FIPLY

Daniel Bersenkowitsch, Andreas Denkmayr, Gerald Irsiegler

19. Februar 2016

Inhaltsverzeichnis

1 Designrichtlinien **4**
- 1.1 Der Splashscreen . . . 4
- 1.2 Animationen und Transactions . . . 4
- 1.3 Navigation . . . 6
- 1.4 Formulare . . . 7
- 1.5 Datenausgabe . . . 8
- 1.6 Style . . . 10

2 Fragments **11**
- 2.1 Was sind Fragments? . . . 11
- 2.2 Der Lifecycle . . . 12
- 2.3 Fragment Transactions . . . 12
- 2.4 Verwendung von Fragments . . . 13

3 Datenanbindung **15**
- 3.1 Verwendungszweck . . . 15
- 3.2 Vorher . . . 15
- 3.3 Nacher . . . 16
- 3.4 Anwendung . . . 17
  - 3.4.1 In der .xml Datei . . . 17
  - 3.4.2 In der .java Klasse . . . 19
  - 3.4.3 Quellen: . . . 20

4 Bundles **21**

5 Permissions **22**

6 Musik **23**
6.1 Lokalisierung der Musikdateien . . . 23
6.2 Verwalten von Playlists . . . 24
6.3 Abspielen der Playlists. . . . 26
6.4 MusicControls . . . 28

7 NavigationDrawer **29**

8 Commercialization **30**
8.1 Donations . . . 30
8.1.1 Flattr . . . 30
8.1.2 PayPal . . . 31
8.1.3 Zweite kostenpflichtige App . . . 31
8.2 Freemium . . . 32
8.2.1 Consumable Items . . . 32
8.2.2 Non-Consumable Items . . . 32
8.2.3 Subscriptions . . . 32
8.3 Advertising mit AdMob . . . 33
8.3.1 Banners . . . 33
8.3.2 Interstitials . . . 33
8.4 PaidVersion . . . 34
8.4.1 Beschreibung . . . 34
8.4.2 Vorteile . . . 34
8.4.3 Nachteile . . . 34
8.4.4 Preis . . . 34
8.4.5 Google-Play-Store . . . 34
8.5 Freemium . . . 35
8.5.1 Beschreibung . . . 35
8.5.2 Vorteile . . . 35
8.5.3 Nachteile . . . 35
8.6 SellApp . . . 36
8.6.1 Beschreibung . . . 36
8.6.2 Vorteile . . . 36
8.6.3 Nachteile . . . 36
8.6.4 Probleme/Schwierigkeiten . . . 36
8.7 Promotion . . . 37
8.7.1 Website . . . 37
8.7.2 Social Media Reputation . . . 38
8.7.3 Presse . . . 39
8.7.4 Konteste . . . 39
8.7.5 Persöhnliche Werbung . . . 39

8.7.6 Reklame 40
8.7.7 In Appstore Optimierung (IAO) 40
8.7.8 Fazit 41

9 Der Trainingsplan 42
9.1 Vorwort 42
9.2 Was ist Was? 42
9.3 Phase 1: Allgemein 43
9.4 Phase 2: Kraftausdauer (Gesundheit) 44
9.5 Phase 2: Muskelaufbau
Phase 3: Kraftausdauer 45
9.6 Phase 2: Maximalkraft
Phase 3: Muskelaufbau 46
9.7 Phase 3: Maximalkraft 47
9.8 Visualisierung 47
9.9 Mobilisation 48

1 Designrichtlinien

Designrichtlinien sind ein wichtiges Element aller Applikationen bei denen der kommerzielle Erfolg stark vom Userinterface und dem visuellen Auftritt abhängt. Das dies bei unseren Zielsetzungen der Fall ist, ist es uns ein großes Anliegen, dass wir uns als Team im Vorhinein über das geplante Erscheinungsbild einig sind. Wir verwenden dabei Tipps und Designinstruktionen von Google, welche auf der Internetseite https://www.google.com/design/spec/material-design/introduction.html nachzulesen sind.

1.1 Der Splashscreen

Der Splashscreen ist der erste Screen den der User sieht. Hier wird meist das Logo dargestellt, während im Hintergrund die Daten geladen werden. Der Splashscreen sollte nicht zu lange sichtbar oder zu langweilig sein. Der Splashscreen kann folgendermaßen umgesetzt werden:

- ImageView: Mithilfe einer ImageView lässt sich ein einzelnes Bild darstellen und dieses Bild ist auch während der Laufzeit einfach austauschbar.
- VideoView: Ähnlich der ImageView lässt sich mithilfe einer VideoView ein einzelnes Video darstellen und man hat die Möglichkeit das Video zu starten oder zu pausieren.
- ProgressBar: Mithilfe einer ProgressBar ist es möglich Fortschritt grafisch darzustellen. Dabei ist auch möglich eine Cycling Animation anzuzeigen bei der dem User kein Fortschritt angezeigt wird, abgesehen davon dass die App etwas berechnet.

Der Splashscreen beinhaltet bei der FIPLY-App eine normale in der Mitte des Bildschirms zentrierte ImageView mit dem FIPLY-Logo, welche Präsenz 3500ms andauert wird. Der Hintergrund dieses Splashscreens ist weiß.

1.2 Animationen und Transactions

Eine Transition ist der Übergange zwischen zwei Activities. Der Übergang muss flüssig verlaufen und darf die User Experience nicht unterbrechen. Der User soll darauf aufmerksam gemacht werden wohin er seine Aufmerksamkeit richten sollte. Dies wird durch persistierende Objekte und präzise Bewegungen in den Übergängen erzielt. Diese Möglichkeiten stehen zur Verfügung:

- ViewSwitcher: Ein ViewSwitchers ist ein ViewAnimator mit dem man zwischen exakt 2 Views hin und herwechseln kann.
- TextSwitcher: Ein TextSwitcher ist ein ViewSwitcher der nur Text-Views auswechseln kann. Dieser wird verwendet um Labels zu animieren.
- ImageSwitcher: Ein ImageSwitcher ist ein ViewSwitcher der nur Bilder auswechseln kann. Dieser wird verwendet um das Wechseln von Bildern zu animieren.
- ViewFlipper: Ein ViewFlipper ist ein ViewAnimator mit dem man den Wechsel zwischen 2 oder mehr Views animieren kann.
- Fragment: Mithilfe von Fragments kann man mehrere Ebenen von Views innerhalb einer einzelnen Activity darstellen. Somit bildet ein Fragment ein auswechselbares Objekt, dass in anderen Activitys wiederverwendet werden kann.

Beschreibung unserer Umsetzung:

- Enter Transition: Die Enter-Transition besteht daraus, dass die aufgerufene Activity aus dem Button der Activity hervorgeht, welche sie ausgelöst hat. Dabei werden vergrößern sich jeweils die Eckpunkte eines Buttons und schmiegen sich der Form des Screens an, wobei die neue Activity gleichzeit durch einen transparenten Fade-In-Effekt erscheint. Es soll so wirken, als ob die geöffnete Activity aus dem Button hervorgeht. Wenn eine neue Activity nicht durch einen Button ausgelöst wird, wird diese einfach von der rechten Seite “eingeschoben”.
- Exit Transition: Die Exit-Transition wird genau so wie die Enter-Transition durchgeführt werden, nur umgekehrt. Wenn die Activity geschlossen werden soll, verschwindet sie mit einem transparenten Fade-Out Effekt und die Eckpunkte der View schmiegen sich zurück an den Button wodurch sie ausgelöst wurde, sodass die andere Activity wieder erscheint. Falls eine Activity nicht durch einen Button ausgelöst wurde und sie geschlossen werden soll, wird sie nach rechts aus dem Screen “hinausgeschoben”.
- Splashscreen Exit Transition: Der Splashscreen wird nach seiner Anzeigedauer durch einen transparenten Fade-Out Effekt verschwinden, sodass das Hauptmenü angezeigt wird.

1.3 Navigation

Als Navigation wird das Wechseln von einer Funktion in der App zur anderen bezeichnet. Dies wird meist durch Buttons erreicht und grafisch mit Transitions dargestellt. Die Buttons sollen auch ohne Text sprechend sein was sie bewirken bzw. wohin man mit ihnen navigiert. Dies kann durch bestimmte Zeichen auf den Buttons erreicht werden, z.B.: ein Zahnrad für die Einstellungen oder eine Lupe für eine Suchfunktion. Es sollen auch andere Methoden zur Navigation wie beispielsweise über den Bildschirm wischen verwendet werden. So lässt sich dies Umsetzen:

- Fragments: Mithilfe von Fragments kann man Navigation vortäuschen. Man wechselt dabei nicht zwischen den Activitys hin und her, sondern wechselt Teile der aktuellen Activitys aus um eine Navigation unnötig zu machen beziehungsweise vorzutäuschen.
- PopupMenu: Ein PopupMenu ist ein Menü das in einer View verankert ist und vor allem für Aktionen innerhalb einer Activity verwendet wird.
- ContextMenu: Ein ContextMenu ist ein Menü das vorallem Aktionen für ein spezielles Item bereitstellt. Ein ContextMenu wird vor allem bei ListViews oder GridViews eingesetzt um Aktionen wie Edit, Share oder Delete für einzelne Items bereitzustellen.
- Buttons: Ist ein Element einer View, das man entweder anklickt oder gedrückt hält, um Aktionen, z.B. Navigation, auszuführen.
- NavigationDrawer: Ist ein Element, dass in der linken Hälfte des Bildschirms angezeigt werden kann. Dies wird mittels einem DrawerLayout realisiert das eine ListView mit den Zielen der Navigation und ein anderes Layout mit dem Inhalt der Activity enthält.
- OptionsMenu: Ein OptionsMenu ist ein Menü das vor allem Aktionen für die aktuelle Activity bereitstellt. Bei älteren Geräten (API level 10 or niedriger) wird dieses OptionsMenu im unteren Teil des Bildschirms angezeigt. Ab API level 11 werden die Elemente des OptionMenus in der AppBar zur Verfügung gestellt.
- Zurücktaste am Gerät: Mithilfe der Zurücktaste, die in allen Androidgeräten verbaut ist, kann man auf die zuletzt besuchte Activity zurückspringen.

In der Applikation wird dies mittels der Technologie des OptionsMenu/AppBar und Buttons umgesetzt, um die Navigation der Applikation möglichst intuitiv für die Benutzer zu gestalten.

- NavigationDrawer: In dem NavigationDrawer werden Navigationen zu den Activities wie "Über uns" und "Profil bearbeiten" zur Verfügung gestellt, dessen Ansicht man in der Hauptansicht der Applikation auslösen kann. Ziel ist es, diesen NavigationDrawer auch innerhalb anderer Activities bereitzustellen.
- Buttons: Weiter Navigationen werden mittels Buttons ausgelöst:
  - Wechseln zwischen Hauptbildschirm und Trainingssessionanzeige.
  - Wechseln zwischen Hauptbildschirm und dem Menü zum generieren eines Trainingsplans.
  - Wechseln zwischen Hauptbildschirm und dem Statistikmenü.
  - Wechseln zwischen Hauptbildschirm und Trainingskatalogsmenü.
- Zurücktaste am Gerät: Mithilfe dieses Button wieder zur vorhergehenden Activity zurückspringen.

