ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung einer konfigurierbaren Crowd-Sensing Anwendung für klinische Studien am Beispiel von Android

Masterarbeit an der Universität Ulm

Vorgelegt von:
Patrick Grau
patrick.grau@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Rüdiger Pryss

Betreuer:
Marc Schickler

2016

Fassung 18. August 2016

© 2016 Patrick Grau

This work is licensed under the Creative Commons. Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/de/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Smartphones sind aus unserer heutigen Zeit nicht mehr wegzudenken. Neben mobilen Anwendungen und Spielen, ist es ebenfalls möglich klinische, wie auch wissenschaftliche Studien auf Smartphones durchzuführen [1], [2]. Mit der Diplomarbeit von Herrn Jochen Herrmann [3] wurde 2012 dabei mit *Track Your Tinnitus* der Grundstein dieser nun ausgebauten Variante gelegt. *Track Your Tinnitus* ist dazu gedacht, Schwankungen des Tinnitus auf dem Smartphones zu überwachen, die Ergebnisse dem Benutzer visualisiert darzustellen, sowie diese Ergebnisse an den Webserver von Track Your Tinnitus zurück zu senden, um diese für Forschungszwecke auszuwerten [1]. Um den Entwicklungsaufwand zu veringern, wurden *Drittbibliotheken* eingesetzt, die unter anderem für das Design wie auch für die Datenübermittlung verantworltich sind.

Im ersten Teil dieser Arbeit wurde die *Website* von Track Your Tinnitus um einen *Studienbereich* erweitert, in dem es möglich ist klinische Studien zu erstellen. Studienleiter können Register Fragebögen entwerfen, die einmalig zu Beginn vom Studienteilnehmer auszufüllen sind. Desweiteren werden auf der Website die sich zu wiederholenden Fragen zur Studie hinterlegt. Der Studienleiter hat ebenfalls die Möglichkeit, die Benachrichtungen selbst zu konfigurieren. Anhand dieser Benachrichtigungen wird der Studienteilnehmer über die App darauf hingewießen, einen Fragebogen erneut auszufüllen.
Im zweiten Teil wurde nun die Android Variante dieser App erweitert, dass die zuvor erstellten klinischen Studien durchgeführt werden können. Um an einer Studie teilzunehmen, erhalten die Probanten einen Studiencode, den der Studienleiter beim Erstellen der Studie generiert hat. Hierbei kommuniziert die Applikation mittels einer so genannten REST-Schnittstelle mit dem Server.
Ein weiterer wichtiger Punkt dieser Arbeit besteht darin, die Applikation auf Googles neuem *Material Design* umzugestalten. Im letzten Teil der Implementierungsphase werden die bisher eingesetzten Drittbibliotheken analysiert und auf ihren Nutzen hin bewertet. Einige dieser Bibliotheken können durch die Einführung des Material Designs abgelöst werden, da diese nun in Googles Android API vorhanden sind.

[1] Ein Resultat dieser Auswertung ist in [4] beschrieben

Danksagung

Mein Dank geht in erster Linie an sämtliche Personen, die mich bei der Erstellung dieser Arbeit unterstützt haben. Vor allem aber möchte ich mich bei Herrn Marc Schickler und Dr. Rüdiger Pryss bedanken, die mir diese Arbeit zur Verfügung gestellt und mich ständig dabei unterstützt haben. Zudem möchte ich mich bei und dem gesamten Track Your Tinnitus Team um Herrn Dr. Winfried Schlee bedanken, die mir eine hervorragende Basis für diese Weiterentwicklung zur Verfügung gestellt haben. Insbesondere gilt mein Dank Herrn Jochen Hermann, der den momentanten Stand dieser Track Your Tinnitus Plattform implementiert hat.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Auch in der heutigen Zeit, in der mehr als die Hälfte aller Deutschen ein Smartphone nutzt [19], werden klinische Studien größtenteils auf Papier durchgeführt. Diese Methode ist meist mit einem sehr hohen Zeitaufwand und Kosten verbunden. Zuerst werden den Studienteilnehmern die Fragebögen ausgehändigt. Die ausgefüllten Fragebögen jedes einzelnen Studienteilnehmers müssen danach von einem Mitarbeiter überprüft und die Antworten ausgewertet werden. Diese Antworten müssen danach aus rechtlichen Gründen sicher und über einen längeren Zeitraum hinweg archiviert werden. Je nach Anzahl der Studien und Teilnehmer der Studie, kann dies hohe Verwaltungs- und Lagerkosten verursachen. Neben des nicht zu unterschätzendem Zeit- und Kostenaufwandes muss auch insbesonders über den Ressourcenverbrauch nachgedacht werden. Eine Möglichkeit diese Probleme zu umgehen ist der Einsatz von mobilen Endgeräten, bzw. Smartphones.
Einen ersten Ansatz lieferte Jochen Herrman mit der Entwicklung von *Track Your Tinnitus* zum Zwecke der Tinnitusüberwachung. Mit dieser Plattform ist es möglich, dass Personen die an Tinnitus leiden, ihre tagesabhängigen Ausprägungen des Tinnitus aufzeichnen können. Jedoch ist man im momentanten Stand der App auf diese Schwankungsaufzeichnung, sowie Übertragung der Ergebnisse beschränkt.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, die bereits bestehende Plattform von Track Your Tinnitus zu erweitern. Es soll damit möglich sein, klinische Studien durchführen zu können. Der Server, auf dem die Website von Track Your Tinnitus betrieben wird, dient dabei als zentraler Speicherort der anfallenden Daten zu den Studien. Die eigentliche Durchführung dieser Studie soll zunächst auf Android Smartphones stattfinden. Es soll möglich sein, dass Studienleiter sich ihre klinische Studie selbst konfigurieren können. Dazu zählen das Erstellen der Studie, wie auch die beinhalteten Register Fragebögen und die statischen Fragen zur klinischen Studie. Zumdem soll eine Möglichkeit geschaffen werden, in dem der Studienleiter die Benachrichtigungszeitpunkte der App an den User selber bestimmen kann. Im zweiten Teil der Arbeit wird die Android Applikation erweitert. Track Your Tinnitus benutzt momentan zahlreiche Drittbibliotheken die dank der Einführung von Googles neuen "Material Designäbgelöst werden können. Weitere Designaspekte dieser *Sprache* sollen in die Applikation von Track Your Tinnitus eingearbeitet werden. Im letzten Schritt wird die Android Applikation erweitert, damit klinische Studien von den Teilnehmern durchgeführt werden können. Ausschlaggeben dabei ist, dass der Studienteilnehmer nur an einer klinischen Studie teilnehmen kann, in die er sich mit einem Studiencode eingeschrieben hat.

1.3 Struktur der Arbeit

Diese Arbeit untergliedert sich in acht Kapitel. Angefangen wurde mit einer kurzen aber prägnanten Einführung und Zielsetzung in das Thema der vorliegenden Masterarbeit. In Kapitel 2 möchte ich auf einige Grundlagen eingehen. Diese dienen dazu, ein grundlegendes Verständis über das Vorgehen während der Implementierung der Arbeit zu schaffen. Zudem werden die eingesetzten Technologien beschrieben. Kapitel 3 beschäftigt sich mit verwandten Arbeiten zum Thema dieser Arbeit. In Kapitel 4 wird der momentane Stand der Track Your Tinnitus Plattform und die Architektur der Systeme näher beschrieben. Kapitel 5 beschäftigt sich mit den funktionalen sowie den nicht-funktionalen Anforderungen, die an das erweiterte Framework gestellt worden

Abbildung 1.1: Gliederung der Masterarbeit

sind. Kapitel 6 stellt den Hauptteil dieser Masterarbeit dar. Hier wird auf die Ergebnisse eingegangen, sowie wichtige Codefragmente beschrieben. Im darauffolgenden Kapitel werden die gestellten Anforderungen aus Kapitel 5 mit den Ergebnissen der Arbeit abgeglichen. Das letzte Kapitel befasst sich mit einer Zusammenfassung, sowie einem Ausblick auf mögliche Erweiterungen des Projekts.

2

Grundlagen

In diesem Kapitel werden Grundlagen behandelt, die für das weitere Verständis der Arbeit unerlässlich sind. Darunter zählt das *Laravel* PHP Framework, mit dem das Backend der Track Your Tinnitus Plattform entwickelt wurde. Desweiteren werden die beiden Begriffe *REST* und *JSON* im Kontext dieser Arbeit näher betrachtet. Track Your Tinnitus setzt *Drittbibliotheken* ein, die im Laufe dieser Arbeit entfernt werden sollen. An dieser Stelle folgt eine kurze Beschreibung dieser Libraries und für was diese verantwortlich sind. Zum Schluss wird noch der *Lebenszyklus* einer Android Activity, der Einsatzzweck von *Sharedpreferences* und der Grundgedanke des „*Material Designs*“ erläutert.

2.1 Laravel

Laravel [21] ist ein Opensource - PHP Web Framework, das dem Model-View-Controller Pattern folgt. Das von Taylor Otwell entwickelte Framework wurde im Juni 2011 in der Version 1 veröffentlicht. 2015 hat es sich, laut einer Umfrage der Seite Sitepoint [24], zum beliebtesten Framework etabliert.
Besonders macht dieses Framework der Einsatz von *Eloquent ORM, Migrations, Blade Templates* und dem Packageverwaltungstool *Composer*. Eloquent ORM ermöglicht es, PHP Klassen, bzw. Objekte auf relationale Datenbanken wie MySQL abzubilden. Für jede Tabelle in einer Datenbank steht somit ein Modell zur Verfügung mit dem man kommunizieren kann. Auch Beziehungen unter diesen Modellen sind abbildbar. Die gängisten sind dabei:

- One To One

- One To Many
- Many to Many

Um solche Beziehungen zu definieren muss eine Methode in der Modelklasse erstellt werden, die eine der folgenden Methoden zurückliefert: *has_one(), has_many()* oder *belongs_to()*. Weitere Vorteile bei der Anwendung von Laravel im Vergleich zu anderen Frameworks sind [5]:

- *Bundels*: Mit Bundels lassen sich weitere Bibliotheken in Laravel integrieren.
- *Applications Logik*: Die Logik der Anwendung kann entweder in den dafür vorhergesehenen Controllern geschehen, oder direkt während der Deklaration der Routen in der routes.php Datei.
- *Reverse Routing*: Hiermit lassen sich Links zu den benannten Routen generieren. Sollte sich an diesen Routen im Verlauf der Entwicklung etwas ändern, setzt Laravel automatisch den richtigen URI für diesen Link.
- *Restful Controllers*: In diesen Controllern lassen sich die GET und POST Request trennen. Die Funktionen erhalten einen Prefix mit dem jeweiligen Request -> get_login() oder post_login().
- *Migrations*: Dienen als Versionierungenverwaltung für Datenbankschemen.

2.2 REST

REST steht für *Representational State Transfer* und beschreibt eine simple und zustandslose Architektur im World Wide Web. Hierbei wird jedes einzelne Objekt als Ressource angenommen. Ansprechbar ist diese Ressource über den *Uniform Resource Identifier* (URI):

```
1 https://www.abc.de/user/1
```

Quelltext 2.1: Beispiel - URI

Zur Übertragung von Daten wird in den meisten Fällen das Protokoll HTTP beziehungsweiße HTTPS genutzt. Ist der Einsatz von HTTP(S) gegeben, stehen folgende Methoden zur Verfügung um Ressourcen vom Webserver anzufragen oder auch zu ändern:

Methode	**Auswirkung**
GET	Fordert eine Ressource vom Webserver an. Dabei wird der Zustand dieser Ressource auf dem Webserver nicht verändert.
POST	Fügt eine neue Ressource hinzu.
DELETE	Löscht die gewünschte Ressource
PUT	Eine neue Ressource wird hinzugefügt. Sollte die Ressource bereits bestehen, wird sie geändert.
PATCH	Eine Ressource (bzw. ein Teil dieser) wird geändert.

Ein möglicher GET - Request wie auch Response werden damit wie folgt dargestellt:

```
1 GET /possible/request/uri HTTP/1.1
2 Accept: application/json
3 Authorization: Basic dGVzdHBkdfjAcxvDRjb25g
4 Host: localhost
5 Connection: keep-alive
```

Quelltext 2.2: HTTP - GET Request

```
1 HTTP/1.1 200 OK
2 Server libnhttpd
3 Connection: Keep-Alive:
4 Content-Type: application/rdf+xml
5 Content-Length: xxx
6 [Ressource]
```

Quelltext 2.3: HTTP - GET Response

2.3 JSON

JSON steht für *JavaScript Object Notation* und dient als Datenformat zum Zweck des Datenaustausches zwischen Webanwendugen. Ein Objekt ist durch eine öffnende und eine schließende geschweifte Klammer gekennzeichnet, sowie einem String und einem

Wert, die durch einen Doppelpunkt getrennt sind. Mehrere Paare in einem Objekt werden durch Kommata getrennt, wie in Abbildung 2.1 zu sehen [20]: Ein komplettes JSON

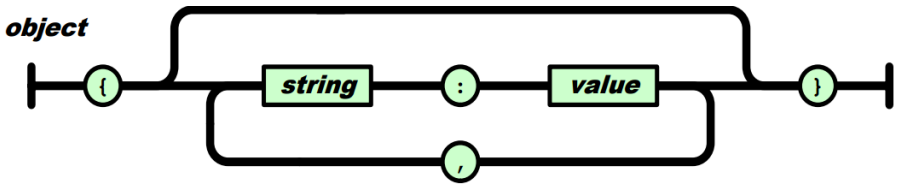

Abbildung 2.1: Schematische Darstellung eines JSON Objekts

Objekt kann beispielsweiße folgendermaßen aufgebaut sein:

```
{
   "Firma": "WunschAG",
   "TelefonNummer": "1234-5678-9012-3456",
   "Stadt": "Ulm",
   "Plz": "89073",
   "Inhaber":
   {
      "Name": "Karl",
      "Vorname": "Heinz",
      "Geburtstag": "31.08.1947",
   }
}
```

2.4 Eingesetzte Drittbibliotheken

Insgesamt fünf Drittbibliotheken sind im momentanen Stand der Track Your Tinnitus Applikation im Einsatz:

1. *ActionBarSherlock* [6] ist eine Opensource Bibliothek und erlaubt die allgemeine Nutzung des Action Bar Design Patterns über eine einzige API Schnittstelle. Die eingeführte Action Bar aus Android 4.0 ist damit auch auf den Android Smartphones mit einer Version kleiner als 4 bis hin zu Version 2 zu nutzen.

2. *SlidingMenu* [14] ist eine Opensource Bibliothek die erlaubt Applikationen mit einem Slidingmenü auszustatten. Dieses Slidingmenü dient zur Navigation in der App um zwischen Ansichten zu wechseln.

3. *SegmentedControl* [15] sind verschiedene Button-, oder Tabelemente, die der Anwendung im Vergleich zu einem herkömmlichen Radiobutton ein besseres Aussehen verleihen.
4. *Android Switch Backport* [16] ist eine Opensource Bibliothek die den Switch-Button (On/Off), der in Android Version 4 veröffentlich wurde, auch unter Android 2.1 zur Verfügung stellt.
5. *Android Async HTTP* [17] ist ein asynchroner HTTP-Client der auf den Apache HTTPClient Bibliotheken aufbaut. Dieser Client ermöglicht es HTTP Request zu senden und Responses des Server zu empfangen.

2.5 Androids Activity Lebenszyklus, Shared Preferences & Material Design

Eine *Activity* in Android ist nichts anderes als die Darstellung einer Bildschirmseite in einer App. Eine solche Activity verfügt über Methoden und Zustände, die in den unterschiedlichen Phasen eines Lifecycles aufgerufen werden können. Abbildung 2.2 zeigt diesen Lebenszyklus im Detail.

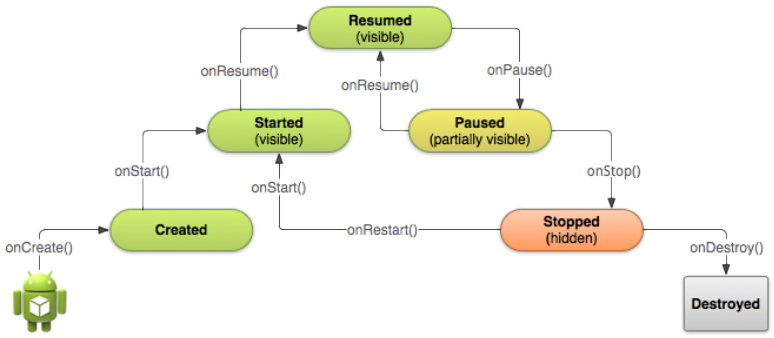

Abbildung 2.2: Lifecycle einer Activity übernommen aus [18]

Sharedpreferences sind eine einfache Möglichkeit in Android, so genannte *Key-Value Paare* zu speichern. Ein Objekt der Klasse Sharedpreferences zeigt auf eine Datei im Filesystem und ermöglicht durch vorgefertigte Methoden in dieser Datei zu lesen oder zu

schreiben. Sharedpreferences können dabei im private-, oder shared Modus vorliegen. Im *private* Modus erlaubt nur die eigene Applikation diese Datei zu lesen. Der *shared* Modus ermöglicht sämtlichen Applikationen das lesen und editieren dieser Datei. Listing 2.4 zeigt einen beispielhaften Aufbau einer Sharedpreference Datei.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">SharedPreferenceDemo</string>
    <string name="name">Hans</string>
    <string name="alter">44</string>
    <string name="wohnort">Ulm</string>
</resources>
```

Quelltext 2.4: Beispielhafte Sharedpreference

Um einen Wert in den Sharedpreferences zu schreiben wird die Sharedpreferences.Editor Klasse verwendet:

```
SharedPreferences settings = context.getSharedPreferences("‘
    PREFS_NAME"’, Context.MODE_PRIVATE);
Editor editor = settings.edit();
editor.putString("key", "value");
editor.commit();
```

Quelltext 2.5: Schreiben in Sharedpreference Datei

Um einen Wert aus der Sharedpreference Datei zu lesen wird die jeweilige Methode (getInt, getString, getBoolean, ...) der Sharedpreference Klasse genutzt:

