FIPLY

Daniel Bersenkowitsch, Andreas Denkmayr, Gerald Irsiegler

19. Februar 2016

Contents

Andreas Denkmayr 19. Februar 2016

1 Musik

1.1 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);
}
```

1.2 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));
```

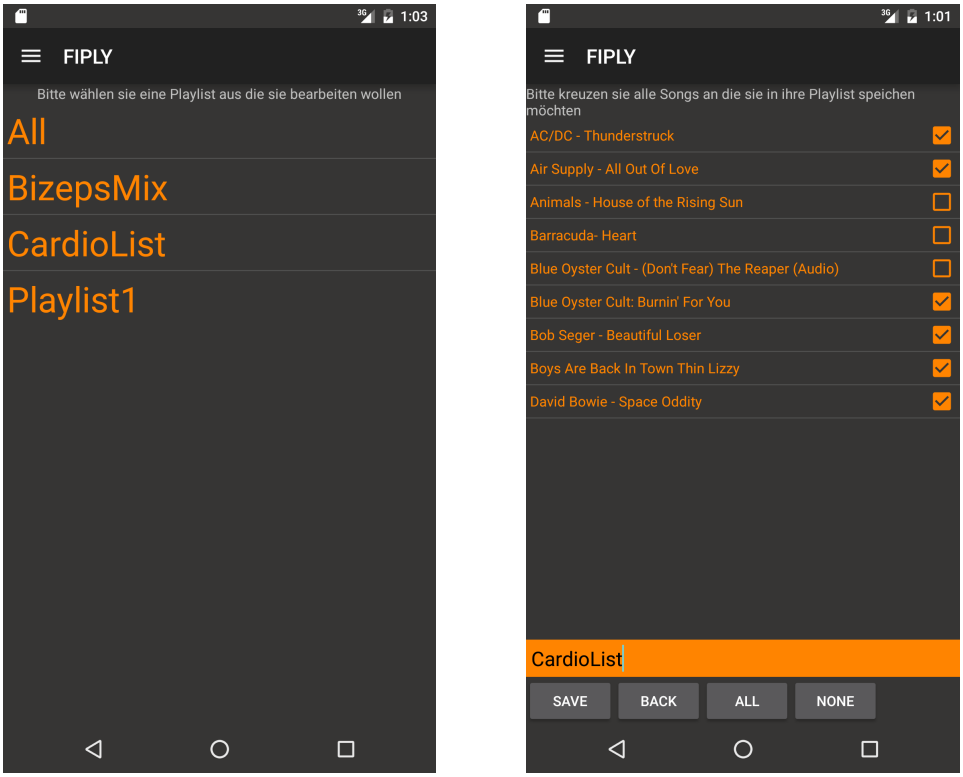


Figure 1: Bei klicken eines Elements in Screenshot 1 werden alle Songs angezeigt (Screenshot 2) und mittels der Checkbox sind alle Songs gekennzeichnet die sich in der ausgewählten Playlist befinden.

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

1.3 Abspielen der Playlists.

[**adMediaPlayer**][**bMediaPlayer**][**wMusicDroid**][**yMP3Player**]
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.

```
1  public void changeSong(int songIndex, String playlist) {
2      aktPlaylist = playlist;
3      setPlaylist(psrep.getByPlaylistName(aktPlaylist));
4      setSongIndex(songIndex);
5      try {
6          mp.reset();
7          mp.setDataSource(getPlaylist().get(getSongIndex()).get("
       songPath"));
8          mp.prepare();
9      } catch (IOException e) {
10         e.printStackTrace();
11     }
12     progressBar.setProgress(0);
13     updateProgressBar();
14     tvSongname.setText(getPlaylist().get(getSongIndex()).get("
       songTitle"));
15     tvTotalDur.setText(millisecondsToHMS(mp.getDuration()));
16 }
17
18 private void updateProgressBar() {
19     mHandler.postDelayed(mUpdateDurTask, 100);
20 }
```

