

FIPLY

Daniel Bersenkovitsch, Andreas Denkmayr, Gerald Irsiegler

19. Februar 2016

Inhaltsverzeichnis

1	Pflichtenheft	7
1.1	Motivation	7
1.2	Ausgangssituation und Zielsetzung	7
1.2.1	Ausgangssituation	7
1.2.2	Beschreibung des Geschäftsfeldes	7
1.2.3	Zielbestimmung	7
1.3	Funktionale Anforderungen	7
1.3.1	Use Case Diagram (1 bis 6)	8
1.3.2	Systemarchitektur	9
1.3.3	Use Case Details	9
1.4	Nicht-funktionale Anforderungen	22
1.5	Mengengerüst	22
1.6	Risikovermeidung	23
1.7	Abnahmekriterien	23
2	Der Trainingsplan	24
2.1	Begriffserklärung	24
2.2	Einleitung	24
2.3	Phase 1: Allgemein	26
2.4	Phase 2: Kraftausdauer (Gesundheit)	27
2.5	Phase 2: Muskelaufbau Phase 3: Kraftausdauer	28
2.6	Phase 2: Maximalkraft Phase 3: Muskelaufbau	29
2.7	Phase 3: Maximalkraft	30
2.8	Mobilisation	31

3	Verwendete Technologien	32
3.1	GitHub	32
3.1.1	Repositories	32
3.1.2	Branches	32
3.1.3	Commits	33
3.1.4	Pull Request	33
3.1.5	Community	33
3.1.6	Issues	33
3.1.7	Integration	33
3.1.8	.gitignore	34
3.1.9	GitHub Desktop	34
3.1.10	IntelliJ	35
3.2	LaTeX	36
4	Latex	36
4.0.1	Subfiles	36
4.0.2	Quellenangaben	36
5	Planung	37
6	Designrichtlinien	37
6.1	Der Splashscreen	37
6.2	Animationen und Transactions	37
6.3	Navigation	39
6.4	Formulare	40
6.5	Datenausgabe	41
6.6	Style	43
7	Datenanbindung	44
7.1	Verwendungszweck	44
7.2	Vorher	44
7.3	Nacher	45
7.4	Anwendung	46
7.4.1	In der .xml Datei	47
7.4.2	In der .java Klasse	49
8	Umsetzung	50
8.1	Permissions	50
8.1.1	Arten von Permissions	50
8.1.2	bis Android 5.1 (API level 22)	51
8.1.3	ab Android 6.0 (API level 23)	51

8.1.4	Permission groups	52
8.2	Fragments	53
8.2.1	Was sind Fragments?	53
8.2.2	Der Lifecycle	54
8.2.3	Fragment Transactions	54
8.2.4	Verwendung von Fragments	55
8.3	Bundles	57
8.4	NavigationDrawer	58
9	Benutzerverwaltung	61
9.1	Beschreibung	61
9.2	Aufteilung der Verwaltung	61
9.2.1	Schritt 1	61
9.2.2	Schritt 2	62
9.2.3	Schritt 3	63
9.3	Implementierung	64
10	Exportieren	65
10.1	Als CSV exportieren - OpenCSV	65
10.1.1	Alternative	65
10.1.2	Anwendung	65
10.2	Emails senden	67
10.2.1	Mittels Intents	67
10.2.2	Vorteile/Nachteile von Intents	67
11	Übungskatalog	69
11.1	Beschreibung	69
11.2	Implementierung	69
11.2.1	Expandable List View	69
11.2.2	List View	69
11.2.3	Einlesen der Übungen	70
11.3	DetailView	72
11.4	Filter	73
11.4.1	Beschreibung	73
11.4.2	Implementierung	74
12	VideoView	77
12.1	Beschreibung	77
12.2	Vorteile	77
12.3	Probleme	77

13 Youtube Android Player API	77
13.1 Beschreibung	77
13.2 Vorteile	77
13.3 Probleme	77
14 WebView	78
14.1 Beschreibung	78
14.2 Vorteile	78
14.3 Probleme	78
15 Nutzwertanalyse	78
15.1 KO-Kriterien	78
15.2 Erfüllung der KO-Kriterien	78
15.2.1 VideoView	78
15.2.2 Youtube Android Player API	78
15.2.3 WebView	78
15.3 Schlussfolgerung	79
15.4 Musik	80
15.4.1 Lokalisierung der Musikdateien	80
15.4.2 Verwalten von Playlists	81
15.4.3 Abspielen der Playlists.	83
15.4.4 MusicControls	85
16 Statistik	86
16.1 Beschreibung	86
16.2 Verfügbare Statistiken	87
16.2.1 Geistige Verfassung	87
16.2.2 Gehobenes Gewicht	87
16.2.3 Insgesamt gestimmtes Gewicht	87
16.3 Implementierung	88
16.3.1 Aufnahme und Speicherung der Werte	88
16.3.2 Darstellung	88
17 Datenbank	89
17.1 SQLite	89
17.2 Zugriff auf die Daten	89
17.2.1 Physische Ebene	89
17.2.2 DB Helper	89
17.2.3 Repositories	90
17.3 Contract	91

18 Social Media	92
18.1 Beschreibung	92
18.2 Verknüpfung mit Facebook	92
18.2.1 FacebookSDK	92
18.2.2 Facebook Login Button	92
18.2.3 Test Account	93
18.3 Notifications	94
18.4 Notifications	95
19 Design	96
20 Kommerzialisierung	97
20.1 Advertising mit AdMob	98
20.1.1 Banner Ads	99
20.1.2 Interstitial Ads	100
20.2 Donations	101
20.2.1 Flattr	101
20.2.2 PayPal	102
20.2.3 Donations über In-App-Käufe	102
20.2.4 Zweite kostenpflichtige App	102
20.3 Freemium	103
20.3.1 Consumable Items	103
20.3.2 Non-Consumable Items	103
20.3.3 Subscriptions	103
20.4 Free/Paid Versions	104
20.4.1 Beschreibung	104
20.4.2 Vorteile	104
20.4.3 Nachteile	104
20.4.4 Implementierung	104
20.5 PaidVersion	105
20.5.1 Beschreibung	105
20.5.2 Vorteile	105
20.5.3 Nachteile	105
20.5.4 Preis	105
20.5.5 Google-Play-Store	105
20.6 SellApp	106
20.6.1 Beschreibung	106
20.6.2 Vorteile	106
20.6.3 Nachteile	106
20.6.4 Probleme/Schwierigkeiten	106
20.7 Promotion	107

20.7.1	Website	107
20.7.2	Social Media Reputation	108
20.7.3	Presse	109
20.7.4	Konteste	109
20.7.5	Persönliche Werbung	109
20.7.6	Reklame	110
20.7.7	In Appstore Optimierung (IAO)	110
20.7.8	Fazit	111

1 Pflichtenheft

1.1 Motivation

Das Projekt wird im Rahmen einer Diplomarbeit durchgeführt.

1.2 Ausgangssituation und Zielsetzung

1.2.1 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.

1.2.2 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.

1.2.3 Zielbestimmung

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

1.3 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 Konditionstraining 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 Gesundheitszustand) abgefragt, um seine Entwicklung akkurat darstellen zu können.

1.3.1 Use Case Diagram (1 bis 6)

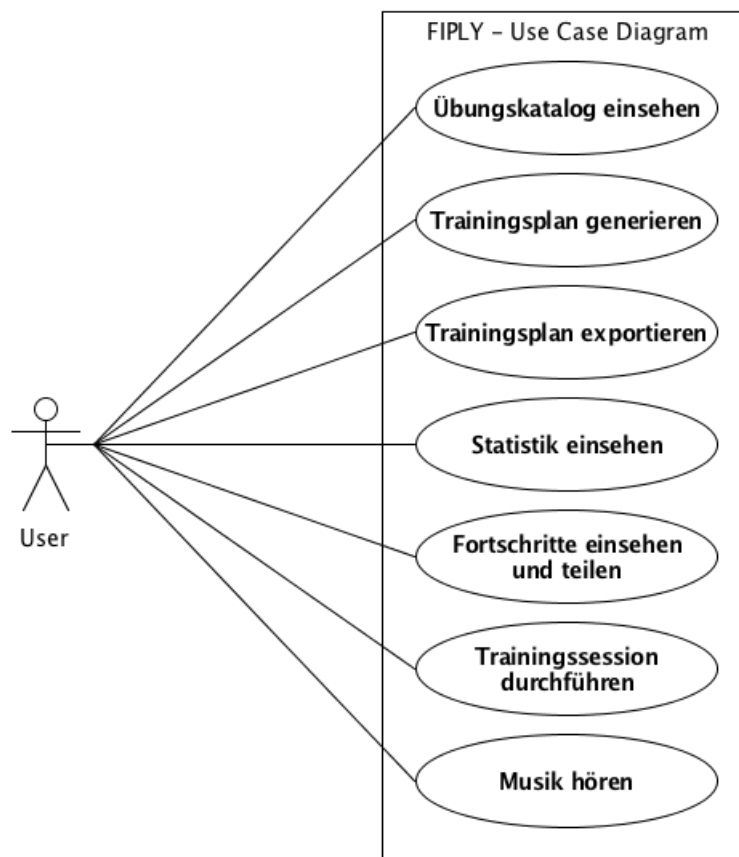


Abbildung 1: Das Use-Case Diagramm zu der Applikation

1.3.2 Systemarchitektur

1.3.3 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“.