```
Sharedpreferences settings = context.getSharedPreferences("PREFS_NAME
    ", Context.MODE_PRIVATE);
String text = settings.getString("key", "default");
```

Quelltext 2.6: Lesen aus der Sharedpreference Datei

Google selbst sagt über sein Material Design:
"We challenged ourselves to create a visual language for our users that synthesizes the classic principles of good design with the innovation and possibility of technology and science. This is material design. This spec is a living document that will be updated as we continue to develop the tenets and specifics of material design." [22]
Google meint somit selbst, es nicht lediglich als Design zu sehen. Es stellt eher eine ganze Designsprache dar. Das Ziel ist es ein einheitliches Design auch innerhalb mehrerer unterschiedlicher Produkte zu kreieren.

Desweiteren zählt zu den Besonderheiten von "Material Design", dass ein *Flat-Design* in den Apps dargestellt werden soll. Da dieses Flat-Design nicht nur x- und y-Koordinaten besitzt, sondern auch eine z-Koordinate, ist hierbei ein kleiner 3D-Effekt wahrzunehmen. Somit wirken die einzelnen Elemente wie Papierblätter, die übereinander gestapelt werden. Google hat unter [22] einen Styleguide zur Verfügung gestellt, der die einzelnen Elemente beschreibt und wie diese eingesetzt werden sollen. Die Theme Material Design ist ab der Version 5 (aka Lollipop) verfügbar. Wenn Version 4 mit dem Material Design ausgestattet werden soll, ist der Einsatz einer Bibliothek namens AppCompat v21 von Google [7] nötig. Die Kapitel 6.2.5 & 6.2.7 gehen dabei näher auf die Gestaltung der Applikation ein. Hier werden die ersten sichtbaren Elemente des Material Designs vorgestellt.

3

Related Work

In diesem Kapitel werden verwandte Arbeiten untersucht, die sich mit mobiler Datenerhebung beschäftigen. Dabei steht der Funktionsumfang dieser Arbeiten im Vordergrund.

3.1 QuestionSys

Das Projekt *QuestionSys* ([9], [36]) der Universitäten Ulm und Konstanz hat es sich zum Ziel gemacht, papierbasierte Fragebögen vollständig zu digitalisieren. Dazu zählen das Erfassen, Verteilen, die Datenerhebung, Evaluierung, Analyse sowie Archivierung und Versionierung dieser Fragebögen. Um dies zu realisieren, basiert *QuestionSys* auf drei wesentliche Komponenten. Der *Konfigurator* dient zum Erstellen und Evaluieren des Fragebogens mit den enthaltenen Fragen. Der *Server* speichert die exportierten Fragenmodelle des *Konfigurators*, verteilt diese Fragebögen an *Clients* und speichert die erhobenen Antworten des Nutzers. Der Client ist für die Darstellung der generischen Fragebögen verantwortlich. Dies kann ein Browser sein, aber auch Smartphones und Tables sind als Client möglich.

3.2 meinDiabetes

meinDiabetes [11] richtet sich in erster Linie an Menschen, die an Diabetes Typ-1 oder Typ-2 leiden. Die Smartphone App des Kircheim-Verlages bietet dabei mehrere Funktionalitäten. Dazu zählen ein *Tagebuch* zur Speicherung von Blutzucker- und Insulinwerten, Mahlzeiten, Sport und Notizen. Zudem enthält diese Anwendung eine

Lebensmittel-Datenbank, einen *Diabetes-Pass* um Untersuchungstermine einzutragen, ein Kohlenhydrate-*Schätzspiel*, sowie eine *Kliniksuche*, um Kliniken in der Umgebung zu finden, die auf Behandlung von Diabetes spezialisiert sind.

3.3 mykind

Das Kooperationsprojekt der Universitäten Konstanz und Ulm *mykind* [10] richtet sich an werdende Mütter. Mit dieser App ist es den Schwangeren möglich ihre Belastungen, mittels eines vorgegebenen Fragebogen während der Schwangerschaft, zu dokumentieren. Die Ergebnisse lassen sich entweder in der App oder auf der Website von *mykind* einsehen. Zudem erhalten die Teilnehmerinnen, basierend auf ihren individuellen Antworten, Tipps, um Risiken für sich und Ihr Kind vorzubeugen.

3.4 mPower

Mobile Parkinson Observatory for Worldwide, Evidence-based Research (mPower) [12] ist eine klinische Studie der Organisation Sage Bionetworks und der University of Rochester, New York. Diese Studie zielt darauf ab, die Parkinsonerkrankung, insbesondere die einzelnen Symptome, besser zu verstehen. An dieser Studie können jedoch bisher nur volljährige Probanten in den Vereinigten Staaten mit einem iPhone teilnehmen. Neben dem Ausfüllen eines Fragebogens über das momentane Befinden des Studienteilnehmers, bietet die Applikation auch das Ausführen von einfachen Aufgaben an. Eine dieser Aufgaben ist die Durchführung eines so genannten *Tapping-Tests* [13].

4

Track Your Tinnitus Plattform

In diesem Kapitel wird nochmals kurz der momentane Stand der Track Your Tinnitus Plattform vorgestellt. Diese besteht aus zwei Bestandteilen. Webserver und Applikationen für Android und iOS. Diese Kommunizieren über die JSON API untereinander wie in Abbildung 4.1 dargestellt. Desweiteren wurden folgende Grundfunktionalitäten bereits

Abbildung 4.1: Architekturübersicht entnommen aus [3]

implementiert

- *Login in der Anwendung*: Die App ist vor erfolgreicher Anmeldung nicht zu benutzen. Der Anwender erhält eine Übersicht in der App und auf der Website, die ihn dazu

auffordert seinen Benutzernamen und Passwort zum authentifizieren anzugeben. Sollte es dabei zu einem Fehler kommen, aufgrund falscher Benutzernamen / Passwort - Kombination, wird ihm dementsprechend eine Fehlermeldung angezeigt. Sollte der Login erfolgreich sein, wird überprüft, ob Antworten zu den Fragebögen zur Überwachung der Tinnitusschwankung offen sind. Sollte dies der Fall sein, wird er dazu aufgefordert, die restlichen Fragebögen auszufüllen.

- *Registrieren in der Anwendung*: Einem neuen Nutzer ist es zudem möglich, sich für Track Your Tinnitus zu registrieren. Dabei muss ein Benutzername, E-Mail Adresse und ein Passwort eingegeben werden. Zusätzlich kann der Nutzer selbst bestimmen, ob seine E-Mail Adresse gespeichert werden soll. Sollte ein Benutzername oder E-Mail Adresse bereits vergeben sein, erhält der Nutzer eine Fehlermeldung.
- *Ausfüllen der statischen Fragebögen*: Nach Neuregistrierung bzw. nicht vollständigem Ausfüllen der statischen Fragebögen, muss der Nutzer diese Fragebögen erst komplett ausfüllen. Erst danach ist es ihm möglich seine Tinnitusschwankungen aufzuzeichnen. Diese statischen Fragebögen kann der Nutzer in der App oder auch auf der Website ausfüllen.
- *Implementierung eines Hauptmenüs in der App*: Durch den Einsatz der SlidingMenu Bibliothek kann ein Hauptmenü für die App über die linke obere Ecke ein- und ausgeblendet werden. Das Hauptmenü erlaubt es dem Benutzer durch die App zu Navigieren um in andere Bereiche zu gelangen (Benachrichtungseinstellungen, Einstellungen, Ergebnisse, About und Logout).
- *Fragebogen zur Überwachung der Tinnituswahrnehmung*: Ein Hauptteil der Anwendung stellt die Aufzeichnung der Schwankungen des Tinnitus dar. Wenn der Nutzer erfolgreich eingeloggt ist, steht ihm diese Funktion zu unterschiedlichen Zeitpunkten des Tages zur Verfügung und kann seine momentane Tinnituswahrnehmung aufzeichnen. Dafür stehen ihm insgesamt acht Fragen zur Verfügung, die fest in der Anwendung implementiert sind. Speichert er seine Antworten über den dafür vorgesehenen Button, beendet sich die App automatisch und seine Antworten werden an den Webserver übertragen.

- *Benachrichtigungseinstellungen*: Dem Nutzer stehen zwei verschiedene Benachrichtigungsarten zur Verfügung. *Standart* und *Benutzerdefiniert*. Bei Standart kann er seine Anzahl an Benachrichtungen pro Tag selbst definieren. Danach kann er einen Start- und Endzeitpunkt der Benachrichtigungen festlegen. Die App sendet dem Nutzer schließlich innerhalb dieser Zeitspanne die Anzahl an Erinnerungen, die Ihn informieren, einen Fragebogen auszufüllen. Im Modus Benutzerdefiniert, erhält der Nutzer einen Kalender angezeigt in dem er selbst definieren kann, an welchen Wochentagen und an welcher Uhrzeit die App eine Notification senden soll. Zudem kann der Benutzer auch das Senden von diesen Erinnerungen komplett ausschalten.
- *Einstellungen*: Hier kann der Nutzer seinen Benachrichtungsklingelton auswählen, sowie die Aufzeichnung des Geräuschpegels durch das Mikrofon an- und ausschalten.
- *Grafische Darstellung der Ergebnisse*: Der Benutzer hat zwei unterschiedliche Möglichkeiten seine Ergebnisse grafisch anzeigen zu lassen. Die *Diagrammansicht* visualisiert die Ergebnisse in einem Koordinatensystem. Dabei ist die x-Achse der zeitliche Verlauf, die Y-Achse das eingetragene Ergebnis. In der *Timeline* Ansicht wird jeder Ergebnisdatensatz in einem eigenen Rahmen angezeigt. In diesem Rahmen sind das Datum und die Uhrzeit des Datensatzes, sowie die Ergebnisse aufgeführt.
- *About - Bereich*: Diese Ansicht liefert dem Benutzer weitere Informationen über das Track Your Tinnitus Projekt. Hier kann Kontakt zum Track Your Tinnitus Team hergestellt, sowie die Datenschutzbestimmungen und Open-Source Lizenzen durchgelesen werden.

5

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an das Backend bzw. die App definiert. Diese unterscheiden sich in funktionale Anforderungen wie auch in nicht-funktionale Anforderungen. Funktionale Anforderungen dienen zur Beschreibung, was die App und das Backend leisten soll. Nicht-funktionale Anforderungen hingegen beschreiben was die App und Backend in Design und Handhabung leisten sollen.

5.1 Funktionale Anforderungen

In diesem Unterkapitel werden die wichtigsten funktionalen Anforderungen, die die Applikation und das Backend bieten sollen, tabellarisch aufgelistet.

Nr.	Beschreibung	Problembeschreibung
1	Erstellen einer Studie	Einem Studienleiter soll es möglich sein, eine Studie zu erstellen. Dies kann nur nach erfolgreicher Authentifikation erfolgen.
2	Generierung eines Codes zur Studie	Um sich als Studienteilnehmer in eine Studie einzuschreiben, muss ein Studiencode in der App eingetragen werden. Diesen Code soll der Studienleiter sich im Backend generieren lassen können.
3	Erstellung von Register-Fragebögen	Das Ausfüllen von konfigurierbaren statischen Fragebögen ist eine Voraussetzung zur Teilnahme an der eigentlichen Studie. Diese Fragebögen müssen im Backend konfigurierbar sein.

4	Erstellung eines statischen Fragebogens zur Durchführung einer klinischen Studie	Nach ausfüllen der Register Fragebögen, sollen nun die statischen Fragen zu einer Studie ausgefüllt werden können. Der Studienleiter muss diese Fragen im Backend pflegen können.
5	Konfiguration von Benachrichtigungen	Der Studienleiter soll im Backend die Möglichkeit haben, selbst zu bestimmen wann ein Studienteilnehmer von der App benachrichtigt werden soll. Diese Benachrichtigung soll eine Motivation darstellen um einen neuen statischen Fragebogen auszufüllen.
6	Starten und Stoppen einer klinischen Studie	Der Studienleiter soll in der Lage sein, selbst zu entscheiden, wann eine Studie beginnt bzw. beendet ist.
7	Exportieren von Ergebnissen	Der Studienleiter soll die Möglichkeit besitzen, bereits eingesendete Informationen und Daten zur Studie (Antworten auf den statischen Fragebogen zur Studie) zu exportieren.
8	Überichts- und Informationsseite zu einer klinischen Studie	Der Studienleiter kann Informationen über seine erstellten Studien auf einer Website herausfinden.
9	Eine klinische Studie darf nicht mehr veränderbar sein, sobald diese gestartet ist	Sollte eine klinische Studie bereits begonnen haben, darf diese vom Studienleiter nicht mehr veränderbar sein. Dies Betrifft die Benachrichtigungen an den Probanden, wie auch die Fragen zu den Fragebögen.
10	Proband soll via App an Studie teilnehmen können	Der Proband erhält von seinem Studienleiter einen Studiencode, der es ermöglichen soll an einer klinischen Studie teilzunehmen.
11	Register- und Statefragebögen sollen in der Android App ausfüllbar sein	Der Proband kann in seiner App einen Fragebogen erfolgreich ausfüllen und an das Backend zur Speicherung weiterleiten.

12	Bei Custom-Notifications soll ein ändern der Benachrichtigungen nicht möglich sein	Der Proband soll bei Custom Notifications nicht berechtigt sein, die Erinnerungen von der App selbst zu definieren.
13	Android App soll Offline Funktion bieten	Es ist nicht immer gewährleistet, dass das Smartphone permanent mit einem Netzwerk verbunden ist. In diesem Fall sollen die Antworten des Probanden in der App zwischengespeichert bleiben und bei erfolgreicher Internetkonnektivität übertragen werden.

Tabelle 5.1: Funktionale Anforderungen

5.2 Nicht-funktionale Anforderungen

In diesem Unterkapitel werden die wichtigsten nicht-funktionalen Anforderungen, die die Applikation und das Backend bieten sollen, tabellarisch aufgelistet. Dazu gehören Anforderungen bezüglich der *Qualität* sowie *Benutzbarkeit* der erweiterten Track Your Tinnitus Plattform.

Nr.	**Beschreibung**	**Problembeschreibung**
1	Der Betrieb von Track Your Tinnitus darf keinerlei Auswirkungen der neuen Variante verspüren	Die Track Your Tinnitus Studie darf von den neuen Funktionen nicht beeinflusst werden und muss daher weiterhin fehlerfrei verwendbar sein.
2	Android Version der App soll dem Styleguid des Material Designs nachgebaut werden.	Um ein modernes Aussehen der App zu gewährleisten, veröffentliche Android mit seiner Version 5 (Lolipo) das Material Design. Die Android Variante soll auf dieses Design umgebaut werden.

3	Teilnahme an der Studie unter der Prämisse, dass die E-Mail Adresse des Probanten gespeichert werden darf	Eine Teilnahme an einer klinischen Studie setzt voraus, dass trackyourtinnitus.org die E-Mail Adresse des Probanten zur Auswertung von klinischen Statistiken speichern darf.
4	Die Anwendung soll Mehrsprachigkeit unterstützen	Momentan sollen Deutsch und Englisch als Sprachvarianten unterstützt werden. Dies soll in Abhängigkeit der eingestellten Sprache des Smartphones realisiert werden.
5	Drittbibliotheken sollen, soweit sinnvoll, von der Android App entfernt werden.	Viele Funktionalitäten dieser Drittbibliotheken werden nun durch das Material Design bzw. seit Android 5.0 (Lolipop) abgedeckt. Hierbei muss untersucht werden, inwieweit sich ein Einsatz dieser Drittbibliotheken noch lohnt.

Tabelle 5.2: Nicht-funktionale Anforderungen

6

Konzept, Entwurf und Implementierung

Dieses Kapitel befasst sich mit der Bearbeitung der einzelnen Anforderungen aus Kapitel 5. Dabei ist es in weitere 2 Unterkapitel getrennt. Das erste Kapitel beschäftigt sich mit der Entwicklung und Realisierung des Webservers. Es wird beschrieben, wie das Erstellen einer Studie sowie deren Bestandteile realisiert und implementiert wurden. Das Zweite erläutert die Abarbeitung der Anforderungen an die Android Applikation. Hierbei wird das neue Berechtigungskonzept von Android 5 näher betrachtet, die neuen Activities und Fragmente der Track Your Tinnitus Applikation beschrieben, sowie der Nutzen von Services, Notifications und Receivern im Kontext von Track Your Tinnitus erläutert.

6.1 Website

In diesem Unterkapitel werden die einzelnen Teile einer klinischen Studie beschrieben. Es wird ausführlichst erläutert, wie eine solche Studie aufgebaut ist und wie diese technisch implementiert wurde. Eine Studie besteht im wesentlichen aus mehreren Modellen und dessen Relationen untereinander. Abbildung 6.1 zeigt einen beispielhaften Aufbau. Eine klinische Studie besteht zuerst einmal aus dessen *Studylocalizationmodel()*. Dieses dient dazu um die klinische Studie in einer weiteren Sprache zur Verfügung zu stellen. Zudem kann es, abhängig der Konfiguration (s. Kapitel 6.1.2), aus mehreren *WeekdaySchedules* bestehen. Ein elementarer Bestandteil einer Studie ist dabei der *Registerquestionnaire* (s. Kapitel 6.1.3). Dieser beinhaltet mehrere *Registerquestions* (s. Kapitel 6.1.4). Diese Register Fragebögen sind vor der eigentlichen Durchführung der Studie zu beantworten. Die Fragen der klinischen Studie sind durch das *Statequestion*

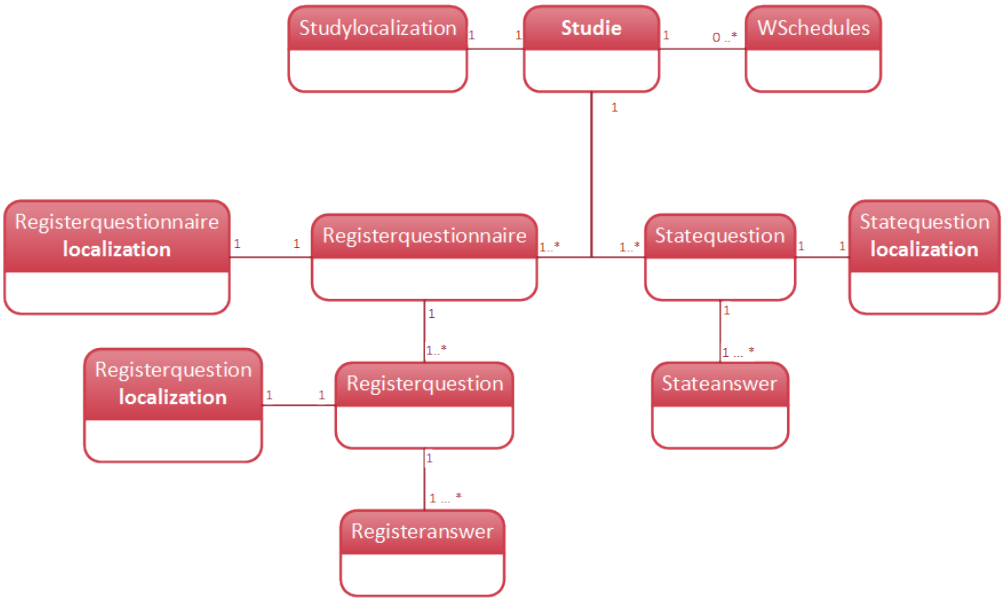

Abbildung 6.1: Aufbau und Bestandteile einer Studie

Model abgebildet (s. Kapitel 6.1.5). Registerquestionnaire, Registerquestion und Statequestion besitzen zudem noch ihre jeweiligen *Localization* Modelle. Die Fragenmodelle sind zuletzt noch ihren jeweiligen Antworttypen, *Registeranswer* und *Stateanswer* (s. Kapitel 6.1.6), zugeordnet. In den jeweiligen Unterkapitel werden nun die einzelnen Bestandteile genauer beschrieben. Dazu zählt der Aufbau eines solchen Models, sowie die jeweilige Ansicht auf der Website.

6.1.1 Study

Abbildung 6.2 zeigt die Studienindex Seite die ein Studienleiter sieht, sobald er sich erfolgreich an der Track Your Tinnitus Website angemeldet und auf den Button in der Menüleiste (1) geklickt hat. Ein Studienleiter im Track Your Tinnitus Kontext ist ein User, der sich in den Gruppen „*studienadmin*“ oder „*admin*“ befinden. Diese können durch Administratoren in die jeweilige Gruppe eingetragen werden. User die sich in der Gruppe „*studienadmin*“ befinden, haben jedoch keinen Zugriff auf die Adminfunktionalitäten von Track Your Tinnitus.

Über den Button (2) kann der Studienleiter eine weitere Studie erstellen. In der Tabelle (3) befinden sich sämtliche Studien, die der Studienleiter angelegt hat. Um herauszufinden, welche Studien zu welchem Studienleiter gehören, besitzt das Studienmodel eine Referenz auf die ID des Users.

Diese Tabelle beinhaltet folgende Informationen: *ID der Studie, Studiencode, Titel der Studie, Ist die Studie aktiv, War die Studie schon aktiv, Startdatum der Studie, Enddatum der Studie, Options-Spalte*. Mit der Options-Spalte ist es dem Studienleiter möglich zu den Registerfragebögen und Statequestions zu gelangen. Dies ist jedoch nur möglich, sollte die Spalte "*Active*oder "*Was Active*"den Wert "*No*enthalten. Möchte er dennoch über die URI darauf zugreifen, wird er auf eine Fehlerseite weitergeleitet, da ein Bearbeiten der Fragen zur Studie nach Start bzw. Beendigung der Studie nicht mehr möglich ist. Desweiteren kann er die Studie starten, stoppen oder löschen.

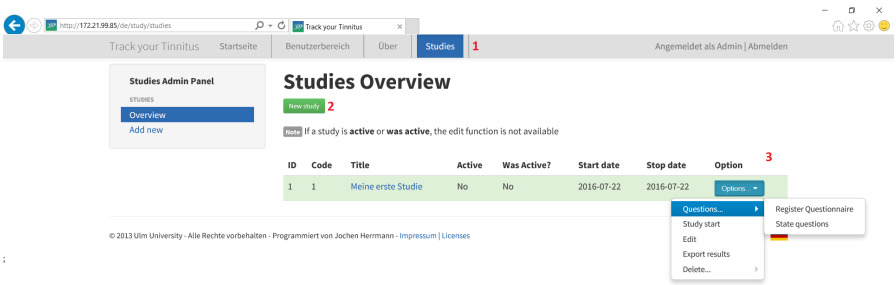

Abbildung 6.2: Indexseite für Studienleiter

Das Studienmodel ist durch folgende Attribute (siehe Zeile 5) definiert:

- *Code*: Enthält den Code den ein Studienteilnehmer benötigt um sich an der Studie anzumelden.
- *Title*: Der Titel der Studie.

- *scheduling_type*: Kann meherere Werte enthalten. Mögliche Werte sind: TYT, daily & weekday.
- *configuration*: Abhängig vom scheduling_type. Sollte der scheduling_type „daily“ sein, stehen hier die Uhrzeiten, an denen die App täglich Benachrichtigungen absetzt. In anderen Fällen ist diese Spalte „null“.
- *description*: Eine optionale Beschreibung der Studie.
- *active*: Boolean Flag, das signalisiert ob eine Studie momentan aktiv ist, oder nicht. Dieses Flag wird gesetzt sobald der Studienleiter die Studie startet. Wenn er die Studie beendet wird dieses wieder zurück auf „false“ gesetzt.
- *was_active*: Boolean Flag, das signalisiert ob eine Studie bereits aktiv war, oder nicht. Dieses Flag wird gesetzt sobald der Studienleiter die Studie startet. Dieses flag bleibt auf „true“ auch wenn der Studienleiter die Studie beendet.
- *was_active_since*: Der Startzeitpunkt der Studie.
- *was_active_till*: Der Stopzeitpunkt der Studie.
- *user_id*: Referenz zum Ersteller der Studie (Studienleiter). Anhand dieser ID erhalten die Studienleiter ihre Studien auf der Indexseite des Studienbereichs. Somit ist ausgeschlossen, dass fremde Studienleiter diese Studie sehen können.

Listing 6.1 zeigt den Quellcode wie Studien den jeweiligen Studienleitern angezeigt werden können:

```
1 $studies = Study::where('user_id', '=', Sentry::user()->get('id'))->
      get();
2 //select * from studies where user_id = 1;
3         if (count($studies) == 0) {
4             return View::make('studies.studies_overview')->with('
                  no_records', true);
5         }
6         return View::make('studies.studies_overview')->with('studies'
              , \$studies);
```

Quelltext 6.1: Quellcode zum Auflisten sämtlicher Studien des angemeldeten Nutzers

Eloquent bietet in Laravel mehrere Methoden an, Einträge aus einer Datenbank zu lesen. In Zeile (1) wird eine einfache *where* Abfrage auf die Tabelle gemacht, die im

Studienmodel hinterlegt ist. *Sentry::user()->get('id')* liefert dabei die ID des momentan angemeldeten Users. Der Kommentar in Zeile 2 symbolisiert dabei das im Hintergrund ausgeführte SQL Query. Die Funktion gibt dabei ein Array an Objekte der Klasse Study zurück. Sollte die Anzahl der Arrayelemente 0 sein, wird dem Studienleiter ein Hinweiß auf der Indexseite angezeigt, dass es von Ihm noch keine erfassten Studien gibt.

Abbildung 6.3 zeigt die View zum erstellen einer neuen Studie. Mit einen klick auf den

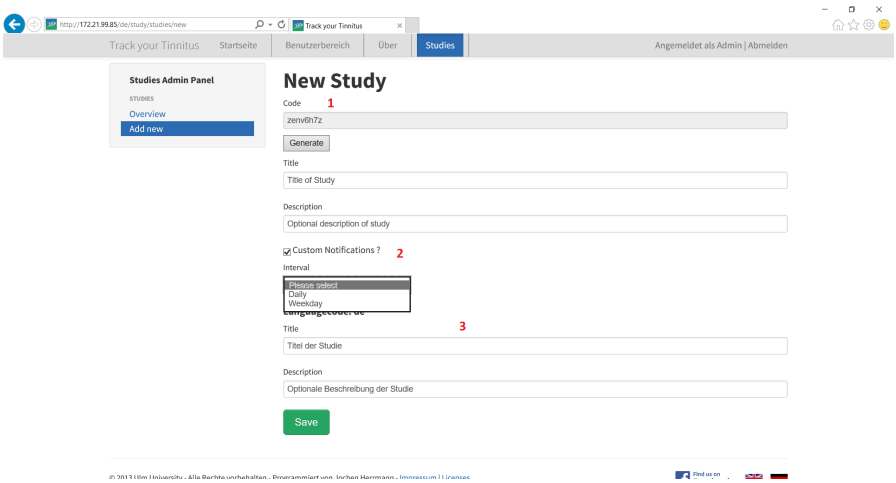

Abbildung 6.3: Erstellen einer Studie

Button „Generieren“ (1) wird der Code zu einer Studie generiert. Dies wurde durch die folgende Javascript Funktion realisiert:

```
1 <script>
2     function generateCode() {
3         var code = Math.random().toString(36).substring(4, 12);
4         document.getElementById("code").value = code;
5     }
6 </script>
```

Quelltext 6.2: Generieren eines Studiencodes

Der Code wird durch die Funktion in Zeile 3 generiert und letztendlich in das vorhergesehene Feld zurückgeschrieben. Im Feld Titel und Description (2) werden der Studientitel und die zugehörige Beschreibung dazu hinterlegt. Mit der Checkbox „Custom Notifications“ können eigene Benachrichtigungen in der App konfiguriert werden. Dem Studienleiter stehen folgende Benachrichtigungsmöglichkeiten zur Verfügung:

- *Daily*: Die App benachrichtigt den Studienteilnehmer täglich zur vorgegebenen Uhrzeit.
- *Weekday*: Die App benachrichtigt den Studienteilnehmer an ausgewählten Wochentagen zu konfigurierten Uhrzeiten.

Sollte diese Checkbox nicht ausgewählt werden, wird der Benachrichtungsalgorithmus (beschrieben in [3]) von Track Your Tinnitus verwendet. In (3) wird die jeweilige *Localization* konfiguriert. Der Titel sowie die Beschreibung der Studie wird hier ins Deutsche übersetzt. In Abhängigkeit der eingestellten Sprache auf dem Smartphone des Anwenders erhält dieser dann den englischen beziehungsweiße deutschen Text. Diese beide Sprachvarianten werden momentan von Track Your Tinnitus unterstützt.
Mit einem Klick auf Speichern wird im StudienController nun ein POST Request ausgelöst. Dieser Request routet auf die *store* Methode (siehe Listing 6.3) im *StudyController*, der einen neuen Eintrag in der Tabelle *studies* anlegt, sowie einen neuer Eintrag in der Studylocalization Tabelle.

```
1  public function post_newstudies(){
2          $input = Input::all();
3          if (!$input['title']) {
4              return Redirect::to('study/studies/new')->with_input()->
                   with('error_message', true);
5          }
6          if (!$input['code']) {
7              return Redirect::to('study/studies/new')->with_input()->
                   with('error_message', true);
8          }
9          $study = new Study();
10         $study->code = $input['code'];
11         $study->was_active = 0;
12         $study->title = $input['title'];
13         $study->description = $input['description'];
14         $study->active = 0;
15         $study->user_id =  Sentry::user()->get('id');
16         $study->save();
17             if(Input::has('custom_notification')){
18                 if($input['interval'] == 'daily'){
19                     $study->scheduling_type = 'daily';
20                     $this->prepareConfigurationDaily($study);
21                 }else{
22                     $study->scheduling_type = 'weekday';
23                     $this->prepareConfigurationWeekday($study, $input
                           );
24                 }
25             }else{
```

```
26              $study->scheduling_type = 'TYT';
27          }
28          $study->save();
29          foreach (Config::get('application.languages') as
                $languagecode) {
30              if (Input::has('title_' . $languagecode) && Input::has('
                    description_'.$languagecode)) {
31                  $localization = new StudyLocalization();
32                  $localization->title = Input::get('title_' .
                        $languagecode);
33                  $localization->description = Input::get('description_
                        ' . $languagecode);
34                  $localization->study_id = $study->id;
35                  $localization->languagecode = $languagecode;
36                  $localization->configuration = $study->configuration;
37                  $localization->save();
38              }else{
39          return Redirect::to('study/studies/new')->with_input()->with('
                missing_translation', true);
40          }
41          }
42          return Redirect::to('study/studies')->with_input()->with('
                record_created', true);
43      }
```

Quelltext 6.3: Anlegen einer neuen Studie in der Datenbank

In Zeile 2-8 wird überprüft ob die Studie einen Titel sowie einen Code besitzt. Wenn nicht, wird der Studienleiter zurück auf die Create-Seite geleitet und auf die fehlenden Attribute hingewießen. Zeile 9 beschreibt das Erstellen eines neuen Studienobjekts. Zeile 10 - 15 füllt die nötigen Attribute zur Studie. Zeile 16 speichert die Studie in die Tabelle. In Zeile 17 - 27 werden die Benachrichtigungen zur Studie konfiguriert (siehe Kapitel 6.1.2). Zeile 29 - 41 legt für jede hinterlegte Sprachvariante die nötige Localization an. Da Track Your Tinnitus momentan nur in Deutsch und Englisch verfügbar ist, gibt es nur eine weitere Übersetzung. Zeile 42 leitet den Studienleiter auf die Studienübersichtsseite zurück, mit einem Hinweiß, dass die Studie erfolgreich angelegt wurde.

6.1.2 Benachrichtungseinstellungen einer Studie

Wie bereits in Kapitel 6.1.1 beschrieben, stehen dem Studienleiter drei verschiedene Benachrichtungstypen zur Verfügung.

- *Track Your Tinnitus*

- *Daily*
- *Weekday*

Je nachdem welcher Benachrichtungstyp gewählt wurde, erhält man unterschiedliche Objekte:

```
1 [{... "title":"Meine erste Studie","scheduling_type":"TYT","
    configuration":null, ...}]
2 [{... "title":"Benachrichtigung DAILY","scheduling_type":"daily","
    configuration":"13:00,15:00,18:00", ...}]
3 [{... "title":"Benachrichtigung WEEKDAY","scheduling_type":"weekday",
    "configuration":[{"Monday":"10:00,13:00"},{"Wednesday":"
    09:00,20:00"},{"Friday":"15:00,17:25"},{"Sunday":"16:00,18:45"}],
    ...}]
```

Quelltext 6.4: Studien mit unterschiedlichen Benachrichtungstypen

Wichtig hierbei sind die Attribute *scheduling_type* und *configuration*. Ersteres enthält den Typ der Benachrichtung. Das zweite Attribut enthält, bei Benachrichtungseinstellung *daily*, nötige Uhrzeiten. In Sourcecode 6.4 enthält das erste Json-Object den Benachrichtungsmodus von Track Your Tinnitus. Das Zweite enthält den Benachrichtungsmodus DAILY. Die App benachrichtigt somit den User täglich um 13, 15 und 18 Uhr. Im Dritten Objekt ist der Benachrichtigungsmodus an speziellen Wochentagen, *weekday*, hinterlegt. Für diesen Benachrichtungstyp steht das Model *study* in einer 1:n Beziehung mit dem Model *WSchedule*. Dieses Model beinhaltet den Wochentag, sowie die dazugehörigen Uhrzeiten als Attribut. Bei einem Request nach einer Studie mit dem Benachrichtigungstyp *weekday* wird im Attribut *configuration* der Studie, sämtliche der Studie zugeordneten Objekte des Models *WSchedule* übertragen. Die App benachrichtigt am Montag um 10 und 13 Uhr. Am Mittwoch um 9 und 20 Uhr. Freitags um 15 und um 17:25 und Sonntags um 16:00 und 18:45 Uhr.

6.1.3 Registerquestionnaire

Ein *Registerquestionnaire* dient als Container für Registerfragen. Diese sind vor der eigentlichen Teilnahme an der klinischen Studie auszufüllen. Klickt der Studienleiter auf der Studienübersichtsseite auf den Punkt *Registerquestionnaiers*, wird er auf die Übersichtsseite der Registerfragebögen weitergeiletet (siehe Abbildung 6.4). In dieser Tabelle

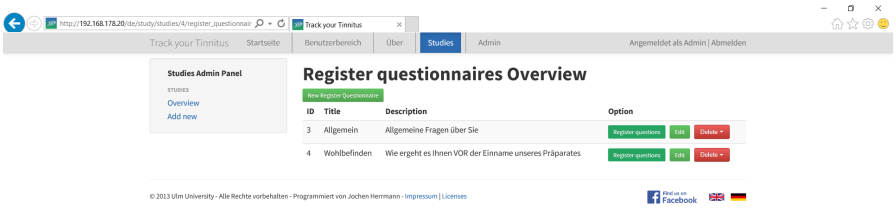

Abbildung 6.4: Übersicht der Registerfragebögen

werden sämtliche Registerfragebögen zur jeweiligen Studie angezeigt. Darunter fällt der Name des Fragebogens, die Beschreibung des Fragebogens, sowie weitere Aktionen die der Studienleiter ausführen kann. Er kann dem Fragebogen Fragen hinzufügen, den Fragebogen editieren oder löschen. Das Editieren ist nur so lange verfügbar, bis diese gestartet wurde. Abbildung 6.5 zeigt das Erstellen eines neuen Registerquestionnaire. Für einen Registerfragebogen sind 4 Pflichtfelder auszufüllen. (1) Zeigt den eigentlichen

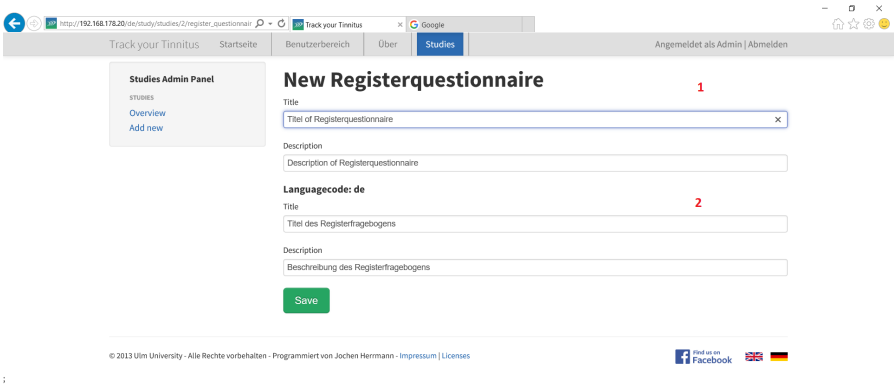

Abbildung 6.5: Erstellen eines Registerfragebogens

Titel und eine Beschreibung für den Fragebogen. (2) Ist die Übersetzung des Titels und

der Beschreibung ins Deutsche, damit die Anforderung der Mehrsprachigkeit gegeben ist. Klickt der Studienteilnehmer auf Speichern, kann er damit beginnen die eigentlichen Fragen zum Fragebogen hinzuzufügen.

6.1.4 Registerquestion

Zuerst gelangt der Studienleiter auf die Übersichtsseite der bisher erstellen Fragen zum Fragebogen (siehe Abbildung 6.6). Ein Registerfragebogen kann 1 - n verschiedene

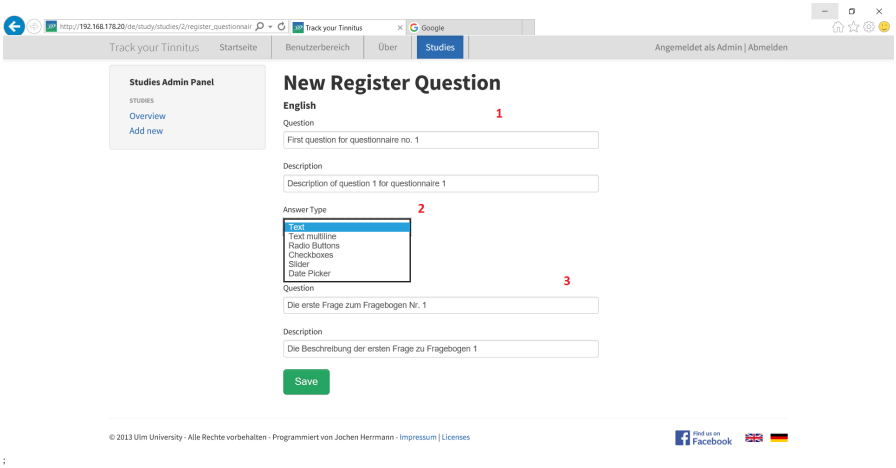

Abbildung 6.6: Übersicht Registerquestions

Registerquestions beinhalten. In dieser Tabelle sieht der Studienleiter den Fragentext, den Typ der Antwort, die vorhanden Localizations der Frage und kann weitere Aktionen durchführen. Ein Editieren der Frage ist so lange möglich, bis die Studie gestartet wurde. Das Löschen der Frage ist ebenfalls möglich, solange die Studie noch nicht gestartet wurde. Eine Registerfrage kann wie in Abbildung 6.7 gezeigt, neu angelegt werden: Der Studienleiter muss zunächst die Frage sowie die Beschreibung zur Frage formulieren (1). Im nächsten Schritt (2) muss der Studienleiter den zugehörigen Antworttyp auswählen. Dabei kann er zwischen 5 verschiedenen Antowrttypen auswählen:

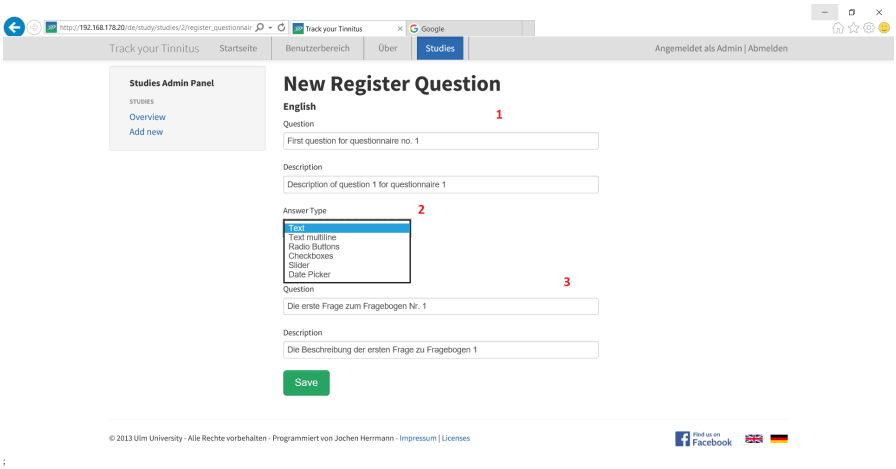

Abbildung 6.7: Erstellen eines Registerquestion

1. *Text*: Hier hat der Studienteilnehmer die Möglichkeit in einem Freitextfeld seine Antwort zu formulieren. Es findet keine Validierung auf den eingegebenen Text statt.
2. *Radio Button*: Bei diesem Antworttyp ist es möglich aus einer Antwortmenge genau eine Antwort auszuwählen. Diese Antwortmenge kann über das aufklappende Konfigurationsfeld angegeben werden.
3. *Checkbox*: Ähnlich wie bei den Radio Buttons ist es hier möglich aus einer Antwortmenge, mehrere Antworten auszuwählen. Auch hier müssen die möglichen Antworten konfiguriert werden.
4. *Slider*: Eine Antwortmöglichkeit um Werte zwischen 0 und 100 % auszuwählen. Hierbei wird initial kein Button angezeigt. Hier können die Achsenbeschriftungen konfiguriert werden.
5. *Datepicker*: Diese Möglichkeit bietet die Auswahl eines Datums als Antwort.

In (3) müssen die Fragen noch auf Deutsch übersetzt werden um Mehrsprachigkeit zu unterstützen. Sollten Fragen mit Konfigurationsmöglichkeiten ausgewählt worden sein, wird auch hier die Konfiguration für deutsche Antworten hinterlegt (Radiobuttons,

Checkboxes, Slider). Listing 6.5 zeigt die Methode im StudienController zum anlegen einer neuen Frage in der Datenbank:

```
1   public function post_registerquestionnew($sid, $qid){
2       if (!Input::has('question')) {
3           return Redirect::to('study/studies/'.$sid.'/
                register_questionnaire/'.$qid.'/register_questions/new
                ')->with_input();
4       }
5       $question_text = Input::get('question');
6       $question_type = Input::get('type');
7       switch ($question_type) {
8           case 'radio_button' :
9               if (!Input::has('configuration')) {
10                  return Redirect::to('study/studies/'.$sid.'/
                        register_questionnaire/'.$qid.'/
                        register_questions/new')->with_input()->with('
                        error_message', true);
11              }
12              break;
13          case 'checkbox' :
14          if (!Input::has('configuration')) {
15              return Redirect::to('study/studies/'.$sid.'/
                    register_questionnaire/'.$qid.'/register_questions
                    /new')->with_input()->with('error_message', true);
16          }
17          break;
18          case 'int_scale' :
19              if (!Input::has('configuration')) {
20                  return Redirect::to('study/studies/'.$sid.'/
                        register_questionnaire/'.$qid.'/
                        register_questions/new')->with_input()->with('
                        error_message', true);
21              }
22              break;
23          default :
24              break;
25      }
26      $question = array();
27      $question['registerquestionnaire_id'] = $qid;
28      $question['question'] = $question_text;
29      $question['type'] = $question_type;
30      $question['description'] = Input::get('description');
31      $question['study_id'] = $sid;
32      switch ($question_type) {
33          case 'radio_button' :
34          case 'checkbox' :
35          case 'int_scale' :
36              if (!Input::has('configuration')) {
37                  return Redirect::to('study/studies/'.$sid.'/
                        register_questionnaire/'.$qid.'/
```

```
                    register_questions/new')->with_input()->with('
                    error_message', true);
38              }
39              $question['configuration'] = Input::get('
                    configuration');
40              break;
41          default :
42              break;
43      }
44      $newquestion = Registerquestion::create($question);
45      foreach (Config::get('application.languages') as
            $languagecode) {
46          if (Input::has('question_' . $languagecode)) {
47              $localization = new RegisterLocalization();
48              $localization->question = Input::get('question_' .
                    $languagecode);
49              $localization->description = Input::get('description_
                    ' . $languagecode);
50              $localization->registerquestion_id = $newquestion->id
                    ;
51              $localization->languagecode = $languagecode;
52              switch ($question_type) {
53                  case 'radio_button' :
54                  case 'checkbox' :
55                  case 'int_scale' :
56                      if (!Input::has('configuration_' .
                            $languagecode)) {
57                          return Redirect::to('/study/studies/'.
                                $sid.'/register_questionnaire/'.$qid.'
                                /register_questions/new')->with_input
                                ()->with('localization_error', true);
58                      }
59                      $localization->configuration = Input::get('
                            configuration_' . $languagecode);
60                      break;
61                  default :
62                      break;
63              }
64              $localization->save();
65          }
66      }
67      return Redirect::to('study/studies/'.$sid.'/
            register_questionnaire/'.$qid.'/register_questions/')->
            with('record_created', true);
68  }
```

Quelltext 6.5: Anlegen einer neuen Registerfrage in der Datenbank

In Zeile 2 - 25 wird überprüft, ob sämtliche Pflichtfelder vorhanden sind. Ist dies nicht der Fall, wird er auf die View zum Erstellen der Frage mit einem entsprechenden Hinweiß weitergeleitet. In Zeile 26 - 44 wird das eigentliche Registermodel erstellt und mit

Registerquestion::create($question) in die Datenbank geschrieben. In Zeile 45 - 66 werden die jeweiligen Localizations zu dieser Frage erstellt.

Wenn ein Registerquestionnaire fertiggstellt wurde, kann man mittels des Studiencode einen HTTP Request an den Webserver stellen, der als Antwort den kompletten Fragebogen mit seinen Fragen und Antworttypen zurückgibt.

Abbildung 6.6 zeigt ein solches Ergebnis:

```
1 [{"id":"2","study_id":"2","title":"Titel of Registerquestionnaire","
    description":"Description of Registerquestionnaire","
    registerquestions":[{"id":"4","study_id":"2","
    registerquestionnaire_id":"2","question":"First question for
    questionnaire no. 1","description":"Description of question 1 for
    questionnaire 1","type":"int_scale","configuration":"Good\r\nn
    good"},{"id":"5","study_id":"2","registerquestionnaire_id":"2","
    question":"Question2","description":"QUESTION DESC 2","type":"
    radio_button","configuration":"Yes\r\nNo\r\nMaybe"},{"id":"6","
    study_id":"2","registerquestionnaire_id":"2","question":"Question
    3","description":"Desc for question 3","type":"checkbox","
    configuration":"0 - 10\r\n10 - 20\r\n20 - 89"},{"id":"7","study_id
    ":"2","registerquestionnaire_id":"2","question":"Question 4","
    description":"Desc question 4","type":"date_of_birth","
    configuration":""}]}]
```

Quelltext 6.6: Registerquestionnaire zur Studie als Json Objekt

6.1.5 Statequestion

Die Übersicht der wiederkehrenden Fragen der klinischen Studie beinhaltet sämtliche Fragen, die dem Studienteilnehmer in der App angezeigt werden (siehe Abbildung 6.8). Der Studienleiter sieht den Fragentext, den Antworttyp der Frage, die vorhandenen Localizations sowie weiter mögliche Aktionen. Ein Editieren und Löschen der Statequestions ist bis zu dem Zeitpunkt des Startes der Studien möglich. Die Statequestions bilden die eigentlichen Fragen der klinischen Studie ab. Der Unterschied zu einer Registerquestion ist, dass diese keinem Fragebogen zugeordnet werden. Zudem besteht zu den möglichen Antworttypen von Statequestions noch der Typ:

- *Feeling*

Dieses wird in der App als *Self-Assessment Manikin* dargestellt und dient als ein sprachfreies Verfahren, um die Dimensionen Freude (Pleasure), Erregung (Arousal) und Domi-

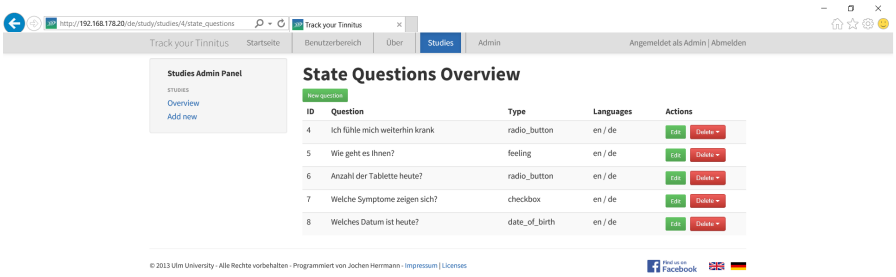

Abbildung 6.8: Übersicht Statequestions

nanz (Dominance) affektiver Reaktionen zu erfassen [25]. Hierbei wird für die Antwortmöglichkeit ein Wert zwischen 0 - 100% übetragen.

6.1.6 Register- & Stateanswers

Die Antworten die ein Studienteilnehmer zu einer Frage gibt, werden im Register- bzw Stateanswer Model gespeichert. Die Datenstruktur einer solchen Antwort ist wie folgt aufgebaut: Jede Registeranswer hat eine Referenz auf den Fragebogen, die Frage,

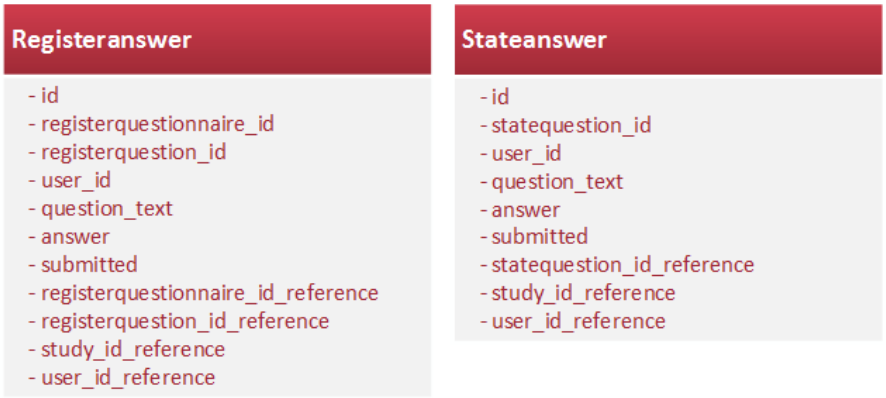

Abbildung 6.9: Datenstruktur Register- und Stateanswer

die Studie, sowie den Studienteilnehmer, von dem diese Antwort stammt. Diese vier Referenzen werden zusätzlich ein weiteres Mal gespeichert, um diese auch zu erhalten, wenn ein zugehöriger Datensatz in den referenzierten Tabellen gelöscht wird. Neben der Antwort als Text wird zusätzlich auch der Wortlaut der Frage gespeichert. Die Stateanswer ist identisch aufgebaut, jedoch fehlen hier die Referenzen auf den jeweiligen Fragebogen.

6.1.7 Editierbarkeit von Studien

Studien sind bis zu dem Zeitpunkt, an dem der Studienleiter sie gestartet hat, editierbar. Somit ist die Konsistenz der klinischen Studie gewährleistet. Sichergestellt wird dies im Backend durch „Route Filter“. Bei diesen Filtern durchläuft jeder Request auf eine bestimmte URI eine Funktion, die eine bestimmte Bedingung erfüllen muss. Im Track Your Tinnitus Umfeld wird dies durch folgende Funktion abgebildet:

```
1 Route::filter('isactive', function(){
2   $study = Study::find(Request::route()->parameters[0]);
3   if($study->active == 1 || $study->was_active == 1){
4     return View::make('error.403');
5   }
6 });
```

Quelltext 6.7: Filterfunktion in Laravel

Sollte ein Studienleiter beispielsweise eine Statefrage zu einer aktiven Studie ändern wollen, wird er auf folgende Seite weitergeleitet (siehe Abbildung 6.10):

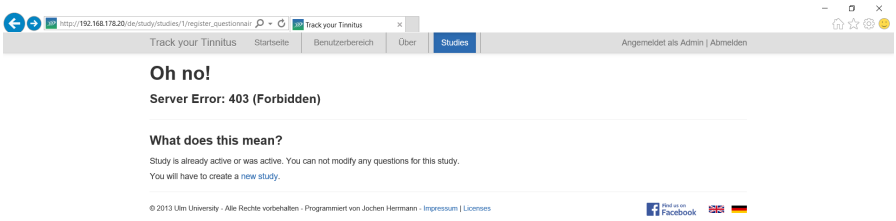

Abbildung 6.10: Versuch eine aktivie Studie zu editieren

6.1.8 Informationsseite zu einer Studie

Der Studienleiter kann Informationen zum verlauf seiner Studie begutachten (siehe Abbildung 6.11). Dies geschieht, indem er auf die jeweilige Studie im Indexbereich klickt. Somit öffnet sich folgende Seite:

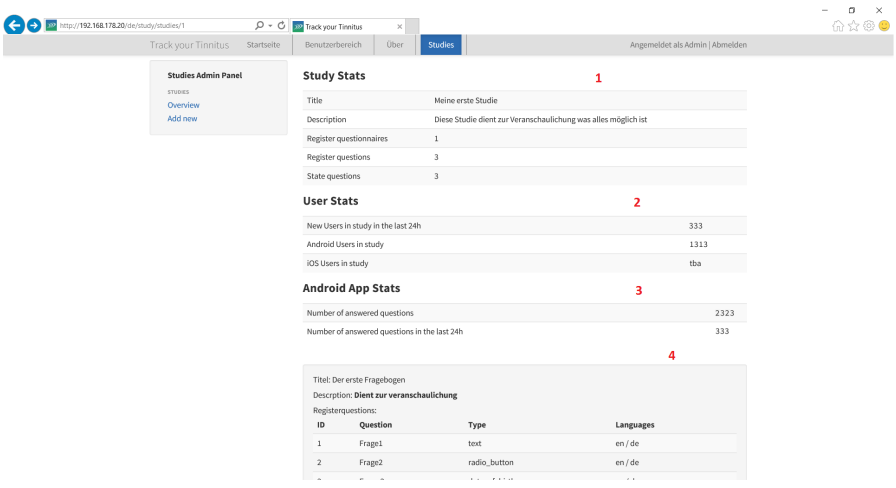

Abbildung 6.11: Studienstatistiken

Diese Seite glieder sich in vier Bereiche. Bereich (1) beinhaltet Informationen allgemein zur Studie. Darunter zählen der Titel, die Beschreibung, die Anzahl an Registerfragebögen, die Anzahl an Registerfragen, sowie die Anzahl der Statefragen. In Bereich (2) sind Benutzerstatistiken aufgeführt. Dazu zählen: Die Anzahl an neu eingeschriebenen Studienteilnehmern in den letzten 24 Studen, die Anzahl der Android-Nutzern und die Anzahl der iOS Nutzern. Im 3. Bereich (3) sind Statistiken zu den übertragenen Stateanswersn in den letzten 24 Stunden sowie die gesamte Anzahl an Stateanswern. In Bereich (4) werden dem Studienleiter nochmal sämtliche Fragen (Register- wie auch Statequestion) aufgelistet.

6.1.9 Export der Daten als CSV Datei

Eine weitere Anforderung ist, die eingengangen Antworten pro klinischer Studie in einer CSV Datei exportieren zu können. in CSV Dateien werden Daten zeilenweiße abgelegt. Die einzelen Daten innerhalb eines Datensatzes werden durch ein Komma separiert und können somit von den meisten Programmen für Auswertungen herangezogen werden. In der Track Your Tinnitus Plattform werden somit sämtliche Stateanswers exportiert. Der Studienleiter klickt auf der Indexseite auf den Button „Option“ und wählt schließlich den Punkt „Export as CSV“. Das Codefragment, das diese CSV Datei erzeugt ist in Listing 6.8 dargestellt:

```
1     public function get_exportstudy($sid)
2     {
3         $headers['Content-Type'] = 'text/csv';
4         $headers['Content-Disposition'] = 'attachment; filename="
              studienergebnisse.csv"';
5         $statequestions = Statequestion::where('study_id', '=', $sid)
              ->get();
6         $csv = "";
7         foreach ($statequestions as $question) {
8             $csv .= $question->question . ', ';
9         }
10        $csv     .= "user_id\n";
11            $stateanswers = Stateanswer::where('study_id', '=', $sid)
                  ->order_by('created_at', 'asc')->order_by('
                  statequestion_id', 'asc')->get();
12            $y = 0;
13            foreach($stateanswers as $sa){
14                $csv     .= $sa->answer.",";
15                $y++;
```

```
16                  if($y == count($statequestions)){
17                      $csv      .= $sa->user_id."\n";
18                      $y = 0;
19                  }
20              }
21          return Response::make($csv, 200, $headers);
22      }
```

Quelltext 6.8: Export der Antworten einer Studie

Im Response Header wird zunächst der *Content-Type* auf *text/csv* gesetzt. Somit ist klar, welche Daten gesendet werden. Das Feld *Content-Disposition* erzwingt den Download der csv Datei, dessen Name auf *studienergebnisse.csv* gesetzt wird. Zeile 7 - 10 sammelt die *Fragentexte* der *Statequestions* der Studie. Zusätzlich wird die Spalte *user_id* hinzugefügt, um eine Referenz von Antworten auf Nutzer zu erhalten. In Zeile 11 - 20 werden nun die Antworten des Studienteilnehmer unter den jeweiligen Fragen aufgelistet. Der gesamelte Content wird letztendlich zurück zum Client gesendet. Das Ergebnis ist in Abbildung 6.12 dargestellt.

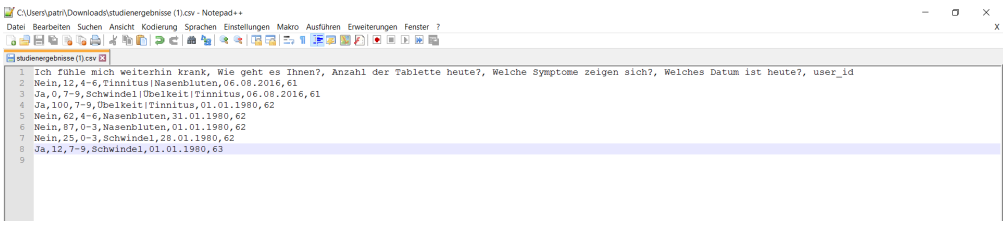

Abbildung 6.12: Ergebnis des Exports der Stateanswer der User

6.2 Android

In diesem Unterkapitel wird die Umsetzung der Anforderungen an die Android Applikation näher erläutert. Dabei wird auch auf die Probleme näher eingegangen, die sich während der Entwicklung ergeben haben. Die aktuelle Version von Track Your Tinnitus wurde unter der Android Version 18 (4.3.1 aka Jelly Bean) erstellt. Die minimale Androidversion, die ein Smartphone installierte haben muss um Track Your Tinnitus verwenden zu können ist 8 (2.2 aka Froyo). Diese Versionen sind zum Teil sehr veraltet und bieten keine API Schnittstellen von neueren Android Versionen. Die neue Applikation erhält somit die targetSDKVersion von 23 (Android 5 aka Marshmallow). Um nun sämtliche Features zu unterstützen muss die minSDKVersion auf 16 (4.1 aka Jelly Bean) geändert werden. Damit Googles Material Design auch auf Smartphones mit Android 4 zur Verfügung steht, ist der Einsatz von Googles *Support Library* [23] nötig. Ein weiterer Grund aktuellere Android Versionen zu unterstützen ist die Verbreitung dieser Versionen wie in Abbildung 6.13 zu sehen ist.

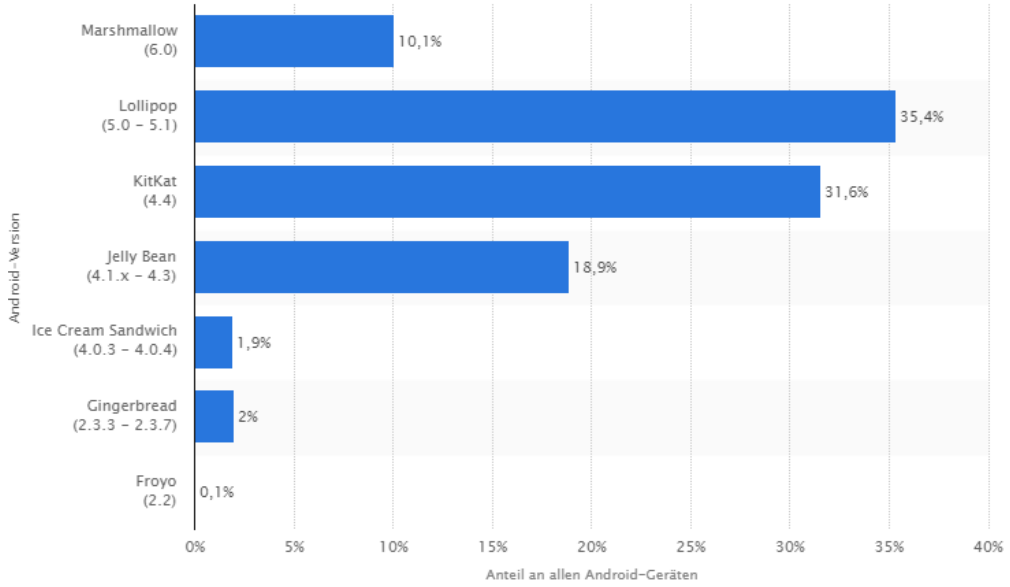

Abbildung 6.13: Anteil der verschiedenen Android-Versionen entnommen aus [27]

Der relevante Marktanteil an zu unterstützenden Smartphones befindet sich ab der Android Version 4.1. Das entspricht 96% aller Androidbenutzer.
Die weiteren Anforderungen werden in den jeweiligen Kapitel erläutert und umgesetzt.

6.2.1 Klassenmodelle der Android Applikationen

Abbildung 6.14 zeigt sämtliche wichtigen Klassen, die neu in die Track Your Tinnitus Applikation gekommen sind. Für die Datenhaltung der Register Fragebögen ist die Klasse *Registerquestionnaire* verantwortlich. Die zugehörigen Register Fragen werden duch die Klasse *RegisterQuestion* repräsentiert. State Fragen werden in der Klasse *StateQuestion* abgebildet. Register und State Antworten erhalten die Klassen *RegisterAnswer* und *StateAnswer*. Jede dieser Klassen besitzt eine eigene SQLiteOpenHelper Subklasse. Diese Klassen übernehmen das Persistieren von Daten in der SQLite Datenbank auf dem Android Gerät.

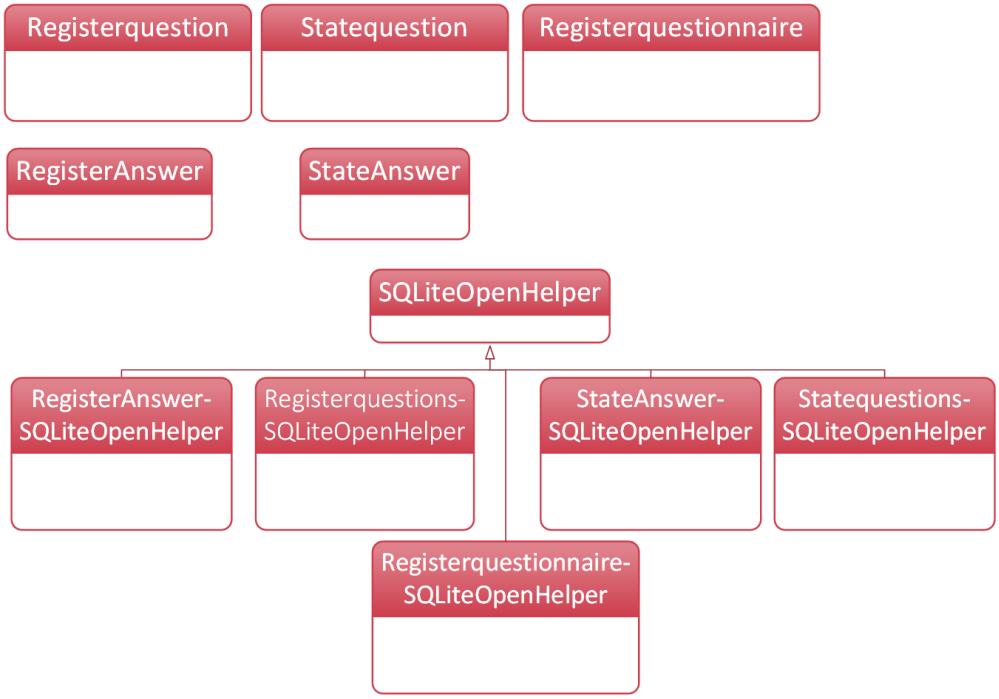

Abbildung 6.14: IntroductionFragmentActivity mit drei Fragmenten

6.2.2 Android Marshmallow's neues Berechtigungskonzept

Android hat mit der Einführung von Version 6 aka Marshmallow ein neues Berechtigungskonzept für Apps eingeführt. In den vorherigen Versionen hat man schon während der Installation aus dem Google Play Store die Berechtigungen angezeigt bekommen, die ein User akzeptieren musste, damit die App ordnungsgemäß funktioniert hat. Mit der neuen Android Version entfällt dieser Schritt nun. Das neue Berechtigungskonzept zielt vielmehr darauf ab, den User entscheiden zu lassen, was er zulassen möchte. Somit muss die App nun in der Lage sein, explizit nach Berechtigungen anzufragen, während die App gestartet ist. Es muss jedoch nicht nach jeder Permission gefragt werden. Android unterscheidet hierbei in 2 große Bereiche:

1. *Normale Berechtigungen*: Hierbei handelt es sich um die Art von Berechtigungen die nicht direkt die Privatsphäre von den Anwendern stört. Sollte so eine Berechtigung in der Manifestdatei der Androidapplikation gelistet sein, wird diese automatisch akzeptiert.
2. *Gefährliche Berechtigungen*: Dieser Typ der Berechtigungen haben unmittelbar mit der Privatsphäre des User zu tun. Diese Berechtigungen muss der User explizit erteilen!

Eine vollständige Liste der gefährlichen Berechtigungen ist in Tabelle 6.1 gelistet.

Tabelle 6.1: Dangerous Permissions in Android aus [26]

Permissions Group	Permissions
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO

PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALLS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
SENSORS	BODY_SENSORS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

Von der Liste der *gefährlichen Berechtigungen* 6.1, nutzt die momentane Track Your Tinnitus Applikation *RECORD_AUDIO* zum Aufzeichnen des Geräuschpegels während des Ausfüllens eines Fragebogens. Sollte diese Berechtigung noch nicht akzeptiert worden sein, beziehungsweise wurde vom Studienteilnehmer wieder in den Einstellungen entzogen, muss die Berechtigungsanfrage während der Ausführung der App gestellt werden. In der Track Your Tinnitus Applikation wird dies in der *MainActivity* in der *onCreate()* Funktion überprüft. Listing 6.15 zeigt, wie Berechtigungen während der Ausführung eingeholt werden können:

```
1  if (ContextCompat.checkSelfPermission(this,android.Manifest.
       permission.RECORD_AUDIO)!= PackageManager.PERMISSION_GRANTED) {
2              if (ActivityCompat.shouldShowRequestPermissionRationale(
                   this,
3                       android.Manifest.permission.RECORD_AUDIO)) {
4                       ...
5              } else {
6                   ActivityCompat.requestPermissions(this,
7                           new String[]{android.Manifest.permission.
                                RECORD_AUDIO},
8                           1);
9              }
10         }
11         ...
12         @Override
13     public void onRequestPermissionsResult(int requestCode, String[]
           permissions, int[] grantResults) {
14         SharedPreferences.Editor editor = preferences.edit();
15         switch (requestCode) {
```

```
16              case 1:
17                  if (grantResults[0] == PackageManager.
                        PERMISSION_GRANTED) {
18                      // Permission Granted
19                      editor.putInt("RecordSoundLevel", 1);
20                  } else {
21                      // Permission Denied
22                      editor.putInt("RecordSoundLevel", 0);
23                  }
24                  editor.apply();
25                  break;
26              default:
27                  super.onRequestPermissionsResult(requestCode,
                        permissions, grantResults);
28          }
29      }
```

Quelltext 6.9: Berechtungsanfrage stellen während der Ausführung der App

In Zeile 1 wird überprüft, ob der Anwender die Genehmigung bereits erteilt hat, dass während des Ausfüllens eines Fragebogens, der Geräuschpegel der Umgebung über das Mikrofon aufgezeichnet werden darf. Wenn dies der Fall ist, wird die Anwendung fortgeführt. Ist dies nicht der Fall, wird überprüft, ob der Anwender diese Berechtigung bereits schon einmal abgelehnt hat (siehe Zeile 3). Sollte dies eintreten zeichnet die Anwendung weiterhin keine Umgebungsgeräusche auf. Anderenfalls (siehe Zeile 5) wird die Berechtigung jetzt angefragt. Abbildung 6.15 zeigt diese Anfrage. Nach Bestätigung

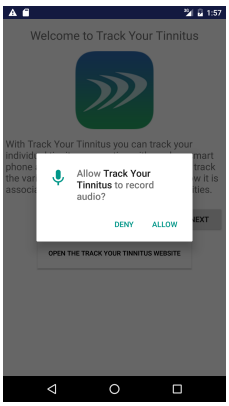

Abbildung 6.15: Anfrage nach benötigter Berechtigung

bzw. Ablehnung dieser Benachrichtigung erhält die Methode *onRequestPermissions-Result()* das Ergebnis zurück. In Abhängigkeit dieses Ergebnisses wird die Aufnahme

über das Mikrofon zugelassen oder auch nicht. Dies wird letztendlich in die Notification Shared Preference zurückgeschrieben.

6.2.3 Entscheidung über den Erhalt der eingesetzten Drittbibliotheken

In Kapitel 2.4 wurden bereits die bis dahin eingesetzten Drittbibliotheken in der Android Anwendung von Track Your Tinnitus vorgestellt. In diesem Kapitel wird beschrieben, für welchen Zweck diese Drittbibliotheken eingesetzt wurden und ob es sich weiterhin lohnt, diese eingebunden zu lassen.

ActionBarSherlock

In der momentanen Track Your Tinnitus Applikation wird eine minimum Android Version von 2.1 gefordert. In dieser Version steht die in Android 4 eingeführte Actionbar jedoch noch nicht zur Verfügung. Eine Actionbar dient als Navigationselement zwischen unterschiedlichen Ansichten in einer Android Applikation. Abbildung 6.16 zeigt den Einsatz von *ActionBarSherlock* [14] in Track Your Tinnitus. In der Hauptansicht der Anwendung

Abbildung 6.16: ActionBarSherlock in Track Your Tinnitus

wurde in der Actionbar ein kleiner Indikator (links in Blau gestaltet), das offizielle Icon von Track Your Tinnitus, sowie die Überschrift der momentanen Ansicht angezeigt. In der Übersichtsseite der Ergebnisse wurde zusätzlich ein Dropdown Menü hinzugefügt, um zwischen den Ansichten *Diagramme* und *Timeline* wechseln zu können (siehe Abbildung 6.17). Seitdem Android 2.x nicht mehr offiziell unterstützt wird in Track Your Tinnitus,

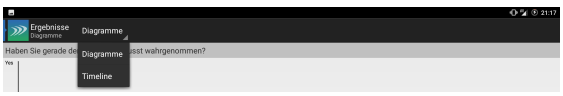

Abbildung 6.17: ActionBarSherlock mit zusätzlichen Menü

wird diese Bibliothek nicht mehr benötigt. Dieses Element befindet sich nun als Build-In Element in der Offiziellen Android API. (Siehe Kapitel 6.2.7 für die Umsetzung)

SlidingMenu

Die Bibliothek *SlidingMenu* [6] wird in der Track Your Tinnitus Applikation eingesetzt um ein modernes seitliches Navigationsmenü einblenden lassen zu können. Abbildung 6.18 zeigt die Verwendung dieser Bibliothek in der Applikation. Um dieses Navigationsmenü

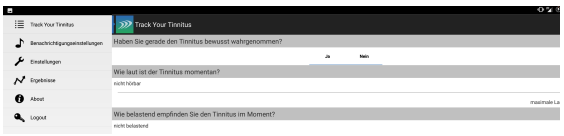

Abbildung 6.18: Einsatz von SlidingMenu mit ActionBarSherlock

zu öffnen, muss der Nutzer auf den kleinen blauen Indikator in der linken Ecke der Actionbar tippen. Das Navigationsmenü klappt auf und der Nutzer kann zwischen weiteren Ansichten der Anwendung wechseln.
Auch diese Bibliothek wird nicht mehr benötigt, da dieses Element nun als fester Bestandteil der Android API seit Version 4.x enthalten ist. Die Umsetzung wird in Kapitel 6.2.7 erklärt.

SegmentedControl

SegmentedControl [15] ist die Implementierung für segmentierte Steuerelemente. In der Track Your Tinnitus Applikation wird diese Bibliothek lediglich für eine Ja/Nein Frage im Fragebogen der Tinnitusschwankung eingesetzt. Abbildung 6.19 zeigt das SegmentControl Element. Dabei handelt es sich um eine benutzer- und touchfreundlichere Version

Abbildung 6.19: SegmentControl Button in Track Your Tinnitus

des Radiobuttons von Android. Während einer Besprechung zum Stand der aktuellen Masterarbeit wurde entschieden, dass für ein einziges Element in der Applikation nicht eine extra Drittbibliothek eingesetzt werden muss. Stattdessen werden die regulären Radiobuttons von Android verwendet.

Android Switch Backport

Die Bibliothek *Android Switch Backport* [16] dient zur Darstellung eines Switch-Buttons. Mit diesem Button lassen sich zwei Zustände darstellen Ein / Aus. Wie in Abbildung 6.20 und 6.21 dargestellt, findet dieser Button Anwendung in zwei Fällen. Erlaubnis der Geräuschpegelmessung (Erlauben / nicht Erlauben) und Senden von Benachrichtungen (Ja / Nein). Seit der Einführung von Android 4 wurde auch dieses Steuerelement in die

Abbildung 6.20: Switch-Button in den Notifications

Abbildung 6.21: SegmentControl Button in Track Your Tinnitus

Android API unter dem Namen *Switchpreference* eingeführt. Somit ist diese Bibliothek ebenfalls nicht mehr nötig.

Android Async HTTP

Der *Android Async HTTP Client* [17] ist eine Bibliothek um HTTP-Requests zu senden und Responses zu verarbeiten. Dies ist in der Track Your Tinnitus Applikation ein sehr häufiger Prozess. Dieser findet beim Login der Anwendung statt, beim synchronisieren der Antworten und Fragen zu einem Fragebogen und beim Senden der Antworten des Tinnitusüberwachungsfragebogens. Ein asynchroner HTTP Request wird mit dem Android Async HTTP Clientwie in Abbildung 6.10 gelistet, implementiert.