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).

```
1  private Runnable mUpdateDurTask = new Runnable() {
2      @Override
3      public void run() {
4          long currentDur = mp.getCurrentPosition();
5          tvCurrentDur.setText(millisecondsToHMS(currentDur));
6          int progress = getProgressPercentage(currentDur, mp.
       getDuration());
7          progressBar.setProgress(progress);
8          mHandler.postDelayed(this, 100);
9      }
10 };
```

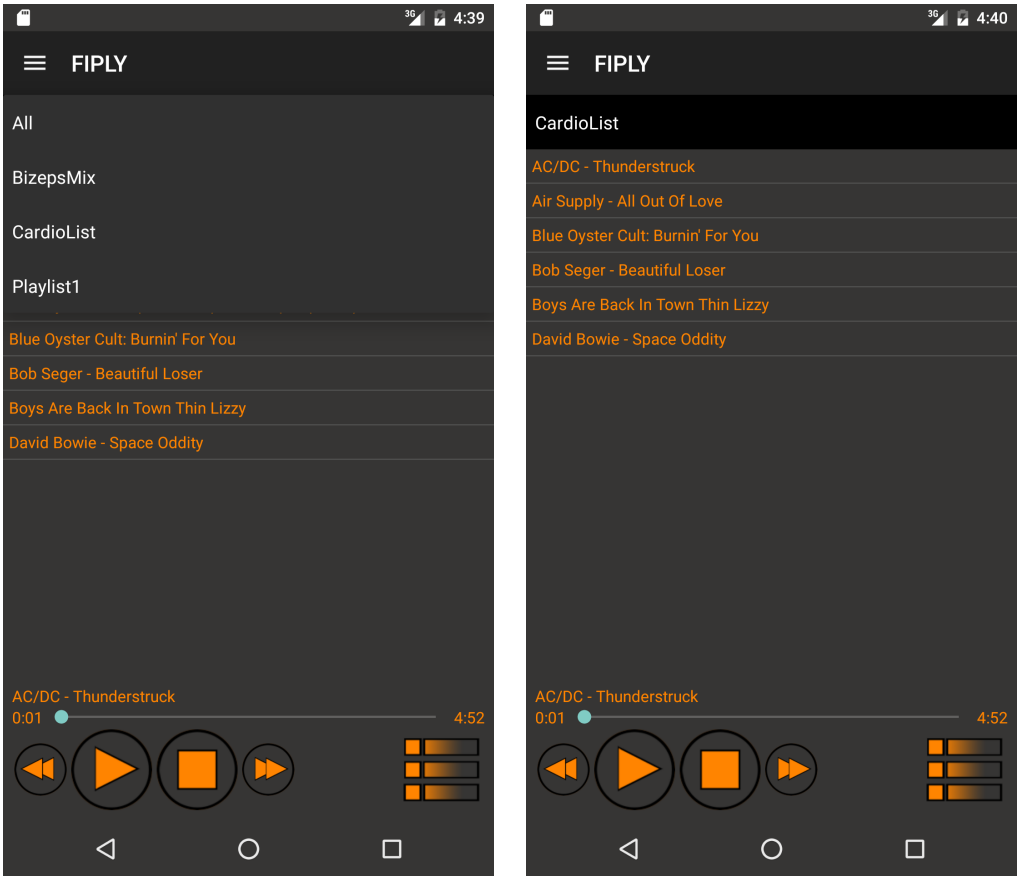


Figure 2: In einem Spinner kann eine Playlist ausgewählt werden. Bei Klick auf einen Song wird dieser abgespielt.

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.

1.4 MusicControls

Figure 3: 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 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.

