ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

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

Prototypische Realisierung einer mobilen Anwendung zur Unterstützung Tinnitus geschädigter Patienten und zur Durchführung Klinischer Studien

Masterarbeit an der Universität Ulm

Vorgelegt von:
Janina Strohm
janina.strohm@uni-ulm.de

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

Betreuer:
Dr. Rüdiger Pryss

2017

Fassung 11. September 2017

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Tinnitus ist eine regelrechte Volkskrankheit. 1999 waren bereits zehn Millionen Bundesbürger jährlich von Ohrgeräuschen betroffen, wovon ca. 1,5 Million mittelgradig bis unerträglich unter anhaltendem Tinnitus litten. Betroffene fühlen sich durch ihren anhaltenden Tinnitus erheblich in ihrer Lebensqualität beeinträchtigt und leiden nicht selten unter psychischen Störungen, wie Depressionen oder Angststörungen. Die individuelle Wahrnehmung des Tinnitus unterliegt im Laufe des Tages Schwankungen, die durch veränderte Umstände, wie zum Beispiel Umgebungsgeräusche oder Stress beeinflusst werden. Es liegen bislang wenig Informationen über die Zusammenhänge dieser Umstände mit den Symptomen vor.

Die Messung von Tinnitusschwankungen kann durch mobile Applikationen vereinfacht werden, indem Benutzer im Kontext Klinischer Studien zum Ausfüllen von Fragebögen aufgefordert werden. Eine in 2011 während einer Diplomarbeit realisierte Android Applikation (App) ermöglicht die Erfassung der Tinnitus Schwankungen. [1]

Diese App wird im Rahmen der vorliegenden Arbeit überarbeitet und erweitert. Dazu werden zunächst die geänderten Anforderungen identifiziert. Daraufhin wird eine geeignete Architektur ausgewählt und beschrieben. Einzelne Implementierungsschritte werden detailliert erläutert. Ein Kapitel widmet sich der ausführlichen Vorstellung und Beschreibung der Applikation aus Nutzersicht.
Abschließend wird die prototypische Realisierung anhand der Anforderungen bewertet und ein Ausblick auf zukünftige Erweiterungen gegeben.

Der mit dieser Arbeit realisierte Android Prototyp soll die neuen Erkenntnisse im Bereich der Tinnitusmessung und technische Neuerungen berücksichtigen und die Basis für eine aktuelle, einfach bedienbare App darstellen, die Patienten eine zuverlässige Unterstützung in ihrem täglichen Umgang mit dem Tinnitus bietet.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich unterstützt und zum Gelingen dieser Masterarbeit beigetragen haben.
Ich danke Prof. Dr. Manfred Reichert für die Möglichkeit, in diesem Themengebiet mitwirken zu können. Ein besonderer Dank gilt meinem Betreuer Dr. Rüdiger Pryss für die Unterstützung bei der Themenfindung und der anschließenden Betreuung der Masterarbeit.

Für die Bereitstellung der zuverlässigen REST-API, für die unzähligen Anpassungen und für die Unterstützung bei der Nutzung der API möchte ich mich ganz besonders bei Johannes Schobel bedanken.

Nicht zuletzt danke ich den anderen Studenten, die während desselben Zeitraums mit dem TrackYourTinnitus-Projekt beschäftigt waren, für ihre kreativen Ideen und den interessanten Austausch.

Es hat mir sehr viel Freude bereitet, in diesem professionellen Umfeld arbeiten zu dürfen.

Natürlich möchte ich mich darüber hinaus auch bei meinem privaten Umfeld, meiner Familie und meinen Freunden dafür bedanken, dass sie mir während meines gesamten Studiums zur Seite gestanden und mich unterstützt haben.

Inhaltsverzeichnis

1

Einleitung

Laut einer Studie des Vereins *Deutsche Tinnitus-Liga e.V.*, waren 1999 zehn Millionen Bundesbürger jährlich von Ohrgeräuschen betroffen. Etwa 1,5 Million Bürger in Deutschland litten mittelgradig bis unerträglich unter anhaltendem Tinnitus. Man geht davon aus, dass sich die Zahl der Betroffenen seitdem weiter erhöht hat. [2]
Die konkrete Ausprägung der Ohrgeräusche wird unterschiedlich wahrgenommen, manche beschreiben den Tinnitus als Rauschen, Brummen oder Surren, andere als Pfeifen oder Klingeln. [3]
Auf dem Tinnitus-Kongress in London 1981 wurde Tinnitus als „eine Tonempfindung, die nicht hervorgerufen ist durch simultanes mechano-akustisches oder elektrisches Signal“, definiert. Bei einem Tinnitus, der erst seit geraumer Zeit (24 Stunden) ununterbrochen besteht, spricht man von *akutem Tinnitus*, dauert er längerfristig (mindestens drei Monate) an, wird er als *chronischer Tinnitus* bezeichnet.

Viele Tinnitus-Patienten fühlen sich durch ihren anhaltenden Tinnitus erheblich in ihrer Lebensqualität beeinträchtigt. In diesem Fall spricht man von *komplexem chronischen Tinnitus*, auf den nicht selten psychische Störungen, wie Depressionen oder Angststörungen zurückzuführen sind. [4]

1.1 Problemstellung

Tinnitus ist ein Symptom, dem unterschiedliche Krankheiten zugrunde liegen können. Die medizinischen Ursachen für Tinnitus sind vielfältig und können zum Beispiel ein Schädelhirntrauma, ein Hörsturz, Lärmschäden, Kiefergelenksbeschwerden, Altersschwerhörigkeit, Drehschwindel, Erkrankungen des zentralen Nervensystems oder auch

Herz-Kreislauf-Erkrankungen sein. [3] Meist, laut der deutschen Tinnitus Liga e.V. bei 45 % der Betroffenen, ist ein organischer Befund als Ursache für die Tonempfindung jedoch nicht eindeutig feststellbar. [2] In diesen Fällen unterdrücken körperlich orientierte Behandlungsverfahren den Tinnitus nur kurzweilig, die Chance auf eine vollständige Genesung ist gering und besteht meist nur bei akutem Tinnitus. [4]

Die individuelle Wahrnehmung des Tinnitus unterliegt im Laufe des Tages Schwankungen, die durch veränderte Umstände, wie zum Beispiel Umgebungsgeräusche oder Stress, beeinflusst werden. Die Symptome lassen sich schwer objektivieren, also eindeutig messen (*subjektiver Tinnitus*). Der tatsächliche Zustand des Tinnitus wird nur vom Betroffenen wahrgenommen und ist somit nur durch diesen mess- und vergleichbar. Tinnitus Geschädigte sind meist in der Lage, relativ präzise Angaben über den aktuellen Zustand des Tinnitus und dessen Schwankungen zu treffen. [5]
Oft werden die Symptome von den Betroffenen jedoch hauptsächlich in ihrem gewohnten Alltag wahrgenommen. Vielen fällt es schwer, im Umfeld einer Studie, die eine physische Präsenz der Teilnehmer erfordert oder bei einem Arztbesuch, den Tinnitus exakt zu beschreiben und eine Beziehung zu potenziellen Auslösern herzustellen. Aus diesem Grund ist es sinnvoll, Betroffenen das Festhalten ihrer Symptome in ihrem gewohnten Umfeld zu ermöglichen.
Mittlerweile benutzen 78 % der deutschen Bevölkerung ein Smartphone. [6] Das Interesse an so genannten Wearables, wie Smartwatches oder Fitnesstrackern, ist mit 46 % ebenfalls sehr groß. [7] Eine mobile Applikaton (App) kann zum Beispiel die zeitlich und örtlich unabhängige Messung bestimmter Faktoren, wie der Umgebungsgeräusche, ermöglichen und Tinnitus geschädigte Personen so unterstützen.