```
1  Requestparams params = new RequestParams();
2  params.put('example', 3);
3  AsyncClient client = new AsyncClient()
4  client.get("www.123.de/status", params, new JsonHttpResponseHandler()
       {
5              @Override
6              public void onSuccess(int statusCode, Header[] headers,
                   JSONObject response) {
7                  // Anfrage war erfolgreich
8              }
9              @Override
10             public void onFailure(int statusCode, Header[] headers,
                   JSONObject failure) {
11                 // Anfrage fehlgeschlagen
12             }
13         });
```

Quelltext 6.10: Asynchroner HTTP-Request mit Android Async HTTP

Mit dieser Bibliothek ist es sehr einfach einen GET / POST Request an einen Server zu schicken und das Ergebnis zu verarbeiten. Android selbst bietet zwar auch eine Möglichkeit Requests zu senden, diese sind aber deutlich komplexer und komplizierter zu implementieren. [8] zeigt ein vollständiges Beispiel, wie man Requests ohne diese Bibliothek sendet und verarbeitet. Der Einsatz von Android Async HTTP vereinfacht hier das Arbeiten und ist auch wartungsfreundlicher. Daher wird diese Bibliothek weiterhin eingesetzt.

6.2.4 Erster Aufruf der App und Teilnahme an einer klinischen Studie

Nach erfolgreichem installieren der App und erstmaligen Ausführen, wird überprüft, ob der Nutzer bereits angemeldet war. War er das noch nicht, öffnet sich die *Introduction-FragmentActivity*. Diese Activity beinhaltet insgesamt vier Fragmente. Abbildung 6.22 stellt davon drei dar.

Der User wird in (1) mit einem Textfragment und dem dazugehörigen Track Your Tinnitus Logo begrüßt. Mit einem klick auf „next“ landet er im nächsten Fragment. In (2) muss dieser seine Logindaten angeben. Sollten diese korrekt verifiziert worden sein, erhält der Nutzer (3) zu sehen. Dies ist ein neues Fragment in dieser Activity. Hier erhält der Nutzer nun die Möglichkeit zwischen zwei Anwendungen zu wählen. Er hat die Auswahl

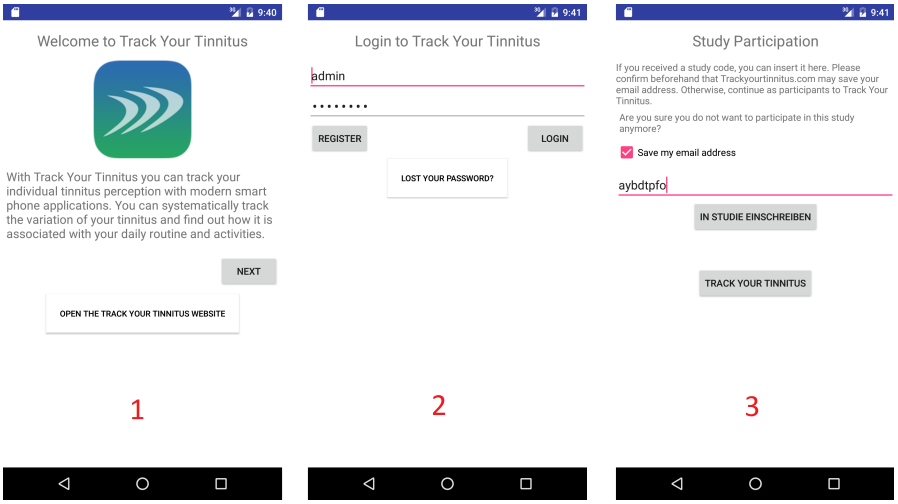

Abbildung 6.22: IntroductionFragmentActivity mit drei Fragmenten

zwischen "*In Studie einschreiben*", oder die Anwendung von "*Track Your Tinnitus*ßu starten.

Um an einer Studie teilnehmen zu können, war es eine Anforderung dies nur zu ermöglichen, wenn der Nutzer damit einverstanden ist, dass Track Your Tinnitus seine E-Mail Adresse speichern darf. Dazu muss der Nutzer die initial nicht ausgewählte Checkbox bestätigen. Erst dann ist er in der Lage einen Studiencode einzugeben. Folgendes Listing beschreibt den Quellcode um diese Anforderung zu realisieren:

```
1  CheckBox agreementCheckbox = (CheckBox) rootView.findViewById(R.id.
       register_save_mail_checkbox);
2          EditText studiencode = (EditText) rootView.findViewById(R.id.
               register_seminarcode);
3          agreementCheckbox.setOnCheckedChangeListener(new
               CompoundButton.OnCheckedChangeListener() {
4              @Override
5              public void onCheckedChanged(CompoundButton buttonView,
                   boolean isChecked) {
6                  if (isChecked) {
7                      studiencode.setEnabled(true);
8                      studybtn.setOnClickListener(new View.
                           OnClickListener() {
9                          @Override
10                         public void onClick(View v) {
11                             if (!studiencode.getText().toString().
                                   isEmpty()) {
12                                 studybuttontouched(v);
13                             } else {
```

```
14                                  }
15                              }
16                          });
17                      } else {
18                          studiencode.setText("");
19                          studiencode.setEnabled(false);
20                          studybtn.setEnabled(false);
21                      }
22                  }
23              });
```

Quelltext 6.11: Auszug aus SelectAppModeFragment

Sollte die *agreementCheckbox* einen *onCheckedChange* Event empfangen, wird überprüft, ob der neue Status der Checkbox *checked* ist, oder nicht. Sollte dies der Fall sein, wird das Studiecode-Textfeld freigeschalten und auf den Studienbutton (In Studie einschreiben) wird ein onClicklistener gesetzt. Sollte die Checkbox nicht mehr aktiviert sein, wird der Text im Studiencode Textfeld zurückgesetzt und der Button auf inaktiv gesetzt.

Wenn nun der User die *agreementCheckbox* auswählt und den erhaltenen Studiencode korrekt eingegeben hat, kann dieser den Button *In Studie einschreiben* betätigen. Hierbei wird ein HTTP Request auf folgende URI gesendet:

```
1 http://trackyourtinnitus.com/api/studies?access_token=
      access_token_des_nutzers&study_code=aybdtpfo
```

Quelltext 6.12: URI zur Anfrage nach Studie

Als erstes wird nun durch die Applikation geprüft, ob eine Studie mit dem eingetragenen Studiencode existiert und ob diese Studie schon gestartet wurde. Sollte dies nicht der Fall sein, wird der Nutzer mit einem Hinweiß darauf informiert.

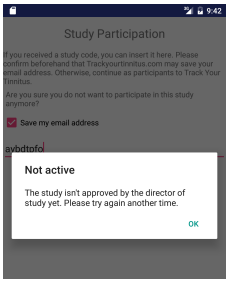

Abbildung 6.23: Hinweiß Studie nicht aktiv

Dies wurde durch einen HTTP Request ausgelöst der als Response eine Fehlermeldung enthalten hat. Folgende Fehler können dabei auftreten:

- Studie *nicht aktiv* (siehe Abbildung 6.23)
- Studie wurde *beendet*
- Die Studie mit dem eingegebenen Studiencode kann nicht gefunden werden

Sollte dies nicht zutreffen, wird der Studienteilnehmer in die Studie eingeschrieben. Dabei werden folgende Schritte durchgeführt:

- Durch das als Parameter mitgesendete *access_token*, das der Nutzer bei der Anmeldung an Track Your Tinnitus erhalten hat, wird der User auf dem Webserver identifiziert.
- Die *Metadaten* des gefundenen Nutzers enthalten das Attribute *study_id*. Dieses Attribut wird auf die Studien ID gesetzt, die auf den Studiencode zutrifft.
- Die Metadaten des Nutzers werden gespeichert.
- Als Response wird nun das Studienmodel zurück an das Smartphone geschickt.

Dieser Response enthält sämtliche Attribute der Studie. Diese *Studiendetails* werden nun in die Sharedpreference Datei *TrackYourTinnitusToken* geschrieben. Listing 6.13 beschreibt das vorgehen dabei:

```
1  private void downloadStudyDetails(RequestParams params) {
2          String countryCode = Locale.getDefault().getCountry();
3          String path = "api/study";
4          if (countryCode.equals("DE")) {
5              params.put("local", "true");
6              path = "de/api/study";
7          }
8          RestClient.get(path, params, new JsonHttpResponseHandler() {
9              @Override
10             public void onSuccess(int i, JSONArray jsonArray) {
11                 try {
12                     JSONObject response = jsonArray.getJSONObject(0);
13                     SharedPreferences settings = getActivity().
                           getSharedPreferences(
14                             PREFS_NAME, 0);
15                     SharedPreferences.Editor editor = settings.edit()
                           ;
16                     editor.putInt("id", response.getInt("id"));
```

```
17                      editor.putString("titel", response.getString("
                            title"));
18                      editor.putString("description", response.
                            getString("description"));
19                      switch (response.getString("scheduling_type")) {
20                          case "TYT":
21                              editor.putString("scheduling", "TYT");
22                              break;
23                          case "daily":
24                              editor.putString("scheduling", "daily");
25                              editor.putString("time", response.
                                    getString("configuration"));
26                              break;
27                          case "weekday":
28                              editor.putString("scheduling", "weekday")
                                    ;
29                              JSONArray timeanddays = response.
                                    getJSONArray("configuration");
30                              for (int y = 0; y < timeanddays.length();
                                     y++) {
31                                  JSONObject object = timeanddays.
                                        getJSONObject(y);
32                                  Iterator<String> iterator = object.
                                        keys();
33                                  while (iterator.hasNext()) {
34                                      String uhrzeit2 = iterator.next()
                                            ;
35                                      editor.putString(uhrzeit2, object
                                            .getString(uhrzeit2));
36                                  }
37                              }
38                              break;
39                      }
40                      editor.apply();
41                  } catch (JSONException e) {
42                      e.printStackTrace();
43                  }
44                  loadRegisterQuestions();
45              }
46              @Override
47              public void onFailure(Throwable throwable, JSONObject
                    errorResponse) {
48                  // show errorResponse in DialogAlert;
49              }
50          });
51      }
```

Quelltext 6.13: Studiendetails in SharedPreferences Datei schreiben

Zuerst muss überprüft werden, welche Sprache auf dem Smartphone eingestellt ist. Sollte das Smartphone auf Deutsch eingestellt sein, wird nach dem Studylocalizationmodel

gefragt. Um die übersetzte Version der Studie zu erhalten, wird darauf die Anfrage URI geändert in:

```
1 http://trackyourtinnitus.com/de/api/studies?study_code=aybdtpfo
```

Quelltext 6.14: URI zur Anfrage nach der übersetzten Studie

Dies enthält statt dem englischem Titel- und Beschreibungstext den Deutschen. Das Ergebnis wird nun in eine SharedPreference Datei geschrieben. Dazu zählen die ID, der Titel, Beschreibung sowie die konfigurierten Zeitpunkte für Notifications (siehe Kapitel 6.2.8 für weitere Details). Im Anschluss daran wird ein Request für Registerquestions gesendet. Dieser findet über die folgende URI statt:

```
1 http://trackyourtinnitus.com/api/registerquestionnaire?study_code=
    aybdtpfo
```

Quelltext 6.15: URI zur Anfrage nach Registerquestionnaires einer Studie

Das Ergebnis wurde in Listing 6.6 gezeigt. Je nachdem welche Sprache auf dem Smartphone eingestellt ist, erhält die Callback-Methode die deutschen oder englischen Texte. Zusammen mit den Register Fragebögen werden auch die dazugehörigen Registerfragen, in der jeweilgen angefragten Sprache, übertragen. Zum speichern dieser Fragebögen wird zunächst mit dem *RegisterquestionnaireSQLiteOpenHelper* eine Verbindung zur SQLite Datenbank hergestellt. Anschließend wird für jeden Fragebogen ein Objekt der Klasse *Registerquestionnaire* erstellt und die nötigen Attribute, ID, Studien ID, Titel und Beschreibung gesetzt. Mit der Funktion *insertIntoDatabase* der Klasse wird der Fragebogen in der Tabelle *registerquestionnaire* gespeichert. Das gleiche geschieht auch bei den Register Fragen. Für jede Frage in einem Fragebogen wird zunächst ein Objekt der Klasse *RegisterQuestion* erstellt. Diesem Objekt werden die Informationen Registerquestion ID, Studien ID, Fragebogen ID, Fragentext, Antworttyp und Konfiguration zugewießen und anschließend mit der Methode *insertIntoDatabase* in die SQLite Datenbank unter der Tabelle *registerquestion* gespeichert. Sind alle Fragebögen mit den zugehörigen Fragen abgearbeitet, sendet die Applikation einen weiteren HTTP-Request um sämtliche, der Studie zugeordneten Statequestions zu erhalten. Im Response erhält die Callback-Methode das JSON-Array mit den zugehören Statequestions. Auch für diese werden jeweils ein Objekt der Klasse *Statequestion* erzeugt und diesem die nötigen Attribute Statequestion ID, Studien ID, Fragentext (englisch oder deutsch, abhängig

der eingestellten Smartphone Sprache), Antworttyp und Konfiguration zugewießen. Die Methode *insertIntoDatabase* der Klasse Statequestion speichert die State Frage in der Tabelle *statequestion*. Zuletzt wird noch ein Request für die evtl. vorhandenen Register Antworten des Nutzers gesendet. Bei Empfang von Antworten wird ein Objekt der Klasse *Registeranswer* erstellt. Registeranswer erhalten Answer ID, Studien ID, Fragebogen ID, Fragen ID, Antwort Text, submitted vom Response. *submitted* ist ein Flag in der Klasse Antwort, das beschreibt, ob die Antwort schon aus der App heraus übertragen wurde.

6.2.5 Ausfüllen eines Registerquestionnaires zur Studie

Als nächstes wird überprüft, ob es noch offene Fragen in einem Register Fragebogen gibt. Listing 6.16 verdeutlicht dies.

