

FIPLY

Daniel Bersenkowitsch, Andreas Denkmayr, Gerald Irsiegler

19. Februar 2016

Inhaltsverzeichnis

1 Eidesstattliche Erklärung	6
2 Abstract	7
3 Danksagungen	8
4 Autoren der Diplomarbeit	9
5 Auftraggeber	10
6 Betreuungslehrer	10
7 Pflichtenheft	11
7.1 Motivation	11
7.2 Ausgangssituation und Zielsetzung	11
7.2.1 Ausgangssituation	11
7.2.2 Beschreibung des Geschäftsfeldes	11
7.2.3 Zielbestimmung	11
7.3 Funktionale Anforderungen	11
7.3.1 Use Case Diagram (1 bis 6)	13
7.3.2 Systemarchitektur	14
7.3.3 Use Case Details	14
7.4 Nicht-funktionale Anforderungen	28
7.5 Mengengerüst	28
7.6 Risikovermeidung	29
7.7 Abnahmekriterien	29

8	Der Trainingsplan	42
8.1	Begriffserklärung	42
8.2	Einleitung	42
8.3	Phase 1: Allgemein	43
8.4	Phase 2: Kraftausdauer (Gesundheit)	44
8.5	Phase 2: Muskelaufbau Phase 3: Kraftausdauer	45
8.6	Phase 2: Maximalkraft Phase 3: Muskelaufbau	46
8.7	Phase 3: Maximalkraft	47
8.8	Mobilisation	47
9	Verwendete Technologien	49
9.1	GitHub	49
9.1.1	Repositories	49
9.1.2	Git	49
9.1.3	Arbeitskopie	50
9.1.4	Lokales Repository	50
9.1.5	Entferntes Repository	50
9.1.6	Commit	50
9.1.7	Push	51
9.1.8	Branch	51
9.1.9	Pull Request	51
9.1.10	Community	51
9.1.11	Integration	51
9.1.12	Issue	52
9.1.13	.gitignore	52
9.1.14	GitHub Desktop	52
9.2	Android Studio	54
9.2.1	Vorteile	54
9.2.2	Nachteile	54
9.2.3	IntelliJ	54
9.3	LaTeX	55
9.3.1	LaTeX	55
9.3.2	TexStudio	55
9.3.3	Subfiles	55
9.3.4	Quellen und Zitate	57
9.4	Adobe Photoshop	58

10 Planung	59
10.1 Designrichtlinien	59
10.1.1 Der Splashscreen	59
10.1.2 Animationen und Transactions	59
10.1.3 Navigation	60
10.1.4 Formulare	61
10.1.5 Datenausgabe	63
10.2 Datenanbindung	65
10.2.1 Verwendungszweck	65
10.2.2 Vorher	65
10.2.3 Nacher	66
10.2.4 Anwendung	67
11 Umsetzung	71
11.1 Permissions	71
11.1.1 Arten von Permissions	71
11.1.2 bis Android 5.1 (API level 22)	72
11.1.3 ab Android 6.0 (API level 23)	72
11.1.4 Permission groups	73
11.2 Fragments	74
11.2.1 Was sind Fragments?	74
11.2.2 Der Lifecycle	75
11.2.3 Fragment Transactions	75
11.2.4 Verwendung von Fragments	76
11.3 Bundles	77
11.3.1 Datenübertragung zwischen Fragments	77
11.3.2 Datenübertragung zwischen Activities	78
11.4 NavigationDrawer	80
11.5 Benutzerverwaltung	83
11.5.1 Beschreibung	83
11.5.2 Aufteilung der Verwaltung	83
11.5.3 Implementierung	87
11.6 Exportieren	88
11.6.1 Als CSV exportieren - OpenCSV	88
11.6.2 Emails senden	90
11.7 Trainingsplan	91
11.7.1 Die Ansicht	91
11.7.2 Das Erstellen	92
11.7.3 Umsetzung des Algorithmus	93
11.8 Übungskatalog	94
11.8.1 Beschreibung	94

11.8.2	Implementierung	94
11.8.3	DetailView	97
11.8.4	Filter	98
11.9	Lösung für die Videodarstellung	102
11.9.1	VideoView	102
11.9.2	Youtube Android Player API	102
11.9.3	WebView	103
11.9.4	Nutzwertanalyse	103
11.9.5	Erfüllung der KO-Kriterien	103
11.9.6	Schlussfolgerung	103
11.10	Musik	104
11.10.1	Lokalisierung der Musikdateien	104
11.10.2	Verwalten von Playlists	106
11.10.3	Abspielen der Playlists.	107
11.10.4	MusicControls	110
11.11	Statistik	111
11.11.1	Beschreibung	111
11.11.2	Verfügbare Statistiken	112
11.11.3	Implementierung	113
11.12	Datenbank	114
11.12.1	Beschreibung	114
11.12.2	SQLite	114
11.12.3	ContentProvider	114
11.12.4	Zugriff auf die Daten	114
11.12.5	Contract	118
11.13	Social Media	119
11.13.1	Beschreibung	119
11.13.2	Verknüpfung mit Facebook	119
11.14	Design	121
11.14.1	FIPLY-Logo	121
11.14.2	Kontrollelemente	123
12	Kommerzialisierung	125
12.1	Advertising mit AdMob	125
12.1.1	Banner Ads	126
12.1.2	Interstitial Ads	127
12.2	Donations	128
12.2.1	Flattr	128
12.2.2	PayPal	129
12.2.3	Donations über In-App-Käufe	129
12.2.4	Zweite kostenpflichtige App	129

12.3	Freemium	130
12.3.1	Consumable Items	130
12.3.2	Non-Consumable Items	130
12.3.3	Subscriptions	130
12.3.4	Gratis/Bezahlte Versionen	131
12.3.5	Fixpreis	132
12.3.6	SellApp	133
12.4	Promotion	134
12.4.1	Website	134
12.4.2	Social Media Reputation	135
12.4.3	Presse	136
12.4.4	Wettbewerbe	136
12.4.5	Persönliche Werbung	136
12.4.6	Reklame	137
12.4.7	In Appstore Optimierung (IAO)	137
12.4.8	Fazit	138

1 Eidesstattliche Erklärung

Wie erklären hiermit an Eides statt, dass wir die vorliegende Diplomarbeit selbstständig angefertigt haben. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Daniel Bersenkowitsch

Gerald Irsiegler

Andreas Denkmayr

April, 2016

2 Abstract

Deutsch

Im Zuge der Diplomarbeit von Daniel Bersenkowitsch, Andreas Denkmayr und Gerald Irsiegler, wird eine Fitnessplan Applikation für den Auftraggeber David Lindenbauer entwickelt. Derzeit muss Herr Lindenbauer den Fitnessplan für seine Kunden immer selbst per Hand erstellen. Dies ist eine zeitaufwändige Tätigkeit, die im Prinzip automatisiert werden kann. Um Zeit zu sparen und Prozesse zu verkürzen, entwickeln wir eine Applikation, die diesen Aufgabenbereich übernimmt. Erweitert wird dieses Produkt durch einen aktive Trainingsunterstützung und einen Übungskatalog zum Nachschlagen.

Englisch

In the course of the mandatory thesis of Daniel Bersenkowitsch, Andreas Denkmayr and Gerald Irsiegler, a fitness planning app is being developed for the customer David Lindenbauer. Currently Mr. Lindenbauer has to create each individual workout plan by hand. This is a very timeconsuming process, that can theoretically be automated. To save time we are developing an application which generates workout plans based on the needs and wants of the trainees. Additionally the application helps with the training itself and features an exercise catalog to refer to.

3 Danksagungen

Wir möchten uns herzlichst bei allen bedanken, die uns bei unserem Projekt unterstützt haben und zu unserem Gelingen beigetragen haben:

- Bei den Lehrern, die uns während des Projektes zur Hilfe bereit standen und uns bei Fragen behilflich waren.
- Bei unserem Betreuungslehrer Mag. Dr. Prof. Thomas Stütz, der sich dazu entschloss uns bei der Diplomarbeit mit Rat und Tat zur Seite zu stehen.
- Bei allen Korrekturlesern der Dokumente, um diese fehlerfrei und ordentlich einzureichen.
- Bei unseren Eltern, die uns während den Arbeiten am Projekt durch Verpflegung unterstützen und Ihre Objektive Meinung zu Gestaltungen einbrachten.
- Bei unseren hilfsbereiten Mitschülern, die uns konstruktive Kritik und wichtiges Feedback gegeben haben.
- Und natürlich bei unserem Projektauftraggeber David Lindenbauer, der diese Diplomarbeit erst möglich gemacht hat und uns durch Engagement und Interesse den Weg erleichterte.

4 Autoren der Diplomarbeit

Alle Autoren der Diplomarbeit gehen in die HTBLA-Leonding und besuchen die 5. Schulstufe des Informatik-Zweiges.

Daniel Bersenkowitsch

Adresse:
Haidbachstraße 31
4061 Pasching
Österreich

Email: daniel@berdanex.net



Gerald Irsiegler

Adresse:
Zentastrasse 8
4061 Pasching
Österreich

Email: gerald.irsiegler@gmail.com



Andreas Denkmayr

Adresse:
Unterstiftung 2
4190 Bad Leonfelden
Österreich

Email: andreas.denkmayr@gmail.com



5 Auftraggeber

David Lindenbauer

Dipl. Fitness- & Personaltrainer

Email: david.lindenbauer@gmx.at



6 Betreuungslehrer

Mag. Dr. Prof. Thomas Stütz

Email: t.stuetz@htl-leonding.ac.at

Unterrichtsfächer:

- Angewandte Datentechnik
- Java Zertifizierung
- Kommunikation und Lernkultur
- Programmieren
- Projektentwicklung
- Virtuelles Lernen mit LM-System



7 Pflichtenheft

7.1 Motivation

Das Projekt wird im Rahmen einer Diplomarbeit durchgeführt.

7.2 Ausgangssituation und Zielsetzung

7.2.1 Ausgangssituation

Fitnessapps gibt es wie Sand am Meer. Die Meisten davon sind nach demselben Prinzip aufgebaut: Es werden eine Reihe von Übungen vorgestellt und deren Ausführung beschrieben, meist nur in Textform. Der User kann sich anhand dieses Übungskatalogs sein Workout selbst zusammenstellen. Jedoch wissen vor allem Anfänger oft nicht, in welcher Intensität und in welcher Reihenfolge ein Training sinnvoll ist.

7.2.2 Beschreibung des Geschäftsfeldes

Die Applikation kann von jedem benutzt werden, der körperlich aktiv sein will. Der Benutzer lädt sich die Applikation vom Google PlayStore herunter, gibt sein persönliches Profil und Trainingsziel an und erhält seinen angepassten Trainingsplan. Die App kann sowohl zu Hause als auch im Fitness-Studio verwendet werden. Nach jeder Trainingseinheit werden seine Fortschritte vermerkt und können eingesehen werden.

7.2.3 Zielbestimmung

Die User soll bei dem Verfolgen seiner Ziele (Muskelaufbau, Gesundheit, Abnehmen) durch die App unterstützt werden.

7.3 Funktionale Anforderungen

Die persönlichen Profildaten werden beim ersten Öffnen der Applikation erfasst. Der Benutzer kann jederzeit einen Trainingskatalog einsehen, in der alle Übungen grafisch nach Muskelgruppen sortiert sind. Jede Übung besitzt eine detaillierte Beschreibung, Anleitung und ein Video der Ausführung. Sobald man alle nötigen Informationen angegeben hat, ist es möglich einen Trainingsplan generieren zu lassen, welcher als Excel-Tabelle abgespeichert und verschickt werden kann. Wenn man zu trainieren beginnen möchte, kann man eine Trainingssession starten. Dabei gibt es einen eingebauten Musikplayer, der die lokale Musikbibliothek wiedergibt. Die aktuelle Übung wird angezeigt

und weiters steht ein Timer zur Verfügung, der vor allem beim Konditions-training zum Einsatz kommt. Nach jedem Trainingsablauf soll der Benutzer ein Feedback geben, um seine Fortschritte zu vermerken. Die Fortschritte können jederzeit eingesehen und auf Social Networks geteilt werden. Zusätzlich werden einmal wöchentlich Userdaten (z.B.: Gewicht und Gesundheits-zustand) abgefragt, um seine Entwicklung akkurat darstellen zu können.

7.3.1 Use Case Diagram (1 bis 6)

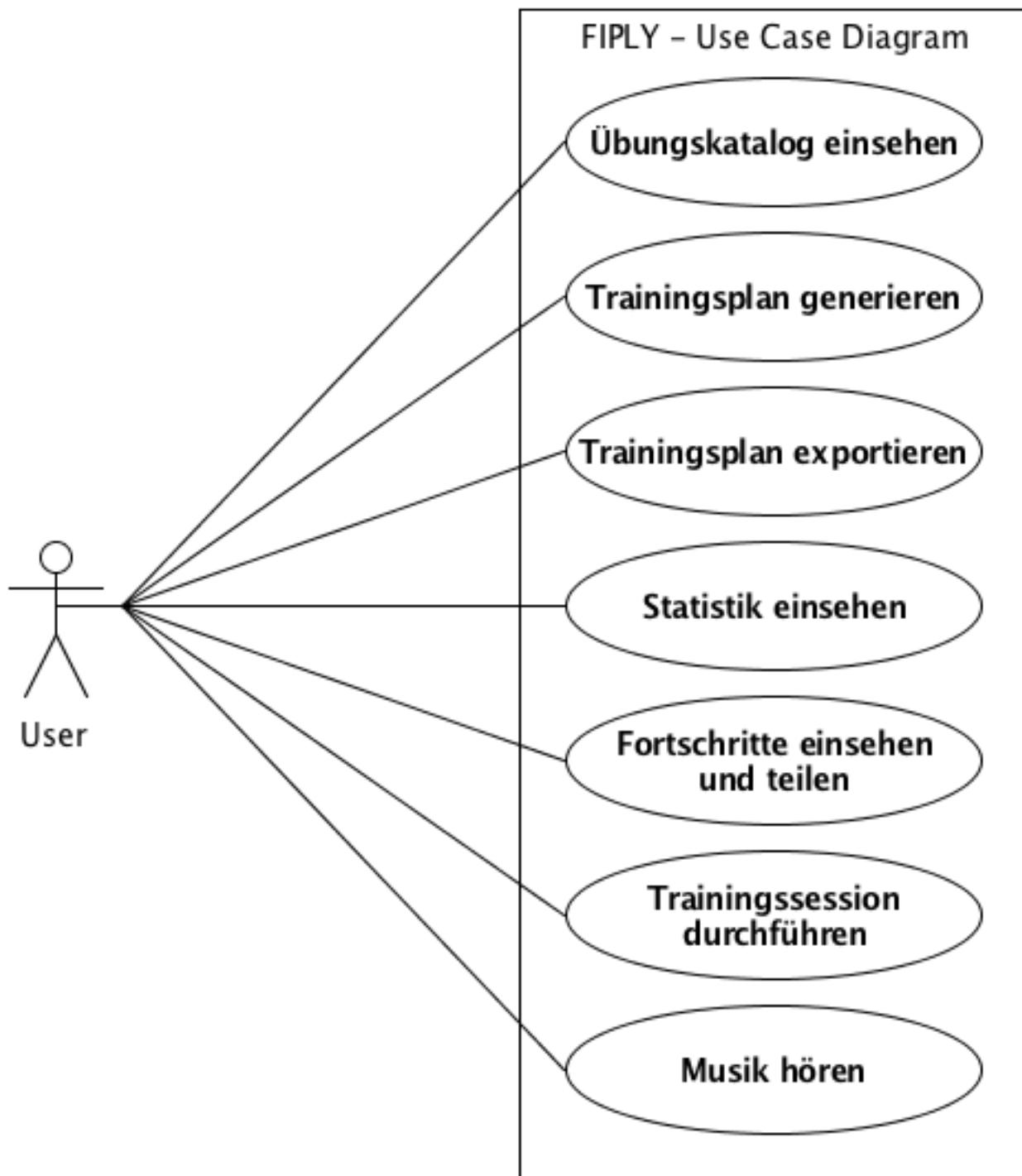


Abbildung 6: Die Systemarchitektur ¹³ der Applikation

7.3.2 Systemarchitektur

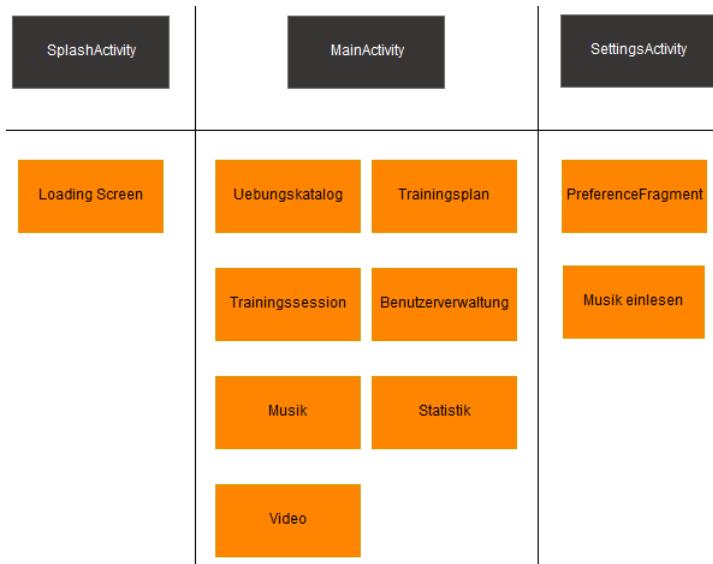


Abbildung 7: Das Use-Case Diagramm zu der Applikation

7.3.3 Use Case Details

UseCase 1: Übungskatalog einsehen

UseCase 1:	Übungskatalog einsehen
Ziel des Use Cases:	Dem Benutzer soll eine Übersicht über alle Übungen geboten werden, falls er manuell einen Trainingsplan zusammenstellen oder sich über Übungen informieren will.
Umgebende Systemgrenze:	Die Applikation selbst ist die Systemgrenze.
Vorbedingung:	Keine.
Nachbedingung bei erfolgreicher Ausführung:	Keine.
Beteiligte Nutzer:	Der Benutzer der App.
Auslösendes Ereignis:	Durch das Betätigen des Knopfes „Übungen“.

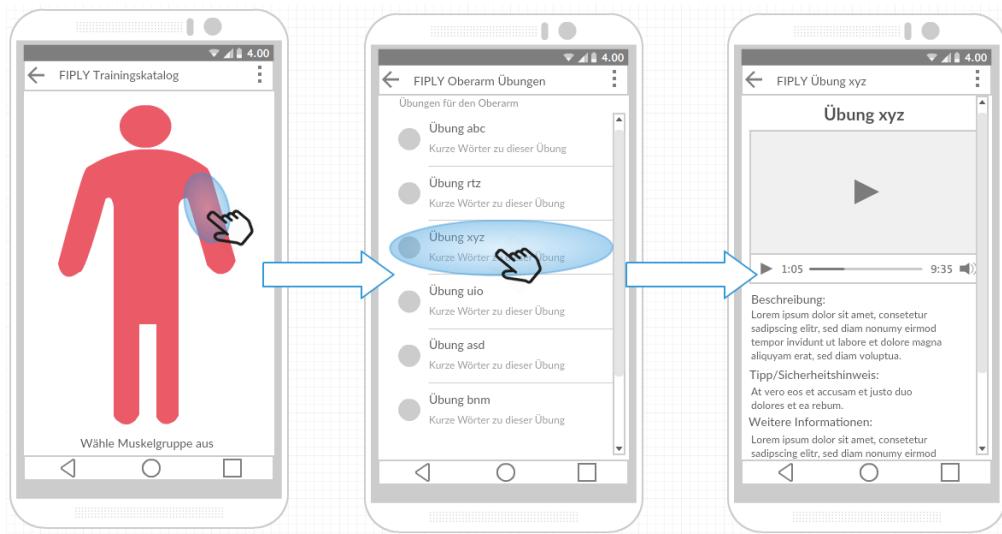


Abbildung 8: GUI für den Aufruf/Ablauf des 1. Use Cases:

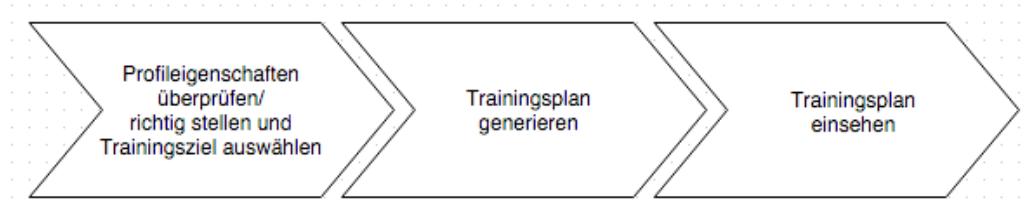


Abbildung 9: Prozesskette des 1. Use Cases:

Szenario für den Standardablauf: (Erfolg)

Schritt	Nutzer	Beschreibung der Aktivität
1: Trainingskatalog Übersicht wird geöffnet.	Der Benutzer der App.	Es wird grafisch eine Menschenfigur angezeigt und der Benutzer wird gebeten einen Körperteil auszuwählen um die spezifischen Übungen einzusehen.
2: Anzeigen der Übungen.	Der Benutzer der App.	Es wird eine Liste von Übungen angezeigt, die der gewählten Muskelgruppe entsprechen.
3: Anzeigen einer ausgewählten Übung.	Der Benutzer der App.	Es wird ein kurzes Video zur korrekten Ausführung der Übung präsentiert, inklusive Beschreibung, Tipps bzw. Sicherheitshinweise und eventuellen weiteren Informationen.

UseCase 2: Trainingsplan

UseCase 2:	Trainingsplan
Ziel des Use Cases:	Dem Benutzer soll ein individuell zusammengestellter Trainingsplan zur Verfügung gestellt werden.
Umgebende Systemgrenze:	Die Applikation selbst ist die Systemgrenze.
Vorbedingung:	Alle benötigten Profildaten (Größe, Gewicht, Geschlecht, Trainingsziel) müssen angegeben werden.
Nachbedingung bei erfolgreicher Ausführung:	Keine.
Beteiligte Nutzer:	Der Benutzer der App.
Auslösendes Ereignis:	Durch das betätigen des Knopfes „Trainingsplan“.

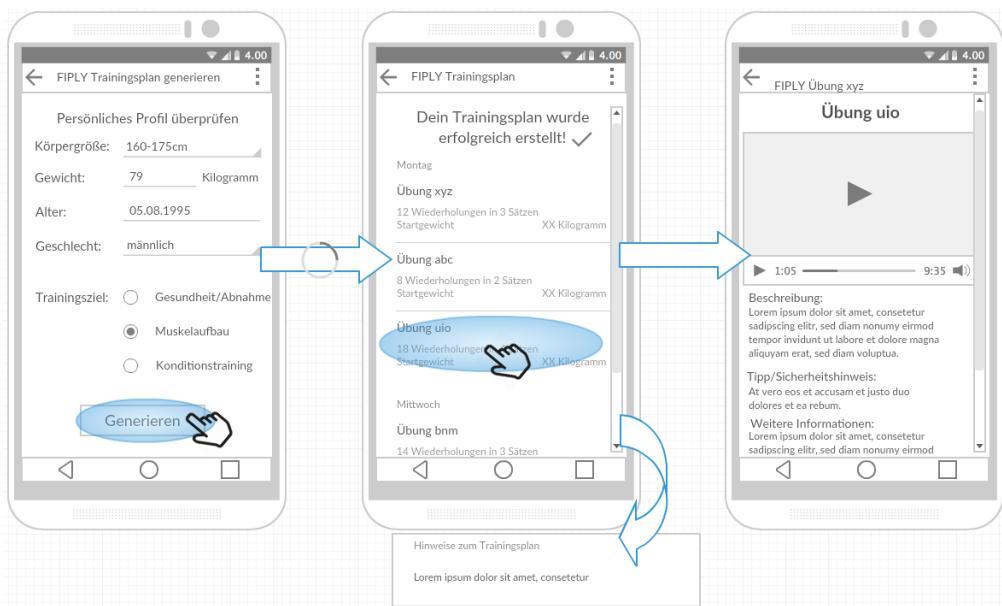


Abbildung 10: GUI für den Aufruf/Ablauf des 2. Use Cases:

Eingabefeld	Erlaubte Eingabewerte
Körpergröße (slider)	100 bis 200 (Zentimeter)
Gewicht (slider)	40 bis 140 (Kilogramm)
Alter (drop-down menu)	16 bis 20, 21 bis 30, 31 bis 40, 41 bis 50, 51+
Geschlecht (Drop-Down Menü)	männlich, weiblich, anderes
Körperbau (Drop-Down Menü)	fit (trainiert), not fit (untrainiert)

Szenario für den Standardablauf: (Erfolg)

Schritt	Nutzer	Beschreibung der Aktivität
1: Daten des persönlichen Profils überprüfen.	Der Benutzer der App.	Die Daten werden vom Benutzer überprüft und wenn nötig angepasst. Nach der Bestimmung des Trainingsziels kann der Trainingsplan generiert werden.
2: Einsehen des generierten Trainingsplan.	Der Benutzer der App.	Der Trainingsplan wird nun angezeigt.
3: Anzeigen einer ausgewählten Übung.	Der Benutzer der App.	Es wird eine detaillierte Beschreibung, eine Anleitung und ein Video der Ausführung angezeigt.