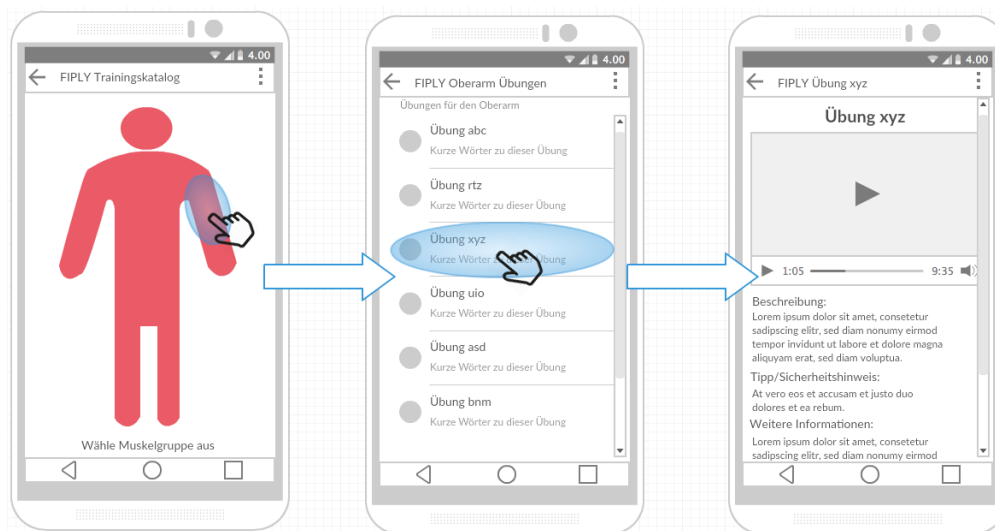


Abbildung 2: GUI für den Aufruf/Ablauf des 1. Use Cases:

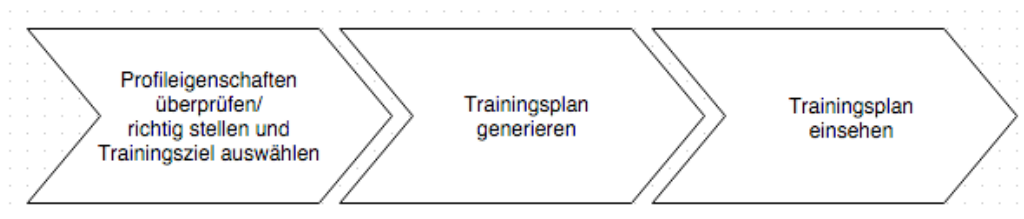


Abbildung 3: Prozesskette des 1. Use Cases:

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“.

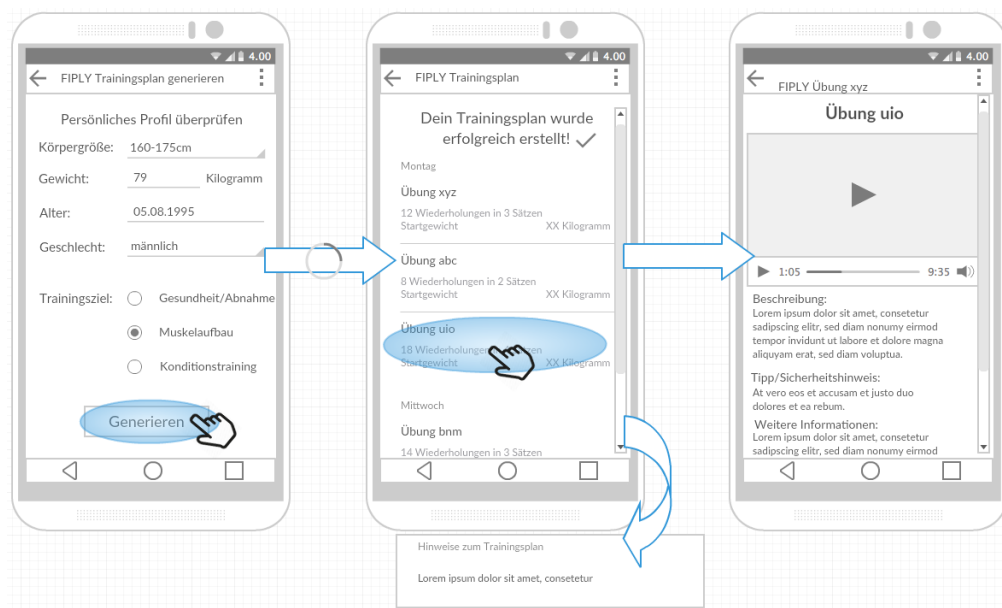


Abbildung 4: GUI für den Aufruf/Ablauf des 2. Use Cases:

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“.

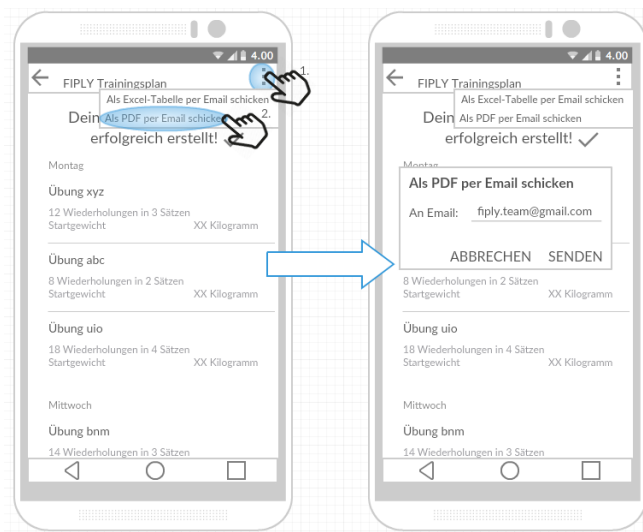


Abbildung 5: GUI für den Aufruf/Ablauf des 3. Use Cases:

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.

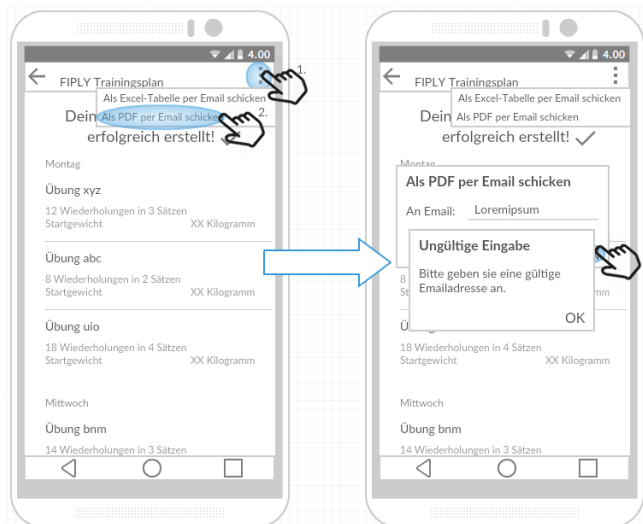


Abbildung 6: GUIs für den alternative Abläufe des Use Cases:

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:

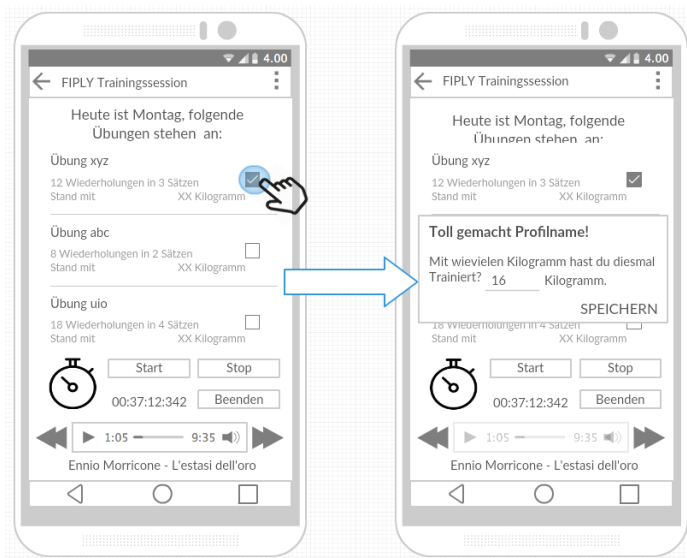


Abbildung 7: GUI für den Aufruf/Ablauf des 4. Use Cases:

Eingabefeld:	Erlaubte Eingabewerte
Kilogramm (Textfeld)	40 bis 140

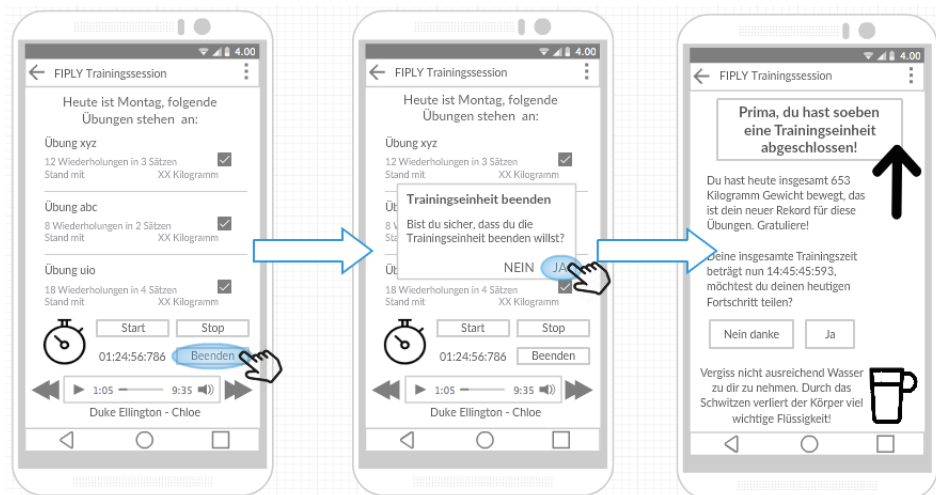


Abbildung 8: GUI für den Aufruf/Ablauf des 4. Use Cases:

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 „Trainingssession 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:

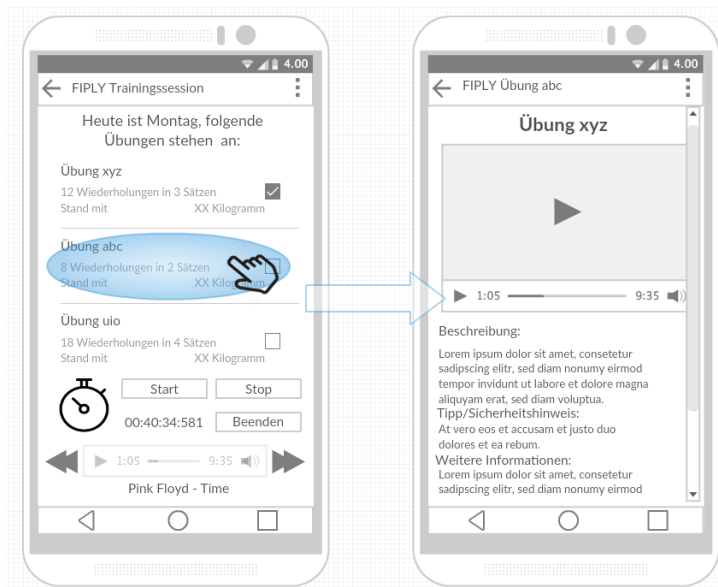


Abbildung 9: GUI für den alternativen Aufruf/Ablauf des 4. Use Cases:

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:

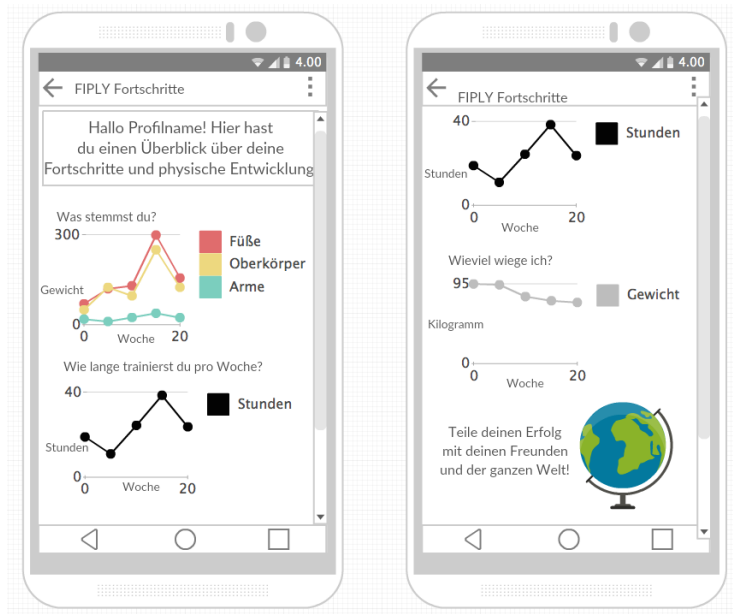


Abbildung 10: GUI für den alternativen Aufruf/Ablauf des 5. Use Cases:

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:

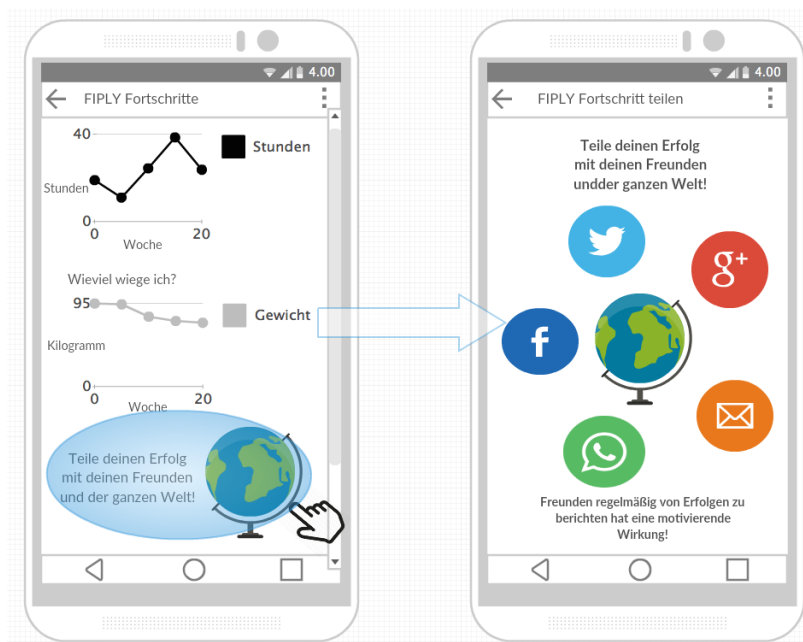


Abbildung 11: GUI für den Aufruf/Ablauf des 6. Use Cases:

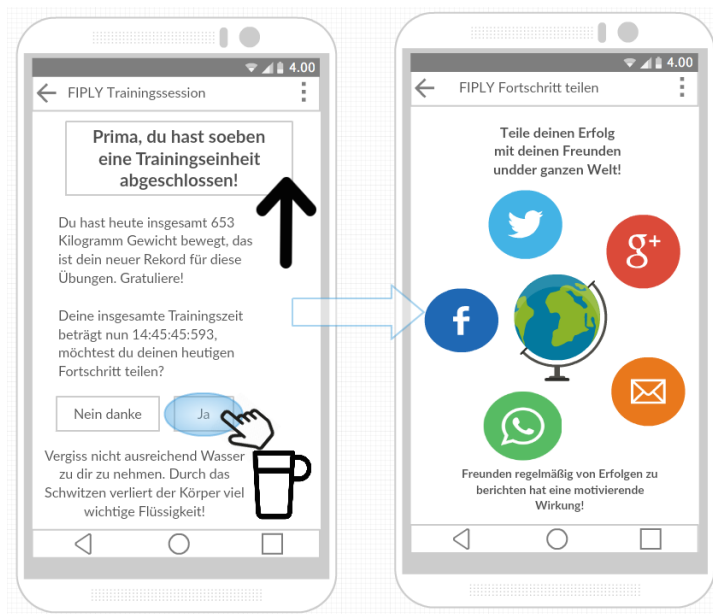


Abbildung 12: GUI für den Aufruf/Ablauf des 6. Use Cases:

1.4 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

1.5 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.

1.6 Risikovermeidung

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

1.7 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.

2 Der Trainingsplan

2.1 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:

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

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):

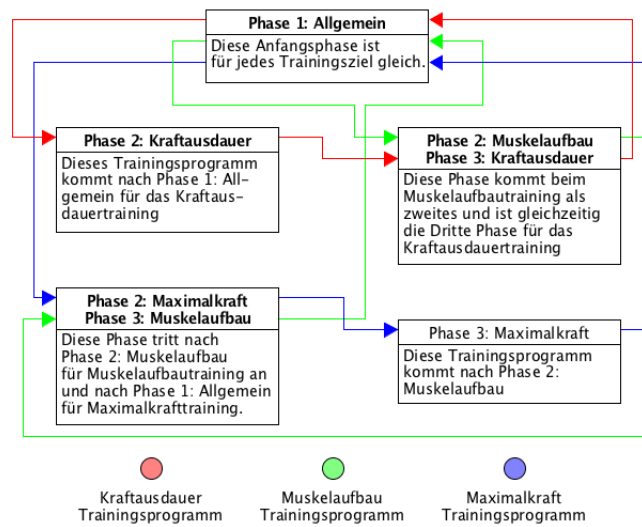
$$\textit{Trainingsgewicht} = \frac{\%RM * (102.78 - (2.78 * \textit{Wiederholungen}))}{100}$$

Mit dem errechneten Gewicht führt man nun die jeweilige zugeordnete Übung aus. Da bei konsequentem 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.

2.2 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 Trainingsziele eine unterschiedliche Reihenfolge haben. Hier eine Visualisierung der Reihenfolge der Trainingsphasen:



2.3 Phase 1: Allgemein

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

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

In Phase 1 werden alle Übungen mit einem Gewicht 55% RM ausgeführt. Es werden Anfangs 3 Trainingstage ausgewählt, an denen der Benutzer Zeit findet um zu trainieren. Dabei 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

2.4 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

2.5 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 eine 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.

2.6 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.

2.7 Phase 3: Maximalkraft

Phase 3: Maximalkraft kommt nach Phase 2: Maximalkraft. Hierbei wird lediglich 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.

2.8 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 größtmöglichem Bewegungsumfang fortlaufend um die eigene Achse drehen.
Becken-Mob	Die Arme über den Köpf führen, Handflächen nach oben schieben, Schultern bleiben tief, das Becken im Uhrzeigersinn, den ganzen Bewegungsumfang ausnutzen, Richtung ändern, die Kreise aus der Hüfte führen, die Beine sind stabil.
Wirbelsäule – Seitneigung	Linken Arm seitwärts hoch heben über den Kopf und Wirbelsäule seitwärts 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]]

3 Verwendete Technologien

3.1 GitHub

GitHub ist ein System zur kollaborativen Versionsverwaltung für Software-Entwicklungsprojekte. Für GitHub gibt es zahlreiche Clients, die meisten von diesen sind Kommandozeilen-Tools. Mit diesen Clients ist es möglich, Änderungen an Projekten zu speichern und jederzeit wieder auf vorherige Versionen zugreifen zu können. GitHub bietet viele Statistiken und ermöglicht auch Benutzern, ohne Erfahrung mit Kommandozeilen-Befehlen, einen sehr einfachen Umgang.

[Stückler [2], GitHub [3]]

3.1.1 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 dieses Repo sehen, 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.