1.4 Formulare

Formulare sind der Ort an dem der User Daten in die App eingibt, wie beispielsweise beim Anlegen eines Userprofils. Formulare sollten so intuitiv wie möglich sein und so wenig wie möglich reine Texteingabe sein, wie z.B.: einen Slider anstatt eines Textfeldes für die Körpergröße. Manchmal ist jedoch eine Texteingabe unumgänglich, wie beispielsweise für die Namenseingabe. Behandlung mit Android Studio: (Necessary Evil-Steuerelemente)

- TextView: Eine TextView ist ein kompletter Texteditor, der in seiner Basisform allerdings kein bearbeiten des Textes erlaubt. Die Einsatzmöglichkeiten von TextViews sind beinahe unbegrenzt da Features wie Rechtschreibprüfung, Klickbare Links, Passwortfelder oder auch Eingabemethoden unterstützt werden.
- EditText: Ein EditText ist eine standardmäßig editierbare TextView.
- AutoComplete TextView: Mithilfe eines DropDownMenus werden dem User Vorschläge für die Textvervollständigung angezeigt die mittels Array definiert sind.
- Button: Ist ein Element einer View, das man entweder anklickt oder gedrückt hält, um Aktionen auszuführen.
- DatePicker: Mithilfe eines Datepickers kann man ein bestimmtes Datum über einen Spinner oder eine CalendarView auswählen.

- TimePicker: Mithilfe eines TimePickers kann man die Zeit auswählen.
- NumberPicker: Ein NumberPicker erlaubt dem User eine Zahl aus einer vordefinierten Zahlenmenge auszuwählen. Dazu werden 2 Buttons zur Verfügung gestellt die jeweils nach oben oder nach unten zählen.
- Switch: Ein Switch hat 2 Zustände (z.B: on and off) und der Zustand kann durch Klicken oder Ziehen geändert werden.
- CheckBox: Eine CheckBox hat 2 Zustände (checked and unchecked) und dieser kann durch Klicken geändert werden.
- RadioButton: Ein RadioButton hat 2 Zustände (checked and unchecked) und dieser kann durch Klicken geändert werden. Anders als bei einer RadioButtons werden meist in einer RadioGroup verwendet. Selektieren eines RadioButtons deselektiert alle anderen RadioButtons dieser Gruppe.
- ToggleButton: Ein ToggleButton hat 2 Zustände (z.B.: on and off) und der Zustand kann durch Klicken geändert werden. Der Zustand wird durch ein Aufleuchten des Buttons angezeigt.
- SeekBar: Eine SeekBar ist eine ProgressBar mit ziehbarem Slider um den Fortschritt zu setzen.
- RatingBar: Eine RatingBar ist eine SeekBar, die den Fortschritt in Sternen anzeigt. Der Fortschritt kann durch Klicken oder Ziehen verändert werden.
- Spinner: Ein Spinner ermöglicht dem User ein Kindelement aus einer Liste von Kindelementen auszuwählen. Dies wird mittels einem DropDownMenu realisiert.

Um den Benutzern eine komfortable Eingabe anzubieten, kommen überall dort intuitive Steuerelemente zum Einsatz, wo es möglich ist.

1.5 Datenausgabe

Datenausgabe ist der Teil der App wo dem User Information ausgegeben wird, wie beispielsweise der Übungskatalog. Die Datenausgabe soll so effizient wie möglich erfolgen. Suchfunktionen bzw. Filter sollen bei jeder Datenausgabe vorhanden sein. Dises Optionen stehen uns zur Auswahl:

- ProgressBar: Mithilfe einer ProgressBar ist es möglich Fortschritt grafisch darzustellen.
- RatingBar: Eine RatingBar ist eine SeekBar, die den Fortschritt in Sternen anzeigt. Optional kann der Fortschritt durch Klicken oder Ziehen verändert werden.
- ImageView: Mithilfe einer ImageView lässt sich ein einzelnes Bild darstellen und dieses Bild ist auch während der Laufzeit einfach austauschbar.
- VideoView/YoutubeAPI: Ähnlich der ImageView lässt sich mithilfe einer VideoView ein einzelnes Video darstellen und man hat die Möglichkeit das Video zu starten oder zu pausieren.
- MediaController: Ein MediaController ist eine View mit Steuerungselementen für den MediaPlayer. Der Controller sorgt dafür, dass die App synchron mit dem MediaPlayer läuft.
- CalendarView: Mithilfe einer CalendarView kann man ein Datum in einem Kalender darstellen oder auswählen.
- Clocks: Mithilfe einer TextClock kann man die aktuelle Zeit als formatierten String sowohl im 12-hour als auch im 24-hour Format anzeigen. Mithilfe einer AnalogClock kann man eine Uhrzeit an einer analogen Uhr anzeigen.
- StackView: Mithilfe einer StackView kann man immer ein Kindelement im Vordergrund anzeigen und sieht die restlichen Kindelemente dahinter angeordnet. Das angezeigte Element kann dabei durch jedes andere Kindelement ausgetauscht werden.
- ListView: Zeigt eine vertikale Liste von Elementen an.
- ExpandableListView: Zeigt eine vertikale Liste von Elementen an. Bei Klicken kann ein Element aufgeklappt werden um mehr Informationen anzuzeigen.
- WebView: Eine WebView kann Webseiten anzeigen ohne in den Webbrowser des Geräts wechseln zu müssen. In der WebView sind auch diverse Steuerelemente zur Navigation oder Textsuche enthalten.

Das ProgressBar Element kommt bei dem Anzeigen der Statistik zum Einsatz. Somit bekommen die User ein visuelles Feedback wie weit sie von ihrem

Ziel noch entfernt sind. Die VideoView/Youtube API wird beim Menüpunkt des Anzeigens der Details einer Übung verwendet. Jede Übung besitzt ein Vorschauvideo, welches durch dieses Element wiedergegeben wird. Die ExpandableListView ist die Anzeige des Übungskatalogs. Dieses Element beinhaltet alle zur Verfügung stehenden Übungen, welche durch Einstellungen gefiltert bzw. durchsucht werden können.

1.6 Style

Die gesamte Applikation bekommt harte Farben um Festigkeit zu symbolisieren. Die Hintergrundgfarbe aller Activities ist ein starkes Grau, die Entscheidung viel schlussendlich auf die Farbe mit dem Farbcode #424242. Alle Steuerelemente innerhalb der Applikation erhalten an den Eckpunkten eine 2px dünnen Konturradius mit unserer Hauptfarbe der Applikation, #009688. Um innerhalb der Elemente dem Farbthema treu zu bleiben, wie es von Google empfohlen wird, haben wir uns für #E0F2F1 entschieden. Diese Farbe wird als Hintergrundfarbe bei den Steuerelementen eingesetzt. Sie repräsentiert eine weiche Neutralität und übt dadurch keinen starken Kontrast mit den anderen Farben aus, wodurch den Usern ein visuell-angenehmes Farbenspektrum geboten wird.

Andreas Denkmayr 20. Dezember 2015

2 Fragments

2.1 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.

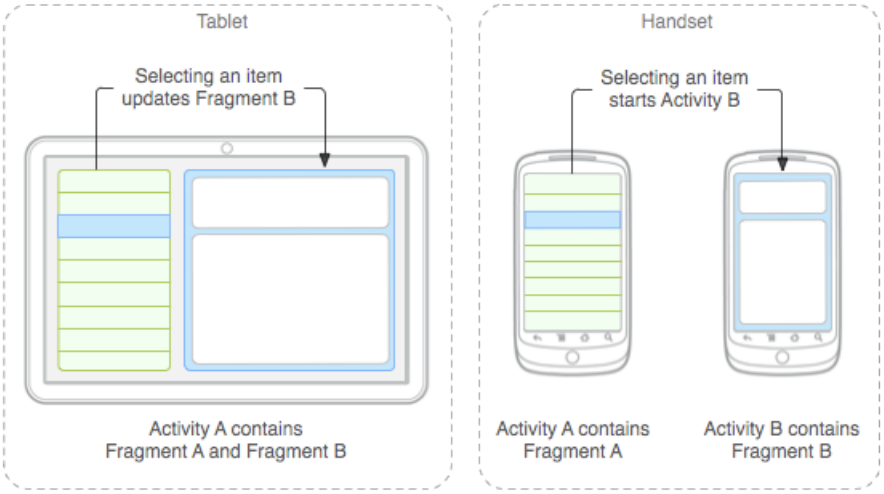


Abbildung 1: Beispiel man kann mithilfe von Fragments Views erstellen, die sowohl auf einem Tablet als auch auf einem Handy ein optimales Benutzerinterface anbieten

[1] [2]

2.2 Der Lifecycle

Ein Fragment ist immer eingebunden in eine Activity und ist direkt vom Lifecycle der übergeordneten Activity abhängig. Wird die übergeordnete Activity pausiert oder zerstört werden auch alle untergeordneten Fragments pausiert oder zerstört. Das Aufbauen der Benutzeransicht erfolgt in der onCreateView() Methode.