Szenarien für alternative Abläufe: (Misserfolg oder Umwege zum Erfolg)

Schritt	Bedingung, unter der Alternative eintritt	Beschreibung der Aktivität
1	Falsche Eingabe bei den Eingabefeldern	Die App fordert den Nutzer auf seine fehlerhaften Daten anzupassen.

UseCase 3: Trainingsplan exportieren lassen

UseCase 3:	Trainingsplan exportieren lassen
Ziel des Use Cases:	Es soll gewährleistet werden, dass der Benutzer auch ohne Smartphone trainieren gehen kann.
Umgebende Systemgrenze:	Die Applikation selbst ist die Systemgrenze.
Vorbedingung:	Ein Trainingsplan muss bereits erstellt worden sein (Use Case 2).
Nachbedingung bei erfolgreicher Ausführung:	Keine.
Beteiligte Nutzer:	Der Benutzer der App.
Auslösendes Ereignis:	Durch das Betätigen des Knopfes „Exportieren“.

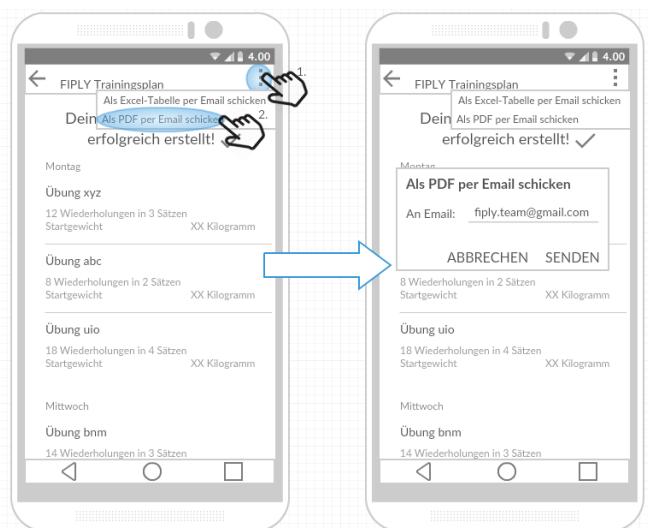


Abbildung 11: GUI für den Aufruf/Ablauf des 3. Use Cases:

Eingabefeld:	Erlaubte Eingabewerte
Email	Korrekte Emailadresse

Szenario für den Standardablauf: (Erfolg)

Schritt	Nutzer	Beschreibung der Aktivität
1: Der Benutzer will den Trainingsplan exportieren.	Der Benutzer der App.	Der Trainingsplan wird als Excel-Tabelle exportiert.
2: Email angeben.	Der Benutzer der App.	Die Emailadresse an die die Datei gesendet werden soll wird angegeben.

Szenarien für alternative Abläufe (Misserfolg oder Umwege zum Erfolg)

Schritt	Bedingung, unter der Alternative eintritt	Beschreibung der Aktivität
2: Email angeben	Irrtum des Benutzers.	In das Email Textfeld wird eine nicht formatkorrekte Email angegeben.

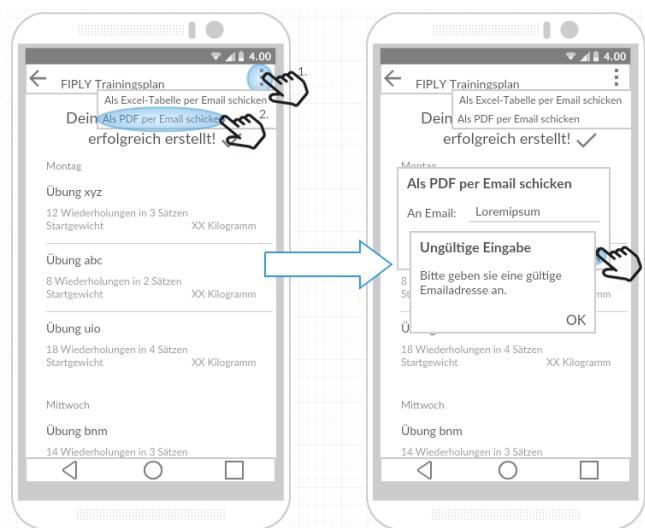


Abbildung 12: GUIs für den alternativen Ablauf des Use Cases:

Eingabefeld:	Erlaubte Eingabewerte
Email (Textfeld)	Gültige ist eine formatkorrekte Emailadresse. Alles andere wird als nicht valide erkannt.

UseCase 4: Trainingssession durchführen und Musik hören.

UseCase 3:	Trainingssession durchführen und Musik hören.
Ziel des Use Cases:	Der Benutzer bleibt beim Trainieren konzentriert, kann seine Fortschritte festhalten und wird zusätzlich durch die Musik motiviert.
Umgebende Systemgrenze:	Die Applikation selbst ist die Systemgrenze.
Vorbedingung:	Ein Trainingsplan muss aufbereitet worden sein (Use Case 3).
Nachbedingung bei erfolgreicher Ausführung:	Keine.
Beteiligte Nutzer:	Der Benutzer der App.
Auslösendes Ereignis:	Durch das Betätigen des Knopfes „Training“.

GUI für den Aufruf/Ablauf des Use Cases:

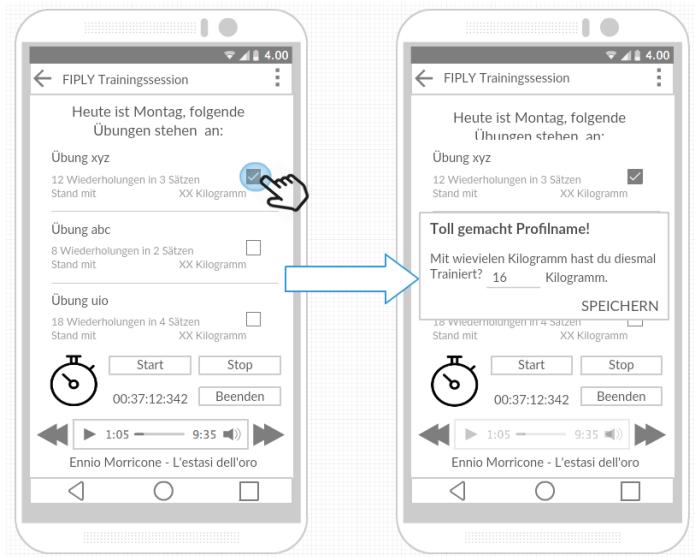


Abbildung 13: GUI für den Aufruf/Ablauf des 4. Use Cases:

Eingabefeld:	Erlaubte Eingabewerte
Kilogramm (Textfeld)	40 bis 140

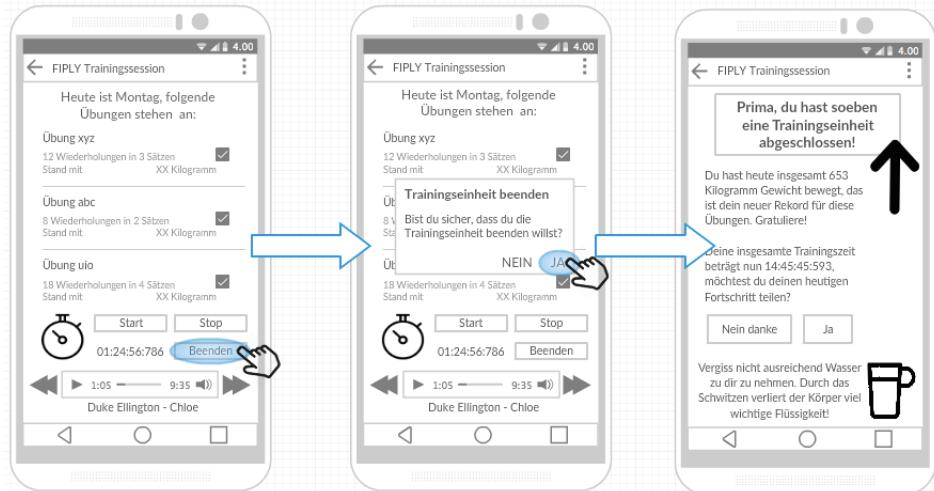


Abbildung 14: GUI für den Aufruf/Ablauf des 4. Use Cases:

Szenario für den Standardablauf: (Erfolg)

Schritt	Nutzer	Beschreibung der Aktivität
1: Starten des Musikplayers.	Der Benutzer der App.	Die Applikation spielt ein Lied aus der lokalen Musikbibliothek ab.
2: Ausführen der Übungen.	Der Benutzer der App.	Der Benutzer führt nun die anstehenden Übungen aus.
3: Eine Übung abschließen.	Der Benutzer der App.	Sobald eine Übung abgeschlossen ist, kann er per Knopfdruck zur nächsten Übung springen.
4: Trainingseinheit abschließen.	Der Benutzer der App.	Wenn alle Übungen abgeschlossen sind und der „Trainingsession beenden“-Knopf gedrückt wurde, wird sein Trainingserfolg wiedergegeben.

Szenarien für alternative Abläufe: (Misserfolg oder Umwege zum Erfolg)

Schritt	Bedingung, unter der Alternative eintritt	Beschreibung der Aktivität
Es wird eine Übung angeklickt.	Der Benutzer kennt die Übung nicht oder will sie einsehen.	Die Übung wird während einer Trainingseinheit angeklickt und die Details dazu werden geöffnet.

GUIs für den alternative Abläufe des Use Cases:

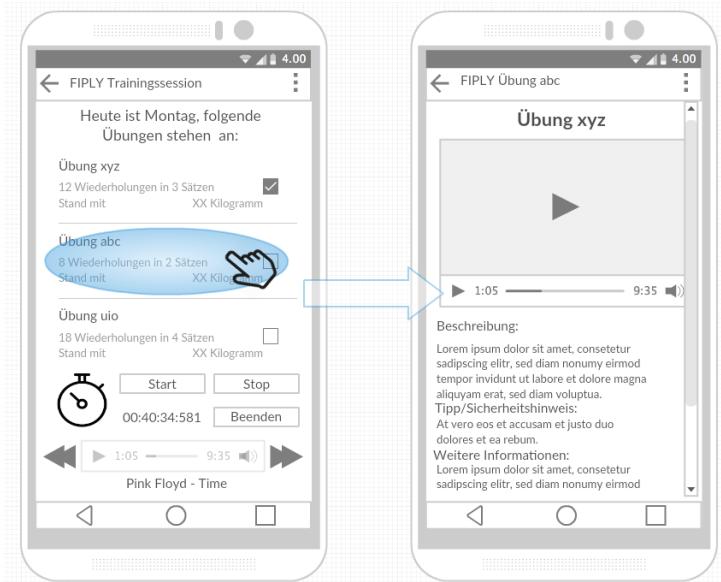


Abbildung 15: GUI für den alternativen Aufruf/Ablauf des 4. Use Cases:

Das Startgewicht ist das Gewicht mit dem der Benutzer am Anfang üben soll. Sobald er mit schwereren Gewichten zu trainieren beginnt, kann er die Veränderung eintragen. Dies wird dann in der Trainingsentwicklung gespeichert.

UseCase 5: Fortschritte/Statistik anzeigen lassen.

UseCase 3:	Fortschritte/Statistik anzeigen lassen.
Ziel des Use Cases:	Der Benutzer soll durch die Visualisierung seiner Fortschritte motiviert werden.
Umgebende Systemgrenze:	Die Applikation selbst ist die Systemgrenze.
Vorbedingung:	Keine.
Nachbedingung bei erfolgreicher Ausführung:	Keine.
Beteiligte Nutzer:	Der Benutzer der App.
Auslösendes Ereignis:	Durch das Betätigen des Knopfes „Statistik“.

GUI für den Aufruf/Ablauf des Use Cases:

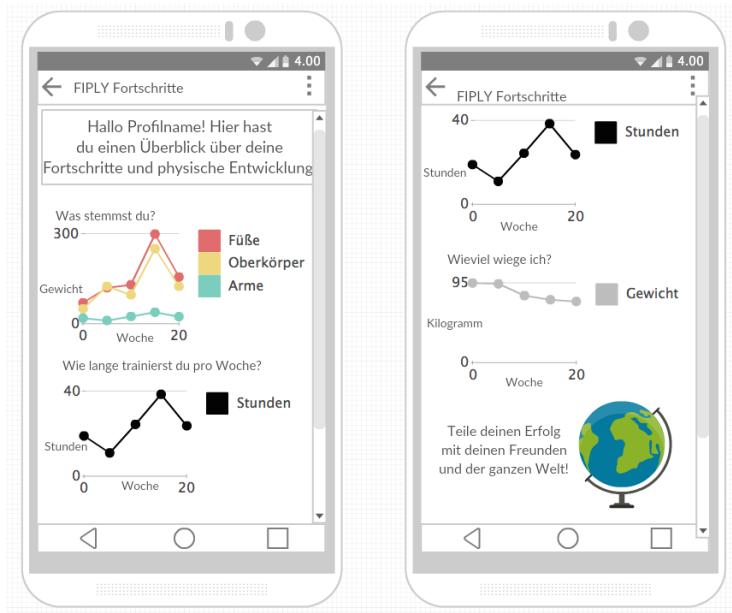


Abbildung 16: GUI für den alternativen Aufruf/Ablauf des 5. Use Cases:

Szenario für den Standardablauf: (Erfolg)

Schritt	Nutzer	Beschreibung der Aktivität
1: Einsehen der Fortschritte.	Der Benutzer der App.	Anzeige der Momentanaufnahmen aus den Trainingssessions.

UseCase 6: Fortschritte auf Facebook teilen.

UseCase 3:	Fortschritte/Statistik anzeigen lassen.
Ziel des Use Cases:	Dadurch soll eine Community aufgebaut werden und andere dazu motiviert werden die Applikation auch zu benutzen.
Umgebende Systemgrenze:	Die Applikation selbst ist die Systemgrenze.
Vorbedingung:	Ein oder mehrere Trainingssessions müssen durchgeführt worden sein.
Nachbedingung bei erfolgreicher Ausführung:	Keine.
Beteiligte Nutzer:	Der Benutzer der App.
Auslösendes Ereignis:	Durch das Betätigen des Knopfes „Erfolg teilen“.

GUIs für den Aufruf/Ablauf des Use Cases:

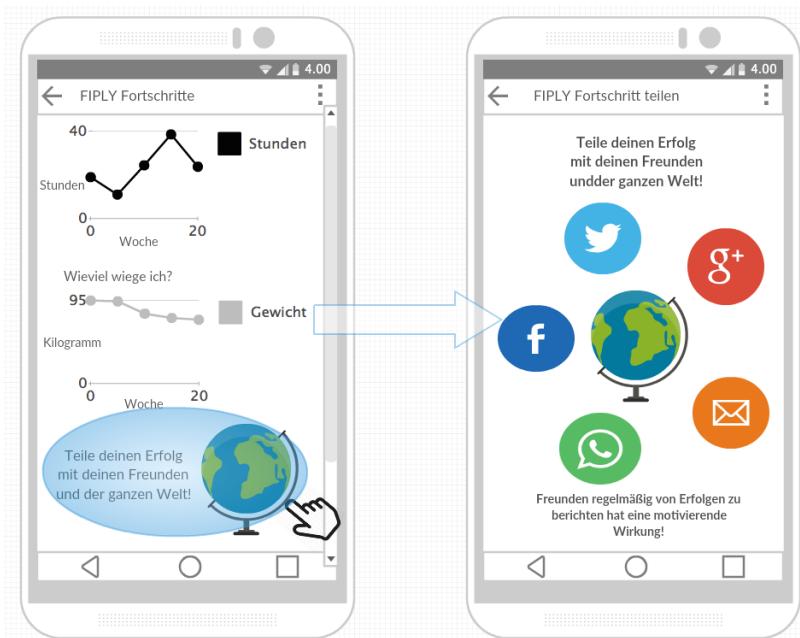


Abbildung 17: GUI für den Aufruf/Ablauf des 6. Use Cases:

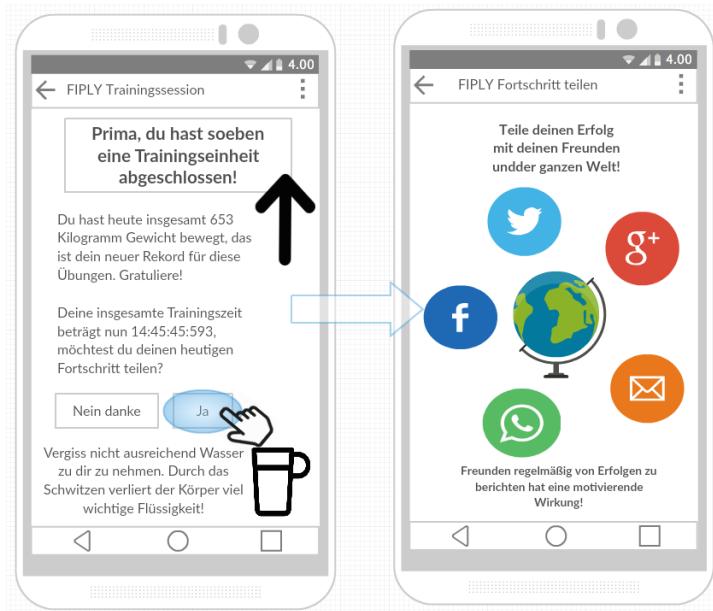


Abbildung 18: GUI für den Aufruf/Ablauf des 6. Use Cases:

7.4 Nicht-funktionale Anforderungen

Name:	Intuitive Oberfläche
Typ:	USE
Beschreibung:	Die Applikation soll leicht bedienbar sein. Übersichtliches Design.
Zugeordnete Use Cases	1,2,3,4,5 und 6.

Name:	Einfach verständlich für Fitnesslaien
Typ:	USE
Beschreibung:	Auch Fitnesslaien sollen sich in der App zurechtfinden und nicht mit Fachbegriffen überfordert werden.
Zugeordnete Use Cases	1,2,4 und 6.

Name:	Community einrichten
Typ:	USE
Beschreibung:	Es soll eine Community auf Facebook aufgebaut werden, um die Nutzer motiviert zu halten.
Zugeordnete Use Cases	1,2,4 und 6.

Name:	Offlineverwendung
Typ:	USE
Beschreibung:	Die Applikation soll auch ohne aktiver Internetverbindung benutzbar sein.
Zugeordnete Use Cases	1,2,4 und 6

7.5 Mengengerüst

Übungskatalog 50-100 Datensätze einmalig in die Übungskatalogtabelle (kaum neue Sätze). Trainingssession 100-200 Datensätze pro Jahr in die Trainingssession-Tabelle. Usersnapshot: 50-55 Datensätze pro Jahr in die Usersnapshot-Tabelle. 50-100 Videos werden von Youtube eingebunden. Zusätzlich werden 50-200 komprimierte Fotos abgespeichert werden.

7.6 Risikovermeidung

Sollte sich ein Nutzer bei Verwendung der Apps psychische oder physische Schäden davontragen trägt er selbst die Verantwortung.

7.7 Abnahmekriterien

Die App ist im Google Playstore verfügbar. Man kann über eine intuitive Oberfläche ein Profil erstellen und sich einen Trainingsplan generieren lassen. Zusätzlich kann eine Trainingssession durchgeführt werden und Informationen über seine Entwicklung stehen zur Verfügung. Erfolge können über Facebook geteilt werden.

Arbeitsunterteilung

Protokoll vom 24.09.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler

Autor des Protokolls: Andreas Denkmayr**Dauer:** 12:40 – 13:45**Menüpunkte:**

- Durchgehen des Pflichtenhefts
- Arbeitsunterteilung

Inhalt:

Durchgehen des Pflichtenhefts, um Missverständnisse zu beseitigen und Klarheit zu schaffen

Grobaufteilung der Bereiche:**Irsiegler:** Formular, User Management und Übungskatalog**Denkmayr:** Trainingssession Ablauf (Musik, Abbildung auf Karte)**Bersenkowitsch:** Trainingspläne generieren, zuteilen und exportieren(Airprint, Excel)**Jeder soll sich hier beteiligen:**

Einbindung von Social Networks in seinem jeweiligen Bereich

Marktforschung

Marketing/Vertrieb

Bemerkungen und Kommentare:

Ob diese Einteilung in Ordnung ist, muss noch abgeklärt werden

Zusammenfassung:

Die Grobunterteilung der Bereiche wurde vorgenommen und unter den Projektmitgliedern aufgeteilt.

Erstes Fitnessconsultant-Treffen

Protokoll vom 11.10.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler,
- David Lindenbauer

Autor des Protokolls: Andreas Denkmayr

Dauer: 15:00 – 17:15

Menüpunkte:

- Erwartete Funktionen
- Pflichtenheftüberarbeitung

Inhalt:

Altes Pflichtenheft wurde gemeinsam mit dem Fitnessconsultant besprochen.

Anschließend gab es eine Beratung mit unserem Fitnessconsultant über die Features, die von unserer App erwartet werden.

Videos:

Videos werden auf Youtube-Channel geladen und eingebunden in den Trainingskatalog.
Videos werden mit Watermark versehen und von unserem Fitnessconsultant erstellt,
wobei Videos sowohl von Männern als auch von Frauen durchgeführt werden sollen.
Videos werden ca. 10 Sekunden Durchschnittslänge aufweisen und sprachlich begleitet sein.

Übungsliste:

Die Übungsliste wird von unserem Fitnessconsultant erstellt und an uns weitergegeben.

Trainingsziele:

Die Trainingsziele: „Maximalkraft“, „Muskelaufbau“, „Ausdauer“, „Abnehmen“ wurden besprochen.

Trainingssession:

Für Ausdauertraining ist zu überlegen, ob ein Pulsmesser mit der App verbunden werden soll um so die aktuelle Herzfrequenz anzuzeigen und in einer Statistik festzuhalten.

Am Ende einer Trainingssession ist es dem Nutzer möglich Feedback über das Training zu geben.
So kann er bestimmen, ob das Training zu einfach / zu schwierig oder genau richtig war.

Somit kann mithilfe des Feedbacks die nächste Session angepasst werden um ein optimales Training zu ermöglichen.

Usermanagement:

Jede Woche wird der User nach dem Gewicht und dem Wohlbefinden gefragt.
So soll mit der Zeit eine Statistik erstellt werden, welche Aussagen über die Entwicklung des Nutzers ermöglicht.

Google Kalender:

Einbindung von Google Kalender um Trainingseinheiten einzutragen wurde besprochen.

Erstellen des Pflichtenhefts

Ausgangslage: Trainingszustand hinzufügen

Ist-Zustand: Auftraggeber ist eine Privatperson mit beschränkten Ressourcen

Zielsetzung: Nutzer soll durch unsere App beim Trainieren unterstützt werden.
Es soll eine Community aufgebaut werden die den Nutzer auf lange Zeit motiviert.
Durch Punktesystem, Social Networks und Rankings soll das erreicht werden.
Im Nachhinein soll der Fortschritt mit den vorigen Wochen vergleichbar sein.
Man kann am Ende der Übung Feedback geben, ob die Übung zu einfach/schwer/
genau richtig war, bzw man erhält Ausgleichsübungen, welche den selben Zweck
erfüllen aber Rücksicht auf die Bedürfnisse des Nutzers nehmen.

Anforderungen: Anforderungen an Kommunikationssysteme: Ob Rest-Server eingebaut wird, kann erst nach Fertigstellung der Systemarchitektur festgesetzt werden

Mengengerüst: 1 Nutzer auf 1 Gerät,
Videos werden Youtube Videos offline gespeichert oder über YouTube eingebettet, bzw. können sie zu Hause vorgeladen werden?)
1 Youtube Video entsprechen 6-10 Sekunden mit sprachlicher Erklärung und vorzeigen der Übung (wobei 50+ Videos anfallen).
Anzahl der Social Media Interactions werden erst in den Betatests bekannt.
Datenbank wird verwendet um Usersessions, Trainingssessions und Übungskatalog abzuspeichern.
1 mal pro Woche Eintrag in die Usersession Tabelle 52 Datensätze pro Jahr.
Pro Training ein Eintrag in die Trainingssessiontabelle=150 Datensätze
Trainingskatalog wird auch in eine Tabelle eingetragen 50-150 Datensätze.