3.1.2 Branches

Innerhalb eines Repositories gibt es mehrere Versionen einer Software. Diese Versionen werden in verschiedenen Branches abgespeichert. Es gibt eine Version auf der 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 einer Branch an größeren Änderungen, zum Beispiel an dem Wechsel auf eine alternative Technologie, gearbeitet werden.

3.1.3 Commits

Ein Commit ist der Vorgang eine neue Version einzureichen. Das bedeutet, dass diese neue Version auf die aktuelle Branch übernommen wird. Zu diesen Commits kann zurückgesprungen werden. Dies ermöglicht eine sehr hohe Transparenz im Entwicklungsvorgang. Das Projekt sollte zum Zeitpunkt eines Commits immer lauffähig sein!

3.1.4 Pull Request

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

3.1.5 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, diese einsetzen und daran mitentwickeln.

3.1.6 Issues

Da GitHub auf Kollaboration ausgelegt ist, bietet die Webseite auch ein Ticketsystem. Jeder mit einem kostenlosen Account kann Tickets erstellen. Eine 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 eine Issue geschlossen werden.

3.1.7 Integration

GitHub lässt sich sehr einfach in die meisten Entwicklungsumgebungen integrieren. Durch den hohen Bekanntheitsgrad wird es von den populärsten IDE's unterstützt. Sollte so ein Support nicht standardmäßig vorliegen, werden zahlreiche Extensions angeboten die so einen Support ermöglichen.

3.1.8 .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 Entwicklern, 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 *.bbl
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 genau diese spezifische Datei.

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

3.1.9 GitHub Desktop

GitHub Desktop ist der Standardclient für GitHub. Ist dieser Client installiert, ist eine enge Zusammenarbeit mit der GitHub Webseite möglich. So kann man 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.

TODO: AndroidStudio

3.1.10 IntelliJ

TODO

3.2 LaTeX

4 LaTeX

todo

TODO: TextStudio

4.0.1 Subfiles

TODO

4.0.2 Quellenangaben

TODO

TODO: Photoshop

TODO: Umlet

5 Planung

6 Designrichtlinien

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

6.1 Der Splashscreen

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

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

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

6.2 Animationen und Transactions

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

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

Beschreibung unserer Umsetzung:

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

6.3 Navigation

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

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

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

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

6.4 Formulare

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

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

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

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

6.5 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 Kindelement im Vordergrund anzeigen und sieht die restlichen Kindelemente dahinter angeordnet. Das angezeigte Element kann dabei durch jedes andere Kindelement ausgetauscht werden.
- **ListView:** Zeigt eine vertikale Liste von Elementen an.
- **ExpandableListView:** Zeigt eine vertikale Liste von Elementen an. Bei Klicken kann ein Element aufgeklappt werden um mehr Informationen anzuzeigen.
- **WebView:** Eine WebView kann Webseiten anzeigen ohne in den Webbrowser des Geräts wechseln zu müssen. In der WebView sind auch diverse Steuerelemente zur Navigation oder Textsuche enthalten.

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

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

6.6 Style

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

7 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* [4]]

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.

7.1 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* [5]]

7.2 Vorher

Hier auf der untenigen Grafik sehen wir eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität ohne Datenanbindung. Wir sehen ein LinearLayout mit zwei enthaltenen TextViews. Diesen wird in der Aktivität den Vornamen und den Nachnamen eines Employee Models zugewiesen. Ein Problem ist es, jedes Element eine ID zuweisen zu müssen. Wenn man 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 sich für jedes Element eine Unterschiedliche ID einfallen lassen, obwohl manche die selbe Funktion haben. Dadurch entstehen lange und unübersichtliche ID-Namen, die man sich nicht merken kann. Das Problem ist: Es muss immer darauf geachtet werden, keine doppelten IDs zu vergeben.

Listing 1: 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 2: Die Activity vor dem Einsatz von DataBinding.

```

1  public class OldWayActivity extends AppCompatActivity {
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 Aktivitätsencode vorfinden. Dieses Zugriffsverfahren ist, wie bereits oben erwähnt, unnötig performancelastig und kann vermieden werden.

7.3 Nacher

Listing 3: 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 4: Die Activity vor dem Einsatz von DataBinding.

```

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

```

Auf dieser Grafik sehen wir nun eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität mit der Verwendung von Datenanbindung. Man muss sich keine eindeutigen IDs für jedes vorkommende Element mehr ausdenken. 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 `Employee`-objektes angezeigt wird und bei der anderen der Nachname.

7.4 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 `dataBin-`

ding auf `enabled=true` (`dataBinding{enabled=true}`) setzen, um Datenanbindung möglich zu machen.

7.4.1 In der .xml Datei

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

Listing 5: 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-Feldern:

Listing 6: 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 7: 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.

7.4.2 In der .java Klasse

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

Listing 8: 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 Änderung des Elements, automatisch. [*Data Binding Guide* [6]]

8 Umsetzung

8.1 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 böartigen 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 [7], Android Developers [8][9]]

```
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>
```

8.1.1 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.

8.1.2 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.

8.1.3 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.

```
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("Permissionmessage",
13                 new DialogInterface.OnClickListener() {
14
15                 @Override
16                 public void onClick(DialogInterface dialog, int which) {
17                     ActivityCompat.requestPermissions(Settings.this,
18                         new String[]{ Manifest.permission.READ_EXTERNAL_STORAGE },
19                         REQUEST_CODE_ASK_PERMISSIONS);
20                 }
21             }, this);
22             return;
23         }
24         ActivityCompat.requestPermissions(this, new String[]{
25             Manifest.permission.READ_EXTERNAL_STORAGE },
26             REQUEST_CODE_ASK_PERMISSIONS);
27         return;
28     }
29     rm.ReadSongsIntoArrayList(context);
30 }
```

```

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.

8.1.4 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.

8.2 Fragments

8.2.1 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 austauschen. Da Fragments in mehreren Activities wiederverwendet werden können, müssen Ansichten wie Detailviews oder Listen nur einmal programmiert werden und können überall eingesetzt werden. Fragments werden ab Android 3.0 (API level 11) zur Verfügung gestellt. [Android Developers [10][11]]

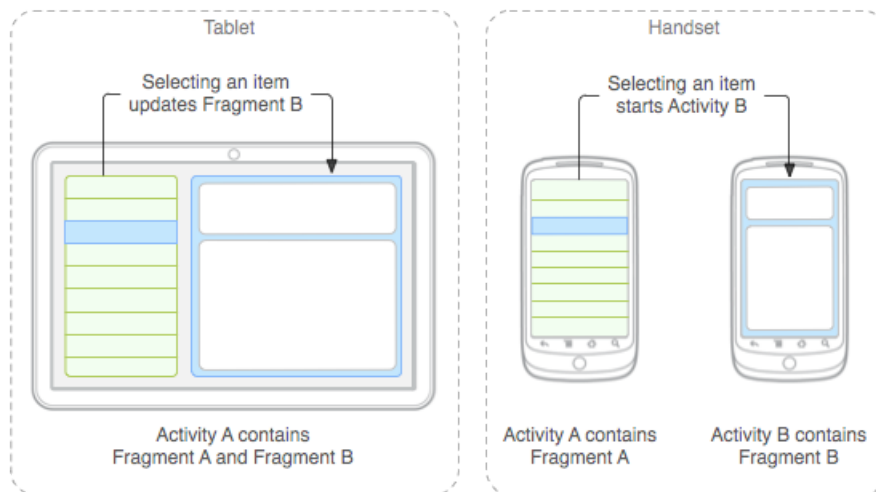


Abbildung 13: Zum Beispiel kann man mithilfe von Fragments Views erstellen, die sowohl auf einem Tablet als auch auf einem Handy ein optimales Benutzerinterface anbieten. [Android Developers [11]]

8.2.2 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.