Um Betroffenen die Dokumentation und Überwachung ihres Tinnitus und dessen Zusammenhang mit dem Tagesablauf zu ermöglichen und Erkenntnisse aus den Messungen zu ziehen, wurde das Track Your Tinnitus Forschungsprojekt von der Tinnitus Research Initiative (TRI) in Kooperation mit dem Institut für Datenbanken und Informationssysteme der Universität Ulm (DBIS) gegründet. Dazu wurde 2014 auf Basis einer Diplomarbeit von Jochen Herrmann unter anderem eine mobile Android Applikationen entwickelt, mit Hilfe derer Betroffene speziell entwickelte Fragebögen zur Erfassung der aktuellen Tinnitus Wahrnehmung ausfüllen können. [1] Die Applikation erlaubt die Messung des

Geräuschpegels, was als wesentlicher Faktor der Tinnitus Schwankungen vermutet wird. Weitere potenzielle Faktoren, wie zum Beispiel die Herzfrequenz oder die Umgebung werden jedoch bislang nicht berücksichtigt. Auch besteht in dieser Version noch keine enge Verknüpfung mit der zentralen Track Your Tinnitus Application Programming Interface (API), weshalb es keine Möglichkeit gibt, ohne eine Anpassung der App, dynamisch weitere Studien und Fragebögen zu ergänzen, beispielsweise auf Basis neu gewonnener Forschungserkenntnisse.

1.2 Zielsetzung

Das Ziel dieser Arbeit besteht darin, die mobile Applikation zur Durchführung Klinischer Studien zur Messung der individuellen Tinnitus Wahrnehmung unter Berücksichtigung erweiterter Anforderungen von Grund auf neu zu konzipieren und prototypisch auf Basis der Android Plattform umzusetzen.

Dabei liegt besonderer Fokus auf einer engen Verbindung mit dem Backend um die App flexibler und erweiterungsfähiger zu machen. Die Bedienbarkeit bei fehlender Internetverbindung darf darunter jedoch nicht leiden. Des Weiteren sollen Möglichkeiten identifiziert und analysiert werden, äußere Faktoren und Umgebungseinflüsse, wie Herzfrequenz, Lautstärke und Standort durch die mobile Applikation einfacher und zuverlässiger messbar zu machen und so Tinnitus geschädigte Patienten noch besser zu unterstützen.
Der mit dieser Arbeit realisierte Android Prototyp soll die neuen Erkenntnisse im Bereich der Tinnitusmessung und technische Neuerungen berücksichtigen und so die Basis für zukünftige Erweiterungen schaffen, um die erste Version der Android App mittelfristig abzulösen.

1.3 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in acht Hauptkapitel. Das nachfolgende Kapitel 2 soll zunächst einen Überblick über die verwandten Arbeiten im Umfeld der Unterstüt-

zung Tinnitus geschädigter Patienten und zur Durchführung Klinischer Studien geben. Daraufhin werden alle Anforderungen an die prototypische Realisierung der Applikation erarbeitet, wobei eine Unterteilung in funktionale und nicht-funktionale Anforderungen vorgenommen wird.
Auf der Basis der Anforderungen wird in Kapitel 3 die Architektur der App beschrieben. Dazu werden als erstes alle vorgesehenen Abläufe des Prototyps modelliert und darauf aufbauend das technische Architekturkonzept vorgestellt. Außerdem werden zusätzlich verwendete Bibliotheken beschrieben.

Das Kapitel 5 geht näher auf einzelne Details der Implementierung des Prototyps ein. Die Hauptaspekte Backendkommunikation, Authentifizierung und Lokalisierung werden beschrieben. Außerdem wird die Umsetzung bezüglich der Benachrichtigungsfunktion und der Geräusch-, Herzfrequenz-, und Standortmessung dargestellt.

Der umgesetzte Prototyp wird schließlich in Kapitel 6 vorgestellt. Dabei werden die einzelnen Ansichten und Funktionalitäten aus Benutzersicht genauer beschrieben.
Nach der Vorstellung erfolgt ein Abgleich der in Kapitel 3 definierten Anforderungen mit der tatsächlichen Realisierung, um die Vollständigkeit und den Status des Prototyps in Kapitel 8 abschließend bewerten zu können. Kapitel 8.2 gibt schließlich einen Ausblick auf potenzielle Folgeprojekte und Erweiterungen des Prototyps.

2

Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über verwandte Arbeiten im Bereich der Unterstützung Tinnitus geschädigter Patienten. Dadurch soll eine bessere Einordnung der vorliegenden Arbeit und des umzusetzenden Prototyps ermöglicht werden.

2.1 Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung Tinnitus geschädigter Patienten

Aufgrund bis dahin mangelnder Möglichkeiten, die Schwankungen der Tinnituswahrnehmung zu überwachen bzw. genauer zu untersuchen, hat Jochen Herrmann 2014 in seiner Diplomarbeit erstmals ein Framework für die Unterstützung Tinnitus geschädigter Patienten vorgestellt und realisiert. [1] Diese Arbeit entstand im Rahmen des Track Your Tinnitus-Forschungsprojektes.

Bestandteile des Frameworks sind eine Webseite, mit der die Nutzer Informationen abrufen, statistische Fragebögen ausfüllen und Ergebnisse anzeigen können und zwei mobile Applikationen, je eine für die Plattform Android und eine für das Betriebssystem iOS. Außerdem wurde ein Backend implementiert, um die Fragebogendaten von den Clients zu empfangen. Zur Überwachung der Tinnitusschwankung wurde ein spezieller Fragebogen entwickelt, der nur über die mobilen Apps ausgefüllt werden kann. Zur Erinnerung an diesen Fragebogen kann ein Benutzer bestimmte Benachrichtigungseinstellungen vornehmen. Wählt er keine fixen Zeitpunkte zur Benachrichtigung aus, werden zufällig unter Berücksichtigung der definierten Menge und des Zeitraumes

Benachrichtigungszeitpunkte anhand eines in der Arbeit beschriebenen Algorithmus generiert. [1] Während des Ausfüllens des Tinnitus-Fragebogens durch den Nutzer wird eine Geräuschpegelmessung durchgeführt, sofern der Benutzer dies nicht deaktiviert hat. Neben dem Fragebogen zur Überwachung der Tinnitusschwankung können Benutzer in der App auch statistische Fragebögen ausfüllen. Eine Darstellung der Ergebnisse aus dem Fragebogen zur Überwachung der Tinnitusschwankung ist ebenfalls möglich. Die Ergebnisse aus den Fragebögen in den Apps werden an das Backend gesendet, um eine weitere Forschung auf Basis der erfassten Tinnitusschwankungen zu ermöglichen. Die Abbildungen 2.1 und 2.2 zeigen jeweils einen Ausschnitt aus der aktuellen Version der Android Applikation.

Abbildung 2.1: Statistischer Fragebogen

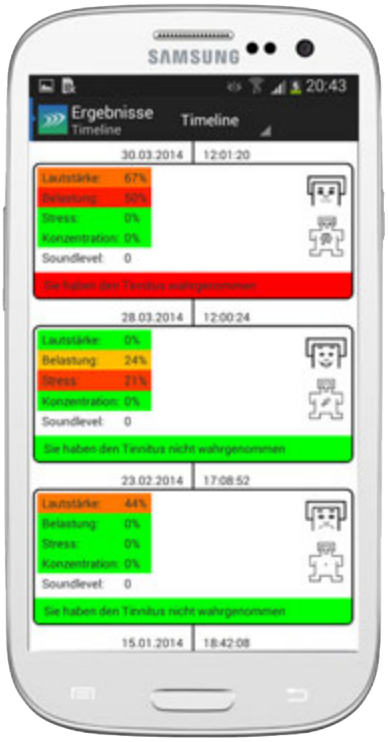


Abbildung 2.2: Ergebnisse

Die Arbeit von Jochen Herrmann stellte die grundlegende Basis für das gesamte Track Your Tinnitus Forschungsprojekt der TRI und der Universität Ulm dar. Jochen Herrmann erarbeite darin ein solides und funktionales Framework für die Unterstützung

[1] Siehe [1], Kapitel 5.2.3. Algorithmus für die Benachrichtigungsverteilung

Tinnitus geschädigter Patienten. Die im Zuge der Arbeit entwickelten Applikationen wurden in den darauffolgenden Jahren erweitert und angepasst und mittlerweile mehr als 1.800 mal (iOS) und mehr als 1.000 mal (Android) heruntergeladen.