`In einem Fragment:`

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

2.3 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         FragmentManager fragmentManager = getFragmentManager();
2         FragmentTransaction fragmentTransaction = fragmentManager
    .beginTransaction();
3         fragmentTransaction.addToBackStack(null);
4         fragmentTransaction.replace(R.id.fraPlace, fragment);
5         fragmentTransaction.commit();
```

2.4 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 MFragment oder dem NavigationDrawer ermöglicht.

`MainActivity.java und MFragment.java:`

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

`activity_main.xml:`

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

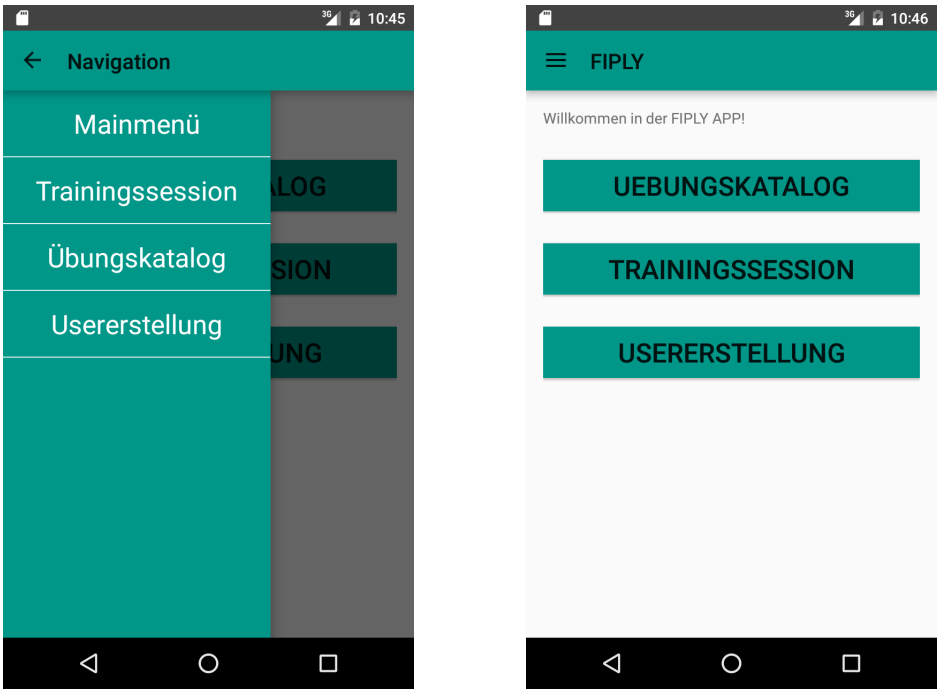


Abbildung 2: Bild des NavigationDrawers und des FMains

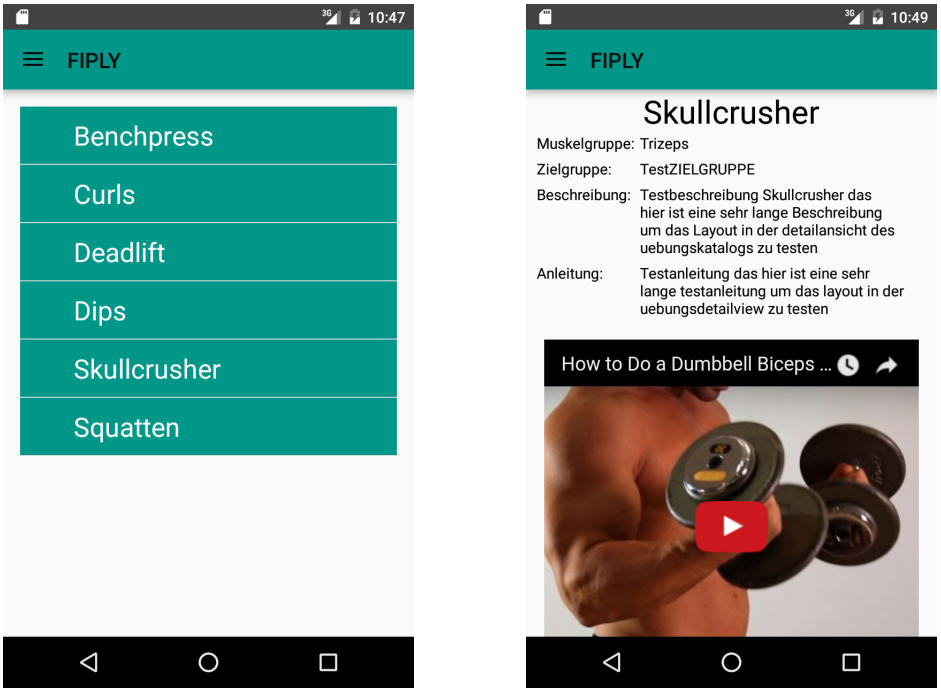


Abbildung 3: Bei Klicken eines Elements in der ListView wird die zugehörige DetailView aufgerufen

3 Datenanbindung

Als Datenbindung (engl. Data Binding) bezeichnet man die automatische Weitergabe von Daten zwischen Objekten. Typischerweise werden Daten aus einem Datenobjekt an ein Steuerelement der Benutzeroberfläche weitergegeben. Aber auch zwischen Steuerelementen ist Datenbindung in einigen Frameworks möglich. [1]
Beim Anzeigen von Daten in einer Listenansicht beispielsweise muss man das Steuerungselement nach jeder Veränderung der Daten aktualisieren. Wenn man aber die Technologie des Datenanbindung verwendet, braucht man nur die Objektliste an das Steuerungselement mit einem einfachen Zuweisungsbefehl anbinden. Dadurch erneuert sich die Listenansicht der Daten jedes Mal automatisch, sobald sich die Objektliste auch verändert.

3.1 Verwendungszweck

In erster Linie werden dem Programmierer dadurch viele Zeilen Code erspart. Datenanbindung trennt einen großen Teil des UI codes von den Aktivitäten und Fragmenten, wodurch eine bessere Übersicht über das Projekt verschafft wird. Zusätzlich, dadurch, dass die XML-Layoutdatei direkt auf die gebundenen Objekte und ihre Attribute zugreift, erspart man sich umständliche findViewById Codiereungen, die sehr performancelastig sind. [2]

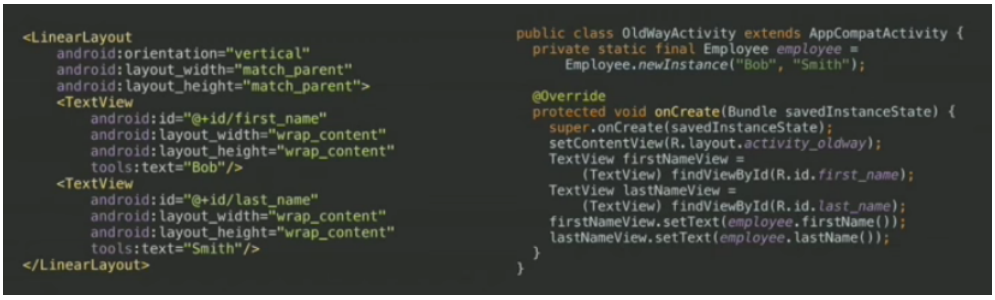
3.2 Vorher

Hier auf der untigen Grafik sehen wir eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität ohne Datenanbindung. Wir sehen ein LinearLayout mit zwei enthaltenen TextViews. Diesen wird in der Aktivität den Vornamen und den Nachnamen eines Employee Models zugewiesen. Ein Problem ist es, jedes Element eine ID zuweisen zu müssen. Wenn man jetzt mehrere Views mit unterschiedlichen Layout XML-Dateien und gleichnamigen IDs hat und man später die Refactor-Funktion verwenden will, benennt man alle neu, ohne das man es will. Man muss sich für jedes Element eine Unterschiedliche ID einfallen lassen, obwohl manche die selbe Funktion haben. Dadurch entstehen lange und unübersichtliche ID-Namen, die man sich

[1] https://www.it-visions.de/

[2] "Droidcon NYC 2015 - Data Binding Techniques"
https://www.youtube.com/watch?v=WdUbXWztKNY

nicht merken kann. Das Problem ist: Es muss immer darauf geachtet werden, keine doppelten IDs zu vergeben.



```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/first_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:text="Bob"/>
    <TextView
        android:id="@+id/last_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:text="Smith"/>
</LinearLayout>
```

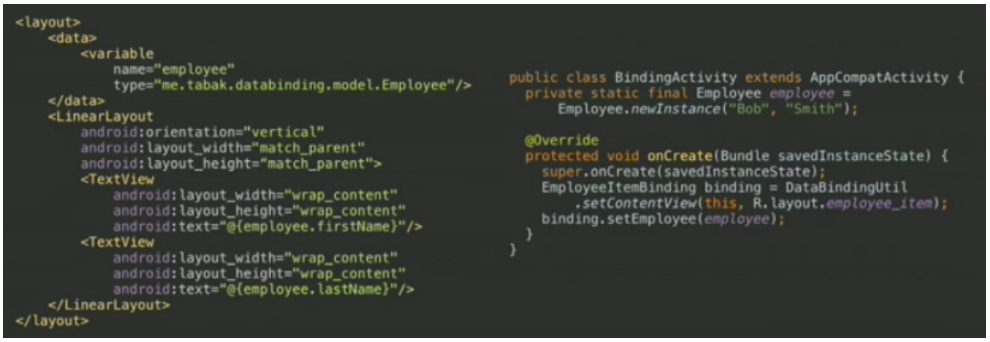
```
public class OldWayActivity extends AppCompatActivity {
  private static final Employee employee =
      Employee.newInstance("Bob", "Smith");

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_oldway);
    TextView firstNameView =
        (TextView) findViewById(R.id.first_name);
    TextView lastNameView =
        (TextView) findViewById(R.id.last_name);
    firstNameView.setText(employee.firstName());
    lastNameView.setText(employee.lastName());
  }
}
```

Abbildung 4: Screenshot des Codes wie er ohne Datenanbindung aussieht

Ein weitere Umständlichkeit ist es, für jedes Element ein findViewById-casting vornehmen zu müssen, wie wir es im Aktivitätencode vorfinden. Dieses Zugriffsverfahren ist, wie bereits oben erwähnt, unnötig performancelastig und kann vermieden werden.

3.3 Nacher



```
<layout>
    <data>
        <variable
            name="employee"
            type="me.tabak.databinding.model.Employee"/>
    </data>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{employee.firstName}"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{employee.lastName}"/>
    </LinearLayout>
</layout>
```

```
public class BindingActivity extends AppCompatActivity {
  private static final Employee employee =
      Employee.newInstance("Bob", "Smith");

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EmployeeItemBinding binding = DataBindingUtil
        .setContentView(this, R.layout.employee_item);
    binding.setEmployee(employee);
  }
}
```

Abbildung 5: Screenshot des Codes wie er mit Datenanbindung aussieht.

Auf dieser Grafik sehen wir nun eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität mit der Verwendung von Datenanbindung. Der Unterschied ist, sich keine einzelnen IDs für jedes vorkommende Element mehr ausdenken zu müssen. Ein großer Vorteil ist es, keine findViewById aufrufe mehr machen zu müssen. Man übergibt nur noch das zu bindende Objekt, an dessen Attribute sich die in der Layoutdatei befindenen Elemente orientieren. Ein weitere Pluspunkt: Man erkennt sofort für was welches Steuerelement zuständig ist. Ein Programmierer der sich gerade in ein

Projekt einarbeitet erkennt sofort, dass bei der ersten TextView der Vorname eines Employeeobjektes angezeigt wird und bei der anderen der Nachname.[3]

3.4 Anwendung

Bei der verwendung von Datenanbindung muss man darauf achten, eine aktuelle Gradle version zu benutzen (min. 1.3). Zusätzlich muss man in der build.gradle (Module App) Datei innerhalb des android{}-Bereichs dataBinding auf enabled=true (dataBinding{enabled=true}) setzen, um Datenanbindung möglich zu machen.

3.4.1 In der .xml Datei

Gehen wir davon aus, unsere .xml Datei besteht aus einem Linearlayout und zwei TextViews: Remo

Listing 1: Layoutcode ohne jedliche Datenanbindung.

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res
    /android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
     <TextView
        android:id="@+id/vorname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Max" />
     <TextView
        android:id="@+id/nachname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Mustermann" />
</LinearLayout>
```

Nun, um Datenanbindung zu ermöglichen, brauchen wir eine Objektklasse auf die wir Referenzieren. In diesem Fall erstellen wir eine Klasse "User"mit den String-Attributen firstName und lastName. Die Klasse besteht weiters aus einem Konstruktorfeld mit den Attributen und jeweiliger getter-Felder:

Listing 2: Unsere Objektklasse die bei der Datenanbindung referenziert wird.

```
package com.example.daniel.showcasedatabinding;
public class User {
```

[3] "Droidcon NYC 2015 - Data Binding Techniques"
https://www.youtube.com/watch?v=WdUbXWztKNY

```
    private final String firstName;
    private final String lastName;

    public User(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    public String getFirstName() {
        return this.firstName;
    }
    public String getLastName() {
        return this.lastName;
    }
}
```

Haben wir diese erstellt, kann bei der Layoutdatei weitergemacht werden. Wir erstellen nun um unser Layout einen "layoutTag. Es folgt ein variable Tag innerhalb eines data Tags. Darin definieren wir eine Variable auf die man innerhalb der Elemente zugreifen kann.

