise noii activități prin apelul metodei membre putExtra("message", value).

În exemplu nostru se dorește trimiterea unui text care să formeze un mesaj ce va fi trimis către un destinatar prin noua activitate.

**Exemplu:**

intent.setType(**"text/plain"**);  
intent.putExtra(Intent.***EXTRA\_TEXT***, mesajText);

* argumentul metodei setType specifică tipul de dată ce va fi prelucrat de către activitatea vizată (MIME data-type text/plain)
* argumentul messageText al metodei putExtra specifică mesajul care va fi transmis

**Observație:** pentru informații suplimentare se poate apela metoda putExtra de mai multe ori. De exemplu, dacă se dorește specificarea unui subiect al mesajului ce va fi trimis pe poate folosi mesajul:

intent.putExtra(Intent.EXTRA\_SUBJECT, subject);

**Bara de Acțiune (ActionBar)**

Bara de acțiune este un element de desing popular pentru aplicațiile mobile, deoarece oferă o descriere vizuală a structurii aplicații, precum și elemente interactive care sunt familiare unui utilizator (cautare, setare etc).

Funcționalitățile principale ale unui action bar sunt:

* oferă un spațiu dedicat pentru a îi oferii utilizatorului locația în aplicație (numele aplicației sau numele activității curente)
* oferă acces câtre acțiuni importante, precum căutarea, setari etc
* oferă suport pentru navigare și comutare (file, listă derulantă)

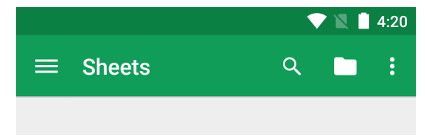


Figura 1 Action bar

În scopul de a adăuga o bară de acțiune aplicaței, se definește o **temă dedicată** care include app bar. Pentru a defini o temă se pot utiliza două modalități:

1. Definirea unei teme într-o manieră hard codată

* în fișierul AndroidManifest.xml

<application

.....  
 android:theme="@style/Theme.AppCompat.Light.DarkActionBar"

>

1. Aplicarea unei teme utilizând resursa style

* în directorul de resurse **res/theme**, fișierul theme.xml conține detalii despre tema selectată
* **<style name**="AppTheme" parent="**Theme.MaterialComponents.DayNight.DarkActionBar**">  
   *<!-- Primary brand color. -->* <item name="colorPrimary">@color/purple\_500</item>  
   <item name="colorPrimaryVariant">@color/purple\_700</item  
   *<!-- Customize your theme here. -->***</style>**

<**style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar"**>  
 *<!-- Customize your theme here. -->  
 <!-- This code is for changing the color of the bar. -->  
 <!-- Type your colour code which you want to set in colorPrimary item -->* <**item name="colorPrimary"**>#0F9D58</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
</**style**>

* Fiecare stil are un nume care este utilizat în fișierul **AndroidManifest.xml** prin proprietatea:

**"@style/AppTheme”**

* Atributul **parent** specifică de unde este preluat stilul selectat, de exemplu "**Theme.AppCompat.Light.DarkActionBar**", care oferă aplicației un fundal deschis la culoare (alb) cu un app bar negru (încercați mai multe teme, precum Theme.AppCompat.Light, Theme.AppCompat.Light.NoActionBar, Theme.AppCompat.NoActionBar)

Observație: **Orice temă poate fi modificată, de exemplu se pot modifica culorile pentru fundalul aplicatiei, a barei de actiune etc.**

O paletă de culori se poate defini în fisierul de resurse **colors.xml**:

*<!-- color for the app bar and other primary UI elements -->*<**color name="colorPrimary"**>#3F51B5</**color**>  
  
  
*<!-- a darker variant of the primary color, used for  
 the status bar (on Android 5.0+) and contextual app bars -->*<**color name="colorPrimaryDark"**>#303F9F</**color**>  
*<!-- a secondary color for controls like checkboxes and text fields -->*<**color name="colorAccent"**>#FF4081</**color**>

<**color name="blueColor"**>#3498db</**color**>  
<**color name="colorPrimary"**>#3F51B5</**color**>  
<**color name="colorPrimaryDark"**>#303F9F</**color**>  
<**color name="colorAccent"**>#FF4081</**color**>

**ToolBar**

**Adăugarea unui app bar se realizează într-o manieră simplă**, dar are dezavantajul de nu include toate proprietățile noi adăugate. O altă modalitate de a adăuga în aplicație un app bar este utilizarea componentei grafice Toolbar.