```
1  private void loadWelcomeScreenForStudy() {
2          RegisterquestionnaireSQLiteOpenHelper helper = new
               RegisterquestionnaireSQLiteOpenHelper(getActivity());
3          Integer[] unansweredQuestionnaires = helper.isSubmitted(
               helper.getReadableDatabase());
4          helper.close();
5          if (unansweredQuestionnaires.length != 0) {
6              Intent i = new Intent(getActivity().getApplication(),
                   QuestionnairesStudyListActivty.class);
7              startActivity(i);
8              getActivity().finish();
9          } else {
10             Intent i = new Intent(getActivity().getApplication(),
                   WelcomeMessageStudyActivity.class);
11             startActivity(i);
12             getActivity().finish();
13         }
14     }
```

Quelltext 6.16: Sourcecode zum Überprüfen offener Fragen

Zuerst wird eine Verbindung zur Datenbank hergestellt. Die Funktion *isSubmitted* der Klasse *RegisterquestionnaireSQLiteOpenHelper* schaut in der Tabelle Registerquestions, ob zu den in der Studie vorhandenen Register Fragebögen noch unbeantworte Register Fragen vorhanden sind. Diese werden ermittelt durch das in der Klasse *RegisterAnswer* vorhandene Attribut *submitted*. Ist dieser Wert gleich 0 gilt die Frage, referenziert durch das Attribut *registerquestion_id*, als noch nicht beantwortet. Alle ID's

der Fragebögen die unbeantwortete Fragen enthalten werden zurückgegeben und befinden sich nun in der Variable *unansweredQuestionnaires*. Sollte die Anzahl an Elementen in diesem Array ungleich 0 sein, wird die Activity *QuestionnairesStudyListActivty* gestartet. Dies ist die Ansicht zum Ausfüllen der Register Fragebögen bzw. Registerfragen.
Die Klasse QuestionnairesStudyListActivty ist eine Subklasse von Androids *AppCompatActivity*. Diese Klasse ermöglich es nun eine konsistente ActionBar einzubinden, die unter den Android Versionen 4, 5 und 6 identisch aussieht. Abbildung 6.24 zeigt den Einsatz dieser Bibliothek in der Ansicht zum Ausfüllen der Register Fragebögen: Der Farbcode (a) ist unter dem Itemnamen *colorPrimaryDark* definiert (#303F9F). (b)

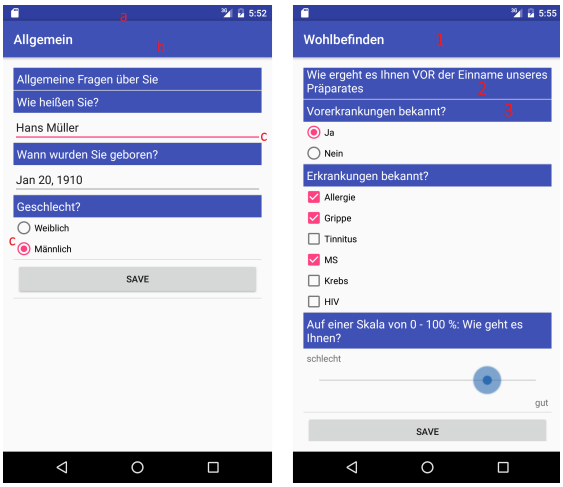

Abbildung 6.24: Aussehen der Actionbar der Supportlibrary

entspricht dem Item *colorPrimary* (#3F51B5). (c) ist unter dem Namen *colorAccent* zu definieren (#FF4081). Diese Farben können in der *..\values \color.xml* angepasst werden. Die QuestionnaireStudyListActivity enthält zudem eine Actionbar, dessen Titel dem Titel des Registerfragebogen entspricht (1). (2) ist ein einzelnes TextView Element das die Beschreibung des Registerfragebogens enthält. In (3) werden zuletzt die einzelnen Fragen des Registerfragebogen gelistet. Die Darstellung dieser Fragen ist Abhängigkeit des Antworttypes:

Abbildung 6.25: Antworttyp: Text

- Fragentext mit primaryColor als Background
- EditText Element

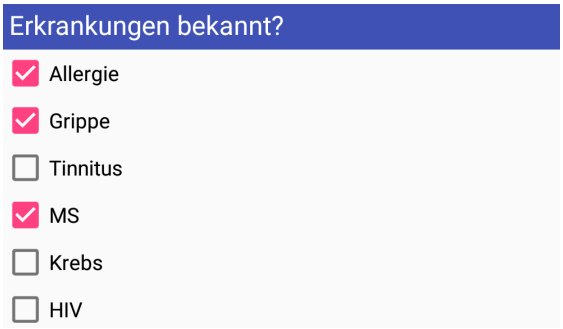

Abbildung 6.26: Antworttyp: Checkbox

- Fragentext mit primaryColor als Background
- Checkbox - Konfigurationstext

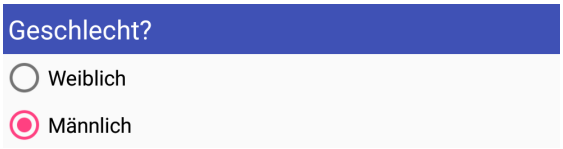

Abbildung 6.27: Antworttyp: Radiobuttons

- Fragentext mit primaryColor als Background
- Radiobutton - Konfigurationstext

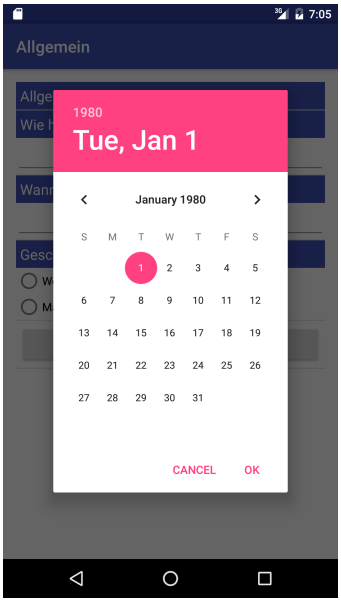

Abbildung 6.28: Antworttyp: Datepicker

- Fragentext mit primaryColor als Background
- EditText
- Bei Auswahl des Textfeldes öffnet sich das *DatepickerDialog* Element

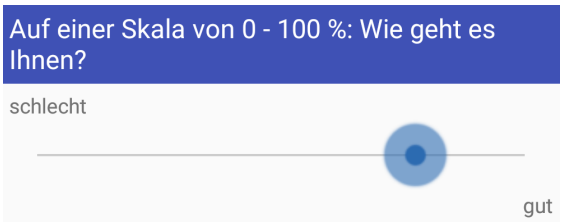

Abbildung 6.29: Antworttyp: Slider

- Fragentext mit primaryColor als Background
- Konfigurationstext - JHSeekBar - Konfigurationstext
- Entwickelt in [3]. Besitzt keinen Initialwert. Erst bei Berührung der Linie wird der Wert gesetzt.

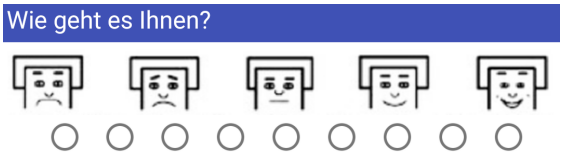

Abbildung 6.30: Antworttyp: Feeling (nur bei Statequestions)

- Fragentext mit primaryColor als Background
- 5 Point SAM Scale
- Radiobuttons

Sobald sämtliche Fragen in einem Registerfragebogen beantwortet sind, können über den Button *save* die Antworten in der Tabelle registeranswer gespeichert werden. Dazu wird zu sämtlichen Fragen jeweils ein Registeranswer Objekt erstellt, die Attribute question_id, questionnaire_id, answer gesetzt und mit der Methode *saveAnswer* der Klasse in die Datenbank gespeichert. Sollten alle Registerfragen in den Registerfragebögen beantwortet sein, öffnet sich eine neue Activity, die lediglich Informationen enthält (s. Abbildung 6.31).

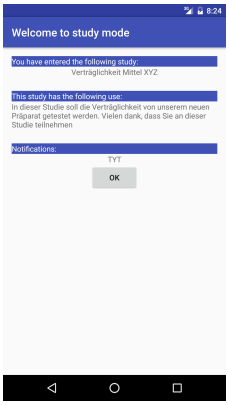

Abbildung 6.31: Informationen zur Studie

Dazu zählen nochmals der Titel der Studie, die Beschreibung und der Typ der Notifications. Mit einem Klick auf OK wird die Activity wieder beendet und die *MainActivity* wird aufgerufen.

6.2.6 Die MainAcitvity von Track Your Tinnitus

Das Hauptmenü von Track Your Tinnitus wird von der *MainActivity* dargestellt. Von hier aus ist es dem Nutzer möglich in sämtliche Bereiche der Anwendung zu gelangen. Dies wird über den neu eingeführten *NavigationDrawer* der Google Support Library [23] implementiert. Kapitel 6.2.7 geht dabei näher auf die Implementierung dieses Menüs ein. Die MainActivity hat in der *onCreate()* mehrere Aufgaben zu erfüllen.

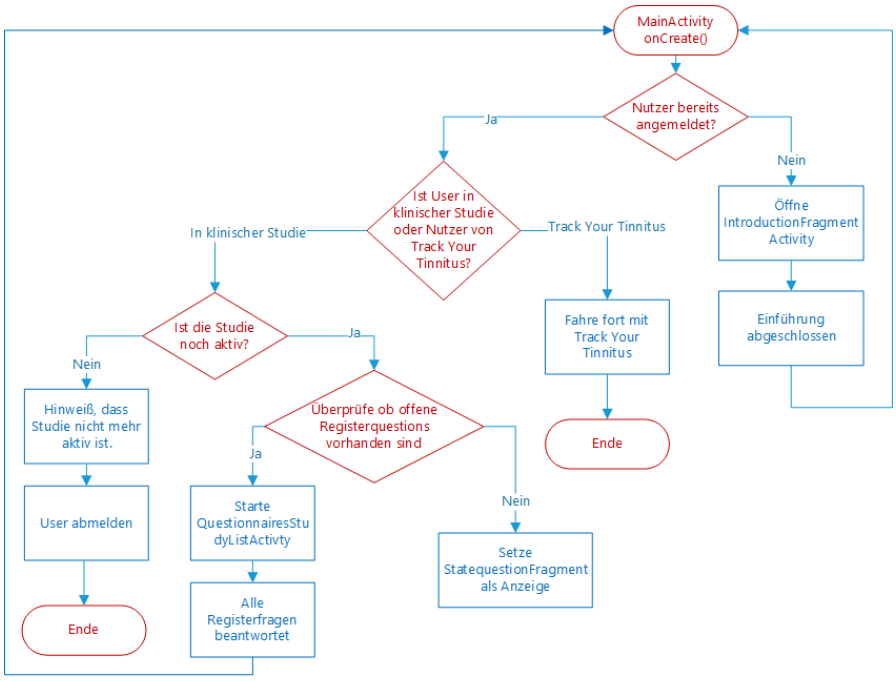

Abbildung 6.32: Flussdiagramm MainActivity

Zuerst muss überprüft werden, ob sich der Nutzer schon einmal an der Anwendung angemeldet hat. Dabei wird in der SharedPreference Datei nach einem Token namens *TrackYourTinnitusToken* geschaut. Ist dieses Token, das der Nutzer nach Anmeldung

an der App erhalten hat, nicht vorhanden, wird die *IntroductionFragmentActivty* (siehe Kapitel 6.2.4) gestartet. Andernfalls wird überprüft, ob sich der Nutzer in einer klinischen Studie befindet, oder als Track Your Tinnitus Nutzer fortfahren möchte. Ist der Nutzer in einer Studie eingeschrieben, wird durch einen HTTP-Request gefragt, ob die Studie noch aktiv ist. Ist die Studie beendet worden, wird dies dem Nutzer durch einen Hinweiß signalisiert. Der User wird abgemeldet und die Applikation schließt sich. Ist die Studie noch aktiv, wird in der Registerquestionnaire Tabelle geschaut, ob noch offene Registerquestions ausgefüllt werden müssen. Ist dies der Fall, wird von der MainActivity die Activity *QuestionnairesStudyListActivty* aufgerufen. Sollten alle Registerfragen bereits beantwortet worden sein, wird von der MainActivity das Fragment *StatequestionFragment* gesetzt. Hier ist es dem User möglich, die Statequestions auszufüllen.

6.2.7 Notification Drawer in der MainAcitvity von Track Your Tinnitus

Um ein Navigationsmenü im Hauptmenü der Anwendung zu realisieren ist die *MainActivity*, wie schon die QuestionnairesStudyListActivty, eine Subklasse der *AppCompatActivity*. Zudem muss das Layout der MainActivity aus dem Element *DrawerLayout* bestehen. Diesem *DrawerLayout* wird als Unterelement eine *NavigationView* hinzugefügt. Diese NavigationView besteht dabei aus einem Header sowie dem Menü. Abbildung 6.33 zeigt die MainActivity mit geschlossenem und geöffnetem Navigationsmenü

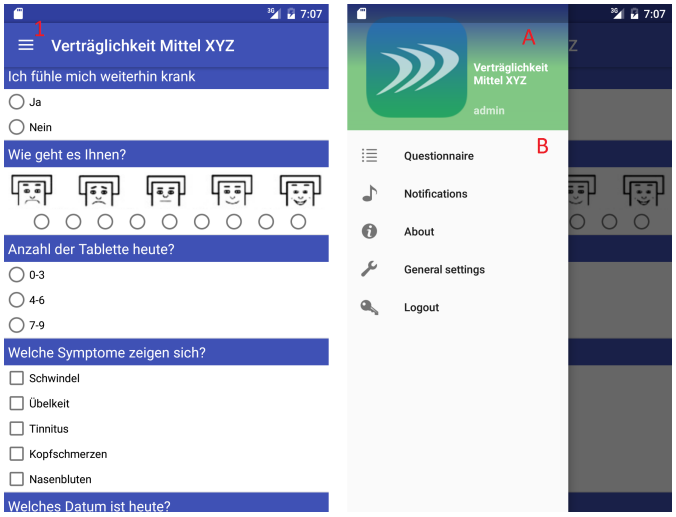

Abbildung 6.33: Navigationdrawer in der MainActivity

Klickt der Nutzer auf den Menübutton (1) in der linken Ecke der Actionbar, öffnet sich das Navigationsmenü. Im Hintergrund ist der eigentliche Inhalt der Aktivity verdunkelt. Dieses Menü besteht aus 2 Teilen. Dem Header (A) und der Menüliste (B). Im Header enthalten sind:

- *Hintergrund* im Farbverlauf von Blau zu Grün
- Track Your Tinnitus *Logo*, leicht transparent
- *Titel* der Studie
- *Name* des angemeldeten Nutzers

In der Menüleiste sind die Menüpunkte aus Track Your Tinnitus übernommen worden. Ein Menüpunkt besteht aus einem Icon und dem Menüpunktnamen. Durch das berühren des jeweiligen Menüpunkts öffnen sich folgende Fragmente, die durch die MainActivity aufgerufen werden.

- *Questionnaire*: Ansicht über die Statequestions der klinischen Studie
- *Notifications*: Da diese Studie als Benachrichtigungstyp „Track Your Tinnitus“ hat, wird dieses Menüitem angezeigt. Sollte der Studienleiter einen anderen Benachrichtungstyp gewählt haben (Daily oder Weekday), ist dieser Menüpunkt nicht

vorhanden. Der Studienteilnehmer hat keine Möglichkeit seine Benachrichtigungen selbst zu bestimmen.

- *About*: Weiter Informationen zu Track Your Tinnitus.
- *Einstellungen*: Klingelton, Privatsphäreneinstellungen und, falls der Nutzer in einer Studie eingeschrieben ist, die Möglichkeit diese Studie zu verlassen.
- *Logout*: Abmelden aus Track Your Tinnitus

Der Menüpunkt Ergebnisse ist bei der Teilnahme an klinischen Studien nicht vorhanden. Den Nutzern der normalen Track Your Tinnitus Anwendung wird dieser Menüpunkt angezeigt. Das Ausfüllen und Speichern der Statequestions ist identisch zu den Registerfragen in der *QuestionnairesStudyListActivty*. Nachdem der Studienteilnehmer sämtliche Statefragen beantwortet hat, übernimmt der Button *save* das Speichern der Antworten in die Tabelle Stateanswers. Dazu wird für jede Frage ein Objekt der Klasse Stateanswer erstellt und die jeweiligen Attribute gesetzt. Dazu gehören die Studien ID, sowie die Antwort auf die Frage. Sind alle Antworten gespeichert, wird die Anwendung geschlossen und ein Uploadprozess wird angestoßen (siehe Kapitel 6.2.9 für weitere Details).

6.2.8 Setzen der konfigurierten Notifications in der App

In Kapitel 6.1.2 wurden die Möglichkeiten beschrieben, die ein Studienleiter hat um seine Studienteilnehmer zu Informieren, wann die Statefragen ausgefüllt werden sollen. Um dies zu realisieren sind mehrere Komponenten erforderlich:

- AlarmManager
- PendingIntent
- NotificationManager

Mit einem AlarmManager in Android, ist es möglich Aufgaben an einem bestimmten Zeitpunkt zu erledigen. Dieser AlarmManager ist dabei nicht abhängig vom Lebenszyklus einer Aktivity und kann daher auch ausgeführt werden, wenn die Applikation nicht läuft

oder das Smartphone benutzt wird. Der AlarmManager wird in Track Your Tinnitus von der Klasse *SuperNotificationManager* erstellt.

```
15 AlarmManager am = (AlarmManager) mContext.getSystemService(Context.
       ALARM_SERVICE);
16 am.set(AlarmManager.RTC_WAKEUP, milliseconds, getPendingIntent(
       request_code));
```

Quelltext 6.17: AlarmManager in Android

Dem AlarmManager müssen insgesamt drei Parameter übergeben werden. Der Parameter *AlarmManager.RCT_WAKEUP* beschreibt, dass das Smartphone beim Eintreten des Alarms aus dem Ruhezustand geholt werden soll. *milliseconds* ist der Zeitpunkt in Millisekunden, wann ein Alarm ausgeführt werden soll. Der Dritte Parameter gibt den PendingIntent an, der bei dem Eintreffen des Alarms ausgeführt werden soll. Dieser PendingIntent ist in diesem Falle ein Broadcast der an das Betriebssystem gesendet wird. Dieses öffnet die App und ruft in der Klasse *NotificationReceiver* die Methode *onReceive()* auf. In dieser Methode wird mittels dem NotificationManager eine Notification gebaut, die dem Nutzer angezeigt wird.

```
17 NotificationCompat.Builder mBuilder =
18             new NotificationCompat.Builder(context)
19             .setSmallIcon(getNotificationIcon())
20             .setContentTitle("Track Your Tinnitus")
21             .setContentText("Bitte fuellen Sie den Fragebogen fuer
                   Ihre Studie aus!")
22             .setTicker(context.getResources().getString(R.string.
                   notification_ticker_text))
23     Intent resultIntent = new Intent(context, MainActivity.class);
```

Quelltext 6.18: NotificationManager

setSmallIcon bekommt aus der Methode *getNotificationIcon()* ein Icon zurück geliefert. Dieses Icon ist abhängig von der installierten Android Version auf dem Smartphone. Ab Android 5 werden nur noch weiße Icons dargestellt (siehe Abbildung 6.34). Dieses Icon enthält die Sillhouette des Track Your Tinnitus Logos. Sollte Android 4 installiert sein, wird das Track Your Tinnitus Logo dem Nutzer angezeigt.

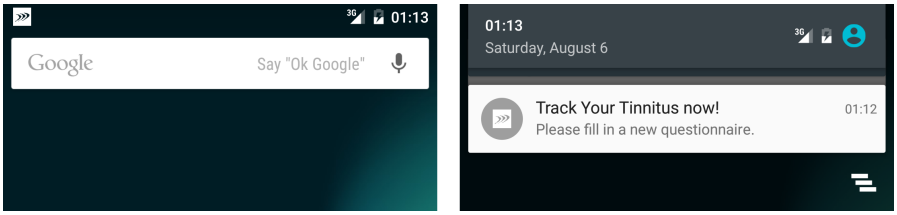

Abbildung 6.34: Notification von Track Your Tinnitus

In Zeile 7 wird beschrieben, welche Activity gestartet werden soll, wenn der Studienteilnehmer auf die Notification tippt. Das Betriebssystem hat nun durch den Broadcast die Erkenntnis, zu welchem Zeitpunkt ein Alarm von der Track Your Tinnitus Applikation abgesetzt wird. Diese Zeitpunkte speichert das System bis zum nächsten Neustart des Gerätes. Sollte das Smartphone neugestartet werden, gehen diese Alarme verloren. Um dem vorzubeugen, mussten weitere Komponenten implementiert werden:

- Die Applikation benötigt eine weitere Berechtigung. Die Permission *android.permission.RECEIVE_BOOT_COMPLETED* erlaubt es der Applikation einen Broadcast der vom Betriebssystem gesendet wird, nachdem es neugestartet wurde, das Flag *ACTION_BOOT_COMPLETED*, zu empfangen.
- Der Broadcastreceiver *BootCompleteReceiver* wurde in der Manifest-Datei der Applikation hinterlegt, der ausgeführt wird, sobald das ACTION_BOOT_COMPLETED Flag empfangen wird.
- In der *onReceive()* Methode dieses Receiver wird nun der *CustomNotificationService* gestartet, der dafür sorgt, dass die AlarmManager wieder beim Betriebssystem bekannt sind.

Somit kann der Studienteilnehmer auch sein Smartphone neustarten, ohne dass Notificatinos verloren gehen. Es sei denn, in dem Zeitpunkt des Neustarts wäre ein Alarm geplant gewesen.

6.2.9 Android Services zum Übertragen der Antworten des Users an das Backend

Service in Android Applikationen dienen dazu Aufgaben im Hintergrund zu erledigen. Dabei findet keinerlei Interaktion mit dem Nutzer des Smartphones statt. Services können in Android zwei verschiedene Ausprägungen haben:

- Started (Die Anwendung startet den Service explizit von sich aus mittels ’ßtartService()")
- Bound (Die Anwendung bindet sich an den Service mittels "bindService()")

In der Track Your Tinnitus Applikation werden Services zur Übertragung der Antworten des Nutzers auf Fragebögen genutzt. Dieser Upload wird einmal beim Starten wie auch beim Schließen der Anwendung gestartet. Um dieses zu realisieren müssen die jeweiligen Methoden der MainActivity überschrieben werden (siehe Listing 6.19).