Mittlerweile wurden neue Erkenntnisse im Bereich der Tinnitusforschung gewonnen und technische Fortschritte erkannt, weshalb eine Überarbeitung der Applikationen erforderlich ist. Nichtsdestotrotz stellt die Diplomarbeit und die daraus entstandene Android Applikation die elementare Grundlage für die hier vorliegende Arbeit dar.

2.2 Unterstützung von Patienten durch die mobile Durchführung regelmäßiger Fragebögen

2.2.1 Praktische Arbeiten

Für die Identifikation weiterer verwandter praktischer Anwendungen wurden die App-Stores von Android und iOS durchsucht. Spezielle Apps, die das Durchführen Klinischer Studien für Tinnitus geschädigte Personen erlauben, wurden dabei keine gefunden.
Allerdings ist eine Vielzahl an Apps zur angeblichen Therapie eines Tinnitus verfügbar, wie zum Beispiel *Tinnitus Klänge* von Oticon A/S, einem Hersteller von Hörgeräten, *Tinnitracks* oder auch *mimi*, das sich nicht nur an Tinnitus geschädigte Menschen richtet. Sie basieren alle auf dem Abspielen bestimmter verfremdeter bzw. an das Gehör speziell angepasster Klänge (*Tailor-made notched music training*) und sollen so „durch frequenzgefilterte Musik die empfundene Lautstärke des Tinnitus, die Tinnitusaktivität der Nervenzellen sowie die psychologische Belastung bei Tinnitus-Patienten statistisch signifikant reduzieren“. [8] Tinnitracks beruft sich dabei im Speziellen auf neurowissenschaftliche Forschungsarbeiten der Universität Münster. [9, 10, 11, 12, 13]
Ob anhand dieser Apps tatsächlich Erfolge in der Tinnitusbehandlung erzielt werden können, ist jedoch kritisch zu bewerten. Anhand einer neueren dreimonatigen Studie zeigte die medizinische Fakultät der Universität Münster, dass diese Therapieansätze, abgesehen von der Lautstärkewahrnehmung, nur wenig Effekt auf die Tinnitusbeschwerden im Gesamten haben. [14]

Bei der weiteren Recherche wurden einige Applikationen identifiziert, die in anderen medizinischen Bereichen eine Erfassung von Patientendaten und die Durchführung von Studien erlauben. So versucht die App *Migraine Buddy* [15] durch Bereitstellung täglicher Fragebögen die Auslöser, Häufigkeit, Dauer und die Symptome von Migräne-Patienten zu identifizieren und zu erfassen. Die App berücksichtigt dabei auch Meta-Informationen, wie Schlafintensität, Wetter und Luftdruck. Sie besitzt hohe Bewertungen, über 100.000 Installationen und wird deshalb als besonders hilfreich und benutzerfreundlich erachtet. Die singapurische Firma Healint, von der *Migraine Buddy* entwickelt wird, plant momentan auch die Unterstützung weiterer Patienten mit chronischen Beschwerden wie Multiple Sklerose, Psoriasis (Schuppenflechte) und Epilepsie. [16] Auch konnten Apps zur Erfassung von Asthma (*AsthmaMD* [17]), Depressionen (*Depressionstest* [18]) oder Rheuma (*RheumaTrack* [19]) in den App Stores gefunden werden. Diese stellen alle bestimmte Fragebögen zur Datenerfassung bereit bzw. bilden ein Tagebuch ab, innerhalb dessen Patienten entsprechende Einträge vornehmen können. Die meisten dieser Apps bieten auch die Auswertung der erfassten Daten bzw. in manchen Fällen auch eine Therapieunterstützung an. Alle Apps sind darauf ausgelegt, im normalen Alltag der Patienten verwendet zu werden.

2.2.2 Theoretische Arbeiten

Eine ausführliche Suche nach Arbeiten zum Thema Unterstützung von Tinnitus-Patienten bzw. Methoden und Werkzeugen zur Durchführung Klinischer Studien lieferte wenig Ergebnisse. Generell ist das Thema Tinnitus in der Literatur relativ wenig präsent. Der Tagungsband Nr. 85 des CIBA Foundation Symposium enthält einen interessanten Artikel zur „Messung von Tinnitus bei Menschen“, welcher jedoch bereits aus dem Jahr 1981 stammt. Zu dieser Zeit waren kontinuierliche bzw. regelmäßige Messungen schwer oder nur durch technisch hohen Aufwand realisierbar. [20]
Ein aus dem Jahr 2004 stammender Beitrag von Craig Newman und Sharon Sandridge im Sammelband „Tinnitus Theory and Management“ geht gezielt auf Fragebogenmöglichkeiten für Tinnitusbeschwerden ein. Dabei ist ein Abschnitt zum Thema „Daily Diaries“ besonders interessant. In diesem heißt es, dass ein tägliches Monitoring der Beschwer-

den Muster und Auffälligkeiten zutage fördern könnte. Im Zusammenhang mit Umgebungslautstärke, Medikation und Diäten könnte die subjektive Bewertung der Lautstärke- und Tonhöheempfindung neue Erkenntnisse in der Tinnitusforschung liefern. Als großes Problem bei täglichen Messungen wird die erhöhte Aufmerksamkeit gegenüber des Tinnitus gesehen, die eine Wahrnehmungsverfälschung erzeugen könnte, insbesondere wenn Messungen für Patienten zwanghafte Züge annehmen. [21]

Aufgrund der wenigen Publikationen bzgl. der tatsächlichen Durchführung und Realisierbarkeit von Tinnitusmessungen wurde die Suche auf Publikationen zur Durchführung wiederkehrender Fragebögen in anderen medizinischen Bereichen erweitert.
Zur Unterstützung von Asthma-Patienten wurde 2005 von D. Ryan et al. eine Studie zum Thema „Mobile phone technology in the management of asthma“ durchgeführt. Darin wurde untersucht, inwiefern in regelmäßigen Abständen durchgeführte Messungen zur Ausatmungsgeschwindigkeit (Peak Flow) mit Hilfe von mobilen Endgeräten zur Zufriedenheit der Patienten und zur Verbesserung der Krankheit beitragen. Es wurde festgestellt, dass die meisten Patienten (69 %) die Messungen und das Feedback über die mobile Applikation positiv bewerteten, mit der Begründung, dass sie durch sie ein besseres Bewusstsein und einen besseren Überblick über ihr Asthma erhielten. [22]

2.2.3 Fazit

Anhand der zuvor aufgeführten Studie wird deutlich, dass Patienten ihren chronischen Beschwerden viel gelassener entgegenstehen, wenn ein hohes Bewusstsein für diese besteht. [22] Dies ist wohl der Grund, weshalb so viele Apps in den App-Stores angeboten werden, die Nutzer in ihrem Alltag bzw. im Umgang mit ihrer Krankheit unterstützen sollen. Sie bilden damit ähnliche Anwendungsfälle ab wie eine App zur Erfassung von Tinnitusschwankungen.