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

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

8.2.3 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.

```
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.

8.2.4 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.

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

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

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

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

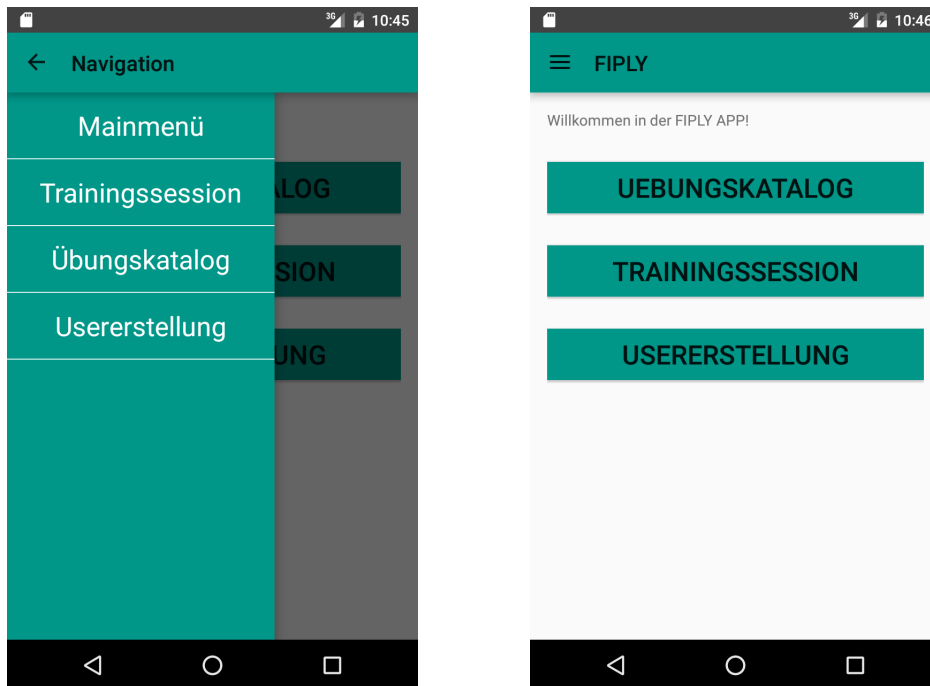


Abbildung 14: Bild des NavigationDrawers und des FMains

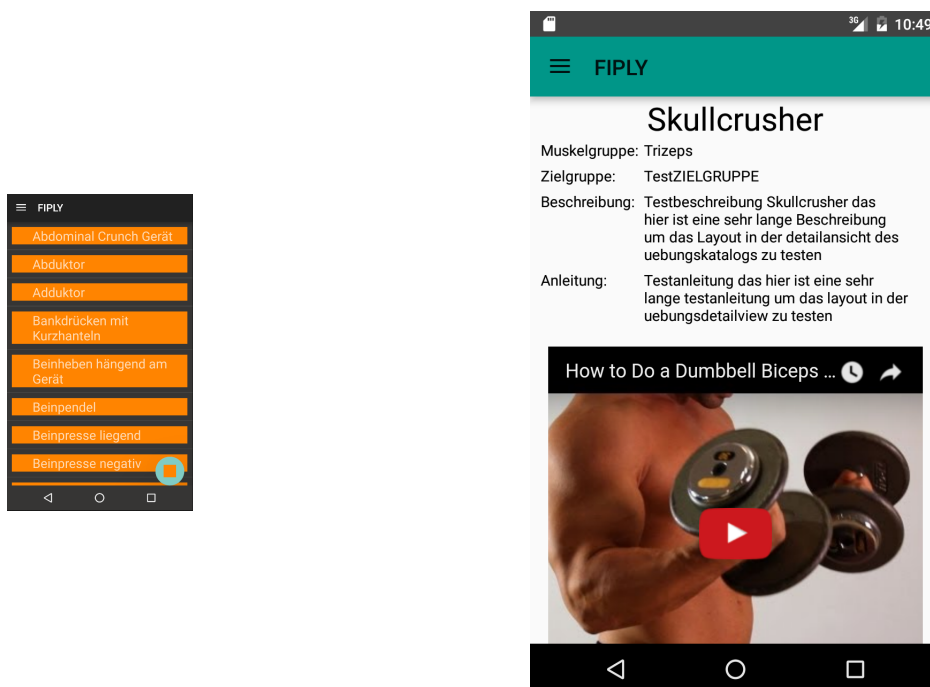


Abbildung 15: Bei Klicken eines Elements in der ListView wird die zugehörige DetailView aufgerufen

8.3 Bundles

TODO

8.4 NavigationDrawer

Der NavigationDrawer ist ein Menü, das die wichtigsten Navigationsoptionen am linken Bildschirmrand darstellt. 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 [12], Tamada [13], Android Developers [14]]

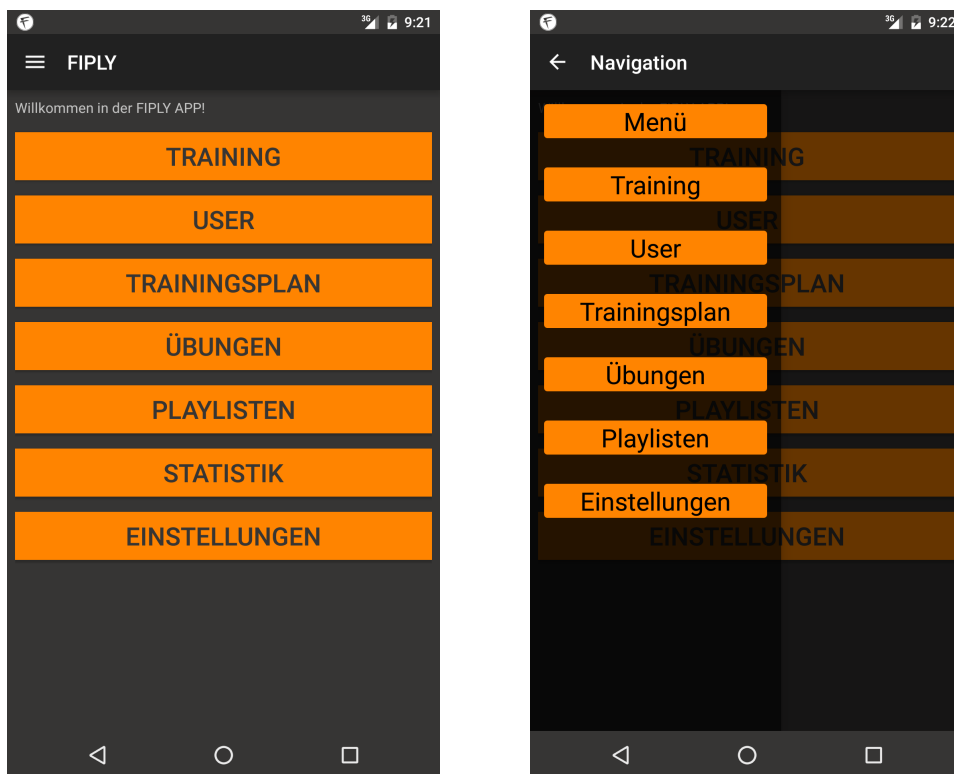


Abbildung 16: Links wird der NavigationDrawer versteckt. Rechts ist der NavigationDrawer geöffnet.

```

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 Views der App. In diesem DrawerLayout wird anschließend eine View angelegt, die dazu verwendet wird den Hauptinhalt der App darzustellen. Dabei ist zu Beachten, dass das 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.

```

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 noch etwas Code nötig. Dieser Code wird ebenfalls im onCreate der Activity, die einen NavigationDrawer erhalten soll, ausgeführt.

```

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);

```

9 Benutzerverwaltung

9.1 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.

9.2 Aufteilung der Verwaltung

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

9.2.1 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.

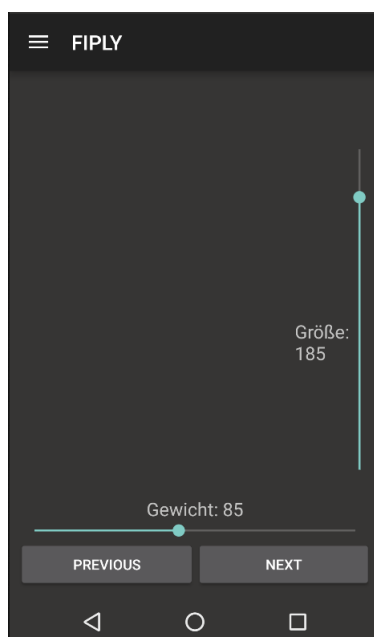


9.2.2 Schritt 2

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.

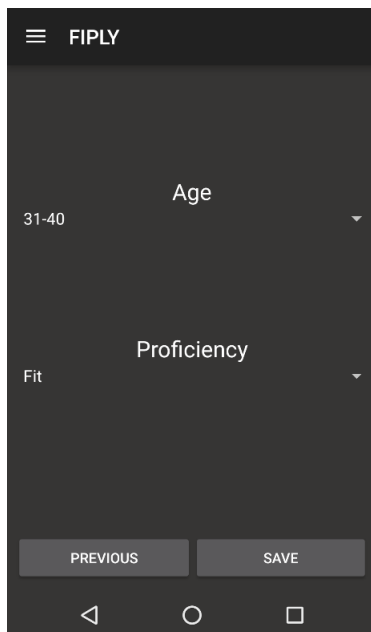


9.2.3 Schritt 3

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.



9.3 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.

10 Exportieren

10.1 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 9: 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.

10.1.1 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.

10.1.2 Anwendung

CSV-Datei lesen:

Listing 10: 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 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 11: 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 öffnet 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* [15]]

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 12: 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 13: 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* [16]]

10.2 Emails senden

10.2.1 Mittels Intents

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

Listing 14: 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* [17]]

10.2.2 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.

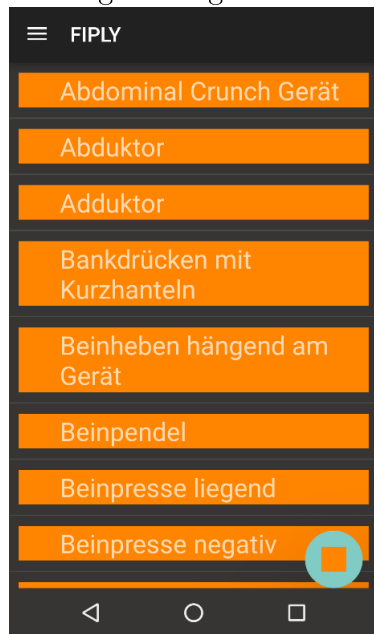
Ohne bereits installierten Client auf dem Android Gerät ist das senden von Emails über diese vorgehensweise nicht möglich. Ein gutes Beispiel dafür ist das Ausführen der Funktion auf einem Android-Emulator auf einem beliebigen Computer. Standardmäßig ist auf einem Android-Emulator kein Emailclient wie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

TODO: TrainingsplanMgt

11 Übungskatalog

11.1 Beschreibung

Der Übungskatalog beinhaltet eine Liste aller verfügbaren Übungen.



11.2 Implementierung

11.2.1 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.

11.2.2 List View

Die List View zeigt eine Liste aus Element an. Ein Klick auf ein bestimmtes Element löst eine Callback-Methode aus, von welcher die Detail View aufgerufen wird.

11.2.3 Einlesen der Übungen

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

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

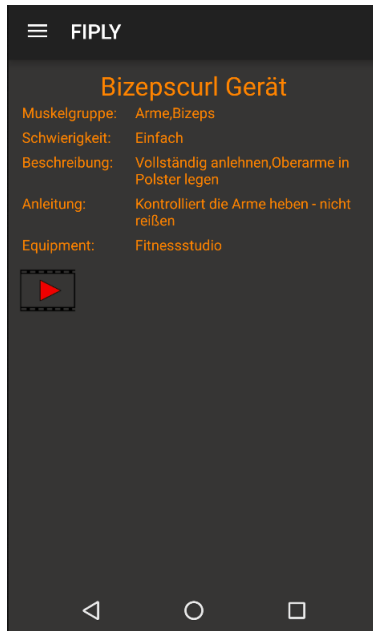
Wie in diesem Exampel erkennbar musste jedes Hochkomma mit einem Backslash escaped werden, da der JSON-String sonst nicht in der strings.xml abgelegt werden könnte.

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

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

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

11.3 DetailView



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.

11.4 Filter

11.4.1 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



11.4.2 Implementierung

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

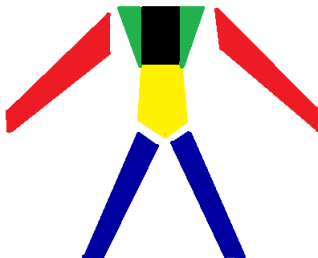
Die folgende Lösung wurde diesem Tutorial entnommen [*Android Images With Clickable Areas – Part 1* [18]]

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.



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



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

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

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

Die erste verwendete Hilfsmethode ermittelt ob zwei Farbwerte ähnlich oder gleich sind.

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

Die zweite verwendete Methode ermittelt den Farbwert des gedrückten Bereichs.

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

12 VideoView

12.1 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 auch vom Internet abgespielt werden.

12.2 Vorteile

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

12.3 Probleme

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

13 Youtube Android Player API

13.1 Beschreibung

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

13.2 Vorteile

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

13.3 Probleme

Für diese Methode wird leider ein Google Developer Key, den wir derzeit nicht besitzen, benötigt. Weiters muss eine externe App installiert sein damit diese Methode funktioniert.

14 WebView

14.1 Beschreibung

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

14.2 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.

14.3 Probleme

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

15 Nutzwertanalyse

15.1 KO-Kriterien

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

15.2 Erfüllung der KO-Kriterien

15.2.1 VideoView

Die Video View ist kostenfrei, jedoch nicht verwendbar aufgrund der beschriebenen Probleme.

15.2.2 Youtube Android Player API

Die Youtube Android Player API ist kostenfrei, jedoch nicht verwendbar aufgrund der beschriebenen Probleme.

15.2.3 WebView

Die WebView ist sowohl verwendbar als auch kostenfrei.

15.3 Schlussfolgerung

Da zwei von drei Alternativen durch die KO-Kriterien ausgeschlossen werden können, bleibt nur die WebView über. Deswegen werden wir für unsere App die WebView verwenden und das Video mit dem embeded-Link von Youtube einbinden und anpassen.

15.4 Musik

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

15.4.1 Lokalisierung der Musikdateien

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

