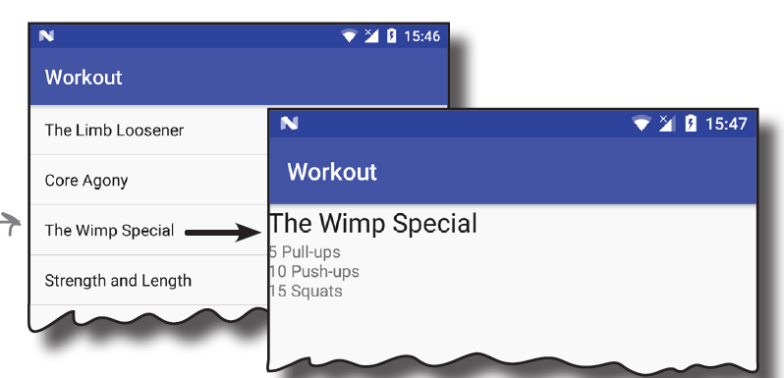
**FRAGMENTE**

O facilitate importantă a unei aplicații Android este dată de faptul ca o aplicație, odată dezvoltată, poate rula pe dispozitive mobile care pot avea rezoluții diferite. Deși, o aplicație trebuie să ruleze în aceași manieră pe orice dispozitiv, există situații în care anumite activități ale aplicației pot fi modificate în raport cu rezoluția dispozitivului mobil. De regulă, o aplicație care rulează pe un telefon mobil, poate să aibă o structură diferită și chiar un design diferit dacă rulează pe o tabletă.

De exemplu, aplicația de mai jos dacă rulează pe un telefon mobil poate să aibă următoarea structură:



Aceeași aplicație dacă rulează pe un dispozitiv de tip tabletă va avea următorul desing:



Se poate observa din figura de mai sus cum detaliile despre un antrenament pot fi listate chiar in activitatea principală, deci nu ar mai fi nevoie de o altă activitate suplimentară. Cu alte cuvinte, același cod (cel are afișează detaliile despre un antrenament selectat) trebuie sa ruleze în activități diferite. În consecintă, este nevoie să se implementeze de două ori același cod!!! Pentru a evita acest lucru, platforma Android introduce noțiunea de **fragment**.

**Fragmentul** reprezintă o porțiune de cod ce poate fi reutilizat de către diferite activități. Un fragment este manageriat prin propriul layout, are un ciclu de viață diferit de cel al activității ce îl încapsulează și poate trata propriile evenimente.

**Ciclu de viață al unui fragment**

Ciclul de viață al unui fragment în Android descrie secvența de stări prin care trece un fragment de la crearea sa până la distrugerea sa.

A diagram of a process

Description automatically generated

1. **Instantierea (Creation)**

* În acest stadiu, fragmentul este creat.
* Se poate utiliza metoda **onCreate()** pentru inițializări generale, cum ar fi inițializarea datelor.

1. **Alăturarea (Attachment)**

* Fragmentul este alăturat unei activități utilizând metoda **onAttach().**
* În acest moment, fragmentul poate accesa contextul activității gazdă.

1. **Crearea interfeței dutilizator (View Creation)**

* Fragmentul își creează interfața de utilizator, în metoda **onCreateView().**
* În această etapă, fragmentul poate să își inițializeze elementele interfeței de utilizator.

1. **Pregătirea pentru interacțiune (Preparation for Interaction)**

* În această fază, fragmentul își pregătește interacțiunea cu utilizatorul.
* Metoda onActivityCreated() poate fi utilizată pentru activități precum interacțiunea cu activitatea gazdă.

1. **Activare (Active)**

* Fragmentul devine activ în această fază.
* Metoda **onStart()** este apelată.

1. **Interacțiune (Interaction)**

* În această fază, fragmentul interacționează cu utilizatorul.
* Evenimente precum atingerea ecranului sau apăsarea butoanelor pot avea loc în această etapă.

1. **Pauză (Pause)**

* Fragmentul poate intra în starea de pauză atunci când o altă activitate sau fragment este pus în fața sa.
* Metoda onPause() este apelată.

1. **Oprire (Stop)**

* În această fază, fragmentul poate fi oprit sau devine invizibil pentru utilizator.
* Metoda **onStop()** este apelată.

1. Detașare (Detachment):

* Fragmentul este detașat de la activitatea gazdă utilizând metoda **onDetach().**

1. Distrugere (Destruction)

* Fragmentul este distrus în această fază.
* Metoda **onDestroy()** este apelată.

**Adăugarea unui fragment în cadrul proiectului**

Operația de adaugarea unui fragment este similară cu cea de adăugare a unei activități New...→Fragment→Fragment (Blank). Pentru fiecare fragment se introduce denumirea sa și se specifică dacă se generează, implicit sau nu, un fișier layout corespunzător.

Codul Java corespunzător unui fragment descrie comportamnetul acestuia.

public class WorkoutDetailFragment **extends Fragment** {

@Override

public View **onCreateView**(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment\_workout\_detail, container, false);

}}

Pentru a crea un fragment se exinde clasa Fragment și se implementează metoda onCreateView() ce va fi apelată la fiecare încluziune a frgmentului în cadrul unei activități.

public View onCreateView(LayoutInflater inflater,

ViewGroup container,

Bundle savedInstanceState) {

**return inflater.inflate(R.layout.fragment\_workout\_detail,**

**container,false);**

}

**Adăugarea fragmentului într-o activitate**

Una dintre modalitățile uzuale de a adauga un fragment într-o activitate (layout) este data de definirea unui layout dedicat **FrameLayout.**

FrameLayout definește un container pentru componente suprapuse (overlapping).

**FrameLayout  
 android:id="@+id/frameLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="40dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"/>**

**Gestionarea unui fragment în cadrul unei activități**

Un fragment (componentă reutilizabilă a interfeței) poate fi adăugat înlocuit, atașat sau detașat în cadrul ciclului de viață a activității gazdă. Android SDK oferă o clasa standard **FragmentManager** care gestionează fragmentele în cadrul unei activități.

Un exemplu simplu de obținere a unui FragmentManager într-o activitate ar fi utilizarea metodei getSupportFragmentManager()

FragmentManager fm = getSupportFragmentManager();

Operațiile specifice care se pot executa asupra unui fragment, cum ar fi atașarea, înlocuirea, detașarea, ascunderea se numesc tranzacții. O tranzacție este modelată prin clasa standard Android.

FragmentTransaction

FragmentTransaction ft = fm.beginTransaction();  
  
*// Inlocuirea unui fragment cu un nou fragment*ft.replace(R.id.*frameLayout*, fragment);  
ft.commit();