Tatsächliche Hilfe für Tinnitus geschädigte Patienten wird dabei jedoch bislang kaum angeboten. Die Track Your Tinnitus App der TRI und der Universität Ulm ist die einzige App, die das Festhalten der Symptome auf täglicher Basis erlaubt und dem Nutzer vor wissenschaftlichem Hintergrund ausgewertete Ergebnisse liefert.

Für den Aufbau von Fragebögen zur Erfassung der Tinnitusbeschwerden und eventueller Zusammenhänge existieren theoretische Vorschläge ([21, 20]), die im Weiteren insbesondere für die Identifikation der Anforderungen an die App berücksichtigt werden sollen.

Abschließend kann zusammengefasst werden, dass im Umfeld der Tinnitus Forschung neue Technologien, wie die Nutzung von Smartphones, Fitnessuhren oder Smartwatches, noch relativ wenig dazu genutzt werden, forschungsrelevante Daten zu sammeln und eventuelle Zusammenhänge mit Alltagssituationen zu identifizieren.

3

Anforderungen

In diesem Kapitel werden die Anforderungen an die Android Applikation des Track Your Tinnitus (TYT) Projekts dargestellt. Dabei werden die Anforderungen in funktionale und nicht-funktionale Anforderungen unterteilt.

3.1 Funktionale Anforderungen

Die folgenden funktionalen Anforderungen definieren die Anforderungen an den Leistungsumfang der Track Your Tinnitus Android Applikation. Zur besseren Übersicht werden diese dabei in drei Bereiche unterteilt.

Allgemeine funktionale Anforderungen

Allgemeine funktionale Anforderungen betreffen Hauptfunktionalitäten der Applikation, die für eine hinreichende Funktionsweise erforderlich sind, ohne dass sie in direkter Verbindung mit Fragebögen oder Studien stehen.

Nr.	Titel und Beschreibung
(1)	**Nutzer können sich in der App registrieren** Die Benutzung der App kann nicht ohne Benutzerkonto erfolgen. Es sollte dem Benutzer daher möglich sein, direkt auf einem Gerät ein solches Benutzerkonto zu erstellen.

(2)	**Nutzer können die App auch ohne Internetverbindung verwenden** Eine funktionierende Internetverbindung auf dem Smartphone sollte keine Voraussetzung für das Ausfüllen von Fragebögen in der App sein, da ein Benutzer potenziell nur schlechten oder gar keinen Empfang hat. Eventuell gespeicherte Werte werden anschließend bei funktionierender Internetverbindung im Hintergrund übermittelt.
(3)	**Nutzer können ihre Sprache auswählen** Wählt ein Benutzer seine Sprache während der Registrierung aus einer Menge an festgelegten Sprachen, werden sämtliche Bestandteile der App in dieser Sprache angezeigt. Ein Benutzer kann seine Sprache nachträglich innerhalb der App ändern. Die jeweils gewählte Sprache wird auf dem Server hinterlegt.

Tabelle 3.1: Allgemeine funktionale Anforderungen

Funktionale Anforderungen an die Studienteilnahme

Alle Anforderungen, die im Zusammenhang mit der Teilnahme an einer Studie stehen, werden nachfolgend dargestellt.

Nr.	Titel und Beschreibung
(4)	**Nutzer können an verschiedenen Studien teilnehmen** Ein Benutzer soll die Möglichkeit haben, sich in mehrere Klinische Studien einzuschreiben und deren Fragebögen auszufüllen. Er soll die Studienteilnahme zudem jederzeit beenden können, sofern es sich nicht um die Standard-Studie handelt.
(5)	**Nutzer können alle verfügbaren Studien sehen** Wird durch eine berechtigte Person über eine entsprechende Plattform eine neue Studie angelegt, erscheint diese, abhängig von ihrem Zustand, in der mobilen Applikation. Aktive Studien werden entsprechend ihres Typs dargestellt (öffentliche, passwortgeschützte, private Studien). Inaktive Studien werden nicht dargestellt.

(6)	**Nutzer können Studieneinladungen annehmen** Liegt eine Einladung zur Teilnahme an einer aktiven privaten Studie vor, wird diese dem Nutzer in der App angezeigt. Der Nutzer kann sich in die Studie einschreiben, indem er die Einladung annimmt.
(7)	**Nutzer können an einer privaten Studie teilnehmen, sofern sie das Passwort kennen** Der Benutzer kann sich per App in eine private Studie einschreiben, sofern er das Passwort kennt.

Tabelle 3.2: Funktionale Anforderungen an die Studienteilnahme

Funktionale Anforderungen an die Fragebögen

Nachfolgend werden alle funktionalen Anforderungen, die das Abrufen, Darstellen und Ausfüllen von Fragebögen betreffen, erläutert.

Nr.	Titel und Beschreibung
(8)	**Nutzer werden zum Ausfüllen statistischer Fragebögen nach dem Einschreiben in eine Studie aufgefordert** Existieren innerhalb einer Studie statistische Fragebögen, werden diese direkt nach dem Einschreiben in die Studie angezeigt. Der Benutzer kann weitere Aktionen erst vornehmen, nachdem er alle statistischen Fragebögen der Studie beantwortet hat.
(9)	**Nutzer erhalten Aktualisierungen bzw. Änderungen von Fragebögen** Die App sollte Änderungen an den Fragebögen regelmäßig vom Backend abrufen und die dem Nutzer angezeigten Daten entsprechend aktualisieren. Neue Fragebögen sollen innerhalb der App sofort nach dem Abruf angezeigt werden.

(10)	**Nutzer werden zum Ausfüllen neuer statistischer Fragebögen aufgefordert** Werden beim Abrufen der Daten vom Backend neue unbeantwortete Fragebögen identifiziert, sollen diese dem Nutzer in der App direkt angezeigt werden. Der Benutzer kann weitere Aktionen erst vornehmen, nachdem er den neuen einmaligen Fragebögen der Studie beantwortet hat.
(11)	**Nutzer können nur aktive Fragebögen ausfüllen** Inaktive Fragebögen dürfen von den Benutzern nicht ausgefüllt werden. Sie dürfen mit einem speziellen Hinweistext zu Übersichtszwecken angezeigt werden, jedoch besteht für den Benutzer keine Möglichkeit inaktive Fragebögen zu öffnen.
(12)	**Nutzer können die Schwankungen ihres Tinnitus und entsprechende Faktoren regelmäßig erfassen** Es existiert eine Standardstudie, die mindestens einen Fragebogen beinhaltet, mit dem ein Benutzer in unregelmäßigen wiederholt Abständen die Schwankungen der Tinnituswahrnehmung festhalten und überwachen kann. Die Teilnahme an dieser Studie kann der Nutzer nicht beenden.
(13)	**Gespeicherte Antworten werden sofort oder nachgelagert an das Backend gesendet** Zur Visualisierung der Ergebnisse aus den Fragebögen auf unterschiedlichen Endgeräten und für Forschungszwecke, sollten die Ergebnisse der Fragebögen aus der App an den Server übertragen werden.
(14)	**Nutzer werden nicht von vorgegebenen Werten beeinflusst** Ein Benutzer lässt sich beim Ausfüllen eines Fragebogens davon beeinflussen, welcher Wert voreingestellt ist. Daher dürfen z.B. Schieberegler in Fragebögen keinen initialen Wert haben.