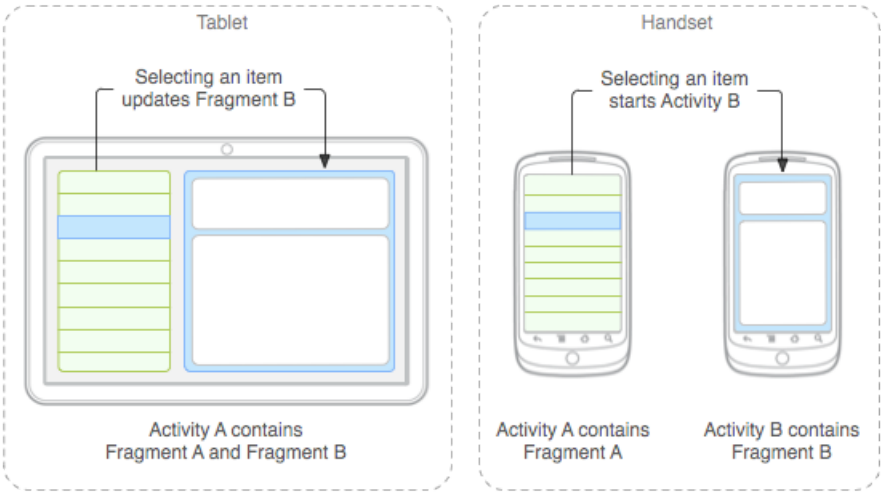


Figure 4: Beispiel man kann mithilfe von Fragments Views erstellen, die sowohl auf einem Tablet als auch auf einem Handy ein optimales Benutzerinterface anbieten

[**adFragments**] [**adFragmentsGuide**]