Bemerkungen und Kommentare:

Erstes Treffen mit dem Fitnessconsultant David Lindenbauer.

Zusammenfassung:

Zusammen mit dem Fitnessconsultant wurde das alte Pflichtenheft besprochen und über die Zusammensetzung eines neuen Pflichtenhefts beraten.

Zusätzlich wurde über die erwartenden Funktionen der App gesprochen und welche davon priorisiert werden sollen.

Teilnehmer: <ul style="list-style-type: none">● Daniel Bersenkowitsch,● Andreas Denkmayr,● Gerald Irsiegler
Autor des Protokolls: Andreas Denkmayr
Dauer: 13:30 – 14:00
Menüpunkte: <ul style="list-style-type: none">● DOM● Pflichtenheft● Code-Guidelines
Inhalt: <p>BIS 23.10.2015 Übungskatalog inklusive Datenbanktabelle Übungskatalog</p> <p>DOM wurde skizziert Pflichtenheft wurde besprochen</p> <p>Es wurde beschlossen, dass während einer Trainings-Session ebenfalls auf die usereigenen Playlists zugegriffen werden kann.</p>
Code-Guidelines: <p>API-Level 18 wird verwendet Wie werden Versionsnummern vergeben?</p>
Bemerkungen und Kommentare: <p>BIS 23.10.2015 Übungskatalog inklusive Datenbanktabelle Übungskatalog</p>
Zusammenfassung: <p>DOM, Pflichtenheft und die Verwendung von Playlists wurden besprochen.</p>

Besprechung nach der SYP-Stunde

Protokoll vom 23.10.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler

Autor des Protokolls: Andreas Denkmayr

Dauer: 13:30 – 15:00

Menüpunkte:

- Review des Fortschritts beim Übungskatalog
- Besprechen der Arbeit bis nächste Woche
- Durchgehen der Datenbank

Inhalt:

Übungskatalog zeigte eine Liste von mehreren Übungen an.

Es wurde zuerst beschlossen, dass bis nächste Woche die ErstellenUserActivity fertiggestellt wird. Später wurde per Whatsappmessage stattdessen mitgeteilt, dass die erste Iteration (Übungskatalog) bis **30.10.2015** fertiggestellt werden soll.

Die Datenbank wurde gemeinsam gesprochen wie Trainingssession abzuspeichern sind.

Am 30.10 kam wegen Terminkollisionen keine Besprechung zusammen.

++

Userdaten werden in key/value-Tabelle gespeichert.

Iterationen(Meilensteinliste) sollen erstellt werden.

Art des Datenbankzugriffs soll verändert werden → Howto: Folien werden vom Herr. Prof. Stütz zur Verfügung gestellt.

Android-Hilfe: udacity.com → Kurse zur Hilfe ansehen

Herr. Prof. Stütz schlag vor IntelliJ Ultimate zu verwenden, wird aber nicht wahrgenommen, da wir uns bereits auf Android Studio festgelegt haben.

Die GUI soll für diese Iteration „perfektioniert“ werden. (Schöne Steuerelemente, schickes design)

Nächsten Iterationen:

- Datenbankstruktur neu überarbeiten
 - GridView einbauen bei Übungskatalog
 - ERD Erstellen
 - SQLiteHelper verwenden
- Systemarchitektur
 - Package Diagram
 - Klassendiagramm
- Eingabe des Profils

++

Bemerkungen und Kommentare:

Zusammenfassung: Bis 30.10.2015 wird der Übungskatalog fertiggestellt.

Besprechung des aktuellen Zustands und weiterer Vorgehensweise

Protokoll vom 20.11.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler

Autor des Protokolls: Daniel Bersenkowitsch

Dauer: 10:00 – 11:00

Menüpunkte:

- GUI des Projekts
- Weiter Iterationen
- Ordnung aller Protokolle

Inhalt:

GUI wurde besprochen. Die GUI der Iteration „Übungskatalog“ soll in den Endzustand bearbeitet werden.

Bis 04.12:

Alle Protokolle auf Vollständigkeit überprüfen und geordnet ausdrucken, mit dem überarbeiteten Pflichtenheft, in eine Mappe ordnen.

GUI: Wie macht man am Besten? Auf was muss man achten? (Ansätze in schriftl. Diplararbeit festhalten)

Essay GUI:

- 1) Was sieht der User?
- 2) Wie sieht es technisch aus?
- 3) Wie sehen richtige Formulare aus?

++

GUI vollständig implementieren inkl. Menüpunkte (Ohne Funktionalität)

Nächste Iteration: Usereingabe.

Systemarchitektur ist gelungen!

+++

Bemerkungen und Kommentare: Ein gemeinsames Treffen des Teams wird erhofft, um obige Punkte untereinander zu besprechen/evaluieren.

Zusammenfassung: Nächste Iteration wurde festgelegt, bis 04.12 implementieren

Lagebesprechung

Protokoll vom 04.12.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Thomas Stütz

Autor des Protokolls: Daniel Bersenkowitsch**Dauer:** 60min**Menüpunkte:**

- Diverse

Inhalt:

Herr Gerald Irsieger konnte nicht anwesend sein, da die Besprechung ausserhalb der Unterrichtszeit stattfindet und er bereits einen anderen Termin für diesen Zeitpunkt hat.

Nächste Iteration: Bereitstellung des NavigationDrawers innerhalb aller Framents & Profileingabefunktion bis 11.12.2015.

Frags werden eingebaut um die Navigation und Transitions leichter zu koordinieren zu können.

Pflichtenheft wurde aktualisiert vorgelegt, weiters ist das GUI Essay abgegeben worden und zusammen mit den anderen Dokumente physisch in einen Ordner sortiert worden. Ebenfalls wurden die Menüpunkte in der Applikation mit den GUI Richtlinien realisiert.

Entwurfsentscheidung: DataBinding verwenden?

Ausprobieren und herausfinden, kleines Beispiel auscodieren bis 11.12.2015.

Entwurfsentscheidung Videos

Wird Youtube API oder WebView oder VideoViewer verwendet? Youtube API ausprobieren.

Bemerkungen und Kommentare: -

Zusammenfassung: Nächste Iteration: Umbauen auf Framents & Profileingabefunktion bis 11.12.2015

Besprechung

Protokoll vom 14.12.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler
- Thomas Stütz

Autor des Protokolls: Daniel Bersenkowitsch

Dauer: 60min

Menüpunkte:

- Inhalte von letztes Mal
- DataBinding

Inhalt:

DataBinding aufschreiben, selbstevalution, Welche Aspekte? Was ist für uns neu,... Was ist das, wozu wird es verwenden?

WebView beim Link die Parameter konfiguieren.

Wie lautet das Vorgehensmodell zur Umsetzung des Algorythmus zum Generieren des Trainingsplans?

Teil von der Diplomarbeit verfassen:

Denkmayr: Fragments

Bersenkowitsch: DataBinding (bis lang wars so – jetzt so, wieso verändert?) + kleines Beispiel, was ist der Unterschied,

Irsieger: VideoAPI: Gratis, funktionieren muss es, Parameter veränderbar! Keine anmeldung, verschiedene Auflösungen.

Gegenüberstellung, welche Alternative hat was, was entspricht meinen Präferenzen?
Was ist mir wichtig? Was will ich? Kriterienkatalog! Welche Alternativen gibt es? + Bewertung

Bemerkungen und Kommentare:

Zusammenfassung: DataBinding Essay verfassen mit den obigen Punkten

Besprechung

Protokoll vom 21.12.2015

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler

Autor des Protokolls: Daniel Bersenkowitsch**Dauer:** 20min**Menüpunkte:**

- Vorstellung der Essays

Inhalt:

Fragments: Was ist das Problem? Heutzutage gibt es viele verschiedene Größe, Proportionen von Bildschirmen, Fragment passen sich automatisch an diese Größen an! Auf Rechtschreibung achten. In der schriftlichen Arbeit gibt es kein „ich“ oder „wir“, nur m.E.: (meines Erachtens) oder u.E.: (unseres Erachtens). (z.B.: „Einsatz von Fragments“ oder „Verwendung von Fragments“).

Konzept zur Erstellung von Trainingsplan ersichtlicher gestalten.

Bemerkungen und Kommentare:**Zusammenfassung:**

Besprechung

Protokoll vom 11.01.2016

Teilnehmer:

- Daniel Bersenkowitsch (Schriftführer),
- Andreas Denkmayr,
- Gerald Irsiegler
- Thomas Stütz

Autor: Daniel Bersenkowitsch

Dauer:

Menüpunkte:

- Konzept zur Erstellung von Trainingsplänen
- API Level updaten
- Dateneinspeisung

Inhalt:

Darstellungen des Konzepts zur Erstellung von Trainingsplänen ersichtlicher machen um sich besser darin zurechtzufinden. Neuüberarbeitung fällig.

Protokolle updaten bei mündlichen Ausmachungen, Protokolle immer mitführen.

Landschapemodus = fullscreen beim Video.

Pflichtenheft updaten!

Immer mitschreiben was braucht man, was braucht man nicht?!

Tablet abholen von Herrn Prof. Thumfahrt um die App darauf zu testen.

Neue Farbschemen werden ausprobiert.

Neuartige „coole“ Sachen laufen nicht auf unserem API level

Usereingabe fortschritt vorgeführt. Für MaterialDesign API level 21 updaten! Das gesamte Projekt neu anlegen – Die Zielgruppe hat eher neuere Apps – daher updaten.

Nächste Iteration:

- Dateneinspeisung in die Datenbank. Bis 18.01.15
- API level auf 21 updaten! 18.01.15
- Usereingabe fertigstellung Bis 18.01.15
- Trainingsplan generieren Konzept ordentlich formulieren. Bis 18.01.15
- Icons erstellen 18.01.15
- ERD erstellen 18.01.15
- Codereview geplant

Bemerkungen und Kommentare: /

Zusammenfassung: /

Diplomarbeitsbesprechung

Protokoll vom 29.01.2016

Teilnehmer:

- Daniel Bersenkowitsch (Schreiber),
- Andreas Denkmayr,
- Thomas Stütz

Autor: Daniel Bersenkowitsch**Dauer: 60min****Menüpunkte:**

- Codereview
- Konzept zur Ausarbeitung eines Trainingsplans

Inhalt:

Das neue Design der Applikation gefällt dem Betreuungslehrer.
Codereview wurde gemacht.

Nächste Iterationen:

- Datenkonstrukt des Trainingsplans und mit dem Algorithmus beginnen zu coden.
- Landscapemode bei Trainingssession konfigurieren.
- Playlistmanagement
- Paketdiagramm erstellen
- Datenbank nur anfüllen wenn sie inkonsistent ist

Databinding nicht verwenden, da es keinen Verwendungszweck bei uns hat, Databinding verwendet man wenn sich Daten von Kontrollelementen zur Laufzeit verändern. Dies ist bei dieser Applikation nicht der Fall.

Bemerkungen und Kommentare:**Zusammenfassung:**

Besprechung

Protokoll vom 25.02.106

Teilnehmer:

- Daniel Bersenkowitsch,
- Andreas Denkmayr,
- Gerald Irsiegler

Autor des Protokolls: Daniel Bersenkowitsch

Dauer: 60min

Menüpunkte:

-
-
-

Inhalt:

Sicherheitsfrage 1:

Haustiername: Senki1

Sicherheitsfrage 2:

Name des ältesten Kindes: Senki2

Ohne Name Zitieren:

android dev 2016

Bemerkungen und Kommentare: -

Zusammenfassung: -

8 Der Trainingsplan

8.1 Begriffserklärung

Eine "Wiederholung" ist das 1-malige Ausüben einer Übung.
Ein "Satz" umfasst alle Übungen und ausgeführten Wiederholungen. 100% RM (=Repetition Maximum = Maximalwiederholung) ist die Ausführung einer bestimmten Übung zu einem bestimmten Gewicht, welches bei der Übung genau ein 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 bei konsequenterem Training die Kraft steigt, sollte man auch immer das Trainingsgewicht erhöhen. Um maximalen Fortschritt zu erzielen, wird 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 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.

8.2 Einleitung

Ein Trainingsplan besteht aus unterschiedlichen Phasen - je nach Trainingsziel (Muskelaufbau, Maximalkraft, Kraftausdauer (=Gesundheit)) unterschiedlich. Jede Trainingsphase besitzt einen empfohlenen 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. Dabei zu bedenken ist, dass verschiedene Trainingsphasen bei verschiedenen Trainingszielen eine unterschiedliche Reihenfolge haben. Hier eine Visualisierung der Reihenfolge der Trainingsphasen:

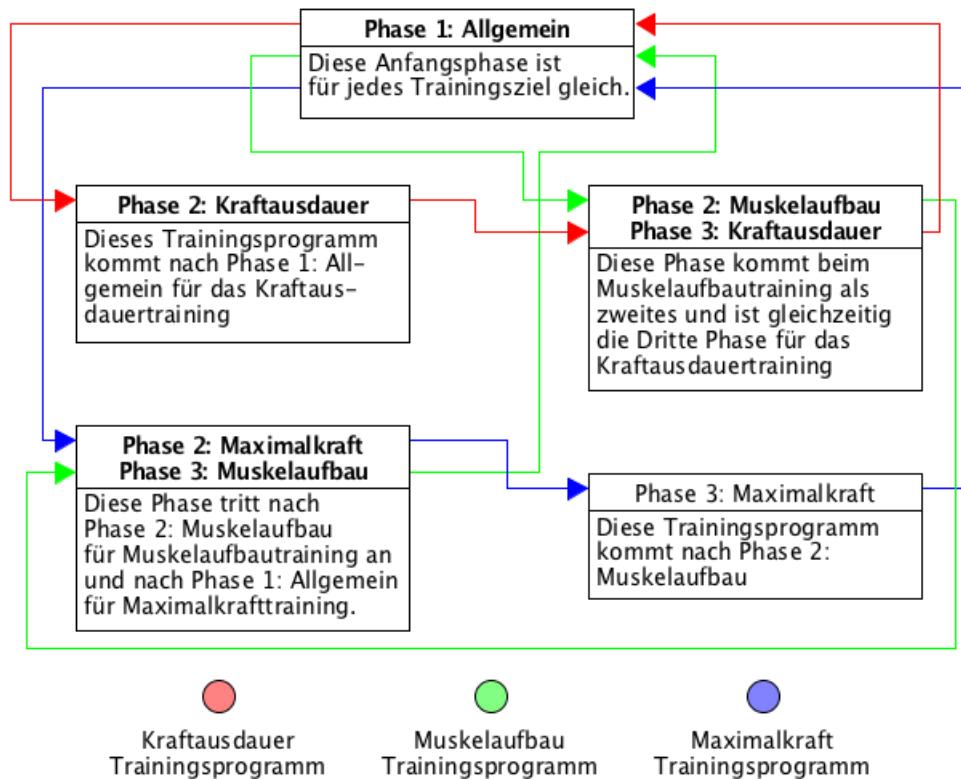


Abbildung 19: Vorgangsvisualisierung der Trainingsplanphasen

8.3 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 Sätze: 2 Wiederholungen: 20	Übungen: 9 Sätze: 3 Wiederholungen: 25
Oberkörper - Arme	Übungen: 6 Sätze: 2 Wiederholungen: 20	Übungen: 8 Sätze: 3 Wiederholungen: 25
Stabilisation	Übungen: 8 Sätze: 2 Wiederholungen: 20	Übungen: 8 Sätze: 3 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 ist zu 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

8.4 Phase 2: Kraftausdauer (Gesundheit)

Phase 2: Kraftausdauer (Gesundheit) besteht aus 2 "Phase 1: Allgemein Trainingstagen in der Woche. Zusätzlich besteht das Training aus entweder ein Mal wöchentlich Phase 2: Muskelaufbau-training oder ein 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

8.5 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 kommt 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 gezielt.

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 fokussierten 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 aufhören oder sich einen neuen Trainingsplan generieren lassen.

8.6 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, die 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.

8.7 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.

8.8 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 größtmöglichen Bewegungsumfang fortlaufend um die eigene Achse drehen.
Becken-Mob	Die Arme über den Kopf 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 beugen, 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.

[Fachkonzept zur Erstellung eines Trainingsplans [1]]

9 Verwendete Technologien

9.1 GitHub

GitHub ist ein System zur kollaborativen Versionsverwaltung für Software-Entwicklungsprojekte. Mit GitHub ist es möglich, Änderungen an Projekten zu speichern und jederzeit wieder auf vorherige Versionen zugreifen zu können. Zusätzlich zur Versionsverwaltung bietet GitHub eine große Community, die ihre Projekte mit der Welt teilt und gemeinsam an solchen Projekten entwickelt. GitHub bietet viele Statistiken und ermöglicht auch Benutzern, ohne Erfahrung mit Kommandozeilen-Befehlen, einen sehr einfachen Umgang.

[Stückler [2]]

9.1.1 Repositories

Jedes Projekt wird in einem Repository, kurz auch Repo, abgespeichert.

Public Repositories

Public Repositories können mit einem kostenlosen GitHub Account erstellt werden. Jeder kann auf die Inhalte des Repos zugreifen, aber der Entwickler entscheidet wer Commits in dieses Repo absetzen kann.

Private Repositories

Will man allerdings ein Private Repository einrichten muss man den Status seines Accounts auf Micro upgraden. Dieses Upgrade kostet \$7.00/Monat. In einem Private Repository, kann der Besitzer entscheiden, wer dieses Repo sehen und wer Commits in dieses Repo absetzen kann.

9.1.2 Git

”At the heart of GitHub is an open source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer.”

[GitHub [3]]

GitHub verwendet Git, ein Programm zur verteilten Versionsverwaltung. Der größte Unterschied dieses Systems gegenüber anderen Systemen der Versionsverwaltung liegt in der Abspeicherung der Projekte. In Git wird eine lokale Kopie des Repositories angelegt und gewartet. Dies benötigt zusätzlichen

Speicherplatz, allerdings ermöglicht dieses System eine schnellere Arbeitsweise, da die Übertragung zwischen dem lokalen und dem entfernten Repository in den meisten Arbeitsschritten eliminiert wird.

9.1.3 Arbeitskopie

Die Arbeitskopie beschreibt die Version eines Projekts, auf der ein Entwickler seine Arbeiten durchführt. Zu Beginn einer Entwicklung ist die Arbeitskopie auf dem Stand des lokalen Repositories. Änderungen werden in der Arbeitskopie durchgeführt. Somit ist das lokale Repo nicht mehr ganz aktuell. Sobald eine Funktion implementiert, oder ein Bug gefixt ist, führt der Entwickler ein Commit durch. Nun ist das lokale Repository wieder auf dem gleichen Stand wie die Arbeitskopie.

9.1.4 Lokales Repository

Das lokale Repository beinhaltet eine lauffähige Version des Projekts. Hier werden alle Commits des Entwicklers zwischengespeichert und können über einen Push auf das entfernte Repository übernommen werden.

9.1.5 Entferntes Repository

Das entfernte Repository ist auf einem Server gespeichert und somit zugänglich für alle berechtigten Personen. Dies kann sowohl auf privaten als auch auf öffentlichen Servern, wie beispielsweise den GitHub Servern, geschehen.

9.1.6 Commit

Der Vorgang einer neuen Version in das lokale Repository einzureichen, wird als Commit bezeichnet. Das bedeutet, dass diese neue Version auf die aktuelle Branch übernommen wird. Zu diesen Commits kann zurückgesprungen werden, was eine sehr hohe Transparenz im Entwicklungsvorgang ermöglicht. Das Projekt sollte zum Zeitpunkt eines Commits immer lauffähig sein!

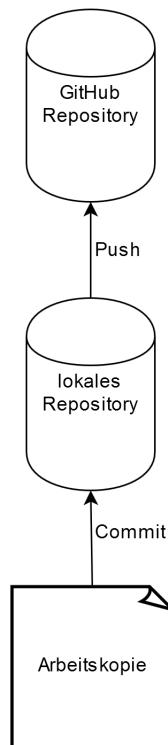


Abbildung 20:
Der Ablauf einer Änderung am Code.

9.1.7 Push

Um Änderungen allen Kollaborateuren zur Verfügung zu stellen, wird ein Push durchgeführt. Dieser übernimmt alle Änderungen des lokalen Repos in das entfernte Repository. Sobald sich die Änderungen auf dem entfernten Repository, beziehungsweise am GitHub Server, befinden, können alle Entwickler die neue Version in ihr lokales Repo übernehmen.

9.1.8 Branch

Innerhalb eines Repositories kann es mehrere Versionen einer Software geben. Diese Versionen werden in verschiedenen Branches abgespeichert. Es gibt eine Version im master-Branch, die lauffähig und funktionstüchtig sein sollte. Neue Versionen werden in Branches entwickelt und können durch Pull Requests in die Version der master-Branch übernommen werden. Auf diese Weise werden parallele Entwicklungen ermöglicht. Beispielsweise kann an einer Version weiterentwickelt und alle Fehler ausgebessert werden. Gleichzeitig kann in einem Branch an größeren Änderungen, zum Beispiel an dem Wechsel auf eine alternative Technologie, gearbeitet werden.

9.1.9 Pull Request

Wurde eine Funktion eingebaut oder ein Bug gefixt, kann ein Entwickler die Änderungen seines Branches in den master-Branch mittels einer Pull-Requests übertragen. Ein Administrator des Repositories begutachtet die Änderungen und entscheidet dann, ob er diesen Code in die master-Branch übernehmen will.

9.1.10 Community

Durch die große Community und Features wie den Entwicklerprofilen ermöglicht GitHub eine große Vernetzung in der Entwicklergemeinde. Personen können einem Entwickler folgen und so alle seine Projekte und Updates verfolgen. Jeder kann sich kostenlos die neueste Version eines Projekts herunterladen, dieses einsetzen insofern die passende Lizenz vorliegt und bei Zustimmung eines Repository Administrators am Projekt mitentwickeln.

9.1.11 Integration

GitHub lässt sich sehr einfach in populäre Entwicklungsumgebungen integrieren. Sollte so ein Support nicht standardmäßig vorliegen, wird dieser über zahlreiche Extensions ermöglicht.

9.1.12 Issue

Da GitHub auf Kollaborationen ausgelegt ist, bietet die Webseite auch ein Ticketsystem. Jeder mit einem kostenlosen Account kann Tickets erstellen. Ein Issue kann von einem Administrator des Repositories einem Entwickler zugeteilt werden. Im Rahmen dieses Systems ist ein Austausch der Entwickler mit der Community möglich. Diese Tickets weisen auf Bugs im Projekt hin, schlagen Verbesserungen des Codes vor, oder stellen den Entwicklern Fragen. Ist das Problem gelöst, kann ein Issue geschlossen werden.

9.1.13 .gitignore

GitHub bietet die Möglichkeit mehrere .gitignore Dateien anzulegen. In diesen .gitignore Files werden Dateien aufgezählt, die von GitHub ignoriert werden sollen. Dies umfasst hauptsächlich generierte Dateien oder Konfigurationsdateien. Dabei werden unnötige Änderungen und Konflikte, mit den Konfigurationsdateien anderer Entwickler, vermieden.

```
1 FIPLY/App/app/app.iml  
2 FIPLY/App/.idea/misc.xml  
3 FIPLY/App/.idea/gradle.xml  
4 FIPLY/App/.idea/vcs.xml  
5 *.log  
6 *.toc  
7 *.aux  
8 *.synctex.gz  
9 *.blg  
10 *.bbt  
11 *.bak  
12 *.bcf  
13 *.run.xml
```

```
1 .gradle  
2 /local.properties  
3 /.idea/workspace.xml  
4 /.idea/libraries  
5 .DS_Store  
6 /build  
7 /captures
```

Das .gitignore im Ordner unseres
Androidprojekts
2015_fiply\FIPLY\App

Das .gitignore im root Ordner

2015_fiply

Einträge wie ”*.log” ignorieren alle Dateien mit dieser Dateiendung.

Einträge wie ”FIPLY/App/app/app.iml” ignorieren eine spezifische Datei.

GitHub bietet standardmäßige .gitignore Dateien an, die auf bestimmte Technologien, wie beispielsweise Android, Latex, JavaScript..., abgestimmt sind.