	Der Pegel der Umgebungsgeräusche eines Nutzers kann erfasst werden
(15)	Um herauszufinden, ob der Tinnitus von den Umgebungsgeräuschen überdeckt oder beeinflusst wird, soll die App während des Ausfüllens eines Fragebogens zur Überwachung der Schwankungen der Tinnituswahrnehmung den Pegel der Hintergrundgeräusche messen und übermitteln können, falls für den entsprechenden Fragebogen vorgesehen.
	Nutzer können die Geräuschpegelmessung deaktivieren
(16)	Der Benutzer sollte in der Lage sein, für die gesamte Applikation sowie für einzelne Fragebögen die Geräuschpegelmessung zu deaktivieren, falls er dies wünscht.
	Die Herzfrequenz eines Nutzers kann erfasst werden
(17)	Um einen potenziellen Zusammenhang zwischen dem Tinnitus und der Herzfrequenz identifizieren zu können, sollte die App während des Ausfüllens des Fragebogens die Herzfrequenz messen und übermitteln können, falls für den entsprechenden Fragebogen vorgesehen.
	Nutzer können die Herzfrequenzmessung deaktivieren
(18)	Der Benutzer sollte in der Lage sein, für die gesamte Applikation sowie für einzelne Fragebögen die Herzfrequenzmessung zu deaktivieren, falls er dies wünscht.
	Der Standort bzw. der Bewegungszustand des Nutzers kann erfasst werden
(19)	Die Wahrnehmung des Tinnitus kann vom aktuellen Bewegungszustand (Reise, Ruhezustand) abhängen. Die App sollte den aktuellen Standort (GPS) in definierten Abständen bestimmen und übermitteln können, falls für den entsprechenden Fragebogen vorgesehen.
	Nutzer können die Standortbestimmung deaktivieren
(20)	Der Benutzer sollte in der Lage sein, für die gesamte Applikation sowie für einzelne Fragebögen die Standortbestimmung zu deaktivieren, falls er dies wünscht.

(21)	**Der Nutzer wird anhand bestimmter Zeitpläne an das Ausfüllen von Fragebögen erinnert** Jeder wiederkehrende Fragebogen kann einen oder mehrere Pläne besitzen, nach denen Benachrichtigungen in bestimmten Intervallen, oder zu vorgegebenen Zeitpunkten vorgesehen werden können. Tritt ein definierter Zeitpunkt ein, wird der Nutzer innerhalb der App benachrichtigt.
(22)	**Nutzer können die Benachrichtigungsfunktion individuell anpassen** Ein Benutzer sollte die Benachrichtigungshäufigkeit und den Zeitraum der Benachrichtigen in der App frei einstellen können, wenn dies vom Studienleiter für den Fragebogen vorgesehen ist.
(23)	**Nutzer können den Klingelton einer Benachrichtigung anpassen** Da der Klingelton einer Benachrichtigung in manchen Fällen vom Tinnitus überdeckt werden kann, sollte einem Benutzer die Möglichkeit gegeben werden, den Klingelton der Benachrichtigung zu ändern.
(24)	**Nutzer können ihre Ergebnisse in der App sehen** Um die zeitliche Entwicklung in der App direkt anzeigen zu können, sollten die Ergebnisse aus dem Fragebogen zur Überwachung der Schwankungen der Tinnituswahrnehmung visualisiert werden.
(25)	**Nutzer können ihre Daten exportieren** Ein Benutzer sollte die Möglichkeit haben, seine Daten (Antworten auf den Fragebogen zur Überwachung der Tinnituswahrnehmung) aus der App zu exportieren, um sie zum Beispiel seinem Arzt zur Verfügung zu stellen.

Tabelle 3.3: Funktionale Anforderungen an die Fragebögen

3.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind Anforderungen, die keinen Einfluss auf die tatsächliche Funktionalität einer Applikation besitzen, aber dennoch von großer Bedeutung für die Nutzungs- und Weitereintwicklungspotenziale der App sind. Die identifizierten

nichtfunktionalen Anforderungen an die Track Your Tinnitus App werden nachfolgend dargestellt.

Nr.	Titel und Beschreibung
(1)	**Technische Anforderungen** Die Applikation ist unter Android 4.1 oder neuer lauffähig.
(2)	**Bedienbarkeit** Die App soll einfach zu bedienen und verständlich sein.
(3)	**Aussehen** Die App soll den aktuellen Gestaltungsprinzipien (Material Design) der Android Plattform entsprechen. [23]
(4)	**Änderbarkeit** Die App soll einfach wart- und änderbar sein.
(5)	**Sicherheit** Die App soll den Anforderungen Vertraulichkeit, Integrität und Informationssicherheit genügen, indem sie aktuelle Sicherheitsstandards berücksichtigt.
(6)	**Fehlertoleranz** Die App weist auf fehlerhafte Benutzereingaben hin und erlaubt deren Korrektur.

Tabelle 3.4: Nichtfunktionale Anforderungen

4

Architektur

In diesem Kapitel wird die gewählte Architektur der Track Your Tinnitus Applikation beschrieben. Dazu wird zunächst der fachlich vorgesehene Ablauf innerhalb der Android App auf Basis der Anforderungen skizziert. Darauf aufbauend wird die entwickelte Datenstruktur erläutert und die Paket- und Klassenstruktur aufgezeigt. Zuletzt werden die ausgewählten Bibliotheken erläutert.

Um die gewählte Architektur der Android Applikation näher beschreiben zu können, ist zunächst ein grober Überblick über die Gesamtarchitektur des Track Your Tinnitus Projektes erforderlich. Abbildung 4 verdeutlicht diese anhand eines Schaubildes.

Das Kernstück des Track Your Tinnitus-Projekts stellt ein serverseitiges Backend dar, das über eine RESTful-API[1] definierte Operationen zulässt. Im Backend werden sämtliche Daten wie Benutzerkonten, Studieninformationen, Fragebögen und die Daten ausgefüllter Fragebögen hinterlegt. So wird erreicht, dass Clients, wie die Android App, die iOS App, Web-Applikationen oder andere Endgeräte wie Smart-Watches, auf einen einheitlichen Datenstand zugreifen. Dies gibt Benutzern die Möglichkeit, die Track Your Tinnitus Funktionalitäten auf unterschiedlichen Endgeräten abzurufen und von überall auf ihre Daten zugreifen zu können. Auf der anderen Seite wird die Auswertung von Fragebögen erleichtert, da Analysen stets auf dem gesamten Datenstand ausgeführt werden können und Antworten nicht erst konsolidiert werden müssen.

Auch die vorigen Versionen der Track Your Tinnitus Anwendungen waren über eine Schnittstelle an ein zentrales Backend angebunden. Aufgrund der beschränkten Funk-

[1] Eine API, die über HTTP Methoden bestimmte Aufrufe zulässt. Eine RESTful API – auch RESTful web service – basiert auf dem Representational State Transfer (REST) Architekturprinzip, dessen Ursprung im durch Roy Fielding beschriebenen *HTTP Object Model* liegt. [24]