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" />
```

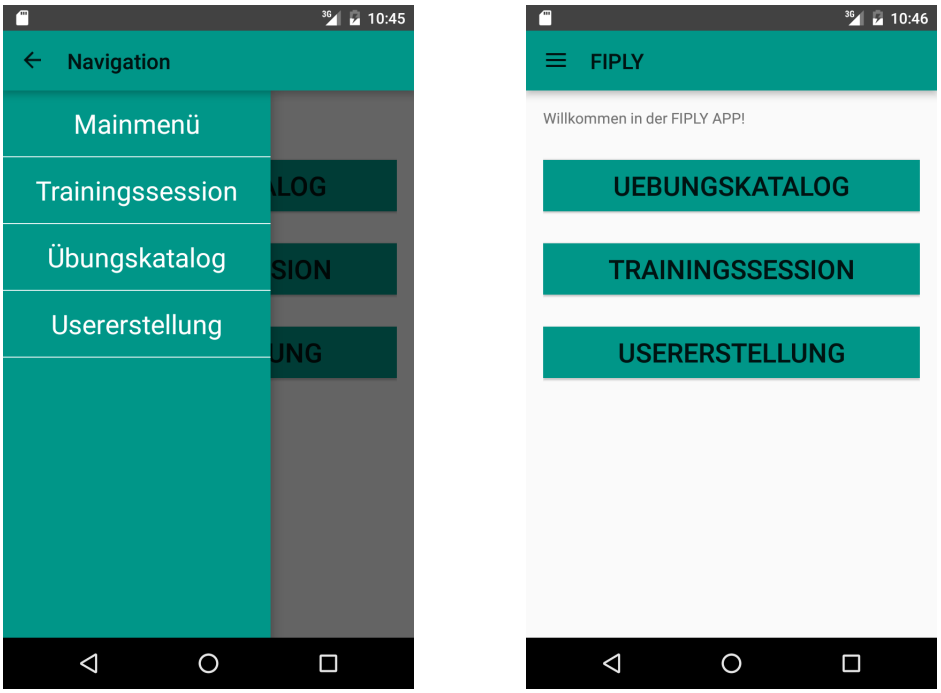


Figure 5: Bild des NavigationDrawers und des FMains

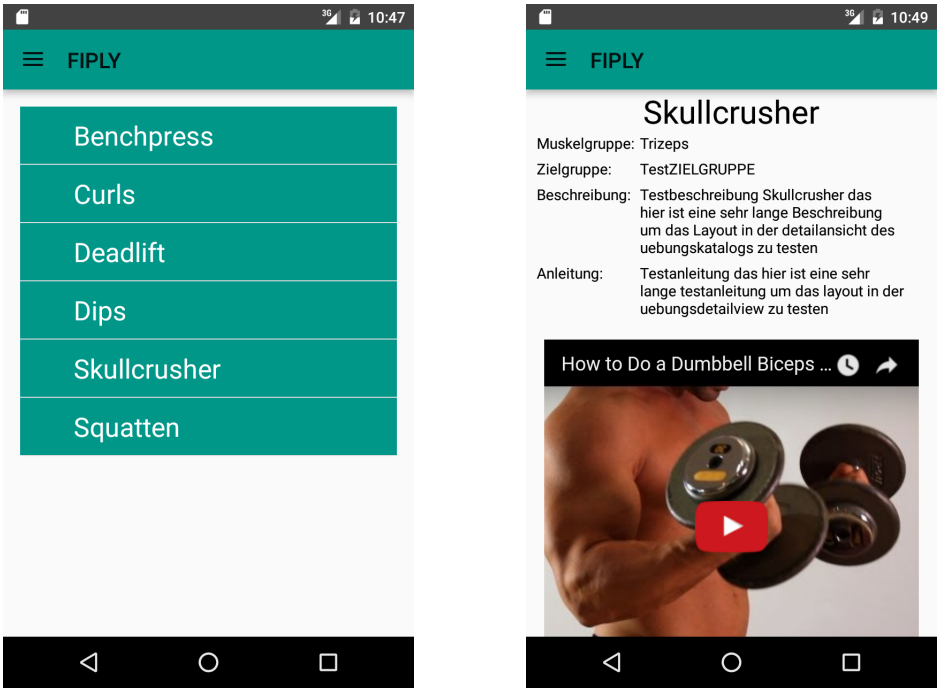


Figure 6: Bei Klicken eines Elements in der ListView wird die zugehörige DetailView aufgerufen

Andreas Denkmayr 21. Februar 2016

3 NavigationDrawer

[**bNavDrawer**]
TODO

Andreas Denkmayr 25. Februar 2016

4 Permissions

TODO

[utf8]inputenc [ngerman]babel graphicx [T1]fontenc listings color
Datenanbindung Daniel Bersenkowitsch 20. Dezember 2015
listings

Contents

5 Was ist 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.

5.1 Wozu Databinding in Android verwenden?

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]

6 Vorher - Nacher

6.1 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 nicht

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

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

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

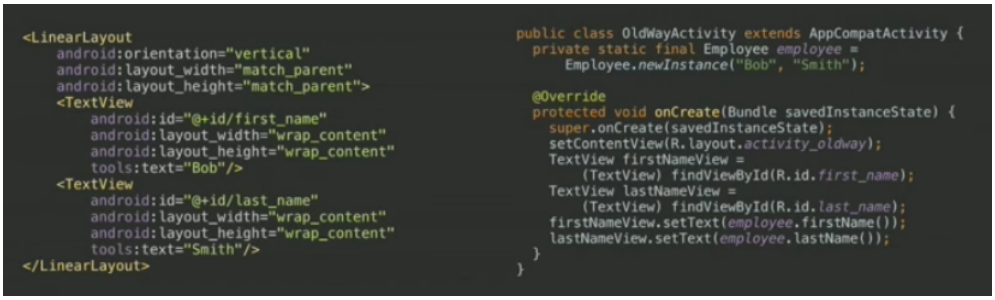


Figure 7: Screenshot des Codes wie er ohne Datenanbindung aussieht

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.

6.2 Nacher

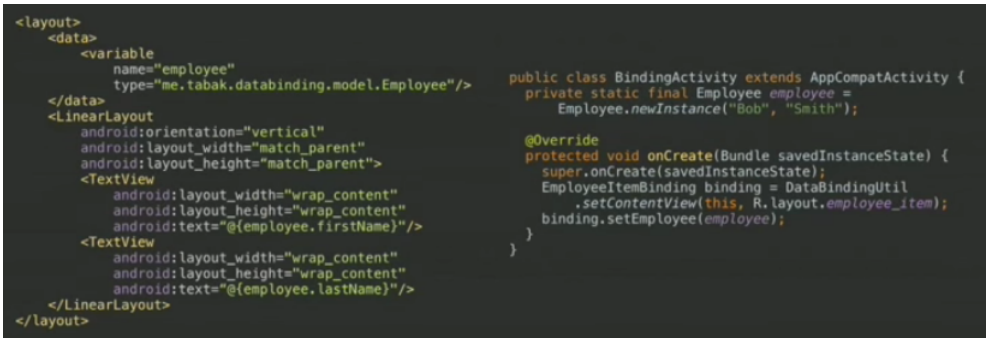


Figure 8: Screenshot des Codes wie er mit Datenanbindung aussieht.

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]

7 Wie wendet man Datenanbindung an?

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.

7.1 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

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

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 {
    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 ”layout”-Tag. 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"
```

```
14              android:text="@{user.firstName}" />
15
16          <TextView
17              android:layout_width="wrap_content"
18              android:layout_height="wrap_content"
19              android:text="@{user.lastName}" />
20          </LinearLayout>
21  </layout>
```