Listing 3: Die XML Datei nach der Integration einer Datenanbindung.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android
    ">
    <data>
      <variable name="user" type="com.example.daniel.
    showcasedatabinding.User"/>
    </data>
    <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@{user.firstName}" />

      <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@{user.lastName}" />
      </LinearLayout>
</layout>
```

Jetzt können wir unsere ID-Vergabe bei den Elementen löschen und im Text-Tag auf die jeweiligen Attribute des Userobjekts verweisen.

3.4.2 In der .java Klasse

Die Datenanbindungstechnologie von Android generiert automatisch spezielle Bindingklasse all jener Layoutdateien die es verwenden. Der Name ergibt sich aus dem Namen der .xml Datei + "binding". Also wenn eine Layoutdatei activity_main.xml benannt ist, wird daraus die Klasse ActivityMainBinding generiert, die wir nun verwenden können:

Listing 4: Die Aktivitätenklasse nach der Integration einer Datenanbindung.

```
1  package com.example.daniel.showcasedatabinding;
2
3  import android.databinding.DataBindingUtil;
4  import android.os.Bundle;
5  import android.support.v7.app.AppCompatActivity;
6
7  import com.example.daniel.showcasedatabinding.databinding.
       ActivityMainBinding;
8
9  public class MainActivity extends AppCompatActivity {
10
11   @Override
12   protected void onCreate(Bundle savedInstanceState) {
13     super.onCreate(savedInstanceState);
14
15     ActivityMainBinding binding = DataBindingUtil.setContentView(
       this, R.layout.activity_main);
16     User user = new User("Max", "Mustermann");
17     binding.setUser(user);
18   }
19 }
```

Es wird eine Instanz der generierten Klasse erstellt, welche dann ein Objekt angebunden bekommt, dessen Attribute die Elemente bekommen. Jede Änderung der verwendeten Eigenschaften der erstellten Userinstanz bedeutet auch eine Änderung des Elements, automatisch. [4]

[4] http://developer.android.com/tools/data-binding/guide.html

3.4.3 Quellen:

- https://www.it-visions.de/
- "Droidcon NYC 2015 - Data Binding Techniques"
  https://www.youtube.com/watch?v=WdUbXWztKNY
- http://developer.android.com/tools/data-binding/guide.html

Andreas Denkmayr 25. Februar 2016

4 Bundles

TODO

Andreas Denkmayr 25. Februar 2016

5 Permissions

TODO

Andreas Denkmayr 19. Februar 2016

6 Musik

6.1 Lokalisierung der Musikdateien

Am Beginn der Arbeit wurde der Music-Ordner nach mp3-Files durchsucht.

`Dabei wurde der Music-Ordner mit Hilfe eines FileExtensionFilters nach mp3-Dateien durchsucht.`

```
File home = new File(Environment.getExternalStorageDirectory().
    getAbsolutePath() + "/Music";);
songs = new ArrayList<>();
if (home.listFiles(new FileExtensionFilter()) != null) {
  for (File file : home.listFiles(new FileExtensionFilter())) {
        HashMap<String, String> hm = new HashMap<>();
                hm.put("songTitle", file.getName());
                  hm.put("songPath", file.getPath());
                   songs.add(hm);
  }
}
```

Im Laufe der Entwicklung stellte sich heraus, dass jeder Benutzer seine Musikdateien in einem anderen Ordner und in verschiedenen Dateiformaten abspeichert. Die Lösung für dieses Problem stellt der Android Mediastore dar. Über diesen können Abfragen nach verschiedenen Medientypen z.B.: Musik, Fotos oder Videos durchgeführt werden. Von diesen Medientypen können Name, Pfad, Dateigröße und vieles mehr ausgelesen werden.

`Die Datenabfrage gegenüber dem Mediastore erfolgt über einen ContentResolver mit Hilfe der query()-Methode.`

```
public final @Nullable Cursor query(@NonNull Uri uri, @Nullable
    String[] projection, @Nullable String selection, @Nullable
    String[] selectionArgs, @Nullable String sortOrder) {
  return query(uri, projection, selection, selectionArgs,
    sortOrder, null);
}
```

`Für den Musicplayer benötigen wir alle Audiodateien die Musik beinhalten.`

```
ContentResolver cr = context.getApplicationContext().
    getContentResolver();
Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
```

```
String selection = MediaStore.Audio.Media.IS_MUSIC + "!= 0";
String sortOrder = MediaStore.Audio.Media.TITLE_KEY + " ASC";
Cursor cur = cr.query(uri, null, selection, null, sortOrder);
```

Für jede Zeile im Cursor cur wird eine HashMap aus 2 Strings erstellt die den Titel und den Pfad eines Songs beinhaltet. Diese HashMaps werden anschließend zu einer ArrayList hinzugefügt.

```
songs = new ArrayList<>();
HashMap<String, String> hm = new HashMap<>();
while(cur.moveToNext())
{
  hm.put("songTitle", cur.getString(cur.getColumnIndex(MediaStore
    .Audio.Media.TITLE)));
  hm.put("songPath", cur.getString(cur.getColumnIndex(MediaStore.
    Audio.Media.DATA)));
  songs.add(hm);
}
```

6.2 Verwalten von Playlists

Im ersten Screenshot sieht man die Auswahl der erstellten Playlists, wobei die Playlist ”All’ nicht bearbeitet werden kann und alle eingelesenen Songs darstellt. Zusätzlich zu der ”All”-Playlist kann der Benutzer eigene Playlists anlegen und diese auch bearbeiten.

Der All und der None Button helfen dem Benutzer schnell alle Songs zu markieren oder die Markierung aller Elemente aufzuheben.

Der Back Button führt zurück zur Playlistauswahl und verwirft alle nicht gespeicherten Änderungen an der aktuellen Playlist.

Wird der Save-Button gedrückt wird für alle Positionen abgespeichert ob das Element an der jeweiligen Position markiert ist. Dies erfolgt über ein SparseBooleanArray. Anschließend wird eine Liste erstellt in der nur die angekreuzten Elemente enthalten sind. Diese Liste wird als eine neue Playlist in die PlaylistSongs-Tabelle gespeichert. Dabei wird als Playlistname der eingetragene Titel in das EditText unter der ListView übernommen.

```
SparseBooleanArray checked = lvSongs.getCheckedItemPositions();
for (int i = 0; i < songs.size(); i++) {
    if (checked.get(i)) {
        checkedSongs.add(songs.get(i));
```

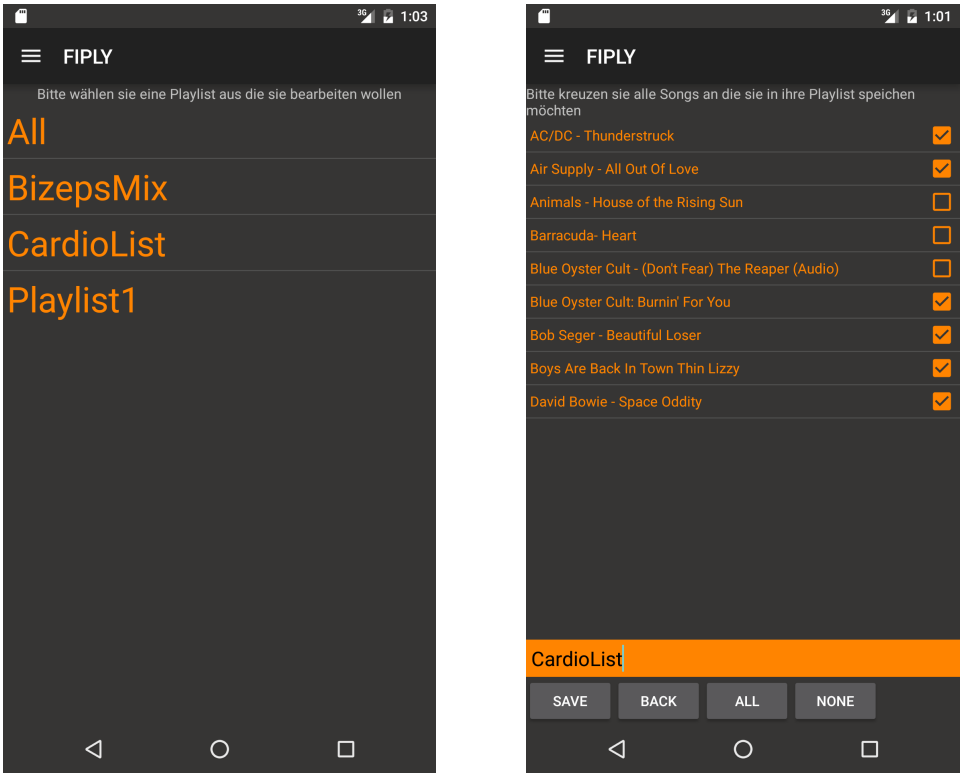


Abbildung 6: Bei klicken eines Elements in Screenshot 1 werden alle Songs angezeigt (Screenshot 2) und mittels der Checkbox sind alle Songs gekennzeichnet die sich in der ausgewählten Playlist befinden.

```
5       }
6   }
7   psrep.reenterPlaylist(etName.getText().toString(), checkedSongs);
```

6.3 Abspielen der Playlists.

[3][4][5][6]
Das Abspielen der Songs erfolgt über den MediaPlayer (API level 1).
Das Wechseln eines Songs wurde mithilfe der changeSong-Methode realisiert. Diese Methode nimmt die Playlist und den Index eines Songs in dieser Playlist entgegen, kümmert sich um das Setzen der Datenquelle für den MediaPlayer und bereitet den MediaPlayer auf die Wiedergabe vor. Zusätzlich wird der neue Songname angezeigt und die laufende Aktualisierung der Fortschrittsanzeigen wird durch den Aufruf von updateProgressBar() eingeschaltet.