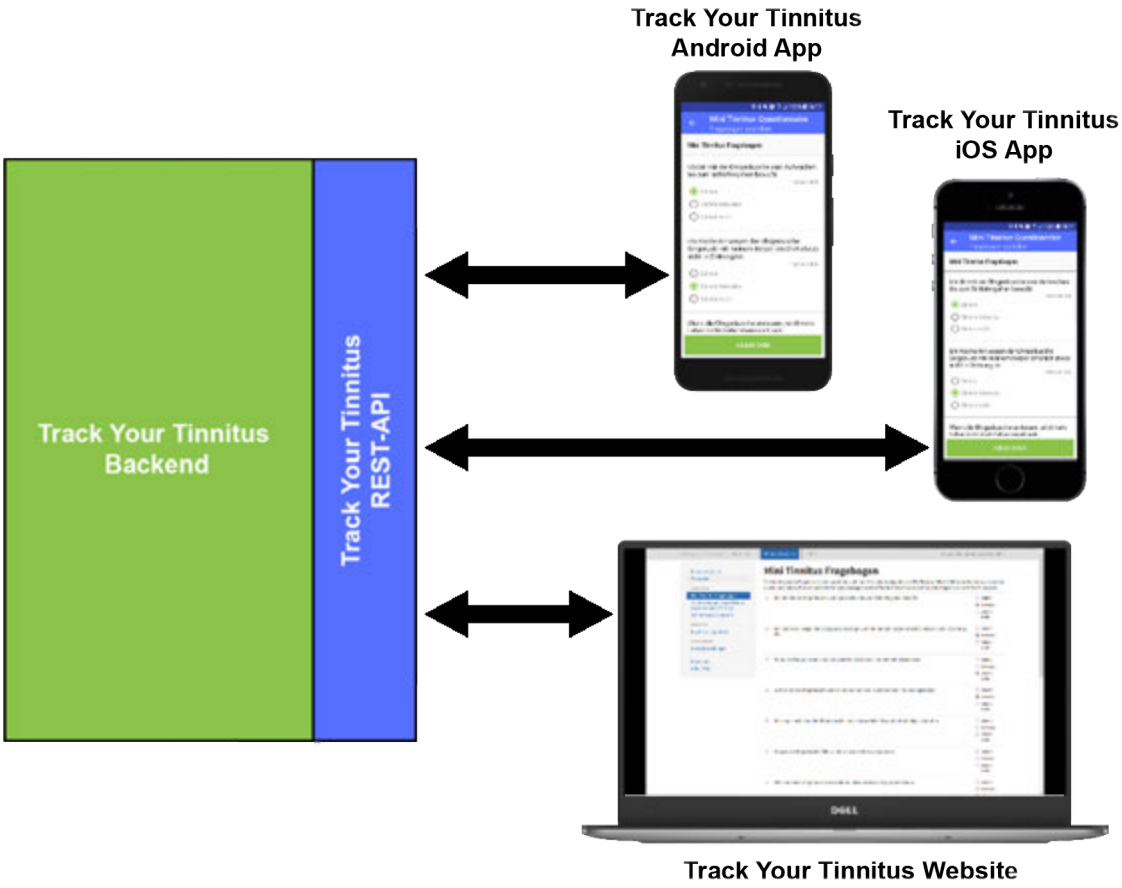


Abbildung 4.1: Gesamtarchitektur des Track Your Tinnitus Projekts

tionalität dieser wurde jedoch parallel zur Umsetzung des Android Prototyps eine neue umfangreiche REST-Schnittstelle entwickelt und zu Testzwecken zur Verfügung gestellt. Diese neue Schnittstelle soll mittelfristig die alte Schnittstelle ablösen und so weitere Funktionalitäten ermöglichen. Die vorliegende Arbeit basiert auf der Neuentwicklung des Backends und dieser entsprechenden REST-Schnittstelle.

4.1 Ablaufbeschreibung

Auf Basis der in Kapitel 3 aufgeführten Anforderungen wurden Prozessdiagramme zur Veranschaulichung der vorgesehenen Abläufe und Zusammenhänge innerhalb der

App erarbeitet. Diese Prozessdiagramme repräsentieren die Basis für weitere architekturelle Entscheidungen wie Datenmodelle und ausgewählte Bibliotheken. Für die Abbildung der Prozesse wurde die Spezifikationssprache Business Process Model and Notation (BPMN) gewählt.

Der Hauptprozess der Applikation ist in Abbildung 4.2 dargestellt. Im Falle, dass die App initial installiert wurde, sich ein Nutzer vollständig ausgeloggt hat, oder die Sitzung eines eingeloggten Nutzers ausgelaufen ist, soll ein Login-Formular angezeigt werden, das den Nutzer zur Eingabe seiner Zugangsdaten auffordert. Für die Authentifizierung sowie für den initialen Abruf der Daten eines Nutzer ist eine bestehende Internetverbindung zwingend erforderlich. Konnte der Nutzer erfolgreich authentifiziert werden, beginnt der Synchronisierungsprozess. Dieser kann im Hintergrund auch periodisch angestoßen werden, wenn ein definierter Zeitrahmen überschritten wird.
Zunächst werden alle bestehenden Fragebögen des Nutzers abgerufen. Besteht zu diesem Zeitpunkt keine Internetverbindung, wird auf die lokale Datenbank zurückgegriffen. Beinhaltet die Antwort einmalige statistische Fragebögen, die zwingend ausgefüllt werden müssen, werden diese dem Benutzer nacheinander angezeigt. Der Nutzer kann währenddessen keine anderen Aktionen in der App ausführen. Statistische Fragebögen beinhalten zum Beispiel demografische Fragen, die zur Einschätzung der Studienteilnehmer erforderlich sind. Nach erfolgreichem Ausfüllen aller statistischen Fragebögen, wird im Hintergrund die Erinnerungsfunktion für die wiederkehrenden Fragebögen des Nutzers aufgesetzt, falls vorgesehen. Es ist wichtig, dass Erinnerungen an wiederkehrende Fragebögen erst nach Abschluss der statistischen Fragebögen erzeugt werden, da der Benutzer vorher ohnehin keine anderen Fragen ausfüllen kann. Daraufhin werden alle Studiendetails des Nutzers abgerufen und in der lokalen Datenbank abgelegt bzw. aktualisiert, falls eine Internetverbindung besteht.

Ab diesem Zeitpunkt bestimmt der Nutzer die Steuerung der App. Wählt er die Studienansicht, werden ihm alle Studien angezeigt, in denen er aktuell eingeschrieben ist. Dieser Prozess ist in Abbildung 4.3 dargestellt. Darüber kann der Nutzer sich auch in neue Studien einschreiben. Der Vorgang des Einschreibens in eine neue Studie wird im Subprozess in Abbildung 4.4 veranschaulicht.