9.1.14 GitHub Desktop

GitHub Desktop, ist der Standardclient für GitHub. Ist dieser Client installiert, so ist eine enge Zusammenarbeit mit der GitHub Webseite möglich. Man kann beispielsweise durch einen einzelnen Klick auf der Webseite ein

Repository oder eine bestimmte Branch herunterladen. Die meisten Basisfunktionen, sind in GitHub Desktop vorhanden und sehr einfach zu bedienen. Im Laufe der Arbeit stellte sich jedoch heraus, dass dieser Funktionsumfang nicht immer ausreicht. Für diese Fälle ist die, bei GitHub Desktop beiliegende, Git Shell zu verwenden. Hier ist ein weitaus größerer Funktionsumfang gegeben. Dieser wird auch benötigt, wenn Probleme, wie komplexe Konflikte bei Commits oder Pull Requests, auftreten.

9.2 Android Studio

Android Studio ist die offizielle Entwicklungsumgebung für die Androidentwicklung und ist plattformunabhängig. 2013 von Google Inc entwickelt, basiert sie auf der IntelliJ IDE von Jetbrains. [*Android Studio Dev [4]*]

9.2.1 Vorteile

Android Studio ist sehr komfortabel. Es ist übersichtlich und sehr einfach zu bedienen. Die automatische Vervollständigung ist meist richtig und führt dadurch zu einer hohen Programmierungseffizienz. Des Weiteren sind keine Plugins nötig und durch das Gradle System sehr mächtig.

9.2.2 Nachteile

Android Studio ist rein für die Entwicklung von Androidprojekten gedacht und ist die am weitesten entwickelte Android IDE. Als einzige “Konkurrenz” gilt die Eclipse IDE. Da Google aber seit der Erscheinung von Android Studio keinen Support mehr für Eclipse bietet, gilt diese als veraltet in der Androidentwicklung. Somit gibt es keine vergleichbare Technologie, die sich in einem positiven Sinne abheben kann und Android Studio einen Nachteil verschafft.

9.2.3 IntelliJ

Teile der App wurden mit IntelliJ entwickelt, da diese Entwicklungsumgebung einen komfortablen Browser für Datenbanken bietet. Dieser Browser ist nicht in Android Studio verfügbar, deshalb wird zu IntelliJ gegriffen, da es als großer Bruder Android Studios zwar nicht optimiert für die Entwicklung von Android Apps ist, aber mehr Funktionen, wie beispielsweise den erwähnten Datenbankenbrowser beinhaltet. Die Bedienung und die Oberfläche ist jener von Android Studio sehr ähnlich.

9.3 LaTeX

9.3.1 LaTeX

LaTeX ist ein Softwarepaket, welches TeX implementiert. TeX ist eine ‘What you mean is what you get’ Programmiersprache zur Erzeugung von Textdokumenten. Am besten ist LaTeX geeignet für wissenschaftliche Artikel und Diplom-, Masterarbeiten [*LaTeX* [5]].

MiKTeX

MiKTeX ist eine TeX-Distribution für Microsofts Windows-Betriebssysteme. Die ergänzenden MiKTeX Tools wurden ebenso auf GNU/Linux portiert [*MiKTeX* [6]].

TeXworks

TeXworks ist ein quelloffenes Anwendungsprogramm für Windows, Unix-Systeme und Mac OS X. Es ist eine graphische Benutzeroberfläche für das Textsatzprogramm TeX und dessen Erweiterungen LaTeX, ConTeXt und XeTeX [*TeXworks* [7]].

9.3.2 TexStudio

TexStudio ist eine im Jahr 2009 herausgekommene Entwicklungsumgebung für Latex dokumente. Entwickelt von Jan Sundermeyer, Daniel Braun und Tim Hoffmann ist sie eine der meistgenutzten Latex IDE für den Mac Computer und basiert auf Texmaker. [*TexStudio* [8]]

9.3.3 Subfiles

Um gleichzeitiges Arbeiten zu erleichtern, werden Subfiles verwendet. Zusätzlich wird das Arbeiten übersichtlicher, da man sich nicht um ein langes Dokument kümmern muss, sondern Kapitel für Kapitel schreiben und kontrollieren kann. Der schriftliche Teil der Diplomarbeit wird in das Basisdokument FIPLY_base.tex und in eine Vielzahl von Subfiles unterteilt.

Dazu wird das Subfiles Package in das Basisdokument eingebunden. Dies erfolgt durch den \usepackage{subfiles} Befehl. Das Basisdokument enthält die Kapitelüberschriften und bindet die einzelnen Subfiles, mit einem Befehl wie \subfile{denkmayr_GitHub}, ein. Dieser Befehl verlinkt auf die denkmayr_GitHub.tex Datei, die sich im selben Ordner wie das Basisdokument befindet. Die Subfiles müssen \documentclass[FIPLY_base.tex]{subfiles}

in ihrem Header definieren. Wird eine PDF aus dem FIPLY_base.tex Dokument generiert, wird der Inhalt aller angeführten Subfiles in das Basisdokument eingefügt.

9.3.4 Quellen und Zitate

Quellen und Zitierungen werden mithilfe des Biblatex Packages durchgeführt. Für die verschiedenen Quellen werden Einträge in der literatur.bib Datei erstellt.

Listing 1: Eintrag in die literatur.bib Datei

```
1 @Misc{bNavDrawer,
2   author = {Ben Jakuben},
3   title = {How to Add a Navigation Drawer in Android},
4   month = feb,
5   year = {2016},
6   url = {http://blog.teamtreehouse.com/add-navigation-drawer-android}
7 }
```

Diese Einträge werden mit dem Programm JabRef verwaltet.

Wie wird zitiert?

Sind die jeweiligen Zitierungen vorhanden, kann im Text mit Befehlen wie
\cite{bNavDrawer},
\citeauthor{bNavDrawer},
\citetitle{bNavDrawer},
\fullcite{bNavDrawer}
eine Zitierung eingefügt werden.

So sehen diese Zitierungen in einem generierten PDF aus:

[1],

Jakuben,

How to Add a Navigation Drawer in Android,

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

Abbildung 21: Aussehen der Zitate im PDF

Um dieses Ergebnis zu erreichen ist es notwendig das Programm biber.exe zu starten bevor eine PDF generiert wird. Dieses Programm sucht und ordnet alle angeforderten Zitierungen und fügt diese in den Text ein.

9.4 Adobe Photoshop

Adobe Photoshop ist ein plattformunabhängiges Bildbearbeitungsprogramm des Entwicklungskonzerns Adobe Systems. Es gilt als das Bildbearbeitungsprogramm schlecht hin und ist in seinem Bereich marktführend. Sämtliche Icons inklusive Logo der Fiply-Applikation wurden mittels diesem Programm illustriert. [*Adobe Photoshop* [9]]

10 Planung

10.1 Designrichtlinien

Designrichtlinien sind ein wichtiges Element aller Applikationen, bei denen der kommerzielle Erfolg stark vom Userinterface und dem visuellen Auftritt abhängt. Es werden dabei Tipps und Designinstruktionen von Google angewendet, welche auf der Internetseite <https://www.google.com/design/spec/material-design/introduction.html> nachzulesen sind.

10.1.1 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. Dieses Bild ist zur Laufzeit austauschbar.
- VideoView: Ähnlich zur ImageView lässt sich mithilfe einer VideoView ein einzelnes Video darstellen. Es besteht die Möglichkeit, das Video zu pausieren.
- ProgressBar: Mithilfe einer ProgressBar ist es möglich, Fortschritte grafisch darzustellen. Es besteht die Option lediglich eine Cycling Animation anzuzeigen, die nur den Verarbeitungsprozess anstatt des Fortschritts wiedergibt.

Der Splashscreen beinhaltet eine zentrierte ImageView mit dem FIPLY-Logo als Bild. Diese wird für 3500ms andauert. Der Hintergrund des Splashscreens ist weiß.

10.1.2 Animationen und Transactions

Eine Transition ist der Übergang zwischen zwei Activities. Der Übergang muss flüssig verlaufen und darf die User Experience nicht unterbrechen. Dies wird durch persistierende Objekte und präzise Bewegungen in den Übergängen erzielt. Diese Möglichkeiten stehen zur Verfügung:

- ViewSwitcher: Ein ViewSwitcher ist ein ViewAnimator, mit dem man zwischen exakt zwei Views hin- und herwechseln kann.

- **TextSwitcher:** Ein TextSwitcher ist ein ViewSwitcher, der nur TextViews 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 zwei oder mehreren Views animieren kann.
- **Fragment:** Mithilfe von Fragments kann man mehrere Ebenen von Views innerhalb einer einzelnen Activity darstellen. Somit bildet ein Fragment eine auswechselbare Ansicht, die 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 vergrößern sich jeweils die Eckpunkte eines Buttons und schmiegen sich an die Form des Screens an, wobei die neue Activity gleichzeitig 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, 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 angezeigt wird. Falls eine Activity nicht durch einen Button ausgelöst worden ist, wird sie beim Schließen 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.

10.1.3 Navigation

Als Navigation wird das Wechseln von einer Funktion der App zur Nächsten 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 Activities hin und her, sondern wechselt Teile der aktuellen Ansicht aus.
- 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 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: Ein Button ist ein Kontrollelement, das man entweder anklickt oder gedrückt hält, um Aktionen auszuführen.
- NavigationDrawer: Ein NavigationDrawer ist ein Element, das in der linken Hälfte des Bildschirms angezeigt werden kann. Dies wird mittels einem DrawerLayout realisiert, welches 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 OptionsMenu in der AppBar zur Verfügung gestellt.
- Zurücktaste am Gerät: Mithilfe der Zurücktaste kann man auf die zuletzt besuchte Activity zurückspringen.

In der Applikation soll dies mittels der Technologie des OptionsMenu, der AppBar und den Buttons umgesetzt werden. Das Ergebnis ist eine intuitive Navigation innerhalb der Applikation.

10.1.4 Formulare

Mittels Formulare kann der Benutzer Daten in das System eingeben, wie beispielsweise bei dem Anlegen eines Userprofils. Formulare sollten so intuitiv

wie möglich sein und so wenig wie möglich reine Texteingabe sein. 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 deseletiert 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 Drop-DownMenu realisiert.

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

10.1.5 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. Diese 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 Kinderelement im Vordergrund anzeigen und sieht die restlichen Kinderelemente dahinter angeordnet. Das angezeigte Element kann dabei durch jedes andere Kinderelement 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.

10.2 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. [*DataBinding Definition* [10]]

Beim Anzeigen von Daten in einer Listenansicht beispielsweise muss man das Steuerungselement nach jeder Veränderung der Daten aktualisieren. Wenn man aber die Technologie der 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.

10.2.1 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 erspart man sich umständliche findViewById Codierungen, die sehr performancelastig sind. [*Droidcon NYC 2015 - Data Binding Techniques* [11]]

10.2.2 Vorher

Auf der unterstehenden Grafik ist eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität ohne Datenanbindung zu sehen. Es besteht aus einem LinearLayout mit zwei enthaltenen TextViews. Diesen wird in der Aktivität der Vornamen und der Nachnamen eines Employee Models zugewiesen. Ein Problem ist es, jedes Element eine ID zuweisen zu müssen. Wenn man nun 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 für jedes Element eine unterschiedliche ID vergeben, obwohl manche die selbe Funktion haben. Dadurch entstehen lange und unübersichtliche ID-Namen. Das Problem ist: Es muss immer darauf geachtet werden, keine doppelten IDs zu vergeben.

Listing 2: XML Datei vor dem Einsatz von Databinding.

```
1 <LinearLayout  
2     android:orientation="vertical"  
3     android:layout_width="match_parent"
```

```

4    android:layout_height="match_parent">
5        <TextView
6            android:id="@+id/first_name"
7            android:layout_width="wrap_content"
8            android:layout_height="wrap_content"
9            tools:text="Bob"/>
10       <TextView
11           android:id="@+id/last_name"
12           android:layout_width="wrap_content"
13           android:layout_height="wrap_content"
14           tools:text="Smith"/>

```

Listing 3: Die Activity vor dem Einsatz von DataBinding.

```

1 public class OldWayActivity extends AppCombatActivity {
2     private static final Employee emp =
3         Angestellter.getInstance("Bob", "Smith");
4
5     @Override
6     protected void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.activity_oldway);
9
10        TextView firstNameView =
11            (TextView) findViewById(R.id.first_name);
12        TextView lastNameView =
13            (TextView) findViewById(R.id.last_name);
14        firstNameView.setText(emp.firstName());
15        lastNameView.setText(emp.lastName());
16    }
17 }

```

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.

10.2.3 Nacher

Listing 4: XML Datei nach dem Einsatz von Databinding.

```

1 <layout>
2     <data>
3         <variable
4             name="employee"
5             type="model.Employee"/>
6     </data>
7     <LinearLayout

```

```

8     android:orientation="vertical"
9     android:layout_width="match_parent"
10    android:layout_height="match_parent">
11    <TextView
12      android:layout_width="wrap_content"
13      android:layout_height="wrap_content"
14      android:Text="@{employee.firstName}"/>
15    <TextView
16
17      android:layout_width="wrap_content"
18      android:layout_height="wrap_content"
19      android:Text="@{employee.lastName}"/>
20  </layout>

```

Listing 5: Die Activity vor dem Einsatz von DataBinding.

```

1 public class BindingActivity extends AppCombatActivity {
2     private static final Employee emp = Angestellter.getInstance("Bob", "Smith");
3
4     @Override
5     protected void onCreate(Bundle savedInstanceState) {
6         super.onCreate(savedInstanceState);
7         EmployeeItemBinding binding = DataBindingUtil
8             .setContentView(this, R.layout.employee_item);
9         binding.setEmployee(emp)
10
11    }
12 }

```

Auf dieser Grafik ist nun eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität bei Verwendung von Datenanbindung zu sehen. Man muss keine eindeutigen IDs für jedes vorkommende Element mehr vergeben. Ein weiterer Vorteil ist es, keine findViewById Aufrufe durchführen zu müssen. Man übergibt nur noch das zu bindende Objekt, an dessen Attribute sich die in der Layoutdatei befindenden Elemente orientieren. Ein weiterer 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.

10.2.4 Anwendung

Bei der Verwendung von Datenanbindung muss man darauf achten, eine aktuelle Gradle Version zu benutzen (min. 1.5). Zusätzlich muss man in der build.gradle (Module App) Datei innerhalb des android{}-Bereichs dataBinding auf enabled=true (dataBinding{enabled=true}) setzen, um Datenan-

bindung möglich zu machen.

In der .xml Datei

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

Listing 6: Layoutcode ohne jedliche Datenanbindung.

```
1 <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
      /android"
3     android:orientation="vertical"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6       <TextView
7         android:id="@+id/vorname"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:text="Max" />
11       <TextView
12         android:id="@+id/nachname"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="Mustermann" />
16   </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 den jeweiligen getter-Methoden:

Listing 7: Unsere Objektklasse die bei der Datenanbindung referenziert wird.

```
1 package com.example.daniel.showcasedatabinding;
2 public class User {
3     private final String firstName;
4     private final String lastName;
5
6     public User(String firstName, String lastName) {
7         this.firstName = firstName;
8         this.lastName = lastName;
9     }
10    public String getFirstName() {
11        return this.firstName;
12    }
13    public String getLastName() {
14        return this.lastName;
15    }
}
```

16 }

Ist diese erstellt, kann bei der Layoutdatei weitergemacht werden. Es wird nun ein Layout-Tag erstellt. Es folgt ein variable Tag innerhalb eines data Tags. Darin wird eine Variable definiert, auf die man innerhalb der Elemente zugreifen kann.

Listing 8: Die XML Datei nach der Integration einer Datenanbindung.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <layout xmlns:android="http://schemas.android.com/apk/res/android"
3   >
4     <data>
5       <variable name="user" type="showcasedatabinding.User"/>
6     </data>
7     <LinearLayout
8       android:orientation="vertical"
9       android:layout_width="match_parent"
10      android:layout_height="match_parent">
11
12       <TextView
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="@{user.firstName}" />
16
17       <TextView
18         android:layout_width="wrap_content"
19         android:layout_height="wrap_content"
20         android:text="@{user.lastName}" />
21     </LinearLayout>
22 </layout>
```

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

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 9: 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 =
16             DataBindingUtil
17                 .setContentView(this, R.layout.activity_main);
18         User user = new User("Max", "Mustermann");
19         binding.setUser(user);
20     }
21 }
```

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 automatische Änderung des Elements. [Data Binding Guide [12]]

11 Umsetzung

11.1 Permissions

Permissions legen fest, auf welche Funktionen des Smartphones eine App Zugriff hat. Ziel dieses Systems ist es, Apps nur so viel zu erlauben, wie unbedingt nötig. Das bedeutet, dass es einer bösartigen App nicht möglich ist, erheblichen Schaden zu verursachen, ohne die entsprechenden Permissions zu haben. Da eine Applikation zu Beginn keine Permissions besitzt, müssen diese im Manifest deklariert werden.

[Phanvilai [13], Android Developers [14][15]]

Listing 10: Deklaration von Permissions im AndroidManifest.xml

```
1 <manifest package="htl_leonding.fiplyteam.fiply"
2     xmlns:android="http://schemas.android.com/apk/res/android" >
3 <uses-permission android:name="android.permission.INTERNET" />
4 <uses-permission android:name="android.permission.WAKE_LOCK" />
5 ...
6 </manifest>
```

11.1.1 Arten von Permissions

Android unterstützt mehrere Levels von Permissions.

- **Normal Permissions** stellen kaum Gefahr für die Privatsphäre des Benutzers oder den Betrieb des Systems dar und werden automatisch gegeben sobald sie im Manifest angefordert werden.
- **Dangerous Permissions** hingegen, können gefährlich für die Privatsphäre des Benutzers werden oder den Betrieb des Systems erheblich stören. Deshalb müssen Dangerous Permissions nicht nur im Manifest angefordert werden, sondern auch explizit vom Benutzer bestätigt werden.
- **Special Permissions** sind die dritte und seltenste Art von Permissions. Diese sind besonders heikel und müssen im Manifest deklariert und über einen Intent angefordert werden. Dieser Intent öffnet ein Fenster, speziell zur Verwaltung dieser Permission.
Zu diesen Special Permissions gehören die WRITE_SETTINGS Permission, die Änderungen der Systemeinstellungen ermöglicht, und die SYSTEM_ALERT_WINDOW Permission, die es ermöglicht Fenster über allen anderen Apps anzuzeigen.

11.1.2 bis Android 5.1 (API level 22)

Wenn auf dem Gerät Android 5.1 (API level 22) oder niedriger installiert ist, werden alle Dangerous und Special Permissions auf der Google Play Store Seite der App angezeigt und müssen vor dem Herunterladen bestätigt werden. Sollten durch ein Update mehr Permissions benötigt werden, müssen diese beim Update bestätigt werden. Permissions können nur durch die Deinstallation der App zurückgenommen werden.

11.1.3 ab Android 6.0 (API level 23)

Wenn auf dem Gerät Android 6.0 (API level 23) oder höher installiert ist, wird beim Download aus dem Google Play Store keine Bestätigung der Permissions verlangt. Die Permissions werden nun während der Laufzeit der App abgefragt. Diese Abfragen muss der Entwickler selbst erstellen und anzeigen lassen.

Listing 11: Das Anfordern von Permissions ab Android 6.0

```
1 final public int REQUEST_CODE_ASK_PERMISSIONS = 123;
2
3 public void CheckMusicPermissionAndReadMusic(Context context) {
4     int readStoragePerm = ContextCompat.checkSelfPermission(this,
5             Manifest.permission.READ_EXTERNAL_STORAGE);
6
7     if (readStoragePerm != PackageManager.PERMISSION_GRANTED)
8     {
9         if (!ActivityCompat.shouldShowRequestPermissionRationale(
10             this, Manifest.permission.READ_EXTERNAL_STORAGE))
11         {
12             showMessageOKCancel("Permission message",
13                     new DialogInterface.OnClickListener() {
14                 @Override
15                 public void onClick(DialogInterface dialog, int which) {
16                     ActivityCompat.requestPermissions(Settings.this,
17                         new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
18                         REQUEST_CODE_ASK_PERMISSIONS);
19                 }
20             }, this);
21             return;
22         }
23         ActivityCompat.requestPermissions(this, new String[]{
24             Manifest.permission.READ_EXTERNAL_STORAGE},
25             REQUEST_CODE_ASK_PERMISSIONS);
26         return;
27     }
28     rm.ReadSongsIntoArrayList(context);
29 }
```

Listing 12: Anzeigen der Permissionsabfrage

```
1 private void showMessageOKCancel(String message, DialogInterface
2     .OnClickListener okListener, Activity activity) {
3     new AlertDialog.Builder(activity)
4         .setMessage(message)
5         .setPositiveButton("OK", okListener)
6         .setNegativeButton("Cancel", null)
7         .create()
8         .show();
9 }
```

Da die Permissions erst während der Laufzeit abgefragt werden, ist es möglich nur manche von diesen zu bestätigen und so die Bereiche, in welche die App eingreifen kann, nach seinen eigenen Bedürfnissen zu kontrollieren. Zudem können Permissions jederzeit in den Systemeinstellungen des Geräts zurückgenommen werden.

11.1.4 Permission groups

Permissions werden in Permission groups zusammengefasst. Beim Anfordern einer Dangerous Permission, wird nicht die einzelne Permission, sondern die jeweilige Permission group angezeigt. Sowohl eine Anforderung der READ_CONTACTS Permission, als auch der WRITE_CONTACTS Permission, zeigt an, dass Zugriff auf die Kontakte des Gerätes benötigt wird. Wird eine Dangerous Permission angefordert, während die App schon eine Dangerous Permission derselben Gruppe besitzt, so wird die Anfrage automatisch bestätigt. Alle Permissions können einer Permission Group zugeordnet werden, allerdings sind nur Permission Groups von Dangerous Permissions relevant für Benutzer und Entwickler.

11.2 Fragments

11.2.1 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 wieder verwendet werden können, müssen Ansichten wie Detailviews oder Listen nur einmal programmiert werden und können überall eingesetzt werden. Fragments werden ab Android 3.0 (API level 11) zur Verfügung gestellt.
[Android Developers [16][17]]

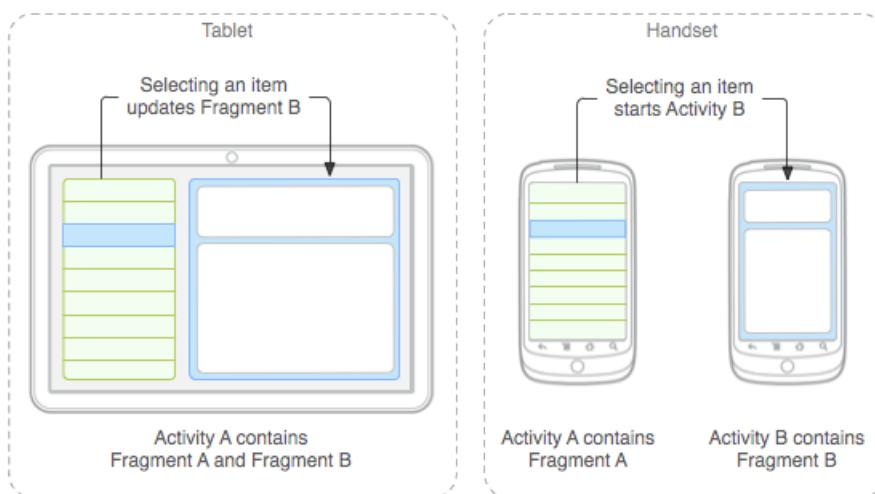


Abbildung 22: Zum Beispiel kann man mithilfe von Fragments Views erstellen, die sowohl auf einem Tablet, als auch auf einem Handy ein optimales Benutzerinterface anbieten. [Android Developers [17]]

11.2.2 Der Lifecycle

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

Listing 13: onCreateView() das ein Fragment anzeigt

```
1 @Override
2 public View onCreateView(LayoutInflater inflater, ViewGroup
3     container, Bundle savedInstanceState) {
4     super.onCreate(savedInstanceState);
5     return inflater.inflate(R.layout.exampleFragment,
6         container, false);
7 }
```

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

11.2.3 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 Activities, 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.

Listing 14: Eine Fragment Transaction

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

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

11.2.4 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 FMain oder dem NavigationDrawer ermöglicht. Zusätzlich werden verschachtelte Fragments verwendet, um komplexere Ansichten darzustellen. In diesem Fall, werden in den Layout XML Files der Fragments ein oder mehrere FrameLayouts erstellt und die verschachtelten Fragments werden dann in diesen neuen FrameLayouts angezeigt.

Listing 15: Die displayView-Methode vereinfacht das Anzeigen eines neuen Fragments

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

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

Listing 16: Die Definition eines FrameLayouts in einem Layout XML File

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

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

11.3 Bundles

Ein Bundle ist ein Objekt, dass einer Vielzahl von Datentypen einen String zuordnen kann, um Daten zwischen Activities und Fragments zu übertragen. Die Übergabe des Bundles, und somit der Austausch der Daten, erfolgt über Intents oder Fragment Transactions. [Logan [18]]

Listing 17: Erstellen und Befüllen von Bundels