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

7.2 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 bedeuted auch eine Änderung des Elements, automatisch. [4]

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

8 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

9 Commercialization

Daniel Bersenkowitsch 25. Februar 2016

9.1 Einleitung

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.

9.2 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:

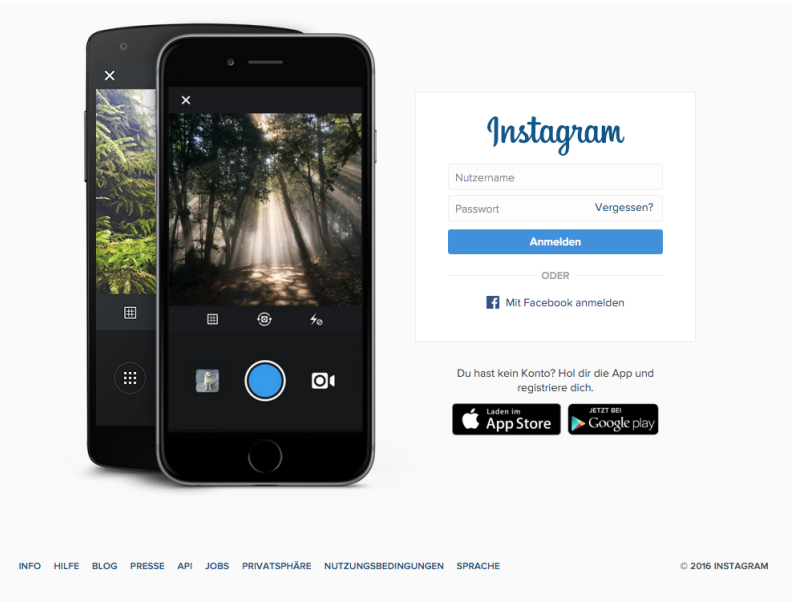


Figure 9: Screenshot von http://instagram.com:

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.

9.3 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.

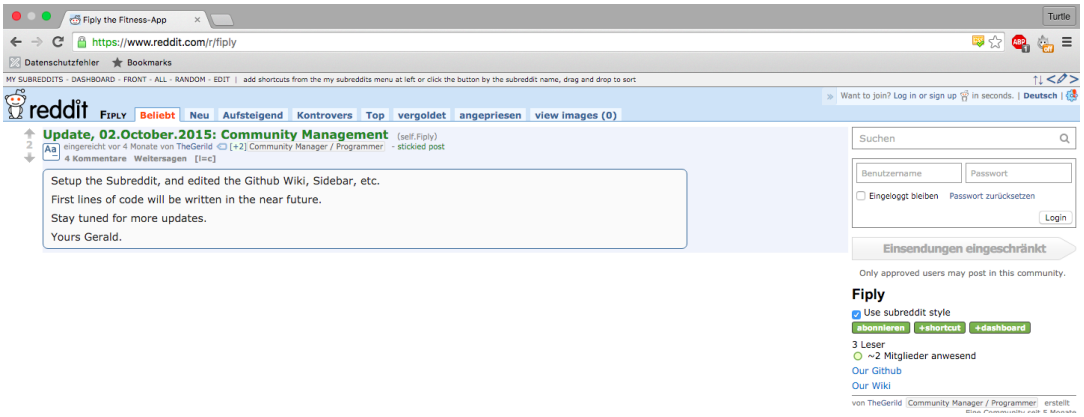


Figure 10: Screenshot von http://reddit.com/r/fiply

9.4 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.