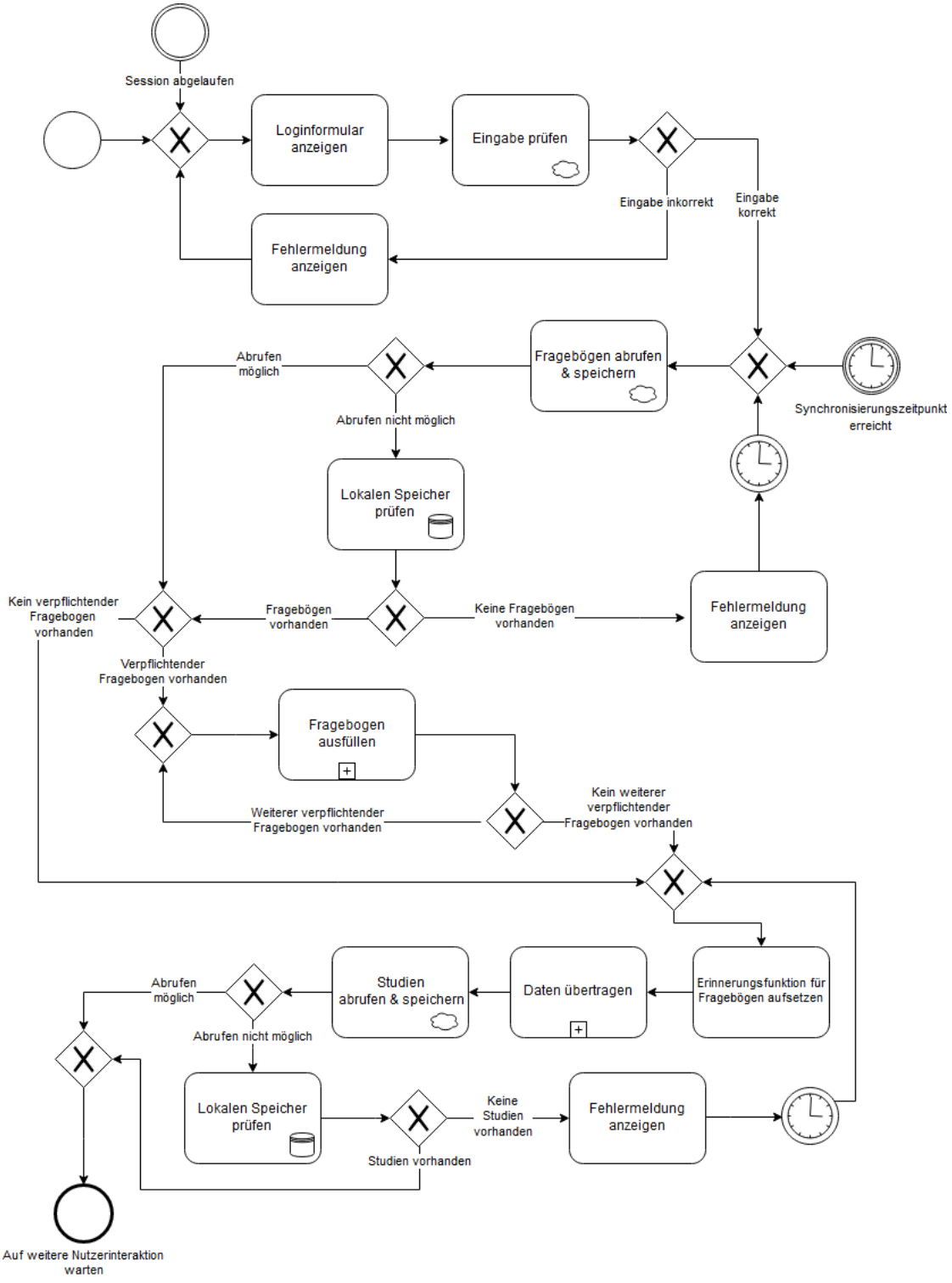


Abbildung 4.2: Haupt- und Synchronisierungsprozess

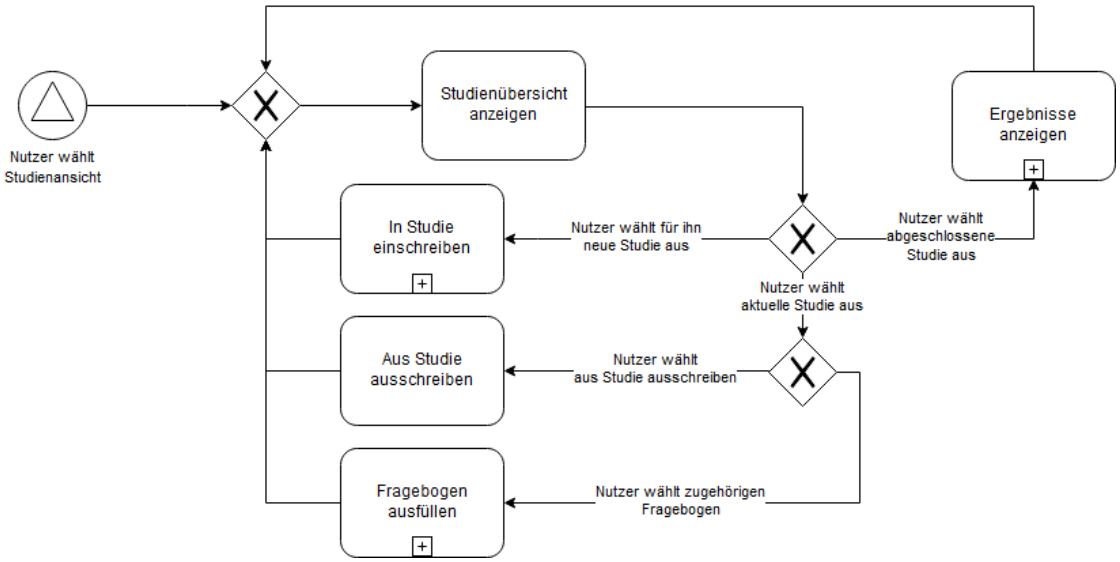


Abbildung 4.3: Studien anzeigen

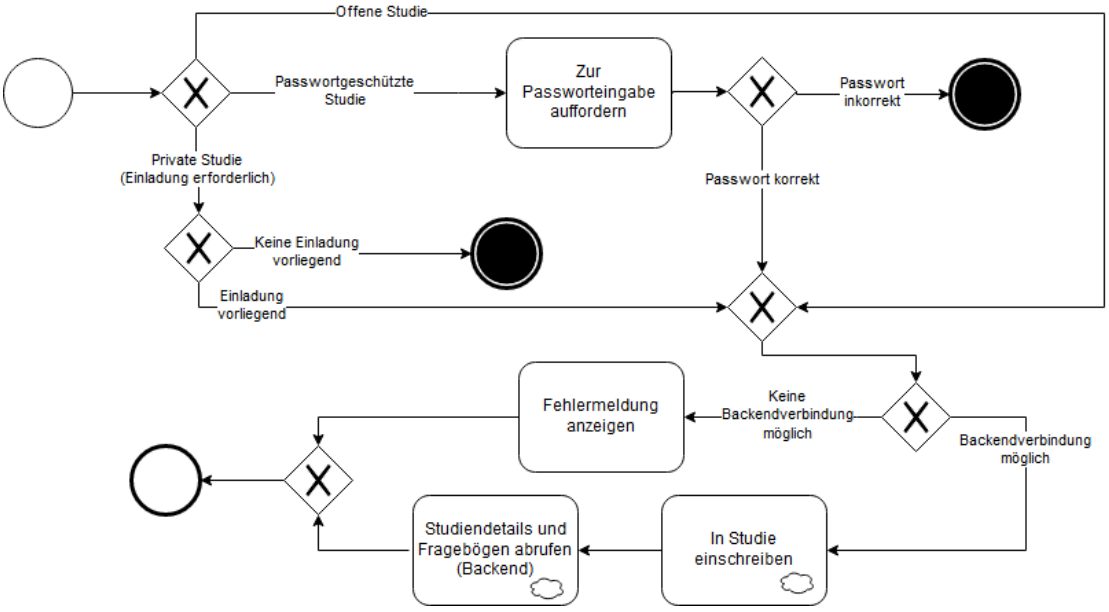


Abbildung 4.4: Subprozess: In Studie einschreiben

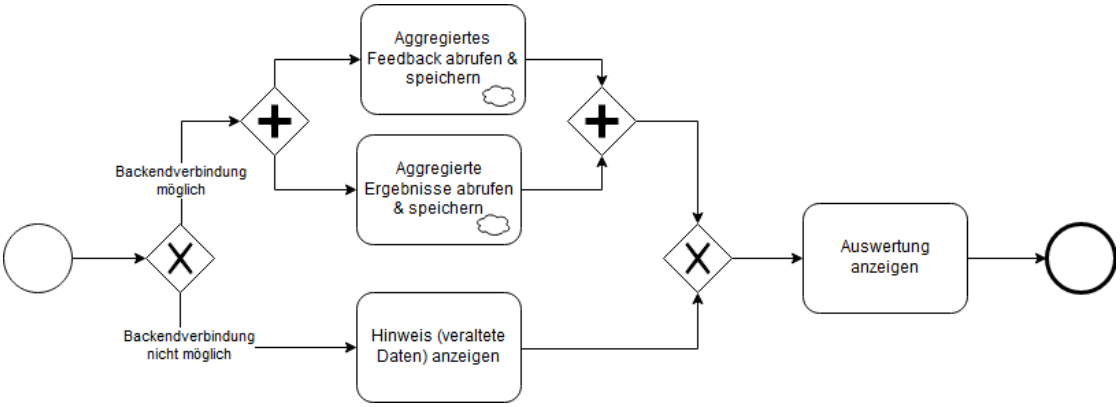


Abbildung 4.5: Subprozess: Ergebnisse anzeigen

Es ist vorgesehen, dass Nutzer eine Gesamtauswertung ihrer Studienteilnahme einsehen können. Abbildung 4.5 zeigt den Prozess zur Anzeige von Ergebnissen.