```
1 Bundle innerBundle = new Bundle();
2 Bundle outerBundle = new Bundle();
3
4 ArrayList<String> arrList = new ArrayList<>();
5 outerBundle.putStringArrayList("bundleArrayList", arrList);
6 outerBundle.putInt("bundleInt", 123);
7 outerBundle.putString("bundleString", "Hallo Welt!");
8 outerBundle.putDouble("bundleDouble", 12.3);
9 outerBundle.putBundle("anotherBundle", innerBundle);
10 outerBundle.putBoolean("bundleBoolean", true);
```

In diesem Beispiel werden Parameter unterschiedlicher Datentypen zu einem Bundle hinzugefügt. Das Hinzufügen erfolgt über die zahlreichen put-Methoden. Der erste Parameter ist die Kennzeichnung. Über diese Kennzeichnung kann später wieder auf die Daten zugegriffen werden. Die zu übertragenden Daten werden als zweiter Parameter in eine put-Methode übergeben.

11.3.1 Datenübertragung zwischen Fragments

Listing 18: Das Übergeben eines Bundles bei einer FragmentTransaction

```
1 Bundle args = new Bundle();
2 Fragment fragment = new FTrainingssession();
3 fragment.setArguments(args);
4
5 FragmentManager fragmentManager = getFragmentManager();
6 FragmentTransaction fragmentTransaction = fragmentManager
7     .beginTransaction();
8 fragmentTransaction.replace(R.id.fraPlace, fragment);
9 fragmentTransaction.commit();
```

Dieses Bundle wird an ein Fragment angehängt bevor es durch eine FragmentTransaction angezeigt wird.

Zugriff auf die Daten

Listing 19: Der Zugriff auf ein Bundle im neuen Fragment

```
1 Bundle bundleFromBefore = this.getArguments();  
2 Boolean ergBool = bundleFromBefore.getBoolean("bundleBoolean");  
3 String ergString = getArguments().getString("bundleString");
```

Die Methode getArguments() ermöglicht einen Zugriff auf ein zuvor übergebenes Bundle. Für jede put-Methode eines Bundles, existiert auch eine get-Methode, die die Daten des jeweiligen Datentyps zurückliefert.

11.3.2 Datenübertragung zwischen Activities

Für die Datenübertragung zwischen Activities gibt es drei mögliche Vorgehensweisen.

Bundle des Intents

Listing 20: Übertragung durch das Bundle des Intents

```
1 Intent intent = new Intent(this, MainActivity.class);  
2 Bundle intentBundle = intent.getExtras();  
3 intentBundle.putString("bundleString", "Hallo Welt!");
```

Das Bundle des Intents wird durch put-Methoden erweitert.

Neues Bundle

Listing 21: Übertragung durch einen Intent mit einem neuen Bundle

```
1 Intent intent = new Intent(this, MainActivity.class);  
2 Bundle newBundle = new Bundle();  
3 newBundle.putString("bundleString", "Hallo Welt!");  
4 intent.putExtras(newBundle);
```

Es wird ein neues Bundle erstellt und dem Intent mithilfe der putExtras()-Methode hinzugefügt.

Abkürzung durch putExtra()

Listing 22: Abkürzung durch die putExtra()-Methode eines Intents

```
1 Intent intent = new Intent(this, MainActivity.class);
2 intent.putExtra("bundleString", "Hallo Welt!");
```

Alternativ kann bei Intents auch die putExtra()-Methode verwendet werden. Diese ist für alle Datentypen, die auch ein Bundle unterstützen, verfügbar.

Zugriff auf die Daten

Listing 23: Zugriff auf die Daten in der neuen Activity durch die getExtras()-Methode

```
1 Bundle bundleFromBefore = this.getIntent().getExtras();
2 bundleFromBefore.getBoolean("bundleBoolean");
3 getIntent().getExtras().getString("bundleString");
```

Die Methode getIntent().getExtras() ermöglicht einen Zugriff auf ein, durch einen Intent übergebenes, Bundle.

Alternativ können die Daten durch getExtra-Methoden abgefragt werden.

Listing 24: Zugriff auf die Daten in der neuen Activity durch getExtra()-Methoden

```
1 getIntent().getBooleanExtra("bundleBoolean", true);
2 getIntent().getStringExtra("bundleString");
```

Bei vielen dieser getExtra-Methoden wird als zweiter Parameter ein Standardwert verlangt.

11.4 NavigationDrawer

Der NavigationDrawer ist ein Menü, das die wichtigsten Navigationsoptionen am linken Bildschirmrand abbildet. Diese Leiste ist meistens versteckt und wird angezeigt sobald der Benutzer mit dem Finger vom linken Rand des Bildschirms in Richtung Bildschirmmitte streicht oder auf das NavigationDrawerIcon klickt. Erneutes Klicken auf dieses Icon oder Streichen nach links versteckt den NavigationDrawer.

[Jakuben [19], Tamada [20], Android Developers [21]]

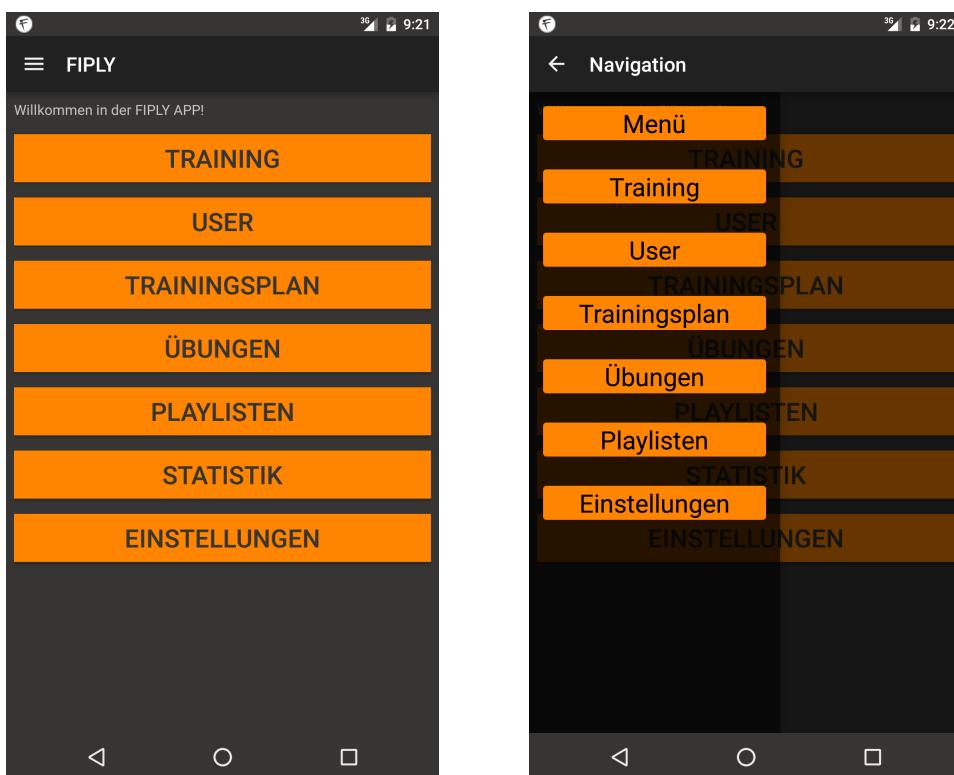


Abbildung 23: Links wird der NavigationDrawer versteckt. Rechts ist der NavigationDrawer geöffnet.

Listing 25: Layout XML File eines NavigationDrawers

```
1 <android.support.v4.widget.DrawerLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:id="@+id/drawer_layout"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context="htl_leonding.fiplyteam.fiply.menu.MainActivity">
8
9     <FrameLayout
10        android:id="@+id/fraPlace"
11        android:layout_width="match_parent"
12        android:layout_height="match_parent" />
13
14     <ListView
15        android:id="@+id/navlist"
16        android:layout_width="250dp"
17        android:layout_height="match_parent"
18        android:layout_gravity="start"
19        android:background="@color/darkNavigationBackground"
20        android:divider="@color/darkNavigationDivider"
21        android:dividerHeight="1dp" />
22
23 </android.support.v4.widget.DrawerLayout>
```

Um einen NavigationDrawer zu einer App hinzuzufügen, wird ein DrawerLayout als Root-View in dem Layout XML File einer Activity deklariert. Eine Root-View beinhaltet alle anderen Ansichten der App. In diesem DrawerLayout wird anschließend eine View angelegt, die dazu verwendet wird den Hauptinhalt der App darzustellen. Dabei ist zu beachten, dass die View, die alle Elemente in sich trägt, als erstes Kind des DrawerLayouts deklariert wird. Da diese View den ganzen Bildschirm befüllen soll müssen die Höhe und die Breite auf "match_parent" gestellt werden.

Zusätzlich wird eine weitere View hinzugefügt, die die Elemente des NavigationDrawers in sich trägt. Diese muss das Attribut layout_gravity überschreiben. Um right-to-left Sprachen zu unterstützen, soll "start" anstatt von "left" dafür spezifiziert werden. Die Höhe soll die Länge des ganzen Bildschirms umfassen, die Breite wird in Density-independent Pixels definiert und sollte 320dp nicht überschreiten da der Benutzer immer Teile des Hauptinhalts sehen soll.

In dieser Arbeit wird für den Hauptinhalt das FrameLayout fraPlace verwendet. Dieses Layout dient als Container für Fragments, die die einzelnen Ansichten der App darstellen. Die Elemente des NavigationDrawers werden in der ListView navlist definiert.

Listing 26: Minimaler Code eines NavigationDrawers

```
1 public class MainActivity extends AppCompatActivity {
2     ListView mDrawerList;
3     ArrayAdapter<String> mAdapter;
4     DrawerLayout mDrawerLayout;
5     String [] navArray = new String [7];
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11
12        mDrawerList = (ListView) findViewById(R.id.navlist);
13        mDrawerLayout = (DrawerLayout) findViewById(
14            R.id.drawer_layout);
15        navArray = res.getStringArray(R.array.navigationArray);
16
17        mDrawerList.setOnItemClickListener(new AdapterView
18            .OnItemClickListener() {
19
20            @Override
21            public void onItemClick(AdapterView<?> parent, View view,
22                int position, long id) {
23                displayView(position);
24            }
25        });
26
27        mAdapter = new ArrayAdapter<>(this, R.layout.navigation_list,
28            R.id.navlist_content, navArray);
29        mDrawerList.setAdapter(mAdapter)
30    }
31 }
```

Das ist der minimale Code, um einen funktionierenden NavigationDrawer zu erstellen. Will man das Design und Verhalten des Icons, zum Öffnen des NavigationDrawers, ändern, ist mehr Code nötig. Dieser Code wird ebenfalls im onCreate der Activity, die einen NavigationDrawer erhalten soll, ausgeführt.

Listing 27: Erweiterung des NavigationDrawers um ihn visuell ansprechender zu machen

```
1 mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
2     R.string.drawer_open, R.string.drawer_close) {
3     ...
4 };
5 mDrawerToggle.setDrawerIndicatorEnabled(true);
6 mDrawerLayout.addDrawerListener(mDrawerToggle);
7 getSupportActionBar().setDisplayHomeAsUpEnabled(true);
8 getSupportActionBar().setHomeButtonEnabled(true);
```

11.5 Benutzerverwaltung

11.5.1 Beschreibung

Die Applikation wurde als Single-User-Application entworfen und umgesetzt. In der Benutzerverwaltung kann der Benutzer seine Daten angeben und umändern. Weiters kann er sein Social Media Profil mit der Applikation verbinden, um später Fortschritte mit seinen Freunden zu teilen. Auch Daten welche essentiell zur Trainingsplanerstellung sind, werden hier aufgenommen.

11.5.2 Aufteilung der Verwaltung

Die Verwaltung bzw. Erstellung des Benutzers ist auf folgende drei Schritte aufgeteilt:

Schritt 1

Im ersten Schritt gibt der Benutzer seinen Namen und sein Geschlecht an. Der Name wird in Plain Text eingegeben, während für das Geschlecht ein Spinner zur Verfügung gestellt wird, wobei man aus einer Liste aus: Male, Female und Other auswählen kann. Weiters kann er sich hier über den Facebook-Login-Button mit seinem Facebook Account anmelden.

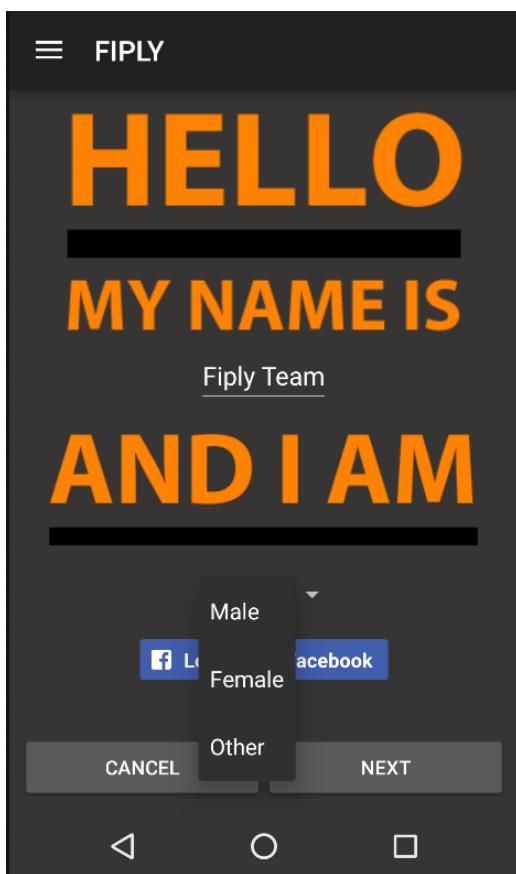


Abbildung 24: Der erste Schritt der Usererstellung.

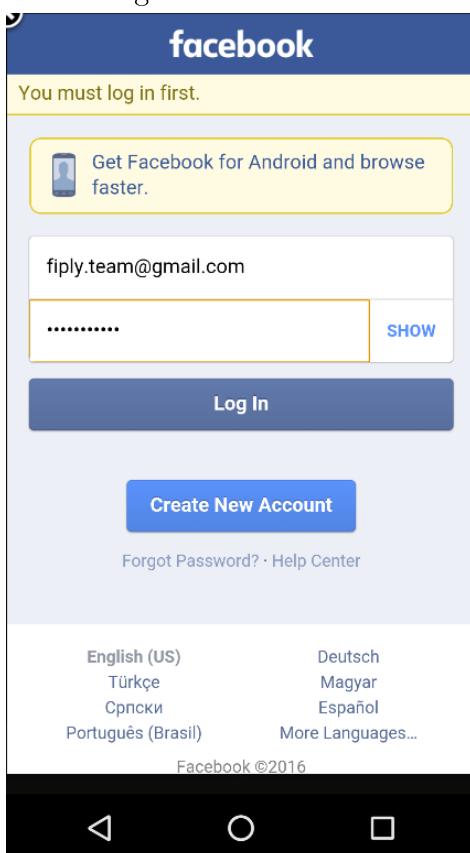


Abbildung 25: Der Loginscreen von Facebook.

Schritt 2

Im zweiten Schritt gibt der Benutzer seine Körpergröße in Zentimetern (cm) und sein Gewicht in Kilogramm (kg) an.

Die Größe kann zwischen 100cm und 200cm auf jeden ganzen Wert eingestellt werden.

Das Gewicht kann zwischen 40kg und 140kg auf jeden ganzen Wert eingestellt werden.

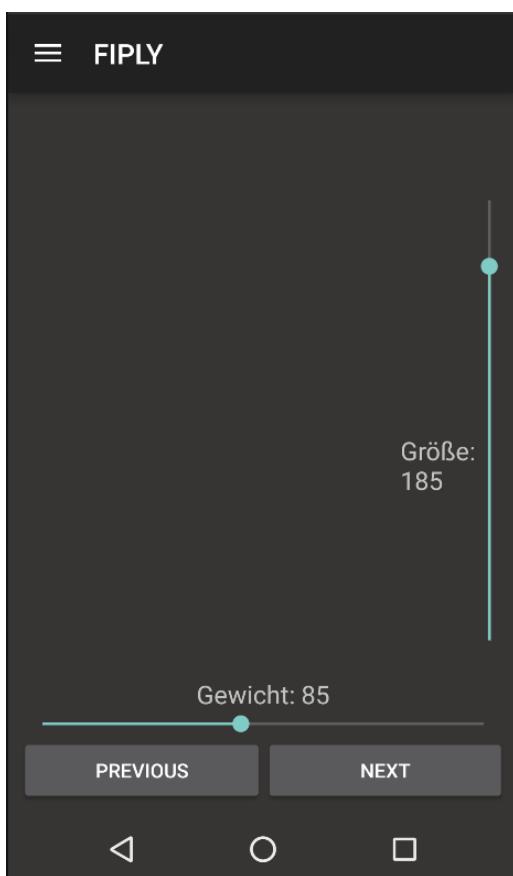


Abbildung 26: Der zweite Schritt der Usererstellung.

Schritt 3

Im dritten Schritt gibt der Benutzer an in welcher Altersgruppe er sich befindet und wie seine körperliche Verfassung derzeit ist. Zur Auswahl stehen bei den Altersgruppen (in Jahren):

- 16-20
- 21-30
- 31-40
- 41-50
- 50+

Bei der derzeitigen Verfassung wird angegeben ob man körperlich fit oder unfit ist.

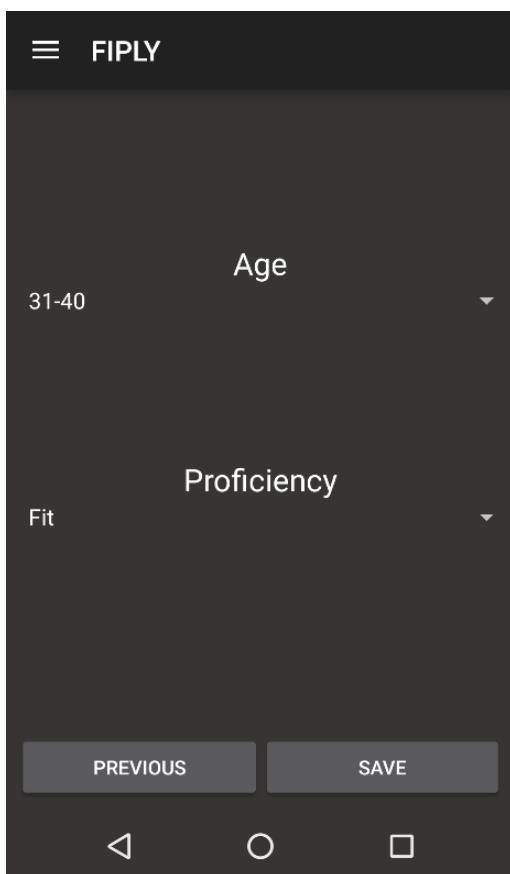


Abbildung 27: Der dritte Schritt der Usererstellung.

11.5.3 Implementierung

Alle Daten über den User werden im Key-Value-Repository der Datenbank als Key-Value-Paare abgespeichert. Weiters werden die derzeit gespeicherten Werte automatisch eingetragen falls der Benutzer eine Seite erneut aufruft.

11.6 Exportieren

11.6.1 Als CSV exportieren - OpenCSV

Die OpenCSV Bibliothek erlaubt dem Java Programmierer eine CSV Datei zu erstellen, speichern, schreiben und lesen. In Bezug auf die Diplomarbeit wird diese Technologie verwendet, um den Trainingsplan in eine CSV-Datei abzuspeichern. Um den aktuellen Pfad im Android-Dateisystem herauszufinden, gibt es die statische getAbsolutePath()-Methode, die folgens verwendet wird:

Listing 28: Methode, um den aktuellen Pfad herauszufinden.

```
1 String path = android.os.Environment  
2         .getExternalStorageDirectory()  
3         .getAbsolutePath();
```

Der String “path“ beschreibt nun den Pfad wo die CSV-Datei eingespeichert wird.

Alternative Eine alternative Möglichkeit zu der OpenCSV Bibliothek wäre, die Funktionen selbst auszuprogrammieren. Da die Verwendung der Bibliothek zeitsparender ist, wird sie der Alternative vorgezogen.

Anwendung CSV-Datei lesen:

Listing 29: Verwendung von CSVReader: Möglichkeit 1, iterativ

```
1 CSVReader reader = new CSVReader(  
2     new FileReader(path + "file.csv"));  
3 String [] nextLine;  
4 while ((nextLine = reader.readNext()) != null) {  
5     // nextLine[] is an array of values from the line  
6     System.out.println(nextLine[0] + nextLine[1] + "etc...");  
7 }
```

Das Objekt reader ist öffnet einen Stream zu der erzielten Datei “file.csv“. Mittels der Standardfunktion .readNext() wird die nächste Zeile in ein eindimensionales String-Array gespeichert. Jede Arraystelle beschreibt den Inhalt eines Spaltenfeldes in der aktuellen Zeile der CSV-Datei.

Listing 30: Verwendung von CSVReader: Möglichkeit 2, alles auf einmal

```
1 CSVReader reader = new CSVReader(  
2     new FileReader(path + "file.csv"));  
3 List myEntries = reader.readAll();
```

Das Objekt reader ist wieder einen Stream zu der erzielten Datei “file.csv“. Durch die Standardfunktion .readAll() werden alle Zeilen bis zum Ende der Datei in eine List gespeichert. Das Listenobjekt myEntries ist eine Liste bestehend aus eindimensionalen Arrays, wobei wieder jede Arraystelle sequentiell den Inhalt eines Spaltenfeldes in der CSV-Datei beschreibt. [Vgl.: *OpenCSV Website* [22]]

CSV-Datei schreiben:

Eine CSV-Datei zu erstellen und schreiben ist genau so einfach wie sie zu lesen. Hierbei wird die Klasse “CSVWriter“ verwendet:

Listing 31: Verwendung von CSVWriter: Möglichkeit 1, iterativ

```

1 CSVWriter writer = new CSVWriter(
2     new FileWriter(path + "file.csv"));
3 String [] country;
4 while ((country = getNextCountries()) != null){
5     writer.writeNext(country);
6 }
7 writer.close();

```

Das Objekt writer ist öffnet einen Stream zu der erzielten Datei “file.csv“. Falls die Datei nicht existiert, wird sie angelegt. Die Methode getNextCountries() liefert in diesem Kontext ein eindimensionales Array zurück. Mit dem Befehl .writeNext() werden Strings in dem Array in eine neue Zeile der Datei gespeichert, wobei jede Arraystelle für eine Spalte der Tabellendatei steht.

Listing 32: Verwendung von CSVWriter: Möglichkeit 2, alles auf einmal

```

1 CSVWriter writer = new CSVWriter(
2     new FileWriter(path + "file.csv"));
3 List<String[]> data = new ArrayList<String[]>();
4 data.add(new String[] {"India", "New Delhi"});
5 data.add(new String[] {"United States", "Washington D.C"});
6 data.add(new String[] {"Germany", "Berlin"});
7 writer.writeAll(data);
8
9 writer.close();
10

```

Das Objekt writer ist öffnet einen Stream zu der erzielten Datei “file.csv“. Falls die Datei nicht existiert, wird sie angelegt. Es wird eine Liste von eindimensionalen Arrays angelegt. Wie auch in den obigen Beispielen steht jede Arraystelle für eine Spalte in der Tabellendatei. Mit dem Befehl .writeAll() werden alle Elemente sequentiell in die Datei gespeichert. [Vgl.: *OpenCSV Writer* [23]]

11.6.2 Emails senden

Mittels Intents Die beste Möglichkeit, um eine Email in Android zu senden, ist das benutzen eines Intents:

Listing 33: Verwendung von CSVWriter: Möglichkeit 2, alles auf einmal