9.5 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 k"onnen auf der App-Homepage aufgeführt werden um Erstbesucher davon zu überzeugen, dass das geworbene Produkt Qualität aufweist.

Gerald Irsiegler 25. Februar 2016

9.6 PaidVersion

TODO

10 Der Trainingsplan

[utf8]inputenc [ngerman]babel [T1]fontenc mathtools pbox graphicx

Konzept zur Erstellung eines Trainingsplans Daniel Bersenkowitsch 17. Jänner 2015

Contents

11 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 "uben kann. Alle empfohlenen Werte (Satzpausen, RM-Wert, Anzahl der Trainingstage,...) sind jediglich eine Option für den Benutzer und können auch frei gewählt werden.

11.1 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 Ausf"uhren einer bestimmten "Uung zu einem bestimmten Gewicht, welches bei der "Ubung genau 1 Mal bew"altigt werden kann:

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

Wenn man sich also das Trainingsgewicht ausrechnen will, mit dem man eine "Ubung 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 "Ubung 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 k"onnen.
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.

12 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 (R"ucken & Gesundheit)) werden in dieser Phase die "Ubungen ausgewählt und die Anzahl der S"atze/Wiederholungen bestimmt:

	Anfänger		Fortgeschrittener	
Baum - Beine - Po	"Ubungen:	9	"Ubungen:	9
	S"atze:	2	S"atze:	3
	Wiederholungen:	20	Wiederholungen:	25
Oberk"orper - Arme	"Ubungen:	6	"Ubungen:	8
	S"atze:	2	S"atze:	3
	Wiederholungen:	20	Wiederholungen:	25
Stabilisation	"Ubungen:	8	"Ubungen:	8
	S"atze:	2	S"atze:	3
	Wiederholungen:	20	Wiederholungen:	25

In Phase 1 werden alle "Ubungen 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 H"alfte.
Im "Uberblick:

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

13 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 w"ochentlich Phase 2: Muskelaufbau -training oder 1 Mal w"ochentlich Stabilit"ats"ubungen. Welche der Benutzer ausf"uhren will kann er selbst am Anfang entscheiden, je nachdem ob er seinen K"orper formen will, oder ob er es als Gesundheits- oder Reha"ubung macht.
Also im "Uberblick:

1. Teil	Phase 1: Allgemein
"Ubungen:	6
Gewicht:	55% RM
S"atze:	2
Wiederholungen:	25
Pausendauer:	30-60 Sekunden
Wochentage:	2
Phasendauer:	8 Wochen

2. Teil	Phase 2: Muskelaufbau	Stabilit"ats"ubungen
"Ubungen:	6	6
Gewicht:	80% RM	Keines
S"atze:	2	3
Wiederholungen:	25	12-20
Pausendauer:	30-60 Sekunden	60-120 Sekunden
Wochentage:	1	1
Phasendauer:	8 Wochen	8 Wochen

14 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 "Ubung egal ist. Es wird davon ausgegangen, dass ein Anf"anger nach der 8-w"ochigen Phase 1 bereits fit genug ist, um alle "Ubungen 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 "Ubungen. Bei dem Splittraining werden "Ubungen zu einer bestimmten Muskelgruppe bei jeder Trainingseinheit durchgeführt. Umgekehrt wird bei dem Ganzk"orpertraining 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 S"atzen und 12 Wiederholungen. Je nach Anzahl der fokusierten Muskelgruppen bestimmt sich die Zahl der "Ubungen, die man bekommt: Bei 2 Muskelbereiche sind es 6 "Ubungen, bei 3 sind es 9 "Ubungen. Nach dieser 8-wöchigen Phase kommt Phase 3: Muskelaufbau.
Im "Uberblick:

	Muskelaufbau	Kraftausdauer
"Ubungen:	6 oder 9	6 oder 9
Gewicht:	80% RM	80% RM
S"atze:	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.

