1 Fragments

1.1 Was sind Fragments?

Ein Fragment stellt einen Teil der Benutzeroberfläche einer Activity zur Verfügung, dabei kann man mehrere Fragments in einer Activity verwenden und diese zur Laufzeit auswechseln. Da Fragments in mehreren Activities wiederverwendet werden können, müssen Ansichten wie Detailviews oder Listen nur einmal programmiert werden und können überall eingesetzt werden. Fragments werden ab Android 3.0 (API level 11) zur Verfügung gestellt. [Android Developers [1][2]]

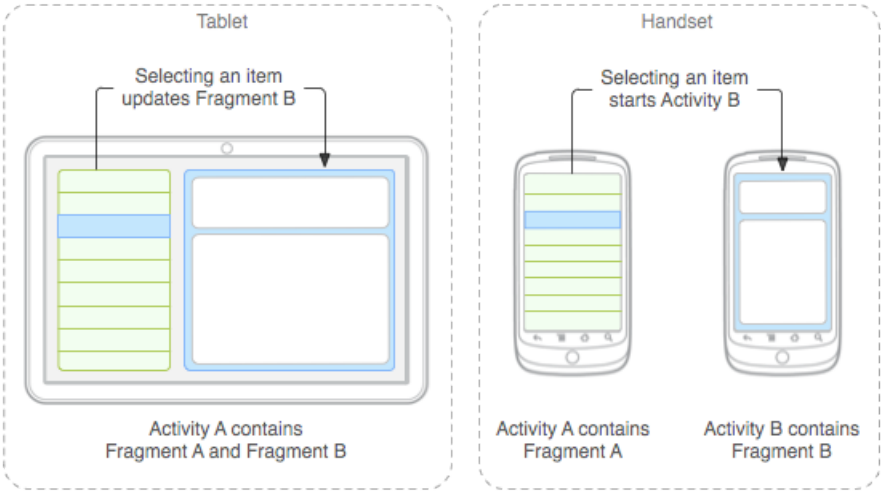

Abbildung 1: Zum Beispiel kann man mithilfe von Fragments Views erstellen, die sowohl auf einem Tablet als auch auf einem Handy ein optimales Benutzerinterface anbieten. [Android Developers [2]]

1.2 Der Lifecycle

Ein Fragment ist immer in eine Activity eingebunden und daher direkt vom Lifecycle der übergeordneten Activity abhängig. Wird die übergeordnete Activity pausiert oder zerstört, werden auch alle zugeordneten Fragments pausiert oder zerstört.

```
1 @Override
2 public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState) {
3         super.onCreate(savedInstanceState);
4         return inflater.inflate(R.layout.exampleFragment,
    container, false);
5     }
```

Das Aufbauen der Benutzeransicht erfolgt in der onCreateView() Methode eines Fragments. In diesem Fall wird das layout XML file exampleFragment angezeigt.

1.3 Fragment Transactions

Mittels Transaktionen lassen sich Fragments hinzufügen, entfernen oder ersetzen. Es werden mehrere dieser Aktionen hintereinander abgesetzt und zusammen nach einem commit() ausgeführt. Ein Fragment kann mittels addToBackStack() auch zum BackStack hinzugefügt werden, um dadurch, ähnlich wie bei Activites, Navigation mit dem BackButton zu ermöglichen. Dabei ist zu beachten, dass alle Aktionen vor einem commit() gemeinsam auf den BackStack gelegt werden und bei drücken des BackButtons alle gemeinsam aufgehoben werden. Wird addToBackStack() nicht aufgerufen, wird ein Fragment beim Schließen oder beim Wechseln auf ein anderes Fragment zerstört und kann nicht mehr aufgerufen werden.

```
1 Fragment exampleFragment = new ExampleFragment();
2 FragmentManager fragmentManager = getFragmentManager();
3 FragmentTransaction fragmentTransaction = fragmentManager
4   .beginTransaction();
5 fragmentTransaction.addToBackStack(null);
6 fragmentTransaction.replace(R.id.fraPlace, exampleFragment);
7 fragmentTransaction.commit();
```

Hier wird ein ExampleFragment erstellt, man ersetzt das Fragment das sich aktuell im FrameLayout R.id.fraPlace befindet mit dem erstellten exampleFragment und fügt es zum Backstack hinzu.

1.4 Verwendung von Fragments

In dieser Arbeit werden Fragments verwendet, um die Benutzeransichten, ausgenommen des NavigationDrawers, anzuzeigen. Dabei wird ein FrameLayout im Layoutfile der MainActivity durch ein Fragment mittels der displayView() Methode ersetzt. Navigation durch diese Fragments wird mittels den Buttons im FMain oder dem NavigationDrawer ermöglicht. Zusätzlich werden verschachtelte Fragments verwendet um komplexere Ansichten darzustellen. In diesem Fall werden in den layout XML files der Fragments ein oder mehrere FrameLayouts erstellt und die verschachtelten Fragments werden dann in diesen neuen FrameLayouts angezeigt.

```
1 private void displayView(Fragment fragment) {
2         FragmentManager fragmentManager = getFragmentManager();
3         FragmentTransaction fragmentTransaction = fragmentManager
4           .beginTransaction();
5         fragmentTransaction.addToBackStack(null);
6         fragmentTransaction.replace(R.id.fraPlace, fragment);
7         fragmentTransaction.commit();
8     }
```

Methoden wie diese werden immer wieder verwendet um die Fragment Transactions an einem Ort zusammenzufassen und den Code so lesbarer zu machen.

```
1 <FrameLayout
2         android:id="@+id/fraPlace"
3         android:layout_width="match_parent"
4         android:layout_height="match_parent" />
```

So sieht ein FrameLayout aus das wir immer wieder verwenden, um darin Fragments anzuzeigen.