```
1 File file = new File(path, "file.csv");
2 Uri u = null;
3 u = Uri.fromFile(file);
4
5 Intent sendIntent = new Intent(Intent.ACTION_SEND);
6 sendIntent.putExtra(Intent.EXTRA_SUBJECT, "Dein FIPLY
    Trainingsplan");
7 sendIntent.putExtra(Intent.EXTRA_STREAM, u);
8 sendIntent.setType("text/html");
9 startActivity(sendIntent);
```

Das senden von Emails mittels Intents erfolgt damit über externe Applikationen, die diese Funktion anbieten. Bei ausführen des obigen Codes wird der Benutzer gefragt, mit welchem Emailclient auf seinem Smartphone er die Aktion durchführen will. Die Emaildetails werden der externen Applikation mitgegeben, worauf sich diese öffnet und der benutzer dann nur noch auf den “senden“-Knopf drücken muss. [*Mails versenden in Android [24]*]

Vorteile/Nachteile von Intents Der große Vorteil dieser Methode ist es, dass man ohne eigenen Emailclient auskommt. Da die Email über eine andere Applikation versendet wird muss sich der Entwickler nicht weiter um aufwendige Emailclient codierungen kümmern.

Ohne bereits installierten Client auf dem Android Gerät ist das senden von Emails über diese vorgehensweise nicht möglich. Ein gutes Beispiel dafür ist das Ausführen der Funktion auf einem Android-Emulator auf einem beliebigen Computer. Standardmäßig ist auf einem Android-Emulator kein Emailclient wie zum Beispiel die Gmail Applikation installiert. Daher kann die Funktion auf einem Emulator nicht getestet werden, bevor man einen manuell installiert.

11.7 Trainingsplan

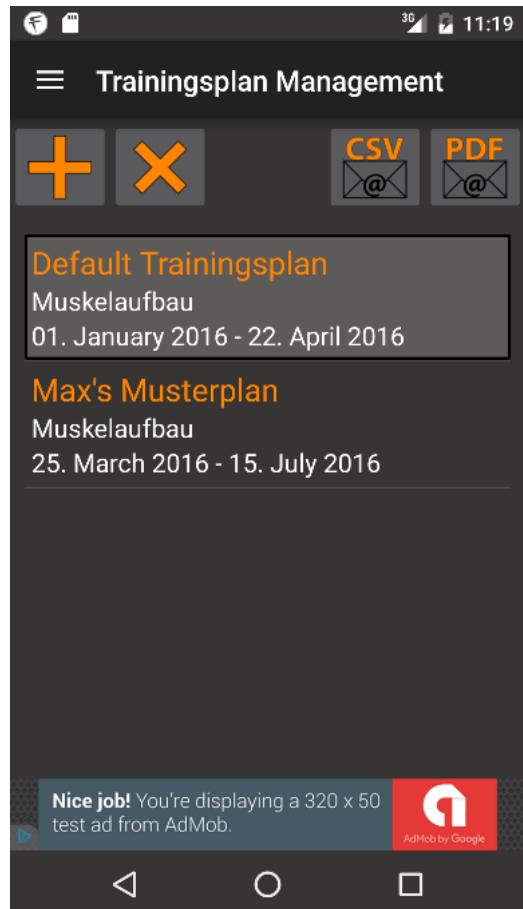
Der Trainingsplan ist das Herzstück des Projekts. Wie in der Theorie beschrieben wird dieser mittels eines Algorithmus umgesetzt.

11.7.1 Die Ansicht

In der Trainingsplan Ansicht kann der Benutzer seine Trainingspläne verwalten.

In der obigen Leiste befinden sich die Optionen "hinzufügen", "löschen", und die Knöpfe zum Exportieren eines ausgewählten Plans.

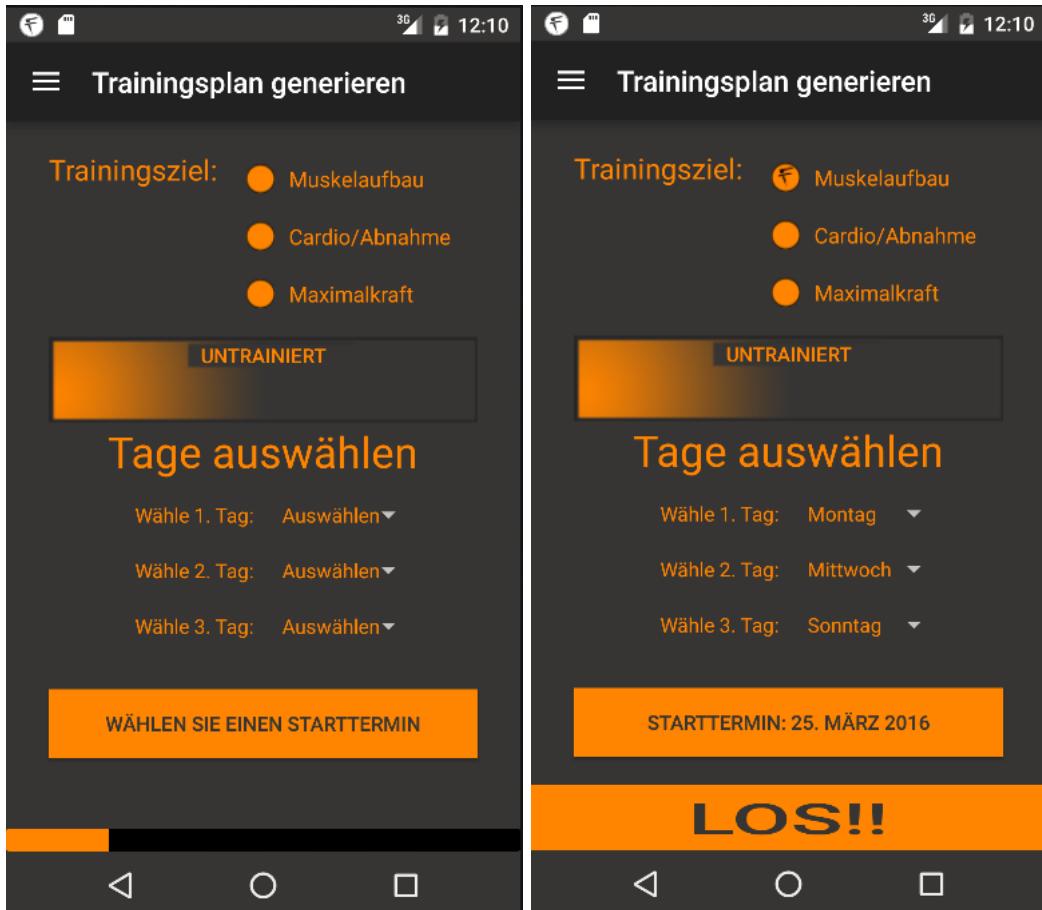
Unterhalb ist eine ListView mit den existierenden Trainingsplänen. Es kann immer nur ein Plan gleichzeitig ausgewählt werden. Der Plan der ausgewählt ist, wird in der Trainingssession herangezogen. Als additionalen Informationen stehen unterhalb des Trainingsplannamens dessen Trainingsziel und Anfangs- und Enddatum. Beim Löschen eines Trainingsplan wird vorher Abgefragt, bittet die Applikation um eine Bestätigung.



(a) Ansicht des Trainingspläne

11.7.2 Das Erstellen

Um einen neuen Trainingsplan zu erstellen drückt man auf den Plus-Knopf ganz links in der Ansichtsleiste. Man wird zur Erstellungsansicht weitergeleitet die folgens aussieht:

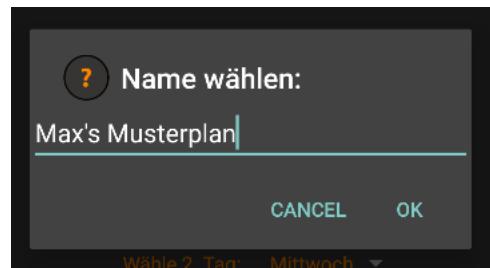


(a) Trainingsplan erstellen: unausgefülltes Formular (b) Trainingsplan erstellen: ausgefülltes Formular

Der Benutzer wählt nun aus, welches Trainingsziel er verfolgen will. Weiters kann er seine Einstellung ändern, ob er trainiert oder untrainiert ist. Danach wählt man 3 Trainingstage aus, an denen man trainieren will. Die jeweiligen Übungen werden dann auf diese Tage verteilt. Gleichzeitig, während des Ausfüllens des Formulars bewegt sich eine ProgressBar mit, die sich je mehr füllt desto mehr Informationen eingegeben worden sind. Bei einer Füllung von 100 Prozent erscheint der "LOS!!"-Knopf, mit denen das Generieren ge-

startet werden kann.

Nach der Betätigung dieses Knopfes wird man wieder zur Trainingsplanübersicht gebracht, wo der neu erstellte Plan sich jetzt befindet. Ab dem Startdatum kann der Plan dann in der Trainingssession in Verwendung gebracht werden.



(a) Erfolgreiche Aktion: Trainingsplan generiert.

11.7.3 Umsetzung des Algorithmus

Der Trainingsplan wird genau so wie in der Theorie beschrieben generiert. Je nach Trainingsziel bestimmt sich die Reihenfolge und Art der Trainingsphasen. Bei jeder Trainingsphase hat der Algorithmus einen bestimmten Übungspool zur Verfügung, die sich aus den Muskelgruppen und dem Schwierigkeitsgrad einer jeweiligen Übung erschließen. Die Anzahl der Wiederholungen & Sätze variieren jeweils in dem angegebenen Bereich in der Theorie, die sich in den Trainingsphasen unterscheiden. Der Algorithmus beachtet alle Bedingungen und verteilt die herangezogenen Übungen auf die angegebenen Trainingstage.

Je nach Dauer der Trainingsphase werden die Übungen eine bestimmte Anzahl an Wochen eingebbracht. Die Summe der Dauer der Trainingsphasen ist die Dauer eines Trainingsplan, wodurch sich mittels Startdatum das Enddatum berechnen lässt. Nach jeder Trainingsphase folgt die nächste, bis der Plan zu Ende ist.

11.8 Übungskatalog

11.8.1 Beschreibung

Der Übungskatalog beinhaltet eine Liste aller verfügbaren Übungen.



Abbildung 31: Der Übungskatalog.

11.8.2 Implementierung

Expandable List View Die Expandable List View ist eine eigene Implementierung der Standard List View. Sie erlaubt es bei Tippen auf ein Element weitere Unterelemente darzustellen. Im Laufe der Entwicklung wurde festgestellt, dass eine Standard List View, mit einem Verweis auf eine Detail-View vorteilhafter wäre.

List View Die List View zeigt eine Liste bestehend aus Elementen an. Ein Klick auf ein bestimmtes Element löst eine Callback-Methode aus, welche die Detail View aufruft.

Einlesen der Übungen Die Informationen über alle Übungen ist in der strings.xml im JSON-Format hinterlegt. Jede Übung hat somit ihr eigenes JSON-Objekt.

Listing 34: Ein JSON-Objekt welches eine Übung beschreibt.

```
1 <string name="exercisecatalog">
2 [
3 ...
4 {
5   \\"Muskelgruppe\\":\\"Brust , Untere Brust\\",
6   \\"Name\\":\\"Negativbankdruecken\\",
7   \\"Beschreibung\\":\\"Mit geradem Ruecken auf Negativbank legen ,
8     Wenn ein Polster vorhanden ist – Beine einklemmen , Stange
9       etwa schulterbreit greifen\\",
10  \\"Durchfuehrung\\":\\"Mit fixierten Schultern die Stange
11    kontrolliert in einer Linie zur Brust und wieder nach oben
12      bewegen , Stange in einer Linie bewegen und Brust nicht
13        beruehren\\",
14  \\"Equipment\\":\\"Negativbank , Langhantel\\",
15  \\"Schwierigkeit\\":\\"Mittel\\"
16 }
17 ...
18 ]
19 </string>
```

Wie in diesem Beispiel erkennbar, muss jedes Hochkomma mit einem Backslash maskiert werden, da der JSON-String sonst nicht in der strings.xml abgelegt werden kann.

Dieser JSON-String wird in einer Methode der Klasse UebungenRepository eingelesen und die einzelnen Übungen werden in die Datenbank eingefügt.

Listing 35: Diese Methode lese den JSON-string ein und fügt die Übungen in die Datenbank ein.

```
1  public void insertAllExercises() throws JSONException {
2      reCreateUebungenTable();
3      String json = repoContext.getResources()
4          .getString(R.string.
5              exercisecatalog);
6
7      JSONArray exercises = new JSONArray(json);
8      JSONObject temp;
9
10     for (int i = 0; i < exercises.length(); i++) {
11         temp = exercises.getJSONObject(i);
12         Log.wtf("Exercise: ", temp.getString("Name"));
13         insertUebung(temp.getString("Name"),
14             temp.getString("Beschreibung"),
15             temp.getString("Durchfuehrung"),
16             temp.getString("Muskelgruppe"),
17             temp.getString("Schwierigkeit"),
18             "https://www.youtube.com/embed/..",
19             temp.getString("Equipment"));
20     }
21 }
```

Diese Methode wird beim Startup der Applikation, während des SplashScreens, in einem Async-Task aufgerufen und ausgeführt.

11.8.3 DetailView

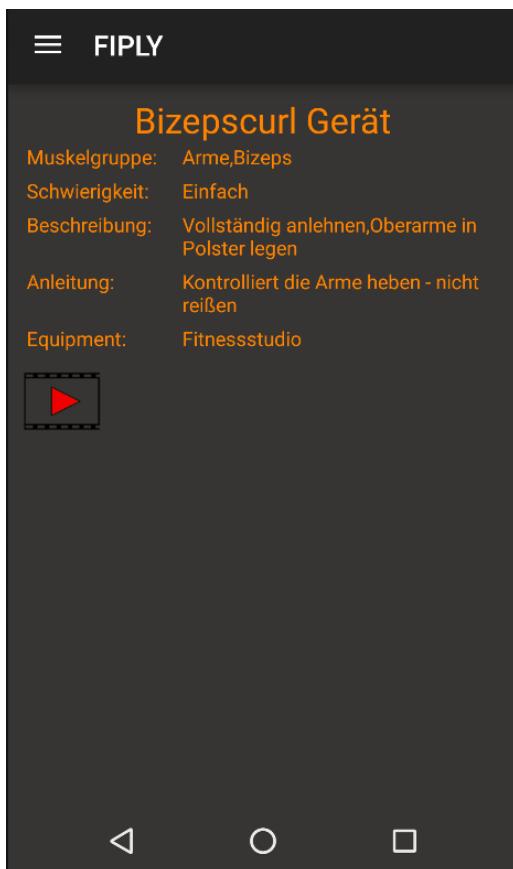


Abbildung 32: Die DetailView der Übung "Bizepscurl".

Für jede Übung gibt es eine Detailansicht, in welcher man genaues über jene Übung erfahren kann. Diese Detail-View wird aufgerufen indem man auf die korrespondierende Übung im Übungskatalog tippt.

Es werden folgende Details bereitgestellt:

- Name der Übung
- Die Muskelgruppe/n welche man mit dieser Übung trainiert.
- Schwierigkeit, wie anspruchsvoll ist diese Übung.
- Beschreibung, die Ausgangslage der Übung.
- Anleitung, wie wird die Übung, von der Ausgangsposition, richtig durchgeführt.

- Benötigtes Equipment, um die richtige Durchführung der Übung zu ermöglichen.
- Ein Video welches die Durchführung beschreibt. Dieses kann im mit Tipp auf das Videosymbol im linken mittleren Bereich der Detail View aufgerufen werden. Die App ändert dann automatisch in den Landschafts-Modus und stellt das Video im Vollbildmodus dar.

11.8.4 Filter

Beschreibung Die Liste kann auch nach Name der Übung und Muskelgruppe gefiltert werden. Der Filter wird über den Floating Action Button aufgerufen, welcher sich in der rechten unteren Ecke des Übungskataloges befindet. Die Muskelgruppen-Auswahl erfolgt über das Tippen auf eine bestimmte Muskelgruppe, beim Namen wird rein nach Text filtriert



Abbildung 33: Die Filter View.

Implementierung Für den Filter für Name und Muskelgruppe gibt es jeweils einen Eintrag in der Key-Value-Repository. Beim Einlesen der Übungen werden die Filter automatisch angewandt.

Die folgende Lösung wurde diesem Tutorial entnommen [*Android Images With Clickable Areas – Part 1* [25]]

Um die geklickte Muskelgruppe zu ermitteln, wurden zwei Auswahlbereiche übereinander gelegt.

Jener Auswahlbereich, welchen der Benutzer zu sehen bekommt. Dieser Auswahlbereich ist nur die visuelle Stütze für den Benutzer, sie hat keinerlei funktionalen Nutzen.



Abbildung 34: Das sichtbare Overlay des Filters.

Und eine farbkodierte Maske, welche der Benutzer nicht sieht, mit welcher er aber eigentlich interagiert. Bei einem Klick auf die Auswahlfäche kann nachvollzogen werden welche Farbe der angeklickte Bereich hat, somit wird festgestellt welche Muskelgruppe der Benutzer ausgewählt hat.

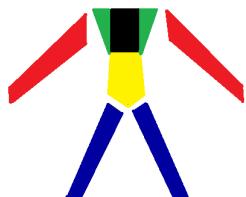


Abbildung 35: Die nicht sichtbare Maske.

Bei einem Klick auf die Auswahlfläche wird folgende Callback-Methode aufgerufen.

Listing 36: Vergleicht die Farben und setzt den Filter.

```
1  KeyValueRepository kvr = KeyValueRepository.getInstance();
2
3  public boolean onTouch(View v, MotionEvent event) {
4      int tolerance = 25;
5      int evX = (int) event.getX();
6      int evY = (int) event.getY();
7      int clickedColor = getHotspotColor(evX, evY);
8
9      //RED area is arms
10     if (closeMatch(getResources().getInteger(R.integer.redInt),
11         clickedColor, tolerance)) {
12         Log.wtf("Area clicked: ", "Arme");
13         kvr.updateKeyValue("filterMuskelGruppe", "Arme");
14     }
15     //BLACK area is Breast
16     else if (closeMatch(getResources().getInteger(R.integer.
17         blackInt), clickedColor, tolerance)) {
18         Log.wtf("Area clicked: ", "Brust");
19         kvr.updateKeyValue("filterMuskelGruppe", "Brust");
20     }
21     //GREEN area is shoulders
22     else if (closeMatch(getResources().getInteger(R.integer.
23         greenInt), clickedColor, tolerance)) {
24         Log.wtf("Area clicked: ", "Schultern");
25         kvr.updateKeyValue("filterMuskelGruppe", "Schultern");
26     }
27     //BLUE are is legs
28     else if (closeMatch(getResources().getInteger(R.integer.
29         blueInt), clickedColor, tolerance)) {
30         Log.wtf("Area clicked: ", "Beine");
31         kvr.updateKeyValue("filterMuskelGruppe", "Beine");
32     }
33     //YELLOW area is core(stomach)
34     else if (closeMatch(getResources().getInteger(R.integer.
35         yellowInt), clickedColor, tolerance)) {
36         Log.wtf("Area clicked: ", "Bauch");
37         kvr.updateKeyValue("filterMuskelGruppe", "Bauch");
38     }
39     return true;
40 }
```

Diese Methode ermittelt die Farbe des gedrückten Bereiches und setzt den Key-Value Eintrag im Repository.

Die erste verwendete Hilfsmethode.

Listing 37: Ermittelt ob zwei Farbwerte ähnlich oder gleich sind.

```
1  public boolean closeMatch(int color1, int color2, int
2      tolerance) {
3          if (Math.abs(Color.red(color1) - Color.red(color2)) >
4              tolerance)
5              return false;
6          if (Math.abs(Color.green(color1) - Color.green(color2)) >
7              tolerance)
8              return false;
9          if (Math.abs(Color.blue(color1) - Color.blue(color2)) >
10             tolerance)
11             return false;
12         return true;
13     }
```

Die zweite verwendete Methode.

Listing 38: Ermittelt den Farbwert des gedrückten Bereiches.

```
1  public int getHotspotColor(int x, int y) {
2      bodyFilterMask.setDrawingCacheEnabled(true);
3      Bitmap hotspots = Bitmap.createBitmap(bodyFilterMask.
4          getDrawingCache());
5      bodyFilterMask.setDrawingCacheEnabled(false);
6      return hotspots.getPixel(x, y);
7  }
```

11.9 Lösung für die Videodarstellung

11.9.1 VideoView

Beschreibung

VideoView ist die native Lösung von Android, Videos in einer App darzustellen. Sie können entweder direkt vom Speicher des Systems oder über einen RTSP-Key (Real-Time Streaming Protocol) auch vom Internet abgespielt werden.

Vorteile

Die native Lösung von Android ist die performanteste aller unserer Optionen.

Probleme

Der RTSP-Key ist sehr umständlich abzurufen und die VideoView ist generell eine etwas ältere Lösung.

11.9.2 Youtube Android Player API

Beschreibung

Die Youtube Android Player API ist die von Google entwickelte Lösung ausschließlich Youtube-Videos in einer Android Umgebung abzuspielen.

Vorteile

Da die Youtube Android Player API rein für das Abspielen von Youtube Videos konzipiert ist, ist es die beste Lösung für unser Problem.

Probleme

Für diese Methode wird leider ein Google Developer Key, welcher derzeit nicht verfügbar ist benötigt.

Weiters muss die YouTube-App auf dem Mobilgerätes des Benutzers installiert sein damit diese Methode funktioniert.

11.9.3 WebView

Beschreibung

Die WebView erlaubt es HTML-Code oder Websites direkt über deren URL in der App darzustellen.

Vorteile

Die WebView ist leicht zu benützen und mit den embeded Links von Youtube können wir unsere Videos leicht einbinden. Weiters ist die WebView sehr flexibel da man auch reinen HTML Code darstellen kann.

Probleme

Da es kein direkter Video-Player ist muss der Youtube-Player in die WebView embeded werden, dadurch wird die Perfomance der App beeinträchtigt. Weiters muss die Vollbild Funktionalität selbst implementiert werden, da es noch keine vorgegebene Lösung gibt.

11.9.4 Nutzwertanalyse

KO-Kriterien

- Die Alternative muss kostenfrei sein.
- Die Alternative muss verwendbar sein.

11.9.5 Erfüllung der KO-Kriterien

VideoView

Die Video View ist kostenfrei und verwendbar.

Youtube Android Player API

Die Youtube Android Player API ist kostenfrei, jedoch nicht verwendbar, da im Moment kein Google-Developer Account verfügbar ist.

WebView

Die WebView ist kostenfrei und verwendbar.

11.9.6 Schlussfolgerung

Die WebView Lösung ist die beste Lösung, nicht nur für die Benutzer sondern auch für die Entwickler, daher wird diese in der Applikation benützt.

11.10 Musik

[Android Developers [26], Tamada [27]]

11.10.1 Lokalisierung der Musikdateien

Listing 39: Ursprünglicher Ansatz zum Lesen aller mp3-Files im Music-Ordner

```
1 File home = new File(Environment.getExternalStorageDirectory()
2   .getAbsolutePath() + "/Music");
3 songs = new ArrayList<>();
4 if (home.listFiles(new FileExtensionFilter()) != null) {
5   for (File file : home.listFiles(new FileExtensionFilter())) {
6     HashMap<String, String> song = new HashMap<>();
7     song.put("songTitle", file.getName());
8     song.put("songPath", file.getPath());
9     songs.add(song);
10  }
11 }
```

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.

Listing 40: Die query-Methode des Android Source Codes in der *ContentResolver.java* Klasse [28]

```
1 public final @Nullable Cursor query(@NonNull Uri uri, @Nullable
2   String[] projection, @Nullable String selection, @Nullable
3   String[] selectionArgs, @Nullable String sortOrder) {
4   return query(uri, projection, selection, selectionArgs,
5     sortOrder, null);
6 }
```

Listing 41: Alle Audiodateien, die Musik beinhalten, werden gesucht.

```
1 ContentResolver cr = context.getApplicationContext()
2   .getContentResolver();
3 Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
4 String selection = MediaStore.Audio.Media.IS_MUSIC + "!= 0";
5 String sortOrder = MediaStore.Audio.Media.TITLE_KEY + " ASC";
```

```
6 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.