15 Phase 2: Maximalkraft Phase 3: Muskelaufbau

Diese Phase ist für das Maximalkrafttrainingsziel die Phase 2 und f"ur das Muskelaufbautrainingsziel die Phase 3. Nur Fortgeschrittene oder User die die Phase 2: Muskelaufbau durchgef"uhrt haben, k"onnen diese Phase beginnen.
Maximalkraft"ubungen sind immer Ganzk"orper"ubungen. Das empfohlene Trainingsgewicht liegt bei 95% RM. Dabei kommen ausschließlich Seilzug- und Hantel"ubungen vor (6). Satzdauer beträgt 90-120 Sekunden, bei 3 S"atzen 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 "Uberblick:

"Ubungen:	6
Gewicht:	95% RM
S"atze:	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), h"ort auf, oder l"asst sich erneut einen generieren.

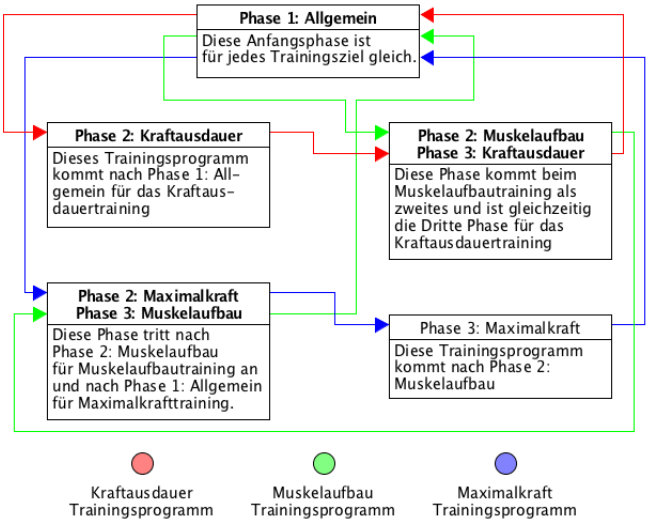
16 Phase 3: Maximalkraft

Phase 3: Maximalkraft kommt nach Phase 2: Maximalkraft. Hierbei wird jediglich 2 Mal w"ochentlich trainiert. Die Tage kann sich der Benutzer wieder aussuchen, einzige Bedingung sind 48 Stunden Erholungszeit. Die Phase besteht wieder aus Seilzug- und Hantel"ubungen, insgesamt 6 mit jeweils 2 Wiederholungen zu 5 Sätzen. Hierbei wird das Trainingsgewicht erneut f"ur ca. 95%RM gew"ahlt.
Im "Uberblick:

"Ubungen:	6
Gewicht:	95% RM
S"atze:	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), h"ort auf, oder l"asst sich erneut einen generieren.

17 Visualisierung



18 Mobilisation

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

Mobilisation	**Beschreibung**
Hals	10cmDen Kopf im Wechsel nach rechts und links drehen.
Schulter	10cmDie Arme neben dem Körper hängen lassen und mit den
Ellbogen	10cmDie Hände auf die Schulter legen, mit den Ellbogen vorwärts
Handgelenk	10cmHände kreisen, beide Hände gleichzeitig mit größtmöglichem
Becken-Mob	10cmDie Arme über den Köpf führen, Handflächen nach oben schi
Wirbelsäule – Seitneigung	10cmLinken Arm seitwärts hoch heben über den Kopf und Wirbels
Wirbelsäule – Rotation	10cmBauchnabel nach innen ziehen, die Arme in U-Form anheben,
Wirbelsäule – Rolldown	10cmAufrechter Stand, den Kopf Richtung Brustbein senken, Bauc