```
public void changeSong(int songIndex, String playlist) {
    aktPlaylist = playlist;
    setPlaylist(psrep.getByPlaylistName(aktPlaylist));
    setSongIndex(songIndex);
    try {
        mp.reset();
        mp.setDataSource(getPlaylist().get(getSongIndex()).get("
   songPath"));
        mp.prepare();
    } catch (IOException e) {
        e.printStackTrace();
    }
    progressBar.setProgress(0);
    updateProgressBar();
    tvSongname.setText(getPlaylist().get(getSongIndex()).get("
   songTitle"));
    tvTotalDur.setText(millisecondsToHMS(mp.getDuration()));
}

private void updateProgressBar() {
    mHandler.postDelayed(mUpdateDurTask, 100);
}
```

Der mUpdateDurTask aktualisiert die Fortschrittsanzeigen 10 mal pro Sekunde. Da mp.getDuration Millisekunden zurückliefert, konvertiert die millisecondsToHMS-Methode die Songdauer in einen Strign im hh:mm:ss Format (ISO 8601).

```
private Runnable mUpdateDurTask = new Runnable() {
    @Override
    public void run() {
        long currentDur = mp.getCurrentPosition();
        tvCurrentDur.setText(millisecondsToHMS(currentDur));
        int progress = getProgressPercentage(currentDur, mp.
   getDuration());
        progressBar.setProgress(progress);
        mHandler.postDelayed(this, 100);
    }
};
```

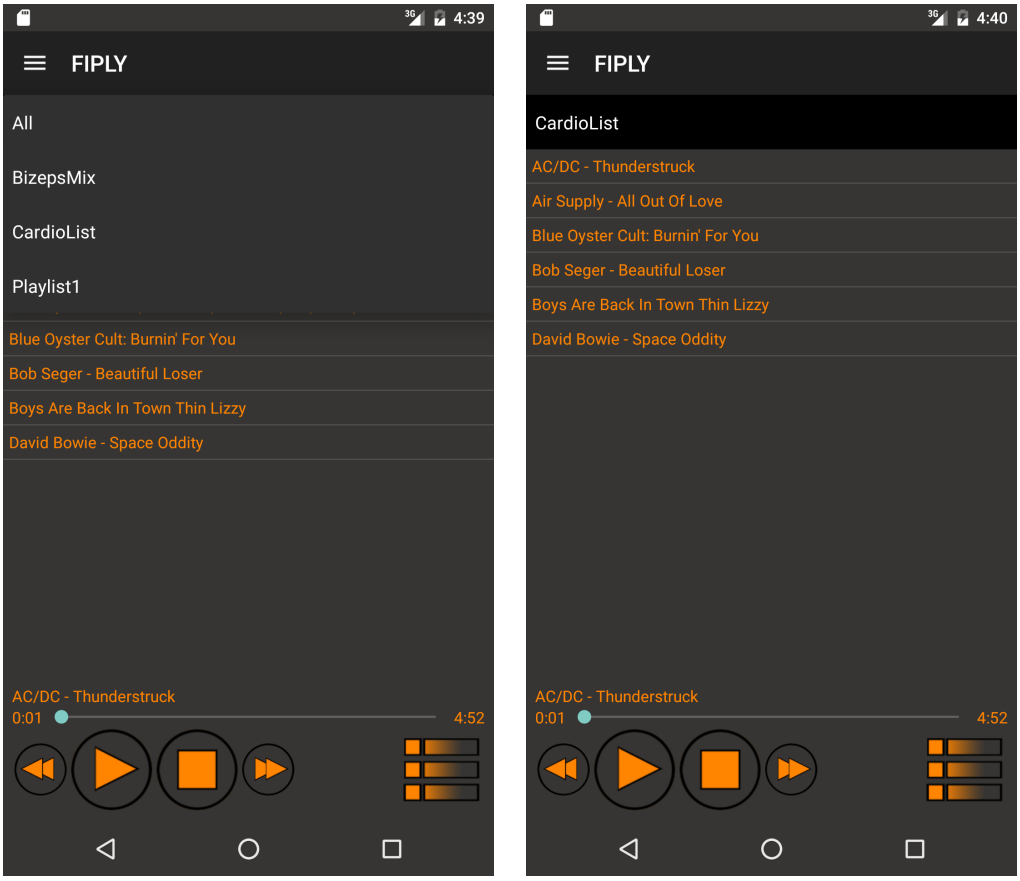


Abbildung 7: In einem Spinner kann eine Playlist ausgewählt werden. Bei Klick auf einen Song wird dieser abgespielt.

Während der Benutzer sich in einer Traingssession befindet kann jederzeit die Musikwiedergabe gestartet werden. Bei Klick auf den Play-Button wird die ”All”-Playlist in alphabetisch aufsteigender Reihenfolge abgespielt.
Die Playlist beziehungsweise der aktuell abgespielte Song kann geändert werden indem in den Musikmodus gewechselt wird.
Im Musikmodus wird über den Spinner die Playlist gewechselt.
In der aktuell ausgewählten Playlist kann man direkt zu einem bestimmten Song wechseln. Dieser wird abgespielt und nach Ende des Songs wird sofort der nächste Playlisteintrag gestartet.

6.4 MusicControls

Abbildung 8: Die MusicControls in Ruhe und während einer Wiedergabe.

- Durch den Zurück und durch den Weiter Button kann auf den vorherigen beziehungsweise auf den nächsten Song gewechselt werden.
- Der Play Button dient dem Starten der Musikwiedergabe.
  Während der Musikwiedergabe erscheint an dieser Stelle der Pause Button mit dem man die Musik pausieren kann.
- Der Stop Button beendet die aktuelle Wiedergabe und setzt den Wiedergabefortschritt auf den Beginn des Songs.
- Der Musikmodus Button befindet sich in der Ecke unten rechts.
  Dieser stellt den aktuellen Modus durch seine Einfärbung dar und ermöglicht einen Wechsel zwischen dem Übungsmodus und dem Musikmodus. Im Übungsmodus wird die aktuelle Übung und die Anweisungen zum Trainieren angezeigt.
  Der Musikmodus hingegen ermöglicht ein Wechseln der Playlist und den manuellen Wechsel auf einen bestimmten Song.
- Links von der Fortschrittsleiste wird der Fortschritt des aktuellen Songs im hh:mm:ss Format (ISO 8601) angezeigt.
- Die Fortschrittsleiste wird durch eine SeekBar über diesen Buttons implementiert. Diese SeekBar stellt den aktuellen Fortschritt des aktuellen Songs dar.
  Bei Klicken auf oder Ziehen an der Fortschrittsleiste kann man den Fortschritt der Wiedergabe manipulieren.
- Rechts von der Fortschrittsleiste wird die Gesamtdauer des Songs im hh:mm:ss Format (ISO 8601) angezeigt.
- Der Name des aktuellen Songs wird in einer TextView über den Fortschrittsanzeigen dargestellt.

Andreas Denkmayr 21. Februar 2016

7 NavigationDrawer

[7]
TODO

8 Commercialization

Andreas Denkmayr 25. Februar 2016

8.1 Donations

Donations stellen eine Möglichkeit dar den Entwicklern einer App Geld zu spenden um die Entwicklung der App oder anderen Apps zu unterstützen. Es gibt viele Möglichkeiten um Donations zu unterstützen.
Zu den populärsten Formen zählen Dienste wie PayPal, Flattr oder das erstellen einer kostenpflichtigen App die keine Funktionen beinhaltet und den selben Namen wie die Gratis App + einen Suffix wie z.B.: (Donation) trägt.

8.1.1 Flattr

> When you're registered to flattr, you add money to your account and set a monthly budget. During the month you flattr creators by clicking the Flattr-button next to their content. At the end of the month, your monthly budget is divided between all the things you flattered and sent to the creators.

[8] *About Flattr*

> Flattr can be used as a complement to accepting donations. Or to having advertising. Or to help getting donations you never get for your open source software, blog, music, film, game etc etc.

[8] *About Flattr*

Flattr eignet sich gut um Spenden durchzuführen, da dieses Vorgehensmodell Entwickler direkt unterstützt anstatt Geld für eine bestimmte Leistung entgegen zu nehmen.

8.1.2 PayPal

PayPal und deren Mobile Payment Libraries unterstützen die einfache Implementierung eines "Pay with Paypal"-Buttons über den Käufe mittels eines PayPal-Account durchgeführt werden können. Da wir unsere App aber in den Google Play Store stellen wollen stehen wir vor dem Problem, dass die Einbindung von Donations in Apps, die über den Play Store vertrieben werden, nur sehr vage in den Google Play-Programmrichtlinien für Entwickler beschreiben sind.

> Käufe im Store: Entwickler, die Gebühren für Apps und Downloads bei Google Play erheben, müssen dies über das Zahlungssystem von Google Play tun.

[9] *Google Play-Programmrichtlinien*

> Here are some examples of products not currently supported by Google Play In-app Billing: [...] One time-payments, including peer-to-peer payments, online auctions, and donations

[10] *Google Play In-app Billing*

8.1.3 Zweite kostenpflichtige App

Es besteht die Möglichkeit eine zweite App zu erstellen die selbst keine Funktionen beinhaltet und den selben Namen wie die Gratis App + einen Suffix wie z.B.: (Donation) trägt. In diesem Falle kann ein zufriedener Benutzer diese zweite App kaufen und so den Entwickler der Gratis App unterstützen.

Andreas Denkmayr 25. Februar 2016

8.2 Freemium

Freemium ist ein Konzept, das das verdienen von Geld über In-App-Käufe als primäre Einkommensquelle vorsieht. Diese In-App-Käufe erfolgen über den Google Play Store.
Dazu werden In-App-Products auf der Google Play Store Website angelegt. Diesen Items wird eine Id, ein Name, ein Preis und einer von 3 Itemtypen zugeteilt. [11] *Implementing Freemium*

8.2.1 Consumable Items

Consumable Items sind Artikel die benutzt werden können und nachgekauft werden können. Beispiele für Consumable Items sind zum Beispiel Tankfüllungen in einem Spiel. Eine Tankfüllung kann nur ein einziges Mal und nur auf einem Gerät verwendet werden. Sollte man noch eine Tankfüllung brauchen muss man das Consumable Item noch einmal kaufen.

8.2.2 Non-Consumable Items

Non-Consumable Items sind Artikel die einmal gekauft werden und dem Benutzer erhalten bleiben. Ein Beispiel für ein Non-Consumable Item ist zum Beispiel eine Upgrade für ein Auto in einem Spiel. Dieses Upgrade bleibt erhalten und ist auch auf anderen Geräten verfügbar solange man mit dem selben Google Play Account eingeloggt ist.

8.2.3 Subscriptions

Bei Subscriptions wird regelmäßig eine Gebühr entrichtet. Diese verlängern sich automatisch und müssen manuell storniert werden falls man die dadurch bereitgestellten Services nicht mehr benötigt. Als Entwickler kann man definieren wie oft eine Gebühr entrichtet werden muss und kann auch eine kostenlose Probezeit zur Verfügung stellen. Ein Beispiel für eine Subscription ist ein Upgrade das unendlich viele Tankfülllungen in einem Spiel zur Verfügung stellt solange diese aktiv ist.

Andreas Denkmayr 25. Februar 2016

8.3 Advertising mit AdMob

Mit dem Schalten von Werbung steht eine weitere Einkommensquelle von Apps zur Verfügung. Es gibt mehrere Dienste die das Schalten von Werbungen unterstützen. Google AdMob ist der populärste dieser Dienste und wird von Google empfohlen. Die Google Developers Seite stellt Tutorials bereit wie Werbungen mit AdMob implementiert werden können.