```
24 @Override
25     public void onStart() {
26     ...
27     Intent i = new Intent(this, StudyUploadService.class);
28                 this.startService(i);
29     ...
30     }
31 @Override
32     public void onPause() {
33     ...
34     Intent i = new Intent(this, StudyUploadService.class);
35                 this.startService(i);
36     ...
37     }
```

Quelltext 6.19: Starten eines Service beim Starten / Schließen der App

Mit diesem Aufruf wird nun im StudyUploadService die Methode „onStartCommand()“ aufgerufen. In dieser Funktion werden nun sämtliche Antworten, die bis jetzt noch nicht synchronisiert sind, zum Backend übertragen. Listing 6.20 zeigt den Code zum Synchronisieren von Antworten:

```
38 private void uploadStateAnswers() {
39         final StateAnswerSQLiteOpenHelper helper = new
               StateAnswerSQLiteOpenHelper(getApplicationContext());
40         SQLiteDatabase database = helper.getReadableDatabase();
41         StateAnswer[] answers = helper.getUnsyncedAnswers(database);
```

```
42          database.close();
43          for (final StateAnswer standardAnswer : answers) {
44              SharedPreferences settings = getSharedPreferences(
                    MainActivity.PREFS_NAME, 0);
45              String accessToken = settings.getString(MainActivity.
                    PREFS_NAME, null);
46              RequestParams params = new RequestParams();
47              params.put("access_token", accessToken);
48              params.put("data", standardAnswer.getJsonRepresentation()
                    );
49              running++;
50              RestClient.post("api/stateanswers", params, new
                    JsonHttpResponseHandler(){
51                  @Override
52                  public void onSuccess(JSONObject response) {
53                      SQLiteDatabase database = helper.
                            getWritableDatabase();
54                      standardAnswer.setSendSuccess(database, 1);
55                      database.close();
56                      running--;
57                      if(running <= 0){
58                          stopSelf();
59                      }
60                  }
61                  @Override
62                  public void onFailure(Throwable throwable, JSONObject
                            errorResponse) {
63                      running--;
64                      if(running <= 0){
65                          stopSelf();
66                      }
67                  }
68              });
69          }
70      }
```

Quelltext 6.20: Upload State Answers

Zuerst wird eine Verbindung zur Datenbank hergestellt. Mit der Methode *getUnsyncedAnswers* werden sämtliche Antworten gesammelt, die im Attribut *sentsuccess* den Wert 0 enthalten. Jede Antwort die in dem Array *answers* steht, wird nun ein HTTP-Request durchgeführt und die Antwort als JSON-Representant zum Backend geschickt.

```
71 [{ data : [{"registerquestionnaire_id":2, "registerquestion_id":4,"
       study_id":4,"answer":"Ja"}]}]
```

Quelltext 6.21: JSON Objekt eines RegisterAnswer Models

```
72 [{data : [{"statequestion_id":4,"study_id":4,"answer":"Nein"}]}]
```

Quelltext 6.22: JSON Objekt eines Stateanswer Models

Im Backend werden die einzelnen Antworten des Nutzers in der Tabelle *register_answers*, bzw. *state_answers* gespeichert. Dabei werden folgende Schritte durchgeführt:

- Anhand dem in der URI mitgesendeten Paramater *access_token*, den der User bei der Anmeldung bei Track Your Tinnitus erhalten hat, wird der User auf dem Webserver identifiziert.
- Für jeden in *data* enthaltenen JSON-Representant einer Antwort, wird auf dem Webserver ein *Registeranswer* bzw. *Stateanswer* Model erstellt.
- Diesem Model werden die Attribute aus dem JSON-Representant zugewießen.
- Das Model wird mit der Methode *save()* aus *Eloquent* in die Datenbank geschrieben.
- Damit diese Antworten nicht erneut gesendet werden, wird in der Datenbank der App das Flag *sentSuccess* in der Tabelle *registeranswer*, bzw. *stateanswer*, auf 1 gesetzt.

7

Anforderungsabgleich

In diesem Kapitel werden die Anforderungen, die in Kapitel 5 definiert wurden, abgeglichen. Auch hier werden diese Anforderungern in funktionale und nicht-funktionale Anforderungen unterschieden.

7.1 Funktionale Anforderungen

Dieser Abschnitt zeigt den Abgleich der funktionalen Anforderungen aus Kapitel 5.1 an das Backend bzw. die Applikation

Nr.	Beschreibung	Umsetzung
1	Erstellen einer Studie	Anforderung umgesetzt. Sobald sich ein User in der Gruppe „Admin“ bzw. „studienadmin“ befindet, kann dieser klinische Studien erstellen.
2	Generierung eines Codes zur Studie	Anforderung umgesetzt. Der Studienleiter kann einen Code generieren auf den seine Studie referenziert. Dieser Code besteht aus ingsgesamt 8 Zeichen (a-z, A-Z, 0-9).
3	Erstellung von Register-Fragebögen	Anforderung umgesetzt. Der Studienleiter kann sich beliebig viele Registerfragebögen generieren, die wiederum 1-n Register Fragen beinhalten.
4	Erstellung eines statischen Fragebogens zur Durchführung einer klinischen Studie	Anforderung umgesetzt.

5	Konfiguration von Benachrichtigungen	Anforderung umgesetzt.
6	Starten und Stoppen einer klinischen Studie	Anforderung umgesetzt.
7	Exportieren von Ergebnissen	Anforderung umgesetzt.
8	Überichts- und Informationsseite zu einer klinischen Studie	Anforderung umgesetzt.
9	Eine klinische Studie darf nicht mehr veränderbar sein, sobald diese gestartet ist	Anforderung umgesetzt.
10	Proband soll via App an Studie teilnehmen können	Anforderung umgesetzt.
11	Register- und Statefragebögen sollen in der Android App ausfüllbar sein	Anforderung umgesetzt.
12	Bei Custom-Notifications soll ein ändern der Benachrichtigungen nicht möglich sein	Anforderung umgesetzt.
13	Android App soll Offline Funktion bieten	Anforderung umgesetzt.

Tabelle 7.1: Funktionale Anforderungen - Abgleich

7.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen aus Kapitel 5.2 werden in folgender Tabelle abgeglichen.

Nr.	Beschreibung	Umsetzung
1	Der Betrieb von Track Your Tinnitus darf keinerlei Auswirkungen der neuen Variante verspüren	Anforderung wurde komplett umgesetzt. Track Your Tinnitus kann neben klinischen Studien seinen vollen Betrieb weiterhin fortsetzten.
2	Android Version der App soll dem Styleguide des Material Designs nachgebaut werden.	Anforderung umgesetzt.
3	Teilnahme an Studie unter der Prämisse, dass die E-Mail Adresse des Probanten gespeichert werden darf	Anforderung wurde umgesetzt. An einer klinischen Studie kann nur teilgenommen werden, wenn der Proband erlaubt, dass seine E-Mailadresse gespeichert werden darf.
4	Die Anwendung soll Mehrsprachigkeit unterstützen	Anforderung umgesetzt. Momentan können die Fragebögen in Deutsch und Englisch ausgefüllt werden.
5	Drittbibliotheken sollen, soweit sinnvoll, von der Android App entfernt werden.	Teilweise umgesetzt. Es wurden sämtliche Drittbibliotheken bis auf den Asynchronous Http Client for Android abgeschafft.

Tabelle 7.2: Nicht-funktionale Anforderungen - Abgleich

8

Zusammenfassung und Ausblick

Im Rahmen dieser Masterarbeit ist eine erweiterte Version des Track Your Tinnitus Frameworks entstanden, dessen Anforderungen klar definiert waren. Es ist nun möglich klinische Studien mit diesem durchzuführen und auszuwerten. Dazu wurde zuerst das Backend mit dem PHP Framework Laravel 3 weiterentwickelt. Hier ist es nun möglich eigene Registerfragebögen und Statequestions anzulegen. Zudem kann man nun auch eigene Benachrichtigungen für die Apps konfigurieren. Im zweiten Schritt ging es an die Anpassung der Android Variante der App. Hier wurde zuerst auf das neue Material Design von Google umgestellt und die Drittbibliotheken der bisherigen Anwendung entfernt. Nachdem die Track Your Tinnitus Applikation in seinem Ursprungszustand komplett funktionierte, wurden die Anforderungen umgesetzt, dass klinische Studien durchgeführt werden können. Da dieses Thema meine Lieblingsgebiete in der Informatik abdeckt, nämlich Web- und Androidentwicklung, hat mir das Erarbeiten dieser Lösung zu jeder Zeit sehr viel Freude bereitet.
Nichtsdestotrotz haben wir auch hier noch keine finale Version geschaffen. Aspekte die noch betrachtet werden können, sind unter anderem:

Anpassung der iOS Variante

Klinische Studien lassen sich momentan nur mit der Android Applikation von Track Your Tinnitus durchführen. Daher ist es unabdingbar auch die iOS Version der App auf den aktuellen Stand zu entwickeln.

Update des Laravelframeworks

Als der Grundstein für das Backend mit der Diplomarbeit von Jochen Herrman 2012 gelegt wurde, war die Basis dazu das Laravel Framework in Version 3. Stand August 2016 ist dieses bereits in Version 5.2 erschienen. Es sind einige Neuerungen und Verbesserungen seitdem hinzugekommen. Dazu zählen neben verschiedensten Sicherheitsupdates auch das Laraveleigene Usermanagementsystem. Mit einem Umstieg von Laravel 3 auf Laravel 5 würden sich weitere Fremdbibliotheken aus dem Backend entfernen lassen. Eine weitere Neuerung die das Arbeiten erleichtert, ist das so genannte Route-Model-Binding. Mit diesem ist es möglich komplette Instanzen eines Models in die Routen einzuschließen anstatt nur den ID's. Ebenfalls wurde der Umgang mit der *routes.php* File sehr stark vereinfacht. Ein Update auf die aktuellste Version von Laravel vereinfacht somit das Arbeiten.

Antwortmöglichkeiten mit Sensoren des Smartphones abbilden

Heutzutage ist ein Smartphone viel mehr, als nur ein Gerät zum Telefonieren und SMS schreiben. Das Smartphone weiß beispielsweise, wo sich der Nutzer momentan befindet. Wie hell es in seiner Umgebung ist, ob sich das Smartphone momentan in ihrer Hosentasche befindet, oder ob es gerade senkrecht, waagerecht, ruhig oder zittrig gehalten wird. All das ist den Sensoren zu verdanken, die im inneren eines Smartphones mittlerweile verbaut werden. Dazu zählen unter anderem:

- Fingerabdrucksensor
- Umgebungslichtsensor
- Pulssensor
- Näherungssensor
- Beschleunigungssensor
- Gyroskop

Interessant hierbei wäre sicherlich der Einsatz von Umgebungslicht- und Pulssensoren. In einigen Highend Smartphones ist auch schon ein Sensor zur Messung der Blutsauerstoffsättigung vorhanden.

Einsatz der Applikation unter Android Wear

Ein weiteres großes Einsatzgebiet bieten so genannte Wearables. Dies sind Geräte wie Smartwaches oder Fitnessarmbänder. Sie bieten weitere Sensoren wie beispielsweiße Schrittzähler und Stockwerkzählen. Auch ein Ausfüllen der Fragebogen unter einer solchen Smartwatch wäre denkbar. Das Institut für Datenbanken und Informationssysteme der Universität Ulm hat sich dabei schon näher mit dem Einsatz von Wearables im Track Your Tinnitus - Umfeld beschäftigt [28].

Literaturverzeichnis

[1] Pryss, R., Reichert, M., Herrmann, J., Langguth, B., Schlee, W.: Mobile crowd sensing in clinical and psychological trials - a case study. In: 28th IEEE Int'l Symposium on Computer-Based Medical Systems, IEEE Computer Society Press (2015) 23–24

[2] Pryss, R., Reichert, M., Langguth, B., Schlee, W.: Mobile crowd sensing services for tinnitus assessment, therapy and research. In: IEEE 4th International Conference on Mobile Services (MS 2015), IEEE Computer Society Press (2015) 352–359

[3] Herrman, J.: Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten (2014)

[4] Probst, T., Pryss, R., Langguth, B., Schlee, W.: Emotional states as mediators between tinnitus loudness and tinnitus distress in daily life: Results from the "tracky-ourtinnitus“ application. Scientific Reports **6** (2016)

[5] Ottwel, T.: Laravel documentation - what makes laravel different? Website (2013) `https://laravel3.veliovgroup.com/docs`; abgerufen am 18. August 2016.

[6] Feinstein, J.: Slidingmenu. Website (2014) `http://jeremyfeinstein.com/SlidingMenu/`; abgerufen am 18. August 2016.

[7] Banes, C.: Appcompat v21 — material design for pre-lollipop devices! Website (2014) `http://android-developers.blogspot.de/2014/10/appcompat-v21-material-design-for-pre.html`; abgerufen am 18. August 2016.

[8] Azzola, F.: Android http client: Get, post, download, upload, multipart request. Website (2013) `https://www.javacodegeeks.com/2013/06/android-http-client-get-post-download-upload-multipart-request.html`; abgerufen am 18. August 2016.

[9] Ulm, U.: Questionsys. Website (2016) https://www.uni-ulm.de/in/iui-dbis/forschung/projekte/questionsys.html; abgerufen am 18. August 2016.

[10] Schobel, J.: mykind. Website (2016) https://www.mykind.info/; abgerufen am 18. August 2016.

[11] Diabetes-Journal, R.: meindiabetes - neue smartphone-app für menschen mit diabetes. Website (2015) http://www.diabetes-online.de/a/1676727; abgerufen am 18. August 2016.

[12] Bionetworks, S.: Mobile parkinson disease study. Website (2015) http://parkinsonmpower.org/; abgerufen am 18. August 2016.

[13] Spektrum: Lexikon der psychologie - tapping-test. Website (2010) http://www.spektrum.de/lexikon/psychologie/tapping-test/15296; abgerufen am 18. August 2016.

[14] Wharton, J.: Actionbarsherlock. Website (2012) http://actionbarsherlock.com/; abgerufen am 18. August 2016.

[15] Ferrari, B.: Segmented control button for android. Website (2012) https://github.com/bookwormat/segcontrol; abgerufen am 18. August 2016.

[16] Lubek, B.: Android switch preference backport. Website (2015) https://github.com/BoD/android-switch-backport; abgerufen am 18. August 2016.

[17] Smith, J.: Asynchronous http client for android. Website (2015) http://loopj.com/android-async-http/; abgerufen am 18. August 2016.

[18] Google: Understand the lifecycle callbacks. Website (2016) https://developer.android.com/training/basics/activity-lifecycle/starting.html; abgerufen am 18. August 2016.

[19] Statista: Anzahl der smartphone-nutzer in deutschland in den jahren 2009 bis 2016 (in millionen). Website (2016) http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/; abgerufen am 18. August 2016.

[20] International, E.: The json data interchange format. Website (2013) http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf; abgerufen am 18. August 2016.

[21] Ottwel, T.: Laravel - the php framework for web artisans. Website (2016) https://laravel.com/; abgerufen am 18. August 2016.

[22] Google: Material design - introduction. Website (2015) https://material.google.com/; abgerufen am 18. August 2016.

[23] Google: Support library. Website (2013) https://developer.android.com/topic/libraries/support-library/index.html; abgerufen am 18. August 2016.

[24] Skvorc, B.: The best php framework for 2015: Sitepoint survey results. Website (2015) https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/; abgerufen am 18. August 2016.

[25] Reichert, S.: Self-assessment manikin. Website (2009) http://www.qu.tu-berlin.de/menue/forschung/laufende_projekte/joyofuse/joy_of_use/joy_of_use/measurement_methods/sam/; abgerufen am 18. August 2016.

[26] Google: System permissions. Website (2015) https://developer.android.com/guide/topics/security/permissions.html; abgerufen am 18. August 2016.

[27] Statista: Anteil der verschiedenen android-versionen an allen geräten mit android os weltweit. Website (2016) http://de.statista.com/statistik/daten/studie/180113/umfrage/anteil-der-verschiedenen-android-versionen-auf-geraeten-mit-android-o abgerufen am 18. August 2016.

[28] Schickler, M., Pryss, R., Reichert, M., Heinzelmann, M., Schobel, J., Langguth, B., Probst, T., Schlee, W.: Using wearables in the context of chronic disorders - results of a pre-study. In: 29th IEEE Int'l Symposium on Computer-Based Medical Systems. (2016) 68–69

[29] Schickler, M., Pryss, R., Reichert, M., Schobel, J., Langguth, B., Schlee, W.: Using mobile serious games in the context of chronic disorders - a mobile game concept for the treatment of tinnitus. In: 29th IEEE Int'l Symposium on Computer-Based Medical Systems (CBMS 2016). (2016) 343–348

[30] Schickler, M., Reichert, M., Pryss, R., Schobel, J., Schlee, W., Langguth, B.: Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health. eXamen.press. Springer Vieweg (2015)

[31] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: Towards flexible mobile data collection in healthcare. In: 29th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2016). (2016) 181–182

[32] Schobel, J., Pryss, R., Schickler, M., Ruf-Leuschner, M., Elbert, T., Reichert, M.: End-user programming of mobile services: Empowering domain experts to implement mobile data collection applications. In: 5th IEEE International Conference on Mobile Services (MS 2016), IEEE Computer Society Press (2016) 1–8

[33] Schobel, J., Pryss, R., Reichert, M.: Using smart mobile devices for collecting structured data in clinical trials: Results from a large-scale case study. In: 28th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2015), IEEE Computer Society Press (2015) 13–18

[34] Schobel, J., Schickler, M., Pryss, R., Reichert, M.: Process-driven data collection with smart mobile devices. In: 10th International Conference on Web Information Systems and Technologies (Revised Selected Papers). Number 226 in LNBIP. Springer (2015) 347–362

[35] Schobel, J., Schickler, M., Pryss, R., Maier, F., Reichert, M.: Towards process-driven mobile data collection applications: Requirements, challenges, lessons learned. In: 10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps. (2014) 371–382

[36] Schobel, J., Ruf-Leuschner, M., Pryss, R., Reichert, M., Schickler, M., Schauer, M., Weierstall, R., Isele, D., Nandi, C., Elbert, T.: A generic questionnaire framework supporting psychological studies with smartphone technologies. In: XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference. (2013) 69–69

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Patrick Grau Matrikelnummer: 766309

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Patrick Grau