**Adăugarea unui Toolbar se realizează prin următoarele etape:**

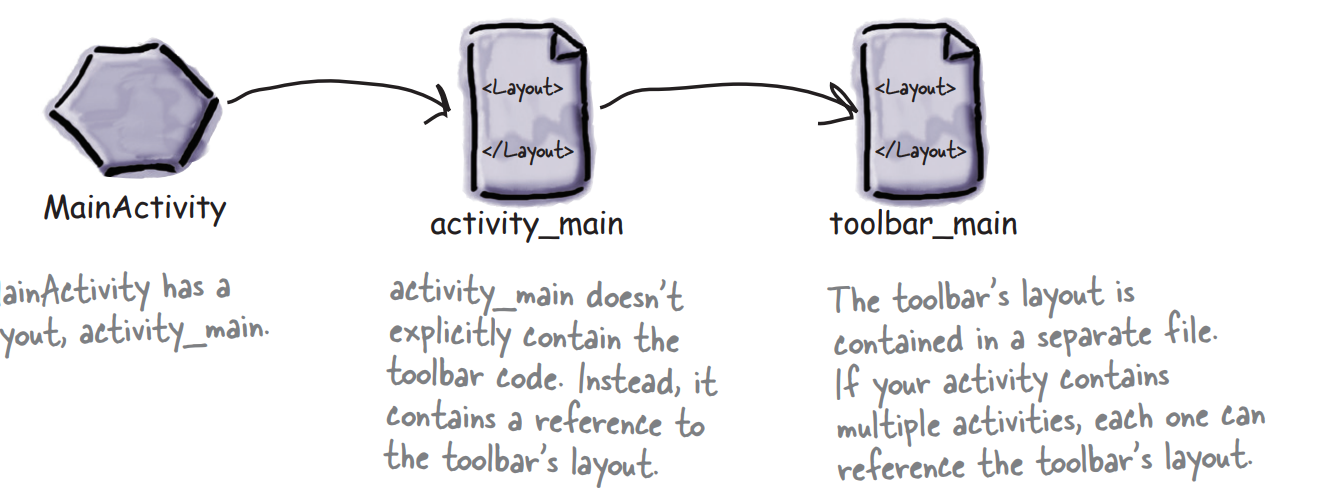
1. Se alege o temă care nu presupune adăugarea unui app bar, în fișierul theme.xml

<style name="Theme.TestActionBar" parent="Theme.AppCompat.Light.NoActionBar">

1. Se adaugă în fișieul layout.xml componenta Toolbar

<androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="409dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:background="?attr/colorPrimary"  
 android:minHeight="?attr/actionBarSize"  
 android:theme="?attr/actionBarTheme"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />

**Observație**: componenta Toolbar se poate insera într-un layout separat, astfel încât să fie inserat ulterior în mai multe activități ale aplicației.



P1. Se definește componenta grafică toolbar într-un fișier xml

**File -> New -> Layout resource file**

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.appcompat.widget.Toolbar  
xmlns:android="http://schemas.android.com/apk/res/android"  
android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"  
android:background="?attr/colorPrimary"/>

P2. În fișierul layout al activității se folosește proprietate include layout pentru a include layoutul ce conține toolbar

<include  
 layout="@layout/toolbar\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/toolbar"  
 />

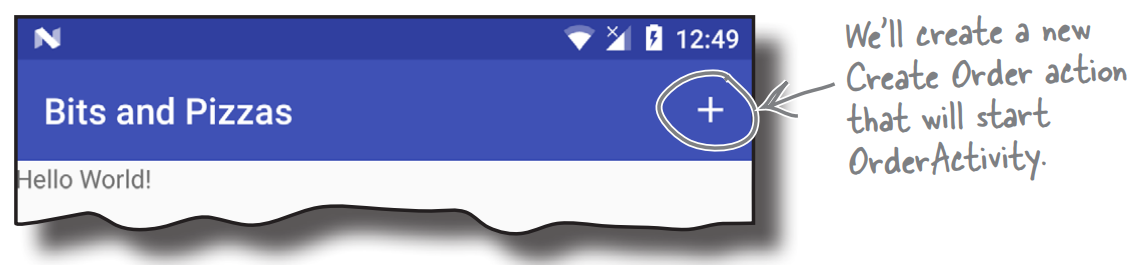
P3. Accesarea componentei toolbar din fișierul java

Toolbar toolbar = (Toolbar) findViewById(R.id.*toolbar*);  
**setSupportActionBar(toolbar);**

**Definirea acțiunilor pentru un app bar/toolbar**

În majoritatea aplicațiilor este nevoie să adăugam acțiuni în bara de aplicații. Acestea pot fi butoane sau text care prin interacțiunea cu utilizatorul conduc la un eveniment. **De exemplu, deschiderea unei noi activități.**