8.3.1 Banners

Banner Ads nehmen einen kleinen Teil des Bildschirms ein. Diese werden meistens in einem Layout File erstellt und dann in einer Activity oder in einem Fragment geladen. Der User kann durch einen Klick auf den Banner auf die beworbene Webseite weitergeleitet werden.

8.3.2 Interstitials

Interstitial Ads bedecken den gesamten Bildschirm. Dabei wird ein InterstitialAd Objekt mit einer einzigartigen Id erstellt. Später wird eine Werbung angefordert und sobald diese geladen ist wird sie angezeigt. Der Benutzer erhält die Entscheidung die Anzeige zu schließen oder dem Link der Werbung zu folgen. Deshalb eignen sich Interstitials in Apps die gelegentlich zwischen 2 Bildschirmen wechseln. Bei diesen Werbungen ist zu beachten, dass die App im Hintergrund weiterläuft also sollten laute Tonwiedergaben und ressourcenintensive Benutzerinteraktionen pausiert werden. Zusätzlich ist es möglich diese Interstitials in einem bestimmten Zeitintervall einzublenden.

Gerald Irsiegler 25. Februar 2016

8.4 PaidVersion

8.4.1 Beschreibung

Die App wird mit einem fixen Preis veröffentlicht bzw. verkauft. Der Kunde bezahlt ein mal und erhält unser Produkt, mit seinem gesamten Funktionsumfang, sofort. Alle zukünftigen Updates sind im Preisumfang enthalten.

8.4.2 Vorteile

- Ein klarer Vorteil bei einem fixen Preis ist, dass es die einfachste Methode der Vermarktung ist. Es ist die bekannteste und weit verbreiteste Methode ein Produkt zu vermarkten, dies ist dem Kunden natürlich auch vertrauter als andere Methoden.
- Weiters ist es die sicherste Methode der Vermarktung, sie garantiert Geld fast sofort, was wiederum für die Weiterentwicklung verwendet werden kann.
- Ein weiterer Vorteil ist der geringe Verwaltungsaufwand. Die App wird am Google-Play-Store vermarktet und kümmert sich um jegliche Details, wie z.B. Steuern.

8.4.3 Nachteile

- Ein Nachteil dieser Vermarktungsmethode ist, dass der Kunde in ein Produkt investiert, welches er vorher nie ausprobieren könnte. Dies könnte möglicherweise abschreckend wirken und potentielle Kunden abwimmeln.

8.4.4 Preis

Der Preis unseres Produkt wird anfänglich bei weniger als einem Euro liegen. Bei einer stetig wachsenden Benutzeranzahl wird der Preis auf einen bis zwei Euro erhöht werden. Dies ist jedoch das Maximum, da ein zu hoher Preis zu niedrigeren Verkaufszahlen führt.

8.4.5 Google-Play-Store

Um eine App in den Google-Play-Store hochzuladen und dort zu vermarkten ist ein Google-Developer Account nötig. Weiters muss die App der EULA von Google zur Vermarktung von Apps in ihrem Store entsprechen.

Gerald Irsiegler 25. Februar 2016

8.5 Freemium

8.5.1 Beschreibung

Es ist eine Basis (Free) Version gratis erhältlich, diese enthält nicht den gesamten Funktionsumhang. Um alle Funktionen freizuschalten ist eine Gebühr zu zahlen.

8.5.2 Vorteile

- Der Kunde kann sich bevor er Geld investiert einen ersten Eindruck verschaffen und muss die App nicht "blind"kaufen.
- Da es keine Einstiegs-Barriere gibt, erreicht die App mehr User und wird sich somit schneller verbreiten.

8.5.3 Nachteile

- Der Hauptteil aller Apps die dieses Modell wählen, verlieren im Laufe ihrer Lebenszeit Geld. Nur ein sehr kleiner Teil kann sich durchsetzen.
- Es gibt im Google-Play-Store etwa vier mal mehr gratis Apps, als bezahlte Apps und somit ist es schwerer sich am Markt durchzusetzen.

Gerald Irsiegler 26. Februar 2016

8.6 SellApp

8.6.1 Beschreibung

Nachdem die App eine größere Benutzerbasis hat, könnte es Interessenten zum Kauf des gesamten Projektes geben.

8.6.2 Vorteile

- Der Gesamtwert einer gut gepflegten App mit großer Userbasis liegt extrem hoch und diese Methode wäre eine der rentabelsten.
- Flexibilität bei der Weiterentwicklung, wir würden entscheiden können ob wir unser Produkt weiterentwickeln oder nicht. Bei Nichtbeiteiligung wäre die Instandhaltung der App aus unseren Händen und das Projekt für uns abgeschlossen.

8.6.3 Nachteile

- Wenn entschieden wird die App mit der Firma, die unsere App erworben hat, weiter zu arbeiten, wäre unsere gestalterische Freiheit eingeschränkt und es wäre schwerer Visionen umzusetzen.

8.6.4 Probleme/Schwierigkeiten

Es werden einige Problem aufgeworfen beim Verkauf einer App an eine größere Firma. Die erste Hürde ist einen Käufer zu finden. Ohne eine große Userbasis wird das Interesse an unserem Produkt sehr niedrig sein, deshalb wird es eine Weile dauern bis uns diese Möglichkeit zur Verfügung steht. Weiters fehlt uns in dieser Hinsicht die Erfahrung, also müssten wir uns auf dritte Personen verlassen den Verkauf für uns abzuwickeln.

Daniel Bersenkowitsch

8.7 Promotion

Es gibt dutzende Mittel um eine Android Applikation zu vermarkten. Ob über das Internet oder durch klassische physische Varianten lässt sich am Besten über Zielgruppenorientatierung bestimmen.

8.7.1 Website

Ein gute Methode um eine positive Reputation für seine Applikation zu schaffen ist es eine Website zu erstellen. Sie soll die Besucher einen kurzen Einblick in das Projekt und die Applikation bieten. Damit kann man bereits vor der Veröffentlichung eine positive Reputation schaffen und Benutzer an Land ziehen, bevor das Produkt überhaupt auf dem Markt ist. Eine solche Teaser-Website sollte nur mit den minimalistischen Beschreibungen der Hauptfunktionen verkleidet werden, ein einfaches Design haben und einen guten Überblick über das Endprodukt aufzeigen: Als Beispiel Die Website der Applikation Instagram:

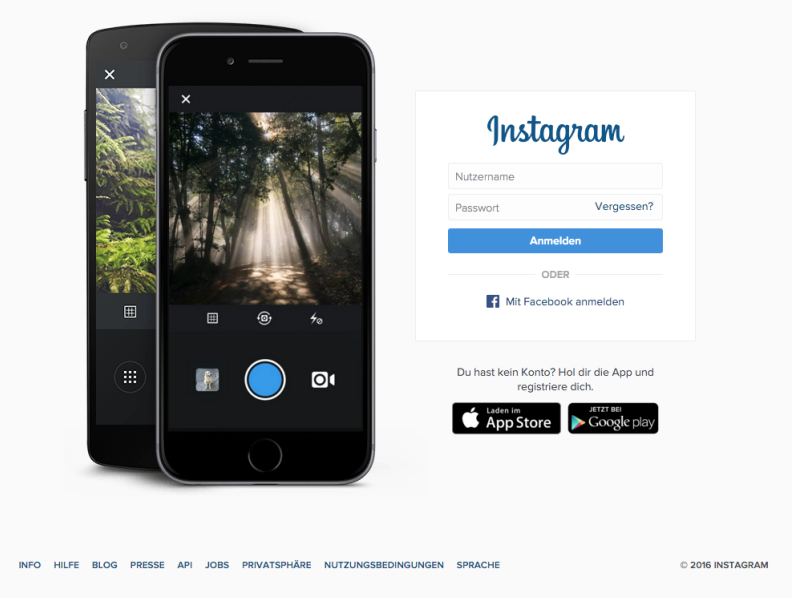


Abbildung 9: Screenshot von http://instagram.com:

Instagram.com ist ein gutes Beispiel für solch eine Teaser-Website. Eine weitere Möglichkeit ist es ein Video in die Website einzubauen, welche die Benutzung und Vorteile der Applikation aufzeigt. Eingebettet in die Seite

kann dies durch verschiedene Videohoster wie: Youtube.com, Vimeo.com oder Vidme.com. Das Video in einem normales HTML5 Mediaplayer einzubetten ist auch ein beliebtes Mittel.

8.7.2 Social Media Reputation

Ein bereits seit langem wichtiges Element in der online Vermarktung ist die Social Media Reputation. Wenn man heutzutage seine Applikation an die Menschen bringen will sollte das über die beliebte Sozialen Netzwerke wie Facebook, Google+, Twitter, Instagram, Vine,... und unzählige mehr. Seiten wie diese bieten die Möglichkeit für das Produkt einen eigene Seite oder Account zu erstellen, über diesen sich dann Benutzer unterhalten und ausstauschen können und neue an Land gezogen werden können. Zusätzlich können Administratoren beliebte Facebook Seiten beispielsweise dafür bezahlt werden, damit sie die App darauf teilen und positive Merkmale dabei unterstreichen.
Ein weiteres gutes Social Media mittel ist Reddit.com. Die Seite inkludiert die Funktion einen eigenen Developer Blog zu führen, worauf updates, bugs und neue Ideen für die Funktionalitäten der Applikationen geteilt werden können. Zusätzlich können bei dieser Webseite angemeldete User auch in diesem Blog posten - sich mit den Entwicklern unterhalten und mithelfen die App zu verbessern. Gute Vorschläge oder Ideen werden von Benutzern mit einem Pfeil nach oben markiert und wandern in der Postingliste hinauf damit die Sichtbarkeit steigt und sie mehr Personen lesen und darüber diskutieren können. Dies schafft eine sogenannte "Below-the-line Werbemöglichkeit für das Projekt.

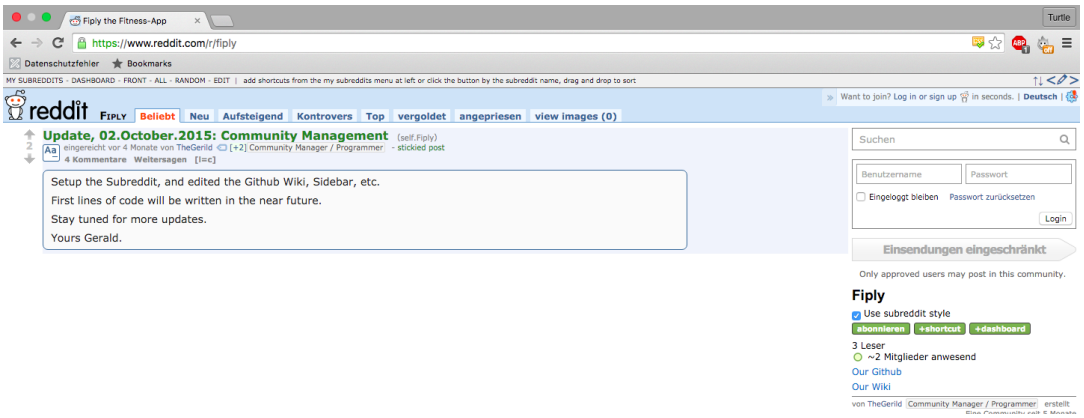


Abbildung 10: Screenshot von http://reddit.com/r/fiply

8.7.3 Presse