```
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 Mediatore 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 Mediatore erfolgt über einen ContentResolver mit Hilfe der query()-Methode.

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

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

```
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.

```

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 }

```

15.4.2 Verwalten von Playlists

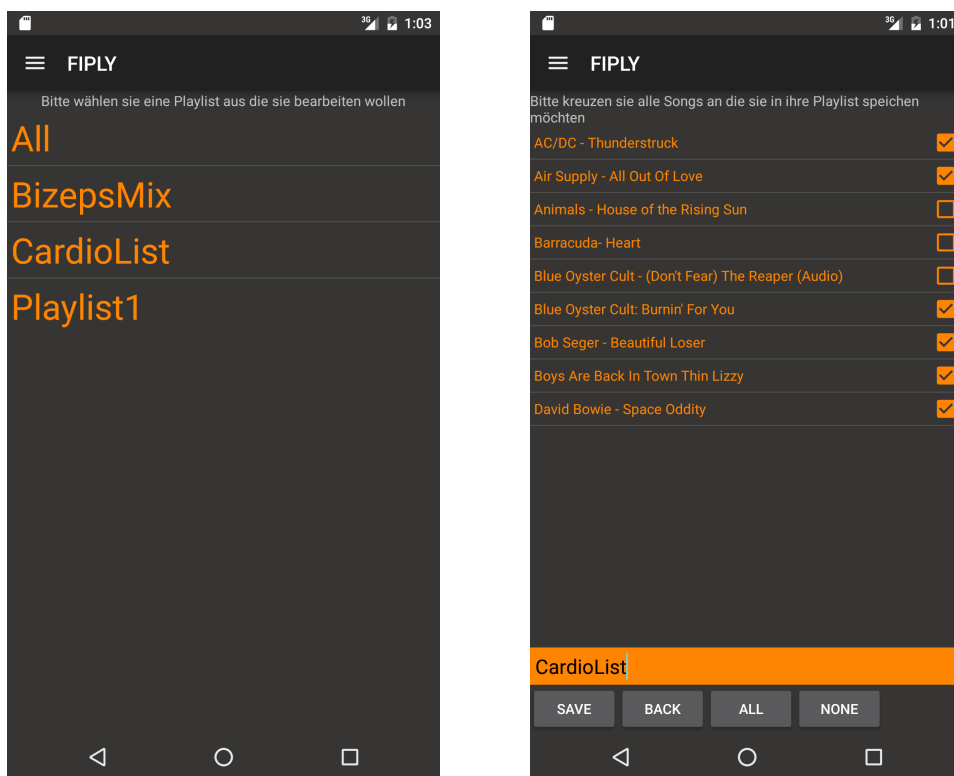


Abbildung 17: Bei Klicken eines Elements in Screenshot 1 werden alle Songs angezeigt (Screenshot 2). Mittels der Checkboxen werden jene Songs markiert, die sich in der ausgewählten Playlist befinden.

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.

```
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);
```

15.4.3 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.

```
1 public void changeSong(int songIndex, String playlist) {
2     aktPlaylist = playlist;
3     setPlaylist(psrep.getByPlaylistName(aktPlaylist));
4     setSongIndex(songIndex);
5     try {
6         mp.reset();
7         mp.setDataSource(getPlaylist().get(getSongIndex())
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 zurückliefert, konvertiert die millisecondsToHMS-Methode die Songdauer in einen String im hh:mm:ss Format (ISO 8601).

```
1 private Runnable mUpdateDurTask = new Runnable() {
2     @Override
3     public void run() {
4         long currentDur = mp.getCurrentPosition();
5         tvCurrentDur.setText(millisecondsToHMS(currentDur));
6         int progress = getProgressPercentage(currentDur, mp.
7             getDuration());
8         progressBar.setProgress(progress);
9         mHandler.postDelayed(this, 100);
10    }
11 };
```

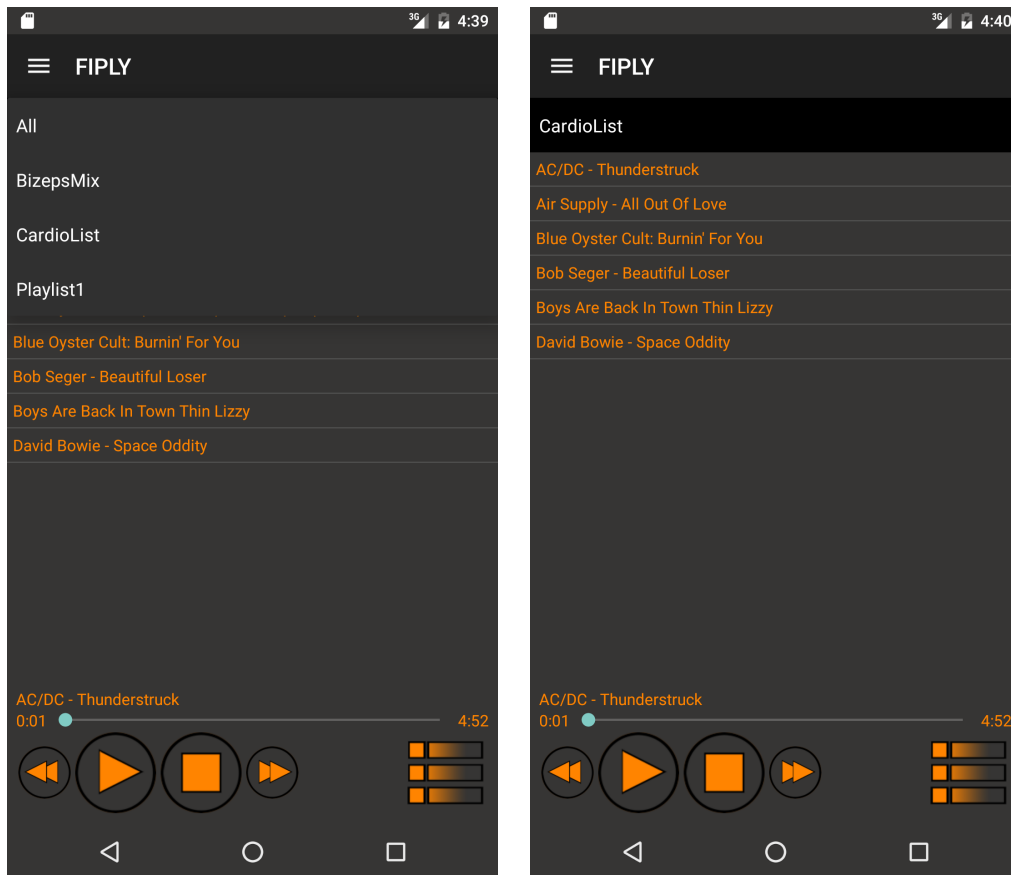


Abbildung 18: In einem Spinner kann eine Playlist ausgewählt werden. Bei Klick auf einen Song wird dieser abgespielt.

Während der Benutzer sich in einer Trainingsession 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.