Aplicație practică: definiți un toolbar de forma de mai jos, care conduce la deschiderea unei noi activități



P1. **Crearea unei noi activități OrderActivity**

**P2. Inlcuderea în layout-ul activității noi a componentei toolbar definită în layoutul anterior**

<include  
 layout="@layout/toolbar\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/toolbar"  
 />

P3. Accesarea componetei Toolbar în fișierul java

Toolbar toolbar = (Toolbar) findViewById(R.id.*toolbar*);  
setSupportActionBar(toolbar);

P4 În fișierul de resurse values se definește o resursă de tip String pentru numele activității în scopul de a schimba eticheta (textul) din componenta Toolbar, în raport cu denumirea activității curente

<string name="create\_order">Create Order</string>

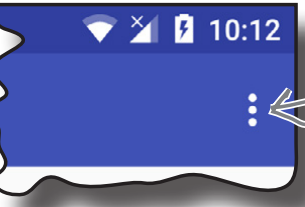
P4. Definirea unui text pentru bara de acțiune

După cum a fost specificat sus, textul care se afișeazp în bara de aplicații presupune utilizarea atributul etichetă din fișierul AndroidManifest.xml.

<activity  
 android:name=".OrderActivity"  
 android:label="@string/create\_order"  
 android:exported="false">

P5. Adăugarea unei resurse unei resurse pentru o acțiune

Pentru a adăuga o acțiune la o bară de aplicații, în general, acesteia i se atribuie o **pictogramă** și un scurt titlu de text. Pictograma este de obicei afișată dacă acțiunea apare în zona principală a barei de aplicații. Dacă acţiunea nu se potrivește în zona principală, este mutată automat în zona de afișare out (cele 3 puncte), iar titlul apare în schimb.



Pentru a adauga un text, se definește o resursă de tip String în fișierul values.xml

<string name ="create\_order\_title">Create Order</string>

**Definirea unui meniu**

Definirea unui meniu în bara de acțiune se realizează într-un director separat, în cadrul directorul de resurse res. Astfel, se creează o referință către resursa meniu în fișierul R, care va permite accesarea ulterioară a sa.

res - > Android resourse file- >resurse type Menu (in directorul menu)

În documenul .xml se definesc opțiunile unui meniu prin proprietatea item, de exemplu:

<**menu xmlns:android="http://schemas.android.com/apk/res/android"**>  
  
<**item android:title="Settings" android:id = "@+id/settings"**>

<**item android:title="Help" android:id = "@+id/help"** ></**item**>item>  
  
</**menu**>

Adăugarea efectivă a unui meniu în bara de acțiune a unei activități se realizează prin adăugarea metodei onCreateOptionsMenu() în clasa care modelează activitatea respectivă.

**public boolean** onCreateOptionsMenu(Menu menu) {  
 MenuInflater menuInflater = getMenuInflater();  
 menuInflater.inflate(R.menu.***main\_menu***, menu);  
  
 **return super**.onCreateOptionsMenu(menu);  
}

Clasa MenuInflater permitea definirea/manipularea unui meniu cu referința definită în fișierul R, accesarea efectivă realizându-se prin apelul metodei getMenuInflater().

Selecția de către utilizator a unei opțiuni impune tratarea evenimentului prin implementarea metodei onOptionsItemSelected()

Exemplu:

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_create\_order:

**Intent intent = new Intent(this, OrderActivity.class);**

**startActivity(intent);**

return true;

default:

return super.onOptionsItemSelected(item);

}

}

**Aplicație practică**: Dezvoltați o aplicație Android cu o activitate ce conține în bara de acțiune un meniu cu două opțiuni, repectiv Settings, Help, și realizează afișarea unui mesaj de tip Toast ce cuprinde denumirea opțiuniii selctate.

**Aplicații propuse**

Dezvoltați aplicația Android StarBuzz (cursul 5) care conține în meniul ActionBar informații despre produsele comercializate (Drinks, Food, Stores). La selecția unei opțiuni se deschide o activitate ce cuprinde o listă cu produsele comercializate din fiecare categorie. De exemplu, dacă utilizatorul selectează opțiunea Drinks se va deschide o activitate ce curinde un logo și o listă cu cele trei opțiuni (Capuccino, Latte, Filter).

Materiale suplimentare:

<https://developer.android.com/guide/components/intents-filters>

<https://www.vogella.com/tutorials/AndroidIntent/article.html>

<https://www.vogella.com/tutorials/AndroidActionBar/article.html>

<https://webplatform.github.io/docs/concepts/Internet_and_Web/mime_types/>