Eine der einfachsten Möglichkeiten ist es, eine Email an selektierte Schreiber die sich auf etwaige Bezugsthemen spezialisiern (Technik Blogs, Youtube-Review Kanäle oder wie bei dem Fiply Beispiel auch Fitnesstrainer/Fitnesscoaches und Fitnesscentern), auszusenden. Zu beachten ist dabei die persönliche Note. Jeder Aussendung sollte individuell zugeschnitten werden um den Empfänger anzusprechen. Inhalt soll knapp und bündig gehalten werden und die wichtigsten Informationen zusammenfassen wie zum Beispiel: die wichtigsten Funktionen, wichtige Links, Kontaktinformationen und Alleinstellungsmerkmale. Weiters ist zu überlegen, ob eventuelle kostenpflichtige Versionen dem Adressaten gratis übermittelt werden um ihn oder sie dafür zu motivern über das Projekt zu schreiben.

8.7.4 Konteste

Kaum mit etwas anderem kann man schneller für Aufmerksamkeit sorgen als mit dem Gewinnen von Kontesten. Auch schon nur bei einer Einreichungen und Nominierung kann man einen hohen an Bekanntheitsgrad für seine App erlangen. Ein positiver Effekt ist auch, dass Gewinner und oft Zweit- und Drittplatzierte sich durch so eine Veranstaltung hin und wieder einen kleinen aber wichtigen Artikel in Zeitungen oder Onlinemagazinen, aber auch Technikblogs u.A. teilen. Konteste sind ein wichtiges Element um das Produkt an die Zielgruppen zu bringen. Am besten sieht man nach welche Art von Kategorien es bei Preisauschreiben gibt und reicht es dann dort mit der höchsten Gewinnchancen ein, oft kann man sein Projekt auch bei mehreren Kategorien gleichzeitig anmelden.
Eventuelle Siege können auf der App-Homepage aufgeführt werden um Erstbesucher davon zu überzeugen, dass das geworbene Produkt Qualität aufweist.

8.7.5 Persöhnliche Werbung

In erster Linie sollte man seine Bekannten und Freunde um ein ehrliches Feeback bitten, ob das Projekt denn wirklich gut bei dem Endbenutzer ankommen kann. Objektive Meinungen von Freunden sind der Grundstein für die Erfolgsvorhersung einer selbst erstellten Applikation. Die Verbreitung an Bekannte oder Verwandte kann durch persönliche Gespräche, Emails oder Social Media Erfolgen. Am Besten bewährt sich dabei jedoch die persöhnliche Vermittlung. Falls das Endprodukt wirklich seinen Funktionsumfang gewährleistet und die Qualität demensprechend gut ist, werden diese Vermittlungspfade auch ihren Bekanntsheitskreis vorstellen. Dies geht weiter weiter.

Mundpropaganda mag nicht sehr schnell sein, dafür aber sehr effektiv.

8.7.6 Reklame

Bei Reklamen/Werbeinschaltungen sollte in erster Linie auf die Zielgruppe geachtet werden. Erfolgreiche Reklame passiert nur durch zielgerichtete Zielgruppendefinition. Dabei sollte man sich Fragen stellen wie: Was will ich genau bewerben? Welche Personen will ansprechen? Durch welche Plattformen komme ich an diese Personen ran? Wie gestalte ich die Werbung möglichst attraktiv?
Potentielle User kann man dort anwerben wo sich diese aufhalten. Bei diversen Onlinecommunities in Foren, Websiten oder Applikationen mit verwandtem Inhalt oder Interessensgruppen. Dabei schränken sich die Möglichkeiten nicht nur auf physische Reklamemöglichkeiten wie Inserate oder Werbeplakate ein. Dienste wie Facebook, Twitter, Google Adsense bieten eine Umfangreiche Zielgruppenschaltung zu entsprechenden Preisen an. Aber auch auf Radio und Televisionswerbung muss nicht verzichtet werden.

8.7.7 In Appstore Optimierung (IAO)

Genau wie bei einer Suchmaschinenoptimierung für Webseiten bei Diensten wie Google.com gibt es dies auch für Applikation im Google Playstore. Ziel ist es, dass der potentielle Benutzer durch verschiedene, möglichst wenige, bestimmte Stichwörter in der Suchanfrage auf die Applikation stoßt. Mehr als 50% der Nutzer einer App entdecken diese durch das erstmalige Suchanfrage bestimmter Stichwörter in dem Store.
Bei der Einrichtung muss speziell auf die Länge des Titels, Beschreibung und Name geachtete werden. Zwecks Wiedererkennungsfunktion sollte die App einen eigentständigen und nicht andersweitig vorkommenden Namen besitzen, damit die Benutzer den Namen eindeutig mit dem Produkt assoziieren. Eine klare, übersichtliche und kurzgehaltene Beschreibung bringt den User eher dazu, die Applikation runterzuladen, womit die kommerzielle Erfolgswahrscheinlichkeit des Produkts am Besten sichergestellt werden kann.
Das Icon sollte möglichst minimalistisch gehalten werden. Der Benutzer assoziiert Farbschemas und Formen des Icons automatisch mit der Applikation, welche möglichst simple und mit der modernen Designrichtlinie "Weniger ist mehrümgesetzt werden soll.
Bei der Einbettung von Medien im Google Playstore sollten die nützlichsten und besten Aspekte und Funktionen der Applikation widergegeben werden. Screenshots mit wenig Aussagekraft sollte vermieden werden, da es ein Limit für eingebundenen Medien für das Appprofil gibt und diese möglichst effizient

genutzt werden sollten.

8.7.8 Fazit

Grundsätzlich kann gesagt werden, dass man sich bei der Bewerbung einer App nicht nur auf die Optimierungsmöglichkeiten im Store verlassen sollte. Es muss in Bezug auf Werbung vielmehr ein ganzheitlicher Ansatz gewählt werden, der schon vorab – also vor Veröffentlichung der App – bei den potenziellen Nutzern einen Anreiz zum Download schafft. Das Wichtigste ist allerdings, seine Annahmen und Maßnahmen regelmäßig auf deren Wirksamkeit und Erfolg hin zu prüfen und gegebenenfalls zu optimieren bzw. im Fall der Fälle auch gänzlich zu überdenken.

9 Der Trainingsplan

Konzept zur Erstellung eines Trainingsplans Daniel Bersenkowitsch

9.1 Vorwort

Ein Trainingsplan besteht aus unterschiedlichen Phasen - je nach Trainingsziel (Muskelaufbau, Maximalkraft, Kraftausdauer (=Gesundheit)) unterschiedlich. Jede Trainingsphase besitzt einen empfohlenenen RM-Wert (0%-100%), mit welchem der Benutzer üben kann. Alle empfohlenen Werte (Satzpausen, RM-Wert, Anzahl der Trainingstage,...) sind jediglich eine Option für den Benutzer und können auch frei gewählt werden.

9.2 Was ist Was?

Eine "Wiederholung" ist das 1-malige Ausüben einer Übung.
Ein "Satz" ist die Abschließung aller Übungen und ausgeführten Wiederholungen. 100% RM =(Repetition Maximum = Maximalwiederholung) ist die Ausführen einer bestimmten Üung zu einem bestimmten Gewicht, welches bei der Übung genau 1 Mal bewältigt werden kann:

$$\%RM = \frac{100 * Trainingsgewicht}{102.78 - (2.78 * Wiederholungen)}$$

Wenn man sich also das Trainingsgewicht ausrechnen will, mit dem man eine Übung ausführen soll, geht man wie folgt vor (Vorgeschlagener %RM-Wert und Wiederholungen sind angegeben):

$$Trainingsgewicht = \frac{\%RM * (102.78 - (2,78 * Wiederholungen))}{100}$$

Mit dem errechneten Gewicht führt man nun die jeweilige zugeordnete Übung aus. Da man bei konsequenten Training die Kraft steigt, sollte man auch immer das Trainingsgewicht erhöhen. Um maximalen Fortschritt zu erziehlen, ist es empfohlen, die Wiederholungen gleich bleiben zu lassen. Der Benutzer testet selbst wieviel Gewicht er mit den Wiederholungen schafft, die Änderung der Gewichts wird von ihm festgehalten, um positive Entwicklungen feststellen zu können.
Das Gewicht kann aber auch selbst abgeschätzt werden. Das Gewicht ist dann optimal gewählt, wenn man damit zwischen 10 und 13 Wiederholungen schafft. Weiters wird immer auf eine Aufwärmphase hingewiesen, welche man vor jeder Trainingseinheit durchführen muss. Sie besteht aus 5-10 Minuten Laufen und Dehnen.

9.3 Phase 1: Allgemein

Phase 1 ist für jeden gleich, unabhängig vom Trainingsziel, und dient dem Eintrainieren. Am Anfang wird festgehalten, ob der Benutzer einen untrainierten oder bereits trainierten Körper besitzt. Anhand dessen und dem ausgewählten Schema (Bauch - Beine - Po, Oberkörper/Arme, Stabilisation (Rücken & Gesundheit)) werden in dieser Phase die Übungen ausgewählt und die Anzahl der Sätze/Wiederholungen bestimmt:

	Anfänger		Fortgeschrittener	
Baum - Beine - Po	Übungen:	9	Übungen:	9
	Sätze:	2	Sätze:	3
	Wiederholungen:	20	Wiederholungen:	25
Oberkörper - Arme	Übungen:	6	Übungen:	8
	Sätze:	2	Sätze:	3
	Wiederholungen:	20	Wiederholungen:	25
Stabilisation	Übungen:	8	Übungen:	8
	Sätze:	2	Sätze:	3
	Wiederholungen:	20	Wiederholungen:	25