Listing 42: Speichern der Songverweise in eine ArrayList von HashMaps

```

1 songs = new ArrayList<>();
2 HashMap<String , String> song = new HashMap<>();
3 while (cur.moveToNext())
4 {
5     song . put("songTitle" , cur . getString (cur
6         . getColumnIndex (MediaStore . Audio . Media . TITLE )) );
7     song . put("songPath" , cur . getString (cur
8         . getColumnIndex (MediaStore . Audio . Media . DATA )) );
9     songs . add (song);
10 }
```

11.10.2 Verwalten von Playlists

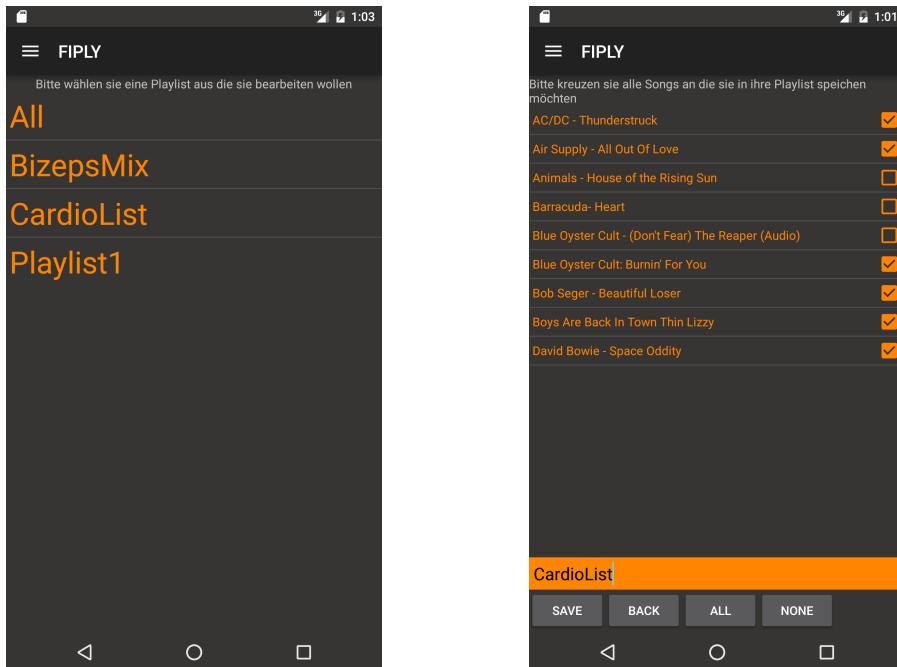


Abbildung 36: Beim Klicken eines Elements in Screenshot 1 werden alle Songs angezeigt (Screenshot 2). Mittels der Checkboxen werden jene Songs markiert, die sich in der ausgewählten Playlist befinden.

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 in das EditText eingetragene Titel unter der ListView übernommen.

Listing 43: Speichern der Songverweise in eine ArrayList von HashMaps

```
1 SparseBooleanArray checked = lvSongs.getCheckedItemPositions();
2 for (int i = 0; i < songs.size(); i++) {
3     if (checked.get(i)) {
4         checkedSongs.add(songs.get(i));
5     }
6 }
7 psrep.reenterPlaylist(etName.getText().toString(), checkedSongs);
```

11.10.3 Abspielen der Playlists.

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.

Listing 44: Die changeSong-Methode

```
1 public void changeSong(int songIndex, String playlist) {
2     aktPlaylist = playlist;
3     setPlaylist(psrep.getByName(aktPlaylist));
4     setSongIndex(songIndex);
5     try {
6         mp.reset();
7         mp.setDataSource(getPlaylist().get(getSongIndex())
8             .get("songPath"));
9         mp.prepare();
10    } catch (IOException e) {
11        e.printStackTrace();
12    }
13    progressBar.setProgress(0);
14    updateProgressBar();
15    tvSongname.setText(getPlaylist().get(getSongIndex())
16        .get("songTitle"));
17    tvTotalDur.setText(millisecondsToHMS(mp.getDuration()));
18 }
19
20 private void updateProgressBar() {
21     mHandler.postDelayed(mUpdateDurTask, 100);
22 }
```

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

Listing 45: Das Runnable aktualisiert die Fortschrittsanzeigen des Musikplayers

```
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.
7             getDuration());
8         progressBar.setProgress(progress);
9         mHandler.postDelayed(this, 100);
10    }
11};
```

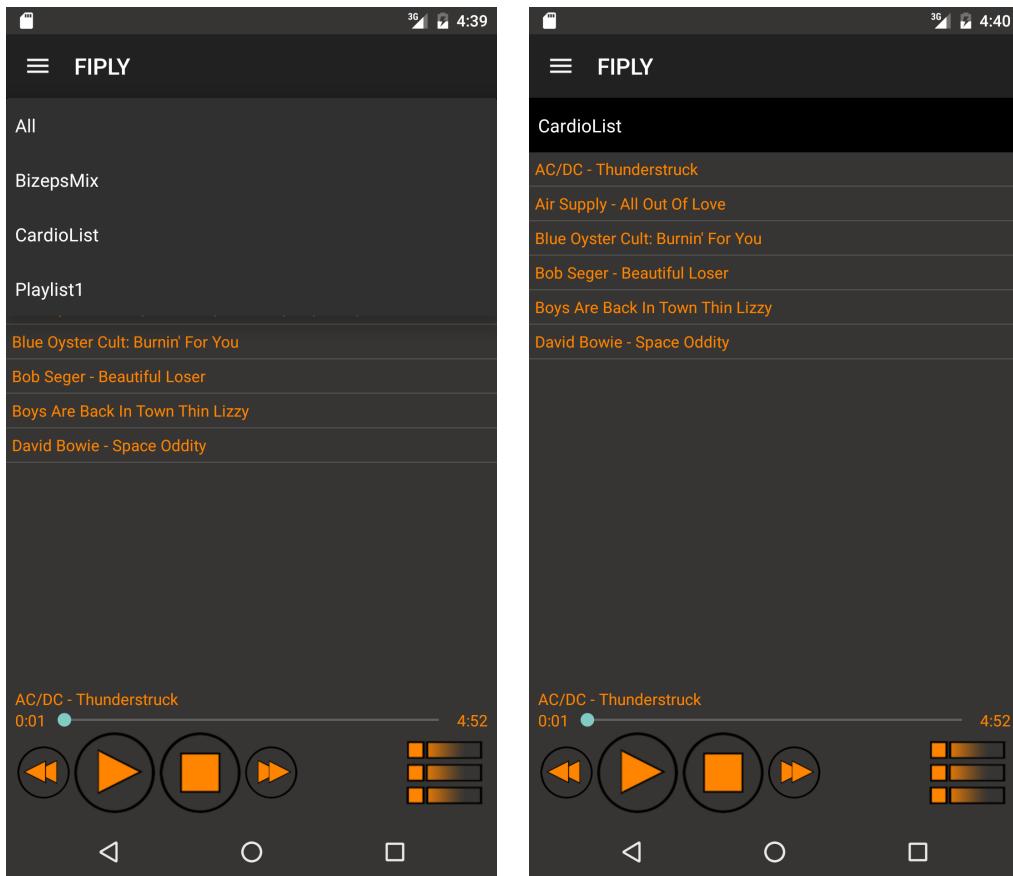


Abbildung 37: In einem Spinner kann eine Playlist ausgewählt werden. Beim Klick auf einen Song wird dieser abgespielt.

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 man durch Klick auf den Musikmodus Button in den Musikmodus gewechselt wird.

In diesem Modus wird über den Spinner die Playlist gewechselt.

In der aktuell ausgewählten Playlist kann man direkt zu einem bestimmten Song wechseln, indem man auf den Songtitel klickt. Dieser wird abgespielt und nach Ende des Songs wird sofort der nächste Playlisteintrag gestartet.

11.10.4 MusicControls



Abbildung 38: Die MusicControls in Ruhe und während einer Wiedergabe.

- Durch den Zurück und 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 werden 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.
- Die Fortschrittsleiste wird durch eine SeekBar über diesen Buttons implementiert. Diese SeekBar stellt den Fortschritt des aktuellen Songs dar. Bei Klicken auf, oder Ziehen an der Fortschrittsleiste, kann man den Fortschritt der Wiedergabe manipulieren.
- Am linken Ende der Fortschrittsleiste wird der Fortschritt des aktuellen Songs im hh:mm:ss Format (ISO 8601) angezeigt.
- Am rechten Ende 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.

11.11 Statistik

11.11.1 Beschreibung

Die Statistik gibt dem Benutzer eine Übersicht zu seinem Training in den letzten Tagen/Wochen/Monaten.

Es werden verschiedene Statistiken zur Verfügung gestellt, um so viel Informationen wie möglich darzustellen.

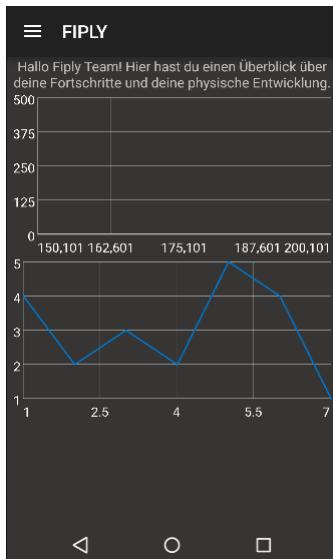


Abbildung 39: Die Statistik.

11.11.2 Verfügbare Statistiken

Geistige Verfassung

Diese Statistik beschreibt wie es dem Benutzer nach jeder Trainingseinheit geht. Er soll auf einer Skala von eins bis fünf seinen geistigen Zustand nach dem Training bewerten, wobei eine höhere Zahl eine bessere Laune beschreibt.

Gehobenes Gewicht

Nach jeder Trainingseinheit wird angegeben mit wieviel Gewicht die jeweiligen Übungen durchgeführt worden sind. Anhand dieser Statistik kann man seine Entwicklung sehr gut verfolgen

Insgesamt gestemmtes Gewicht

Hier wird festgehalten wieviel Gewicht der Benutzer insgesamt an einem Tag, einer Woche, einem Monat gestemmt hat.

11.11.3 Implementierung

Aufnahme und Speicherung der Werte

Die aufzunehmenden Werte werden während bzw. nach der Trainingssession aufgenommen und danach in ihr jeweiliges Repository abgespeichert. Die einzelnen Werte für die Statistik werden in einer SQLite Datenbank in ihren jeweiligen Repositories gespeichert.

Dabei ist jeder Eintrag eine eigene Entität welche das Interface DataPointInterface implementiert.

Listing 46: Ein Datenpunkt für die Graphen.

```
1 public class MoodTime implements DataPointInterface {
2     private double _timestamp;
3     private double _mood;
4
5     public MoodTime(double timestamp, double mood) {
6         _timestamp = timestamp;
7         _mood = mood;
8     }
9
10    @Override
11    public double getX() {
12        return _timestamp;
13    }
14
15    @Override
16    public double getY() {
17        return _mood;
18    }
19 }
```

Darstellung

Zur Darstellung der Statistiken wird GraphView verwendet. GraphView ist eine open-source Library für Android zur Erstellung von Diagrammen. Verfügbar sind eine Vielzahl von Diagramm-Arten wie z.B.: Linien-, Kuchen- und Punktdiagrammen.

11.12 Datenbank

11.12.1 Beschreibung

In der Datenbank werden alle für die Applikation essentiellen Daten gespeichert.

11.12.2 SQLite

SQLite ist eine open-source Library, sie implementiert ein unabhängiges, serverloses, "zero-configuration" und transaktionales SQL-database-engine [SQLite[29]]. SQLite ist die beste Lösung da die Daten nur in einer Applikation verwendet werden.

11.12.3 ContentProvider

Der ContentProvider kapselt den Datenzugriff vom lokalen Speicher des Gerätes, er wird hauptsächlich benutzt falls Daten zwischen Applikationen ausgetauscht werden.

11.12.4 Zugriff auf die Daten

Datenbank Architektur

Der Zugriff auf die abgespeicherten Daten erfolgt über 4 Schichten:

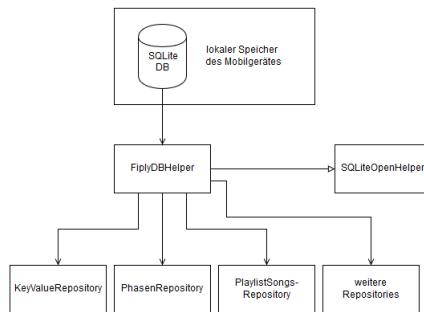


Abbildung 40: Die Datenbank-Architektur.

Physische Ebene

Alle Daten werden im lokalen Speicher des Mobilgerätes abgelegt und persistiert. Die Speicherung übernimmt das DBMS von SQLite.

Datenbankobjekt

Das Datenbankobjekt ist jenes, welches im lokalen Speicher abgelegt ist und von der nächsten Schicht instanziert wird. Es enthält die Informationen über alle in der Datenbank abgespeicherten Daten und erlaubt es auf diese zuzugreifen.

DB Helper

Der DB Helper kapselt das Datenbankobjekt und stellt es den Repositories zur Verfügung. Der DB Helper ist als Singleton implementiert und wird in den Repositories instanziert.

Repositories

Die Repositories dienen als Puffer zwischen dem Datenbankobjekt und der Businesslogik. Jedes Repository ist als Singleton implementiert und um darauf zugreifen zu können muss es vorerst im Code instanziert werden.

Listing 47: Repository wird instanziert.

```
1 Repository repository = Repository.getInstance();
```

In den Repositories muss das Datenbankobjekt mithilfe des DB Helpers instanziert werden ...

Listing 48: Das Datenbankobjekt db wird instanziert über den FiplyDBHelper

```
1 SQLiteDatabase db = getWritableDatabase();  
2  
3 private SQLiteDatabase getWritableDatabase() {  
4     if (repoContext == null)  
5         throw new IllegalStateException();  
6  
7     return FiplyDBHelper.getInstance(repoContext)  
8             .getWritableDatabase();  
9 }
```

... um über dieses Objekt dann mithilfe von SQL-Statements auf die Daten zugreifen bzw. die Daten manipulieren zu können.

Listing 49: db wird instanziert und ein SQL Statement wird ausgeführt.

```
1 SQLiteDatabase db = getWritableDatabase();  
2  
3 return db.query("SQL-STATEMENT");
```

Folgende Repositories sind in der Applikation vorhanden:

- Instruktionen-Repository
- Key-Value-Repository
- Phasen-Repository
- Plan-Repository

- Playlist-Songs-Repository
- Statistik-Repository
- Uebungen-Repository

11.12.5 Contract

Der Contract ist eine Datei in welcher die Metadaten der Datenbank zu finden sind. Dieser "Vertrag" existiert um den makellosen Zugriff auf die Datenbank sicherzustellen. Für jede Tabelle in der Datenbank wird im Contract eine eigene Klasse angelegt, diese beinhaltet den Tabellennamen und die Namen aller ihrer Attribute als strings.

Listing 50: Eine Contract Klasse, mit allen Metadaten.

```
1  public static final class UebungenEntry implements
2      BaseColumns {
3          public static final String TABLE_NAME = "uebungen";
4          public static final String COLUMN_ROWID = "_id";
5          public static final String COLUMN_NAME = "name";
6          public static final String COLUMN_MUSKELGRUPPE = "
7              muskelgruppe";
8          public static final String COLUMN_BESCHREIBUNG = "
9              beschreibung";
10         public static final String COLUMN_ANLEITUNG = "anleitung"
11         ;
12         public static final String COLUMN_SCHWIERIGKEIT = "
13             schwierigkeit";
14         public static final String COLUMN_VIDEO = "video";
15         public static final String COLUMN_EQUIPMENT = "equipment"
16         ;
17     }
```

11.13 Social Media

11.13.1 Beschreibung

Mithilfe der Verknüpfung zu Social Media kann der Benutzer seine Trainingsfortschritte einfach mit seinen Freunden teilen.

11.13.2 Verknüpfung mit Facebook

Die Verknüpfung der App mit Facebook erfolgt über die FacebookSDK und den FacebookLoginButton.

FacebookSDK

Mithilfe dieser SDK kann man den Login verwalten und danach im Namen des Benutzers, mit der Einverständnis des Benutzers, posten. Um die Integrität des Logins und des Datenaustausches zu gewähren, hat jede Applikation welche die Facebook SDK implementiert ihre eigene Serien-Nummer. Diese wird in der strings.xml gespeichert.

Listing 51: Die Facebook App ID in der strings.xml.

```
1 <string name="facebook_app_id">1541961082763294</string>
```

Facebook Login Button

Der Facebook Login Button wird von der Facebook SDK zur Verfügung gestellt und kann ganz einfach in das Layout eingefügt werden.

Listing 52: Das xml-layout des FacebookLoginButtons.

```
1 <com.facebook.login.widget.LoginButton  
2     android:id="@+id/fbLoginButton"  
3     android:layout_width="wrap_content"  
4     android:layout_height="wrap_content"  
5     android:layout_below="@+id/spGender"  
6     android:layout_centerHorizontal="true" />
```

Test Account

Da die Applikation während der Entwicklung noch nicht veröffentlicht ist, wird ein Facebook-Test-Account benötigt.

Die Privatsphäre der Entwickler wird somit auch garantiert, da sonst eine nicht getestete Applikation Postings im Namen der Entwickler, auf deren persönlichen Facebook Seiten machen könnte.

Um einen Test Account anzulegen muss man zuerst einen normalen Facebook Account anlegen und ihn dann in den Einstellungen zu einem Test-Account umändern.

11.14 Design

Die Hintergrundfarbe aller Activities ist ein starkes Grau, die Entscheidung viel schlussendlich auf die Farbe mit dem Farbcode #363534. Die dazu kontraste Signalfarbe ist ein eindeutiges Orange, #ff8400. Der Kontrast dieser zwei Farben ist ein guter Blickfang und angenehm für das Auge. Als Kontur wird ein einfaches Schwarz (#000000) herangezogen. Durch diese drei Hauptfarben wird dem Benutzer ein visuell-angenehmes Farbenspektrum geboten, wodurch eine klare Übersicht über alle Elemente entsteht.

- Hauptfarbe/Kontrastfarbe: #ff8400 (Orange)
- Hintergrundfarbe: #ff8400 (Grau)
- Konturfarbe: #000000 (Schwarz)

Alle weitere Farben entsprechen einem verwandten Farbton, die jedoch nicht erwähnenswert sind.

11.14.1 FIPLY-Logo

Am Anfang war das Ziel des FIPLY-Logos, dem Benutzer bei der Ansicht des Bildes intuitiv zu vermitteln, dass es sich um eine Fitness Applikation handelt. Dies geschieht einfach mit verschiedenen fitnessstypischen Formen, wie zum Beispiel eine Hantel, ein laufender Mensch, ein Muskulöser Arm oder auch nur einen Turnschuh. Da die Applikation das Generieren eines Fitnessplans anbietet, wurde entschieden, eine Art Kalender-Logo daraus zu machen.



(a) Version 1 des FIPLY Logos

In der ersten Version des FIPLY Logos (Grafik a dieses Abschnitts) umfasst nur einen Schwarz-Weiß Farbstil mit einem zentrierten “FIPLY“ Schriftzug. Darunter findet sich das Bild eines typischen Kalenders, dass die Involvierung eines Plans suggestieren soll. Da es in den Augen der Projektmitglieder aber zu wenig Wiedererkennungswert aufweist, wurde entschieden, ein neues Logo zu entwerfen.

Ziel ist es diesmal, ein Logo zu erstellen, das großen Wiedererkennungswert besitzt und sich gleichzeitig von anderen Logos auf eine individuelle Art und Weise heraussticht. Dabei kann die Suggestierung an den Fitnessbereich vernachlässigt werden. Ein möglichst einfach gehaltenes Logo mit ebenso einfachen Farben, ohne Farbverlauf. Dabei wird auf das Standardfarbschema des FIPLY Projekts zurückgegriffen:



(a) Version 2 des FIPLY Logos

11.14.2 Kontrollelemente

Als Hintergrundfarbe der Kontrollelemente wird die Hauptfarbe angewendet. Bei normalen Textelementen kommt die Hauptfarbe zum Einsatz, bei jeden anderen texthaltigen Elementen wird etwa die Konturfarbe oder Hintergrundfarbe eingesetzt. Dies wird mittels Icons umgesetzt. Das jeweilig erstellte Icon wird als Hintergrundbild eines Kontrollelements programmiert. Folgende wichtige Icons wurden für Kontrollelemente eingesetzt, das jeweilige Icon wird eingesetzt für *:



(a) * den play-Knopf des Musikplayers



(b) * den pause-Knopf des Musikplayers



(c) * den voriger Song-Knopf des Musikplayers



(d) * den nächster Song-Knopf des Musikplayers



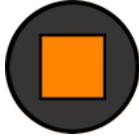
(e) * eine Erfolgsmeldung



(f) * eine Abfrage



(g) * eine Fehlermeldung



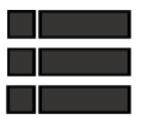
(h) * den stop-Knopf des Musikplayers



(i) * einen Informations-Knopf



(j) * einen leeren Bewertungstern



(k) * eine (spezielle) Listenbearbeitung, ungedrückt



(l) * die Stoppuhr



(m) * den exportieren-Knopf



(n) * einen vollen Bewertungstern



(o) * eine (spezielle) Listenbearbeitung, gedrückt



(p) * eine countdown Uhr



(q) * das löschen eines Elements, ungedrückt



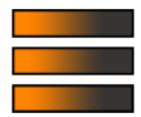
(r) * das löschen eines Elements, gedrückt



(s) * das hinzufügen eines Elements, ungedrückt



(t) * das löschen eines Elements, gedrückt



(u) * eine (normale) Listenbearbeitung, gedrückt



(v) * eine (normale) Listenbearbeitung, ungedrückt

124w) * einen Toggle-Knopf, ein

(x) * einen Toggle-Knopf, aus

12 Kommerzialisierung

12.1 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.

AdMob unterstützt zwei verschiedene Arten von Werbungen. Zum einen gibt es, die sich am Rand des Bildschirms befindlichen, Banner Ads, zum anderen, die den ganzen Bildschirm abdeckenden, Interstitial Ads.

[Google Developers [30]]

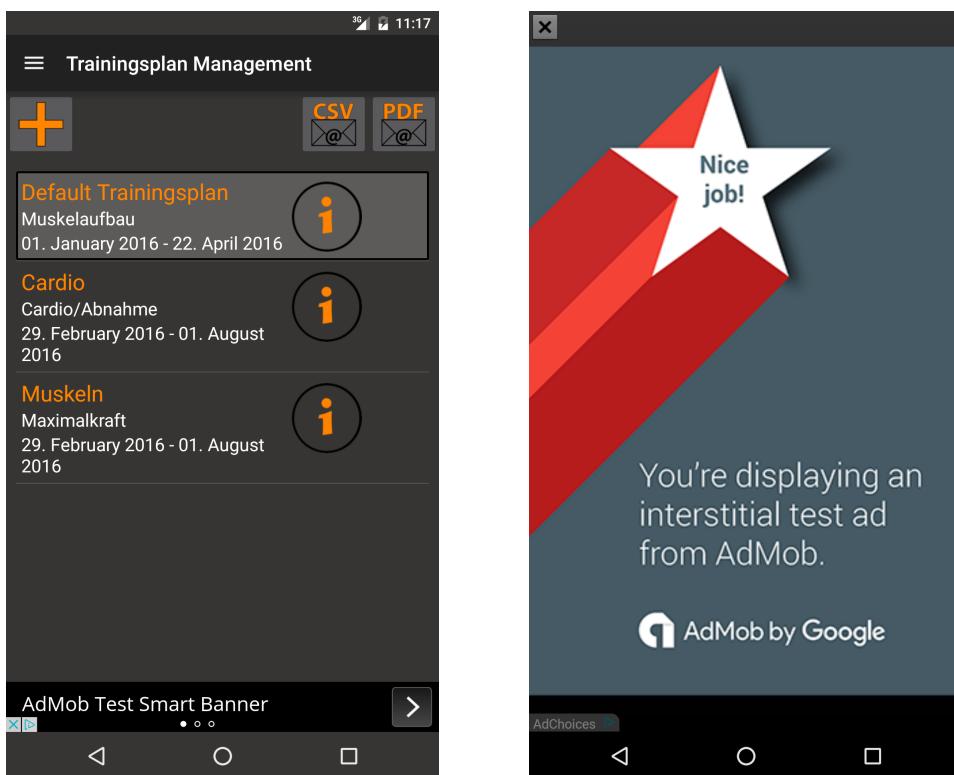


Abbildung 44: Links ist ein Beispiel für eine Banner Ad zu sehen. Rechts sieht man ein Beispiel für eine Interstitial Ad.

12.1.1 Banner Ads

Banner Ads nehmen einen kleinen Teil des Bildschirms ein. Diese werden in einem Layout XML File erstellt und dann in einer Activity, oder in einem Fragment geladen. Der Benutzer wird durch einen Klick auf das Banner, auf die beworbene Webseite weitergeleitet.