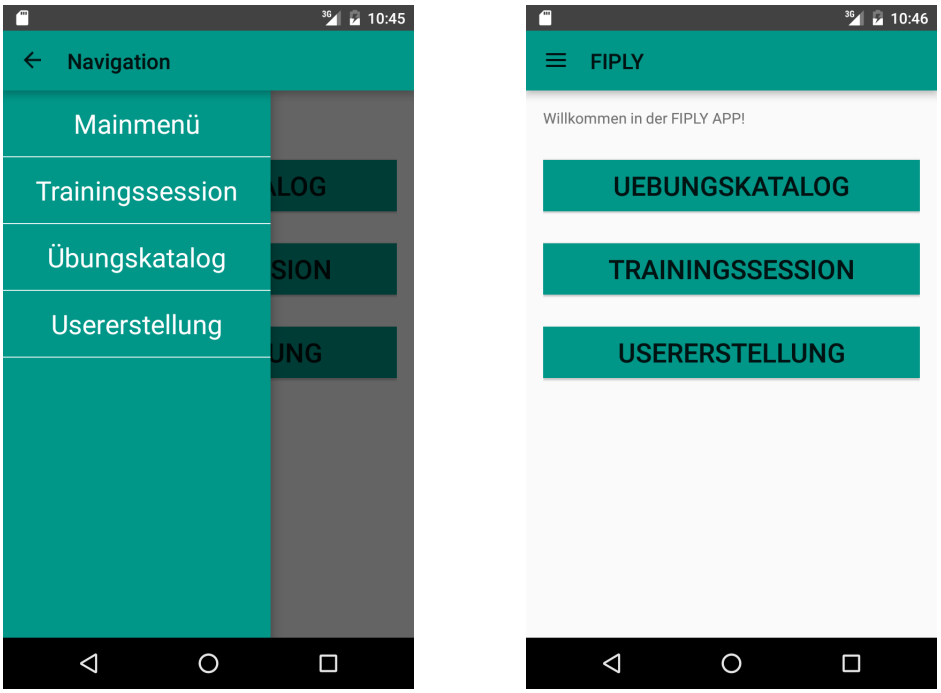

Abbildung 2: Bild des NavigationDrawers und des FMains

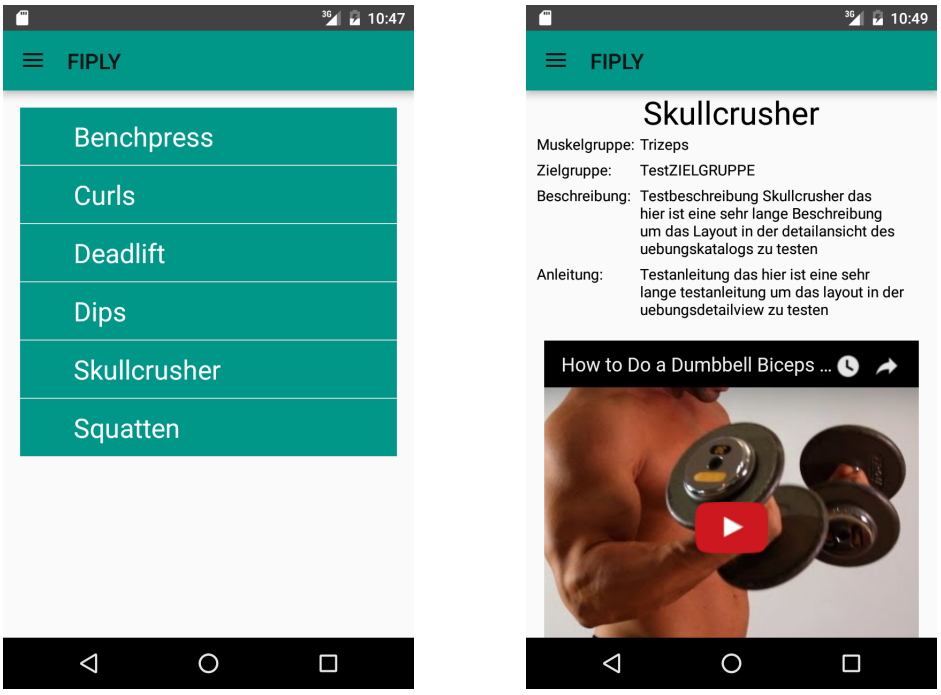

Abbildung 3: Bei Klicken eines Elements in der ListView wird die zugehörige DetailView aufgerufen