In Phase 1 werden alle Übungen mit einem Gewicht 55% RM ausgeführt. Es werden Anfangs 3 Trainingstage ausgewählt, an denen der Benutzer Zeit findet um zu trainieren. Dabei wird beachten, dass zwischen den Trainingstagen mind. 36 Stunden Pause eingelegt werden soll, um den Kreislauf zu schonen. Die empfohlene Satzpause liegt bei 30-60 Sekunden, kann aber auch frei bestimmt werden.
Die Phase 1 dauert bei einem Anfänger 8 Wochen und bei einem Fortgeschrittenen nur die Hälfte.
Im Überblick:

Übungen:	6 bis 9
Gewicht:	55% RM
Sätze:	2 oder 3
Wiederholungen:	20 oder 25
Pausendauer:	30-60 Sekunden
Wochentage:	3
Phasendauer:	4 oder 8 Wochen

9.4 Phase 2: Kraftausdauer (Gesundheit)

Phase 2: Kraftausdauer (Gesundheit) besteht aus 2 Phase 1: Allgemein Trainingstagen in der Woche. Zusätzlich besteht das Training aus entweder 1 Mal wöchentlich Phase 2: Muskelaufbau -training oder 1 Mal wöchentlich Stabilitätsübungen. Welche der Benutzer ausführen will kann er selbst am Anfang entscheiden, je nachdem ob er seinen Körper formen will, oder ob er es als Gesundheits- oder Rehaübung macht.
Also im Überblick:

1. Teil	Phase 1: Allgemein
Übungen:	6
Gewicht:	55% RM
Sätze:	2
Wiederholungen:	25
Pausendauer:	30-60 Sekunden
Wochentage:	2
Phasendauer:	8 Wochen

2. Teil	Phase 2: Muskelaufbau	Stabilitätsübungen
Übungen:	6	6
Gewicht:	80% RM	Keines
Sätze:	2	3
Wiederholungen:	25	12-20
Pausendauer:	30-60 Sekunden	60-120 Sekunden
Wochentage:	1	1
Phasendauer:	8 Wochen	8 Wochen

9.5 Phase 2: Muskelaufbau
Phase 3: Kraftausdauer

Wenn man als Trainingsziel “Muskelaufbau“ gewählt hat, kommt diese nach Phase 1, oder wenn man Phase 2: Kraftausdauer (Gesundheit) abgeschlossen hat. In dieser Phase kommen viel Hantel- und Seilzugtraining zum Einsatz. Dabei ist zu beachten, dass die Schwierigkeit der Übung egal ist. Es wird davon ausgegangen, dass ein Anfänger nach der 8-wöchigen Phase 1 bereits fit genug ist, um alle Übungen die sich im Trainingskatalog befinden zu meistern.
Der User kann sich beginnend aussuchen, welche 2-3 Muskelgruppen er trainieren will. Am Phasenanfang wird zwischen Splittraining und Ganzkörpertraining unterschieden. Der Unterschied zwischen den Trainingsarten liegt bei der zeitlichen Ausführung der Übungen. Bei dem Splittraining werden Übungen zu einer bestimmten Muskelgruppe bei jeder Trainingseinheit durchgeführt. Umgekehrt wird bei dem Ganzkörpertraining in jeder Trainingseinheit auf einen bestimmte Muskelgruppe geziehlt.
Diese Phase besteht wieder aus 3 Trainingstagen pro Woche und das empfohlene Trainingsgewicht liegt bei 80% RM. Die Satzpausendauer beträgt 90-120 Sekunden, bei 3 Sätzen und 12 Wiederholungen. Je nach Anzahl der fokusierten Muskelgruppen bestimmt sich die Zahl der Übungen, die man bekommt: Bei 2 Muskelbereiche sind es 6 Übungen, bei 3 sind es 9 Übungen.
Nach dieser 8-wöchigen Phase kommt Phase 3: Muskelaufbau.
Im Überblick:

	Muskelaufbau	Kraftausdauer
Übungen:	6 oder 9	6 oder 9
Gewicht:	80% RM	80% RM
Sätze:	3	3
Wiederholungen:	12	12
Pausendauer:	90-120 Sekunden	90-120 Sekunden
Wochentage:	3	3
Phasendauer:	8 Wochen	4 Wochen

Nach Phase 3: Kraftausdauer ist wieder von Anfang an (Phase 1: Allgemein) zu beginnen. Man kann aber auch aufh"ören oder sich einen neuen Trainingsplan generieren lassen.

9.6 Phase 2: Maximalkraft
Phase 3: Muskelaufbau

Diese Phase ist für das Maximalkrafttrainingsziel die Phase 2 und für das Muskelaufbautrainingsziel die Phase 3. Nur Fortgeschrittene oder User die die Phase 2: Muskelaufbau durchgeführt haben, können diese Phase beginnen.
Maximalkraftübungen sind immer Ganzkörperübungen. Das empfohlene Trainingsgewicht liegt bei 95% RM. Dabei kommen ausschließlich Seilzug- und Hantelübungen vor (6). Satzdauer beträgt 90-120 Sekunden, bei 3 Sätzen und 5 Wiederholungen. Diese Phase besteht wieder aus 3 Trainingstagen pro Woche und einer Mindesterholungszeit von 48h!
Das Maximalkrafttraining besteht aus einem zusätzlichen Schritt, der Mobilisation, der nach dem Aufwärmen beginnt. Danach kann mit dem Training begonnen werden.
Im Überblick:

Übungen:	6
Gewicht:	95% RM
Sätze:	3
Wiederholungen:	5
Pausendauer:	90-120 Sekunden
Wochentage:	3
Phasendauer:	Muskelaufbau: 4 Wochen Maximalkraft: 6 Wochen

Nach Phase 3: Muskelaufbau ist der Trainingsplan zu ende. Nun kann man ihn erneut starten (vom Anfang an), hört auf, oder lässt sich erneut einen generieren.

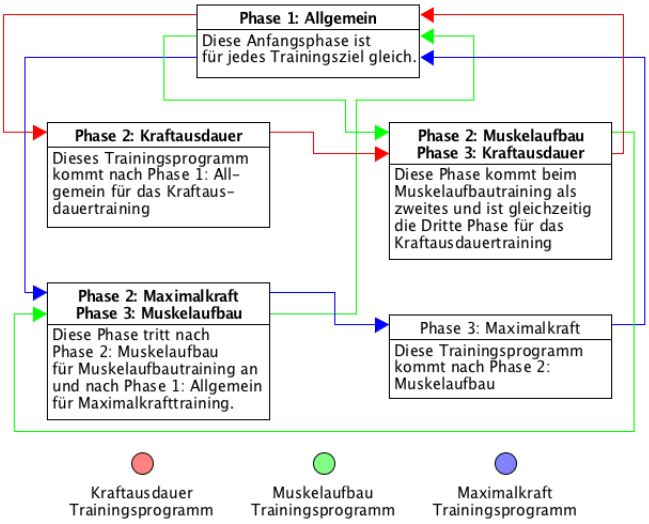
9.7 Phase 3: Maximalkraft

Phase 3: Maximalkraft kommt nach Phase 2: Maximalkraft. Hierbei wird jediglich 2 Mal wöchentlich trainiert. Die Tage kann sich der Benutzer wieder aussuchen, einzige Bedingung sind 48 Stunden Erholungszeit. Die Phase besteht wieder aus Seilzug- und Hantelübungen, insgesamt 6 mit jeweils 2 Wiederholungen zu 5 Sätzen. Hierbei wird das Trainingsgewicht erneut für ca. 95%RM gewählt.
Im Überblick:

Übungen:	6
Gewicht:	95% RM
Sätze:	5
Wiederholungen:	2
Pausendauer:	120-180 Sekunden
Wochentage:	2
Phasendauer:	8 Wochen

Nach Phase 3: Maximalkraft ist der Trainingsplan zu ende. Nun kann man ihn erneut starten (vom Anfang an), hört auf, oder lässt sich erneut einen generieren.

9.8 Visualisierung



9.9 Mobilisation

Die Mobilisation beim Maximalkrafttraining dient dazu, den Körper auf das kommende schwere Training vorzubereiten. Sie besteht aus:

Mobilisation	**Beschreibung**
Hals	Den Kopf im Wechsel nach rechts und links drehen.
Schulter	Die Arme neben dem Körper hängen lassen und mit den Schultern nach rückwärts kreisen.
Ellbogen	Die Hände auf die Schulter legen, mit den Ellbogen vorwärts und rückwärts kreisen, die Schultern dabei nach hinten und unten bewegen.
Handgelenk	Hände kreisen, beide Hände gleichzeitig mit größtmöglichem Bewegungsumfang fortlaufend um die eigene Achse drehen.
Becken-Mob	Die Arme über den Köpf führen, Handflächen nach oben schieben, Schultern bleiben tief, das Becken im Uhrzeigersinn, den ganzen Bewegungsumfang ausnutzen, Richtung ändern, die Kreise aus der Hüfte führen, die Beine sind stabil.
Wirbelsäule – Seitneigung	Linken Arm seitwärts hoch heben über den Kopf und Wirbelsäule seitwärts beigen, gegengleich, Handflächen nach oben.
Wirbelsäule – Rotation	Bauchnabel nach innen ziehen, die Arme in U-Form anheben, Daumen zeigen nach hinten und sind leicht nach außen gedreht, den Oberkörper vorbeugen, Gesäß nach hinten und zur Seite drehen, zur Mitte kommen, zur anderen Seite drehen, zur Mitte, immer im Wechsel, der Rücken bleibt gestreckt, die Schulterblätter sind zusammengezogen, das Becken bleibt stabil.
Wirbelsäule – Rolldown	Aufrechter Stand, den Kopf Richtung Brustbein senken, Bauchnabel nach innen ziehen, einatmen und beim Ausatmen die Wirbelsäule Wirbel für Wirbel in Richtung Boden abrollen , einatmen und wieder Wirbel für Wirbel aufrollen, der Rücken ist locker, der Nacken ist entspannt.

Literatur

[1] *Fragment*. Feb. 2016. URL: http://developer.android.com/reference/android/app/Fragment.html.

[2] *Fragments Guide*. Feb. 2016. URL: http://developer.android.com/guide/components/fragments.html.

[3] *MediaPlayer*. Feb. 2016. URL: http://developer.android.com/reference/android/media/MediaPlayer.html.

[4] Ravi Tamada. *Android Building Audio Player Tutorial*. Feb. 2016. URL: http://www.androidhive.info/2012/03/android-building-audio-player-tutorial/.

[5] *MusicDroid - Audio Player Part I*. Feb. 2016. URL: http://www.helloandroid.com/tutorials/musicdroid-audio-player-part-i.

[6] *Android: Build an MP3-Player*. Feb. 2016. URL: https://www.youtube.com/watch?v=W1JE-jUisVU.

[7] Ben Jakuben. *How to Add a Navigation Drawer in Android*. Feb. 2016. URL: http://blog.teamtreehouse.com/add-navigation-drawer-android.

[8] *About Flattr*. Feb. 2016. URL: https://flattr.com/about.

[9] *Google Play-Programmrichtlinien*. Feb. 2016. URL: https://play.google.com/intl/ALL_de/about/developer-content-policy.html.

[10] *Google Play In-app Billing*. Feb. 2016. URL: https://support.google.com/googleplay/android-developer/answer/6151557.

[11] Jarek Wilkiewicz. *Implementing Freemium*. Feb. 2016. URL: https://www.youtube.com/watch?v=UvCl5Xx7Z5o.