[Google Developers [31]]

Listing 53: AdView in einem Layout XML File

```
1 <com.google.android.gms.ads.AdView
2     android:id="@+id/planAdView"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:layout_centerHorizontal="true"
6     android:layout_alignParentBottom="true"
7     ads:adSize="SMART_BANNER"
8     ads:adUnitId="ca-app-pub-3940256099942544/1033173712"
9 ></com.google.android.gms.ads.AdView>
```

Hier wird ein Banner in einem Layout XML File definiert. Dieses Banner befindet sich am unteren Ende der App und überspannt die gesamte Breite des Bildschirms. Die oben eingetragene Ad Unit Id liefert TestAds und dient zum Testen.

Listing 54: Anfordern einer Banner Ad

```
1 int gender = AdRequest.GENDER_MALE;
2
3 AdView mAdView = (AdView) getActivity()
4     .findViewById(R.id.planAdView);
5 AdRequest adRequest = new AdRequest.Builder()
6     .addTestDevice(AdRequest.DEVICE_ID_EMULATOR)
7     .setGender(gender)
8     .build();
9 mAdView.loadAd(adRequest);
```

Dieser Code beinhaltet das Anfordern einer Werbung und anschließend wird das Laden derselben eingeleitet. Das Banner wird erst dann sichtbar, wenn die Werbung komplett geladen ist. Die angeforderte Werbung kann durch Daten wie Geschlecht, Geburtstag oder Position präziser auf den Benutzer abgestimmt werden. Diese Abstimmung erfolgt durch die Methoden `.setGender()`, `.setLocation()` und `.setBirthday()`.

12.1.2 Interstitial Ads

Interstitial Ads bedecken den gesamten Bildschirm. Wird die Werbung angezeigt, kann der Benutzer die Anzeige schließen oder dem Link der Werbung folgen. Deshalb eignen sich Interstitials für Apps die gelegentlich zwischen mehreren Bildschirmen wechseln. Bei diesen Werbungen ist zu beachten, dass die App im Hintergrund weiterläuft, also sollten laute Tonwiedergaben und ressourcenintensive Benutzerinteraktionen pausiert werden.

[Google Developers [32]]

Listing 55: Erstellen und Anfordern einer Interstitial Ad

```
1 mInterstitialAd = new InterstitialAd(this);
2 mInterstitialAd =
3     setAdUnitId("ca-app-pub-3940256099942544/1033173712");
4 requestNewInterstitial();
```

Hier wird eine Interstitial Ad angelegt und anschließend eine Werbung für das Interstitial angefordert. Die oben eingetragene Ad Unit Id liefert TestAds und dient zum Testen.

Listing 56: Die requestNewInterstitial()-Methode

```
1 public void requestNewInterstitial() {
2     int gender = AdRequest.GENDER_MALE;
3
4     AdRequest adRequest = new AdRequest.Builder()
5         .addTestDevice(AdRequest.DEVICE_ID_EMULATOR)
6         .setGender(gender)
7         .build();
8     mInterstitialAd.loadAd(adRequest);
9 }
```

Mit diesem Code wird eine neue Werbung angefordert. Die angeforderte Werbung kann durch Daten wie Geschlecht, Geburtstag oder Position präziser auf den Benutzer abgestimmt werden. Diese Abstimmung erfolgt durch die Methoden `.setGender()`, `.setLocation()` und `.setBirthday()`.

Listing 57: Das Anzeigen einer Interstitial Ad, sobald sie geladen ist

```
1 if (mInterstitialAd.isLoaded())
2     mInterstitialAd.show();
```

Ist die Werbung geladen kann sie angezeigt werden. Ebenso sind zahlreiche Methoden im AdListener vorhanden, die es ermöglichen, das Verhalten nach einer Interaktion mit der Werbung zu verwalten. Deshalb liegt es nahe, die nächste Werbung im `onAdClosed()` eines AdListeners zu laden, um diese jederzeit auf Abruf anzeigen zu können.

12.2 Donations

Donations stellen eine Möglichkeit für Benutzer dar den Entwicklern einer App Geld zu spenden, um Ihren Dank auszudrücken oder die Entwicklung der App zu unterstützen. Es gibt viele Möglichkeiten, um die Implementierung von Donations zu unterstützen.

Zu den populärsten Formen zählen Dienste wie PayPal, Flattr, das Implementieren von In-App-Käufen, die keinen Gegenwert liefern, oder das Erstellen einer kostenpflichtigen App, die keine Funktionen beinhaltet und denselben Namen wie die Gratis App aber einen Suffix wie z.B.: Donation trägt.

12.2.1 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."

[Flattr [33]]

"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."

[Flattr [33]]

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

12.2.2 PayPal

PayPal und deren Mobile Payment Libraries unterstützen die einfache Implementierung eines "Pay with Paypal"-Buttons, über den Käufe mittels eines PayPal-Accounts 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 beschrieben sind.

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

[Google Play [34]]

"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."

[Google Play [35]]

12.2.3 Donations über In-App-Käufe

Nach diesem Modell werden In-App-Käufe zur Verfügung gestellt, die dem Benutzer die Möglichkeit geben Geld zu bezahlen, ohne einen Gegenwert zu erhalten. Die In-App-Käufe werden im Kapitel Freemium näher beschrieben.

12.2.4 Zweite kostenpflichtige App

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

12.3 Freemium

Freemium ist ein Konzept, welches 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.

[Android Developers [36], Youtube [37]]

12.3.1 Consumable Items

Consumable Items sind Artikel, die benutzt 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.

12.3.2 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 demselben Google Play Account eingeloggt ist.

12.3.3 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 Tankfüllungen in einem Spiel zur Verfügung stellt, solange diese aktiv ist.

12.3.4 Gratis/Bezahlte Versionen

Beschreibung

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

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.

Nachteile

- Der Hauptteil aller Applikationen die dieses Modell wählen, sind während ihrer Lebenszeit nicht profitabel. 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 mit einer App am Markt durchzusetzen.

Implementierung

Android Studio bietet eine Möglichkeit eine bezahlte und eine gratis Version in einem Projekt zu entwickeln. Dies macht es trivial dieses Konzept in die Realität umzusetzen.

12.3.5 Fixpreis

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.

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 (Mehrwertsteuer, etc.)

Nachteile

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

Preis

Der Preis unseres Produkts 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.

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.

12.3.6 SellApp

Beschreibung

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

Vorteile

- Der Gesamtwert einer gut gepflegten App mit großer Userbasis liegt extrem hoch und diese Methode wäre eine der rentabelsten.
- Flexibilität bei der Weiterentwicklung, die Entwickler können entscheiden ob wir sie das Produkt weiterentwickeln oder nicht. Bei Nichtbeteiligung ist die Instandhaltung der Applikation aus ihren Händen und das Projekt abgeschlossen.

Nachteile

- Falls entschieden wird die Applikation mit der Firma, welche sie erworben hat, weiter zu arbeiten, ist die gestalterische Freiheit der Entwickler eingeschränkt und es ist schwerer Visionen umzusetzen.

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.

12.4 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 Zielgruppenorientierung bestimmen. [*Möglichkeiten der App Vermarktung: Teil 1* [38]]

12.4.1 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 von dem Projekt überzeugen, 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:

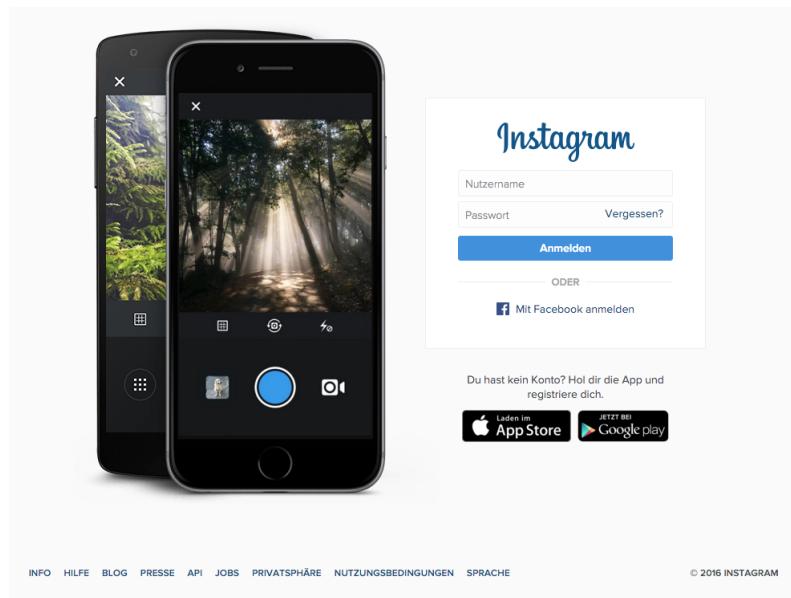


Abbildung 45: Screenshot von <http://instagram.com>:

Instagram.com ist ein gutes Beispiel für solch eine Teaser-Website. Eine weitere Möglichkeit ist es ein Video in die Website einzubauen, welche die Nutzung 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 Medioplayer einzubetten ist auch ein beliebtes Mittel.

12.4.2 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 beliebten sozialen Netzwerke wie Facebook, Google+, Twitter, Instagram, Vine,... und unzähligen mehr passieren. 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 gewonnen werden können. Zusätzlich können Administratoren beliebter 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.

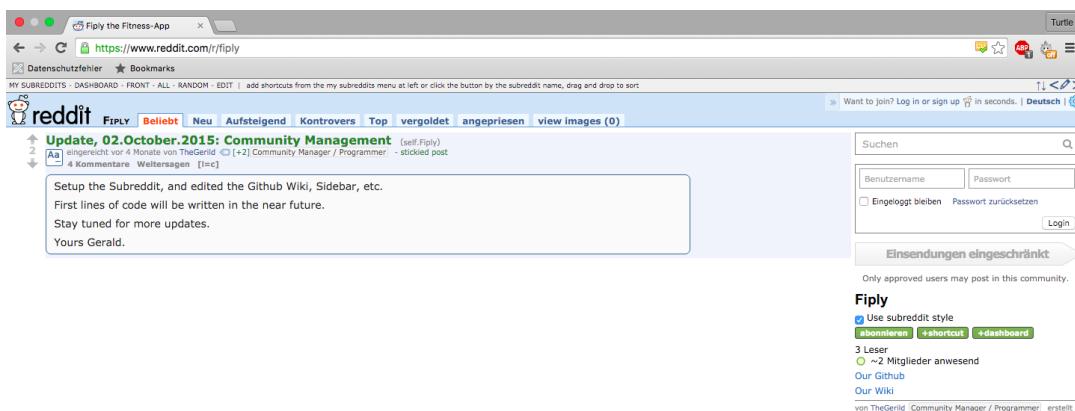


Abbildung 46: Screenshot von <http://reddit.com/r/fiply>

12.4.3 Presse

Eine der einfachsten Möglichkeiten ist es, eine Email an selektierte Schreiber die sich auf etwaige Bezugsthemen spezialisieren (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 motivieren über das Projekt zu schreiben.

12.4.4 Wettbewerbe

Kaum mit etwas anderem kann man schneller für Aufmerksamkeit sorgen als mit dem Gewinnen von Wettbewerben. Auch schon nur bei einer Einreichungen und Nominierung kann man einen hohen 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. Wettberwerbe 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.

12.4.5 Persönliche Werbung

In erster Linie sollte man seine Bekannten und Freunde um ein ehrliches Feedback bitten, ob das Projekt denn wirklich gut bei dem Endbenutzer ankommen kann. Objektive Meinungen von Freunden sind der Grundstein für die Erfolgsvorhersehung 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 persönliche Vermittlung. Mundpropaganda mag nicht sehr schnell sein, dafür aber sehr effektiv.

12.4.6 Reklame

Bei Reklamen/Werbeinschaltungen sollte in erster Linie auf die Zielgruppe geachtet werden. Erfolgreiche Reklame passiert nur durch zielgerichtete Zielgruppdefinition. 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, Websites 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.

12.4.7 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 stößt. 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 geachtet werden. Zwecks Wiedererkennungsfunktion sollte die App einen eigenständigen 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 Farbschemata und Formen des Icons automatisch mit der Applikation, welche möglichst simple und mit der modernen Designrichtlinie “Weniger ist mehr“ umgesetzt werden soll.

Bei der Einbettung von Medien im Google Playstore sollten die nützlichsten und besten Aspekte und Funktionen der Applikation wiedergegeben werden. Screenshots mit wenig Aussagekraft sollten vermieden werden, da es ein Limit für eingebundene Medien für das Appprofil gibt und diese möglichst effizient genutzt werden sollten. [*Möglichkeiten der App Vermarktung: Teil 2 [39]*]

12.4.8 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.

[*Möglichkeiten der App Vermarktung: Teil 2 [39]*]

Listings

1	Eintrag in die literatur.bib Datei	57
2	XML Datei vor dem Einsatz von Databinding.	65
3	Die Activity vor dem Einsatz von DataBinding.	66
4	XML Datei nach dem Einsatz von Databinding.	66
5	Die Activity vor dem Einsatz von DataBinding.	67
6	Layoutcode ohne jedliche Datenanbindung.	68
7	Unsere Objektklasse die bei der Datenanbindung referenziert wird.	68
8	Die XML Datei nach der Integration einer Datenanbindung. .	69
9	Die Aktivitätenklasse nach der Integration einer Datenanbindung.	70
10	Deklaration von Permissions im AndroidManifest.xml	71
11	Das Anfordern von Permissions ab Android 6.0	72
12	Anzeigen der Permissionsabfrage	73
13	onCreateView() das ein Fragment anzeigt	75
14	Eine Fragment Transaction	75
15	Die displayView-Methode vereinfacht das Anzeigen eines neuen Fragments	76
16	Die Definition eines FrameLayouts in einem Layout XML File	76
17	Erstellen und Befüllen von Bundels	77
18	Das Übergeben eines Bundles bei einer FragmentTransaction .	77
19	Der Zugriff auf ein Bundle im neuen Fragment	78
20	Übertragung durch das Bundle des Intents	78
21	Übertragung durch einen Intent mit einem neuen Bundle . .	78
22	Abkürzung durch die putExtra()-Methode eines Intents . .	79
23	Zugriff auf die Daten in der neuen Activity durch die getExtras()-Methode	79
24	Zugriff auf die Daten in der neuen Activity durch getExtra()-Methoden	79
25	Layout XML File eines NavigationDrawers	81
26	Minimaler Code eines NavigationDrawers	82
27	Erweiterung des NavigationDrawers um ihn visuell ansprechender zu machen	82
28	Methode, um den aktuellen Pfad herauszufinden.	88
29	Verwendung von CSVReader: Möglichkeit 1, iterativ	88
30	Verwendung von CSVReader: Möglichkeit 2, alles auf einmal .	88
31	Verwendung von CSVWriter: Möglichkeit 1, iterativ	89
32	Verwendung von CSVWriter: Möglichkeit 2, alles auf einmal .	89
33	Verwendung von CSVWriter: Möglichkeit 2, alles auf einmal .	90

34	Ein JSON-Objekt welches eine Übung beschreibt.	95
35	Diese Methode lese den JSON-string ein und fügt die Übungen in die Datenbank ein.	96
36	Vergleicht die Farben und setzt den Filter.	100
37	Ermittelt ob zwei Farbwerte ähnlich oder gleich sind.	101
38	Ermittelt den Farbwert des gedrückten Bereiches.	101
39	Ursprünglicher Ansatz zum Lesen aller mp3-Files im Music- Ordner	104
40	Die query-Methode des Android Source Codes in der <i>Content- Resolver.java</i> Klasse [28]	104
41	Alle Audiodateien, die Musik beinhalten, werden gesucht.	104
42	Speichern der Songverweise in eine ArrayList von HashMaps .	106
43	Speichern der Songverweise in eine ArrayList von HashMaps .	107
44	Die changeSong-Methode	108
45	Das Runnable aktualisiert die Fortschrittsanzeigen des Musik- players	108
46	Ein Datenpunkt für die Graphen.	113
47	Repository wird instanziert.	116
48	Das Datenbankobjekt db wird instanziert über den FiplyDB- Helper	116
49	db wird instanziert und ein SQL Statement wird ausgeführt. .	116
50	Eine Contract Klasse, mit allen Metadaten.	118
51	Die Facebook App ID in der strings.xml.	119
52	Das xml-layout des FacebookLoginButtons.	119
53	AdView in einem Layout XML File	126
54	Anfordern einer Banner Ad	126
55	Erstellen und Anfordern einer Interstitial Ad	127
56	Die requestNewInterstitial()-Methode	127
57	Das Anzeigen einer Interstitial Ad, sobald sie geladen ist . .	127

Abbildungsverzeichnis

6	Die Systemarchitektur zu der Applikation	13
7	Das Use-Case Diagramm zu der Applikation	14
8	GUI für den Aufruf/Ablauf des 1. Use Cases:	15
9	Prozesskette des 1. Use Cases:	15
10	GUI für den Aufruf/Ablauf des 2. Use Cases:	17
11	GUI für den Aufruf/Ablauf des 3. Use Cases:	19
12	GUIs für den alternativen Abläufe des Use Cases:	20
13	GUI für den Aufruf/Ablauf des 4. Use Cases:	22
14	GUI für den Aufruf/Ablauf des 4. Use Cases:	22
15	GUI für den alternativen Aufruf/Ablauf des 4. Use Cases:	24
16	GUI für den alternativen Aufruf/Ablauf des 5. Use Cases:	25
17	GUI für den Aufruf/Ablauf des 6. Use Cases:	26
18	GUI für den Aufruf/Ablauf des 6. Use Cases:	27
19	Vorgangsvisualisierung der Trainingsplanphasen	43
20	Der Ablauf einer Änderung am Code.	50
21	Aussehen der Zitate im PDF	57
22	Zum Beispiel kann man mithilfe von Fragments Views erstellen, die sowohl auf einem Tablet, als auch auf einem Handy ein optimales Benutzerinterface anbieten. [Android Developers [17]]	74
23	Links wird der NavigationDrawer versteckt. Rechts ist der NavigationDrawer geöffnet.	80
24	Der erste Schritt der Usererstellung.	84
25	Der Loginscreen von Facebook.	84
26	Der zweite Schritt der Usererstellung.	85
27	Der dritte Schritt der Usererstellung.	86
31	Der Übungskatalog.	94
32	Die DetailView der Übung "Bizepscurl".	97
33	Die Filter View.	98
34	Das sichtbare Overlay des Filters.	99
35	Die nicht sichtbare Maske.	99
36	Beim Klicken eines Elements in Screenshot 1 werden alle Songs angezeigt (Screenshot 2). Mittels der Checkboxen werden jene Songs markiert, die sich in der ausgewählten Playlist befinden.	106
37	In einem Spinner kann eine Playlist ausgewählt werden. Beim Klick auf einen Song wird dieser abgespielt.	109
38	Die MusicControls in Ruhe und während einer Wiedergabe.	110
39	Die Statistik.	111
40	Die Datenbank-Architektur.	114

44	Links ist ein Beispiel für eine Banner Ad zu sehen. Rechts sieht man ein Beispiel für eine Interstitial Ad.	125
45	Screenshot von http://instagram.com :	134
46	Screenshot von http://reddit.com/r/fiply	135

Literatur

- [1] *Fachkonzept zur Erstellung eines Trainingsplans*. Jan. 2016. URL: <https://www.facebook.com/Lindenpower-Fitness-174332482640559/>.
- [2] Moritz Stückler. *Was ist eigentlich dieses GitHub?* März 2016. URL: <http://t3n.de/news/eigentlich-github-472886/>.
- [3] *Set Up Git*. März 2016. URL: <https://help.github.com/articles/set-up-git/>.
- [4] *Android Studio Dev*. März 2016. URL: <https://developer.android.com/tools/studio/index.html>.
- [5] *LaTeX*. März 2016. URL: <https://de.wikipedia.org/wiki/LaTeX>.
- [6] *MiKTeX*. März 2016. URL: <https://de.wikipedia.org/wiki/MiKTeX>.
- [7] *TeXworks*. März 2016. URL: <https://de.wikipedia.org/wiki/TeXworks>.
- [8] *TeXStudio*. März 2016. URL: <http://latex.tugraz.at/programme/texstudio>.
- [9] *Adobe Photoshop*. März 2016. URL: <http://www.adobe.com/>.
- [10] *DataBinding Definition*. März 2016. URL: https://www.it-visions.de/glossar/alle/6875/Data_Binding.aspx.
- [11] *Droidcon NYC 2015 - Data Binding Techniques*. Sep. 2015. URL: <https://www.youtube.com/watch?v=WdUbXWztKNY>.
- [12] *Data Binding Guide*. März 2016. URL: <http://developer.android.com/tools/data-binding/guide.html>.
- [13] Sittiphol Phanvilai. *Everything every Android Developer must know about new Android's Runtime Permission*. März 2016. URL: <http://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en>.
- [14] *Requesting Permissions at Run Time*. März 2016. URL: <http://developer.android.com/training/permissions/requesting.html>.
- [15] *System Permissions*. März 2016. URL: <http://developer.android.com/guide/topics/security/permissions.html>.
- [16] *Fragment*. Feb. 2016. URL: <http://developer.android.com/reference/android/app/Fragment.html>.
- [17] *Fragments Guide*. Feb. 2016. URL: <http://developer.android.com/guide/components/fragments.html>.

- [18] Jeremy Logan. *Passing a Bundle on startActivity()*? März 2016. URL: <http://stackoverflow.com/questions/768969/passing-a-bundle-on-startactivity>.
- [19] Ben Jakuben. *How to Add a Navigation Drawer in Android*. Feb. 2016. URL: <http://blog.teamtreehouse.com/add-navigation-drawer-android>.
- [20] Ravi Tamada. *Android Sliding Menu using Navigation Drawer*. März 2016. URL: <http://www.androidhive.info/2013/11/android-sliding-menu-using-navigation-drawer/>.
- [21] *Creating a Navigation Drawer*. Feb. 2016. URL: <http://developer.android.com/training/implementing-navigation/nav-drawer.html>.
- [22] Vgl.: *OpenCSV Website*. März 2016. URL: <http://opencsv.sourceforge.net/>.
- [23] Vgl.: *OpenCSV Writer*. März 2016. URL: <http://viralpatel.net/blogs/java-read-write-csv-file/>.
- [24] *Mails versenden in Android*. März 2016. URL: <http://stackoverflow.com/questions/5401104/android-exporting-to-csv-and-sending-as-email-attachment>.
- [25] Bill Lahti. *Android Images With Clickable Areas – Part 1*. März 2016. URL: <https://blahti.wordpress.com/2012/06/26/images-with-clickable-areas>.
- [26] *MediaPlayer*. Feb. 2016. URL: <http://developer.android.com/reference/android/media/MediaPlayer.html>.
- [27] Ravi Tamada. *Android Building Audio Player Tutorial*. Feb. 2016. URL: <http://www.androidhive.info/2012/03/android-building-audio-player-tutorial/>.
- [28] 2006 The Android Open Source Project. *ContentResolver.java*. März 2016.
- [29] *SQLite*. März 2016. URL: <https://www.sqlite.org>.
- [30] *Adding AdMob into an Existing App*. Feb. 2016. URL: <https://developers.google.com/admob/android/existing-app>.
- [31] *Banner Ads*. Feb. 2016. URL: <https://developers.google.com/admob/android/banner>.
- [32] *Interstitial Ads*. Feb. 2016. URL: <https://developers.google.com/admob/android/interstitial>.

- [33] *About Flattr*. Feb. 2016. URL: <https://flattr.com/about>.
- [34] *Google Play-Programmrichtlinien*. Feb. 2016. URL: https://play.google.com/intl/ALL_de/about/developer-content-policy.html.
- [35] *Google Play In-app Billing*. Feb. 2016. URL: <https://support.google.com/googleplay/android-developer/answer/6151557>.
- [36] *In-App-Billing*. Feb. 2016. URL: https://developer.android.com/google/play/billing/billing_admin.html.
- [37] Jarek Wilkiewicz. *Implementing Freemium*. Feb. 2016. URL: <https://www.youtube.com/watch?v=UvC15Xx7Z5o>.
- [38] *Möglichkeiten der App Vermarktung: Teil 1*. Apr. 2016. URL: <http://www.app-entwicklung.info/2015/04/moglichkeiten-der-app-vermarktung-teil-1-website-social-media-presse-events-co/>.
- [39] *Möglichkeiten der App Vermarktung: Teil 2*. Mai 2015. URL: <http://www.app-entwicklung.info/2015/05/moglichkeiten-der-app-vermarktung-teil-2-die-app-store-optimierung-aso/>.