15.4.4 MusicControls

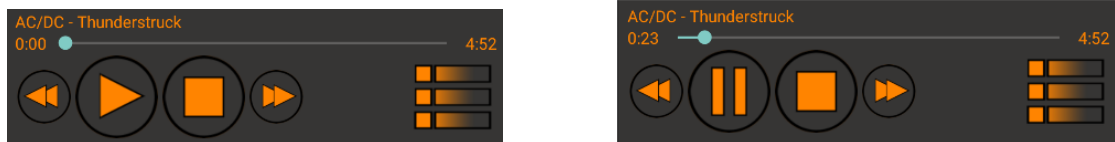


Abbildung 19: Die MusicControls in Ruhe und während einer Wiedergabe.

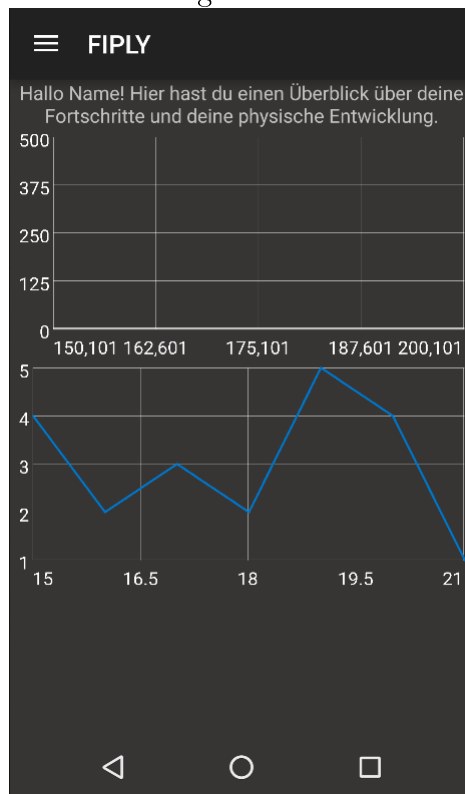
- Durch den Zurück und durch den Weiter Button kann auf den vorherigen beziehungsweise auf den nächsten Song gewechselt werden.
- Der Play Button dient dem Starten der Musikwiedergabe. Während der Musikwiedergabe erscheint an dieser Stelle der Pause Button mit dem man die Musik pausieren kann.
- Der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

16 Statistik

16.1 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.



16.2 Verfügbare Statistiken

16.2.1 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.

16.2.2 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

16.2.3 Insgesamt gestemmttes Gewicht

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

16.3 Implementierung

16.3.1 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 unserer SQLite Datenbank in ihren jeweiligen Repositories gespeichert.

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

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

16.3.2 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.

17 Datenbank

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

17.1 SQLite

SQLite ist eine open-source library, sie implementiert ein unabhängiges, serverlos, zeroconf und transaktionales SQL-database-engine [*SQLite*[21]]

17.2 Zugriff auf die Daten

Der Zugriff auf die abgespeicherten Daten erfolgt über 3 Schichten:

17.2.1 Physische Ebene

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

17.2.2 DB Helper

Der DB Helper stellt die Datenbank als Objekt zur Verfügung, welches in den Repositories instanziiert wird.

17.2.3 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 instanziiert werden.

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

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

```
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.

```
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
- Playlis-Songs-Repository
- Statistic-Repository
- Uebungen-Repository

17.3 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.

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

18 Social Media

18.1 Beschreibung

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

18.2 Verknüpfung mit Facebook

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

18.2.1 FacebookSDK

Mithilfe dieser SDK kann man den Login verwalten und danach im Namen des Benutzers, mit der Einwilligung 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.

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

18.2.2 Facebook Login Button

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

```
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" />
```

18.2.3 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.

18.3 Notifications

TODO: Notifications

18.4 Notifications

TODO

19 Design

TODO: Icons und Style

20 Kommerzialisierung

20.1 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 [22]]

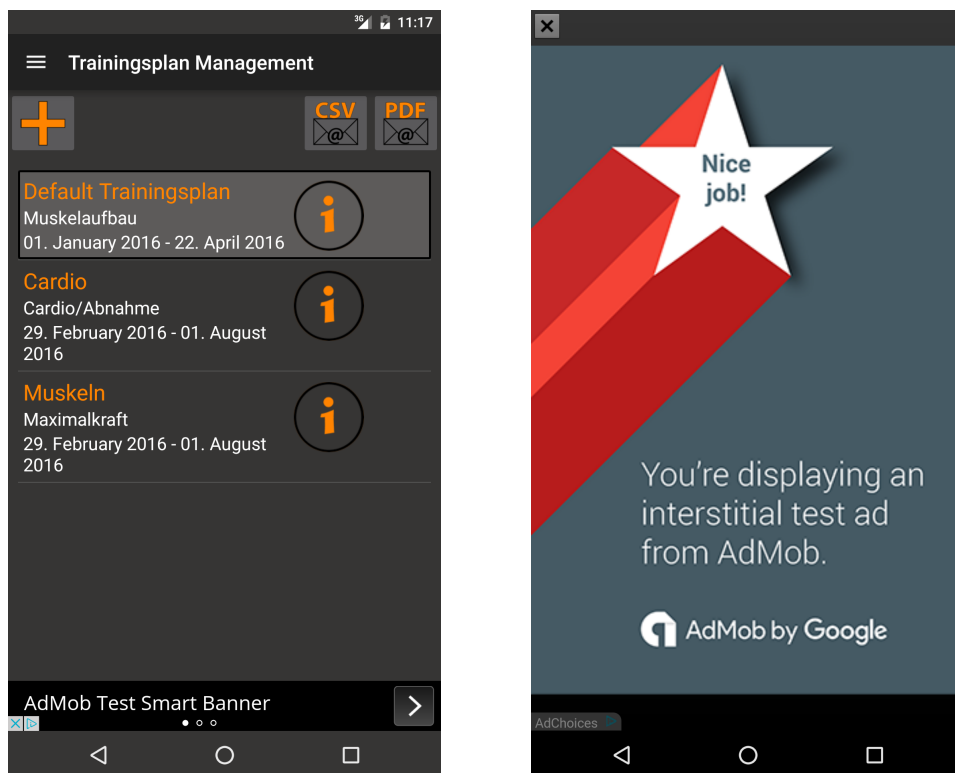


Abbildung 20: Links ist ein Beispiel für eine Banner Ad zu sehen. Rechts sieht man ein Beispiel für ein Interstitial.

20.1.1 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 kann durch einen Klick auf das Banner auf die beworbene Webseite weitergeleitet werden.

[Google Developers [23]]

```
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 Boden der App und überspannt die gesamte Breite des Bildschirms. Die oben eingetragene ad unit Id liefert uns TestAds und dient zum Testen.

```
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 dieser eingeleitet. Das Banner wird erst dann sichtbar, wenn die Werbung komplett geladen worden 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()`.

20.1.2 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 [24]]

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

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

```
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()`.

```
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 `AdListener` zu laden, um diese jederzeit auf Abruf anzeigen zu können.

20.2 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 und einen Suffix wie z.B.: Donation trägt.

20.2.1 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 [25]]

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 [25]]

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

20.2.2 PayPal

PayPal und deren Mobile Payment Libraries unterstützen die einfache Implementierung eines "Pay with Paypal"-Buttons über den Käufe mittels eines PayPal-Account durchgeführt werden können. Da wir unsere App aber in den Google Play Store stellen wollen stehen wir vor dem Problem, dass die Einbindung von Donations in Apps, die über den Play Store vertrieben werden, nur sehr vage in den Google Play-Programmrichtlinien für Entwickler 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 [26]]

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 [27]]

20.2.3 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.

20.2.4 Zweite kostenpflichtige App

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

20.3 Freemium

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

[Android Developers [28], Youtube [29]]

20.3.1 Consumable Items

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

20.3.2 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.

20.3.3 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.

20.4 Free/Paid Versions

20.4.1 Beschreibung

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

20.4.2 Vorteile

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

20.4.3 Nachteile

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

20.4.4 Implementierung

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

20.5 PaidVersion

20.5.1 Beschreibung

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

20.5.2 Vorteile

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

20.5.3 Nachteile

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

20.5.4 Preis

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

20.5.5 Google-Play-Store

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

20.6 SellApp

20.6.1 Beschreibung

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

20.6.2 Vorteile

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

20.6.3 Nachteile

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

20.6.4 Probleme/Schwierigkeiten

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

20.7 Promotion

Es gibt dutzende Mittel um eine Android Applikation zu vermarkten. Ob über das Internet oder durch klassische physische Varianten lässt sich am Besten über Zielgruppenorientierung bestimmen. [*Möglichkeiten der App Vermarktung: Teil 1* [30]]

20.7.1 Website

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

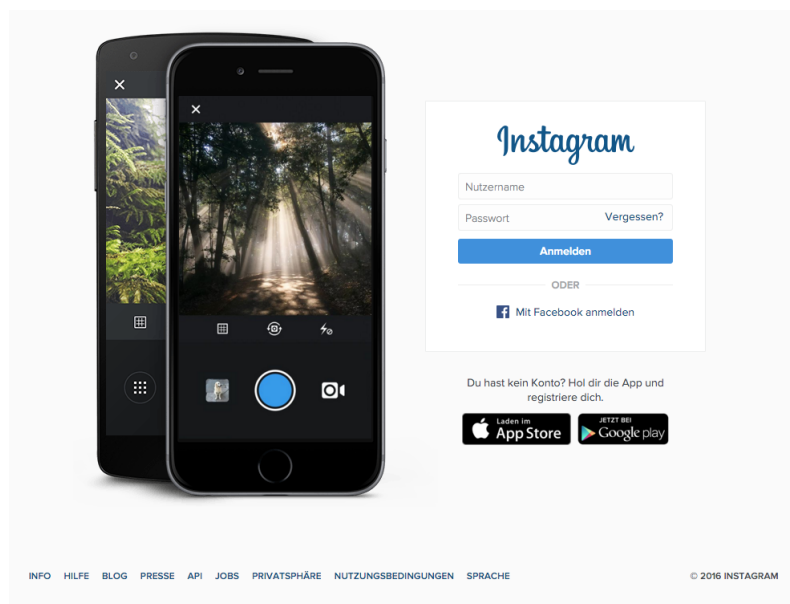


Abbildung 21: Screenshot von <http://instagram.com>:

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

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

20.7.2 Social Media Reputation

Ein bereits seit langem wichtiges Element in der online Vermarktung ist die Social Media Reputation. Wenn man heutzutage seine Applikation an die Menschen bringen will sollte das über die beliebte Sozialen Netzwerke wie Facebook, Google+, Twitter, Instagram, Vine,... und unzählige mehr. Seiten wie diese bieten die Möglichkeit für das Produkt einen eigene Seite oder Account zu erstellen, über diesen sich dann Benutzer unterhalten und austauschen können und neue an Land gezogen werden können. Zusätzlich können Administratoren beliebte Facebook Seiten beispielsweise dafür bezahlt werden, damit sie die App darauf teilen und positive Merkmale dabei unterstreichen.

Ein weiteres gutes Social Media mittel ist Reddit.com. Die Seite inkludiert die Funktion einen eigenen Developer Blog zu führen, worauf updates, bugs und neue Ideen für die Funktionalitäten der Applikationen geteilt werden können. Zusätzlich können bei dieser Webseite angemeldete User auch in diesem Blog posten - sich mit den Entwicklern unterhalten und mithilfe 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.

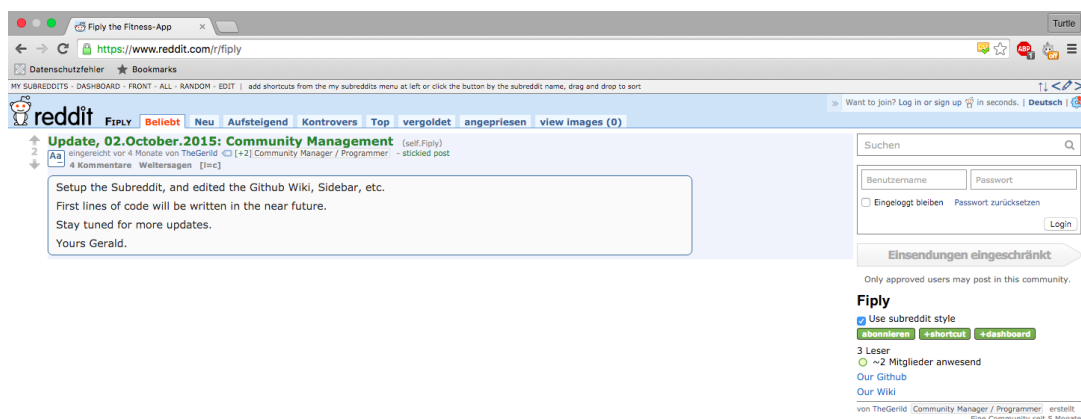


Abbildung 22: Screenshot von <http://reddit.com/r/fiply>

20.7.3 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.

20.7.4 Konteste

Kaum mit etwas anderem kann man schneller für Aufmerksamkeit sorgen als mit dem Gewinnen von Kontesten. Auch schon nur bei einer Einreichungen und Nominierung kann man einen hohen Bekanntheitsgrad für seine App erlangen. Ein positiver Effekt ist auch, dass Gewinner und oft Zweit- und Drittplatzierte sich durch so eine Veranstaltung hin und wieder einen kleinen aber wichtigen Artikel in Zeitungen oder Onlinemagazinen, aber auch Technikblogs u.A. teilen. Konteste sind ein wichtiges Element um das Produkt an die Zielgruppen zu bringen. Am besten sieht man nach welche Art von Kategorien es bei Preisausschreiben 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.

20.7.5 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 Erfolgsvorhersung einer selbst erstellten Applikation. Die Verbreitung an Bekannte oder Verwandte kann durch persönliche Gespräche, Emails oder Social Media erfolgen. Am Besten bewährt sich dabei jedoch die persönliche Vermittlung. Falls das Endprodukt wirklich seinen Funktionsumfang gewährleistet und die Qualität dementsprechend gut ist, werden diese Vermittlungspfade auch ihren Bekanntheitskreis vorstellen. Dies geht weiter weiter.

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

20.7.6 Reklame

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

20.7.7 In Appstore Optimierung (IAO)

Genau wie bei einer Suchmaschinenoptimierung für Webseiten bei Diensten wie Google.com gibt es dies auch für Applikation im Google Playstore. Ziel ist es, dass der potentielle Benutzer durch verschiedene, möglichst wenige, bestimmte Stichwörter in der Suchanfrage auf die Applikation 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 eigentständigen und nicht andersweitig vorkommenden Namen besitzen, damit die Benutzer den Namen eindeutig mit dem Produkt assoziieren. Eine klare, übersichtliche und kurzgehaltene Beschreibung bringt den User eher dazu, die Applikation runterzuladen, womit die kommerzielle Erfolgswahrscheinlichkeit des Produkts am Besten sichergestellt werden kann.

Das Icon sollte möglichst minimalistisch gehalten werden. Der Benutzer assoziiert Farbschemas und Formen des Icons automatisch mit der Applikation, welche möglichst simple und mit der modernen Designrichtlinie "Weniger ist 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 sollte vermieden werden, da es ein Limit für eingebundenen Medien für das Appprofil gibt und diese möglichst effizient

genutzt werden sollten. [*Möglichkeiten der App Vermarktung: Teil 2* [31]]

20.7.8 Fazit

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

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] *GitHub*. März 2016. URL: <https://github.com/>.
- [4] *DataBinding Definition*. März 2016. URL: https://www.it-visions.de/glossar/alle/6875/Data_Binding.aspx.
- [5] *Droidcon NYC 2015 - Data Binding Techniques*. Sep. 2015. URL: <https://www.youtube.com/watch?v=WdUbXWztKNY>.
- [6] *Data Binding Guide*. März 2016. URL: <http://developer.android.com/tools/data-binding/guide.html>.
- [7] 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>.
- [8] *Requesting Permissions at Run Time*. März 2016. URL: <http://developer.android.com/training/permissions/requesting.html>.
- [9] *System Permissions*. März 2016. URL: <http://developer.android.com/guide/topics/security/permissions.html>.
- [10] *Fragment*. Feb. 2016. URL: <http://developer.android.com/reference/android/app/Fragment.html>.
- [11] *Fragments Guide*. Feb. 2016. URL: <http://developer.android.com/guide/components/fragments.html>.
- [12] Ben Jakuben. *How to Add a Navigation Drawer in Android*. Feb. 2016. URL: <http://blog.teamtreehouse.com/add-navigation-drawer-android>.
- [13] Ravi Tamada. *Android Sliding Menu using Navigation Drawer*. März 2016. URL: <http://www.androidhive.info/2013/11/android-sliding-menu-using-navigation-drawer/>.
- [14] *Creating a Navigation Drawer*. Feb. 2016. URL: <http://developer.android.com/training/implementing-navigation/nav-drawer.html>.
- [15] *Vgl.: OpenCSV Website*. März 2016. URL: <http://opencsv.sourceforge.net/>.

- [16] *Vgl.: OpenCSV Writer*. März 2016. URL: <http://viralpatel.net/blogs/java-read-write-csv-file/>.
- [17] *Mails versenden in Android*. März 2016. URL: <http://stackoverflow.com/questions/5401104/android-exporting-to-csv-and-sending-as-email-attachment>.
- [18] Bill Lahti. *Android Images With Clickable Areas – Part 1*. März 2016. URL: <https://blahti.wordpress.com/2012/06/26/images-with-clickable-areas>.
- [19] *MediaPlayer*. Feb. 2016. URL: <http://developer.android.com/reference/android/media/MediaPlayer.html>.
- [20] Ravi Tamada. *Android Building Audio Player Tutorial*. Feb. 2016. URL: <http://www.androidhive.info/2012/03/android-building-audio-player-tutorial/>.
- [21] *SQLite*. März 2016. URL: <https://www.sqlite.org>.
- [22] *Adding AdMob into an Existing App*. Feb. 2016. URL: <https://developers.google.com/admob/android/existing-app>.
- [23] *Banner Ads*. Feb. 2016. URL: <https://developers.google.com/admob/android/banner>.
- [24] *Interstitial Ads*. Feb. 2016. URL: <https://developers.google.com/admob/android/interstitial>.
- [25] *About Flattr*. Feb. 2016. URL: <https://flattr.com/about>.
- [26] *Google Play-Programmrichtlinien*. Feb. 2016. URL: https://play.google.com/intl/ALL_de/about/developer-content-policy.html.
- [27] *Google Play In-app Billing*. Feb. 2016. URL: <https://support.google.com/googleplay/android-developer/answer/6151557>.
- [28] *In-App-Billing*. Feb. 2016. URL: https://developer.android.com/google/play/billing/billing_admin.html.
- [29] Jarek Wilkiewicz. *Implementing Freemium*. Feb. 2016. URL: <https://www.youtube.com/watch?v=UvCl5Xx7Z5o>.
- [30] *Möglichkeiten der App Vermarktung: Teil 1*. Apr. 2016. URL: <http://www.app-entwicklung.info/2015/04/moeglichkeiten-der-app-vermarktung-teil-1-website-social-media-presse-events-co/>.
- [31] *Möglichkeiten der App Vermarktung: Teil 2*. Mai 2015. URL: <http://www.app-entwicklung.info/2015/05/moeglichkeiten-der-app-vermarktung-teil-2-die-app-store-optimierung-aso/>.