Wiederkehrende Fragebögen können von Nutzern zu beliebigen Zeitpunkten ausgefüllt werden. Diese Fragebögen beinhalten meist einen oder mehrere vorgesehene Erinnerungszeitpunkte, zu denen der Benutzer zur Eingabe aufgefordert werden soll. Diese Erinnerungszeitpunkte sind jedoch nicht zwingend und können auch vom Nutzer ignoriert werden. Im Gegensatz dazu werden einmalige Fragebögen direkt von der App angezeigt und können nach einmaligem Ausfüllen nicht mehr vom Nutzer geöffnet werden. Der Prozess in Abbildung 4.6 zeigt den Ablauf zum Ausfüllen eines Fragebogens. Dieser Prozess ist für einmalige und wiederkehrende Fragebögen identisch.

Während des Ausfüllens eines Fragebogens werden im Hintergrund bestimmte Messungen durchgeführt. Dies ist zum einen davon abhängig, welche Messungen für einen Fragebogen vorgesehen sind, und zum anderen davon, ob der Nutzer bestimmte Messungen für Fragebögen deaktiviert hat. Die Einstellungen eines Fragebogens kann der Nutzer individuell anpassen. Dies ist im Prozess in Abbildung 4.7 beschrieben.

Im Falle, dass während des Abschlusses eines Fragebogens durch den Nutzer keine Internetverbindung besteht, werden die Daten nicht direkt an den Server gesendet, sondern lokal abgelegt. Periodisch wird daraufhin auf eine bestehende Internetverbindung geprüft und bei Bestehen die Daten nachgelagert übertragen. Dieser Prozess ist in Abbildung 4.8 verdeutlicht.

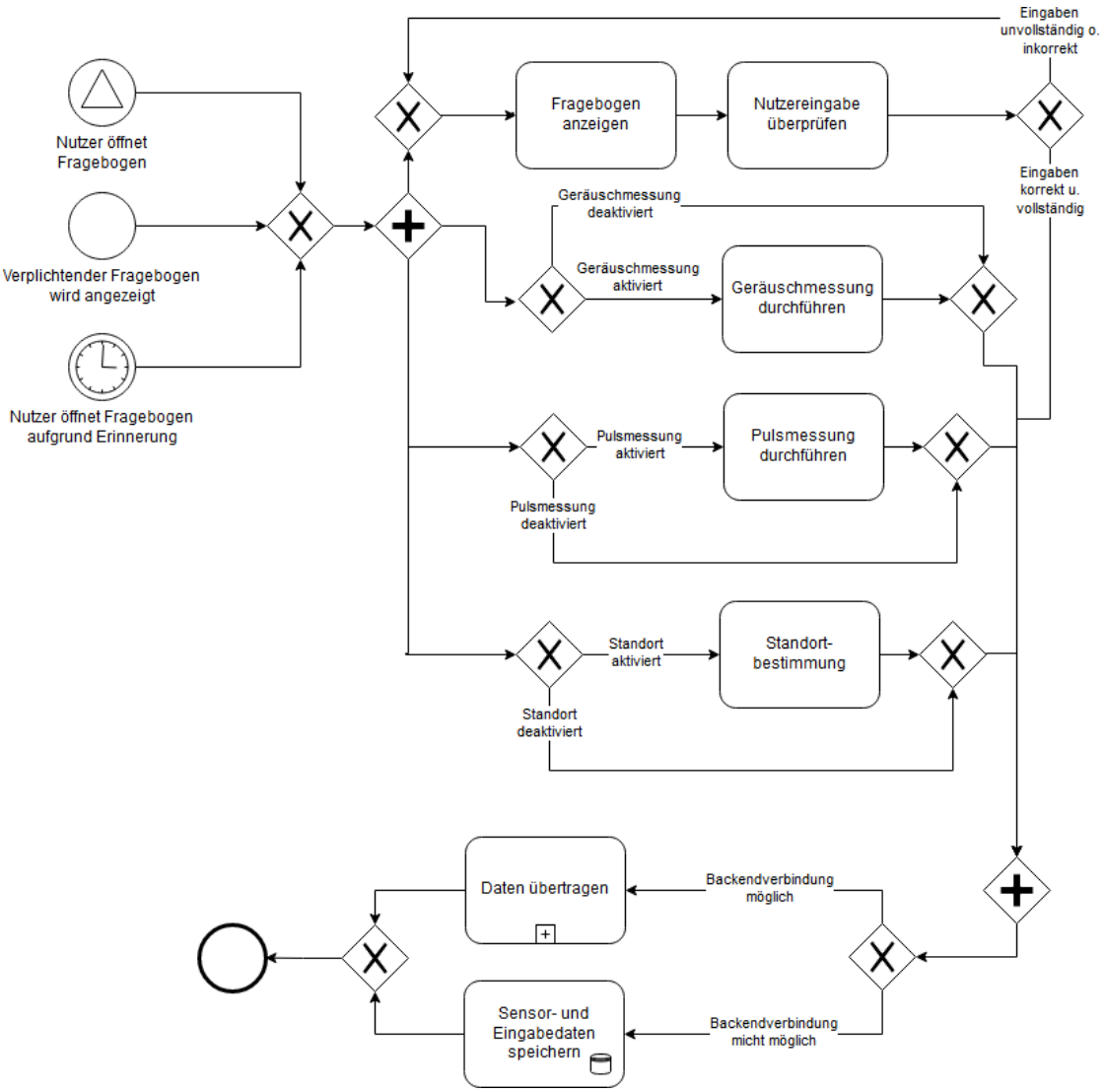


Abbildung 4.6: Subprozess: Fragebogen ausfüllen

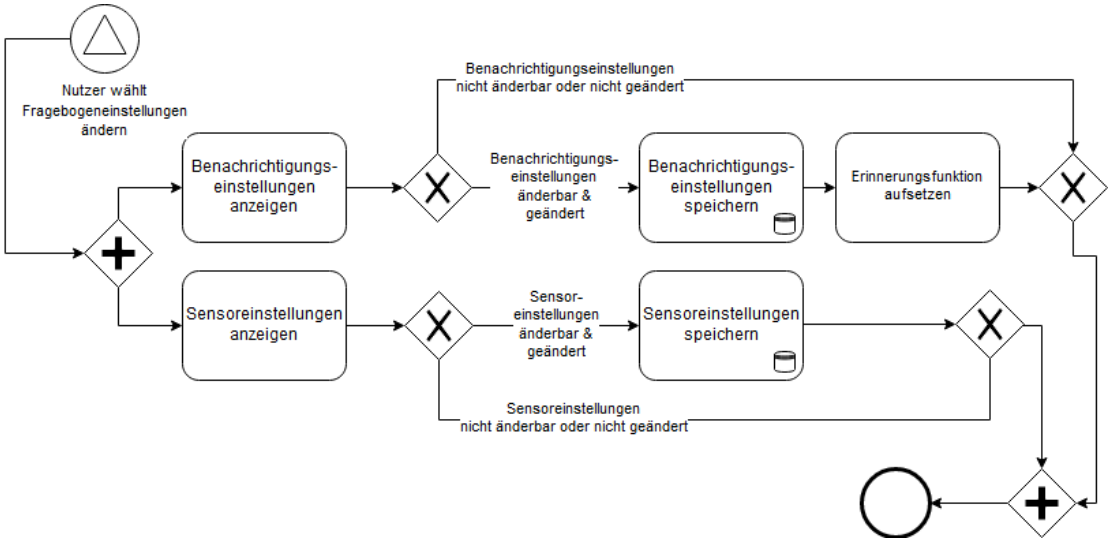


Abbildung 4.7: Subprozess: Fragebogeneinstellungen ändern

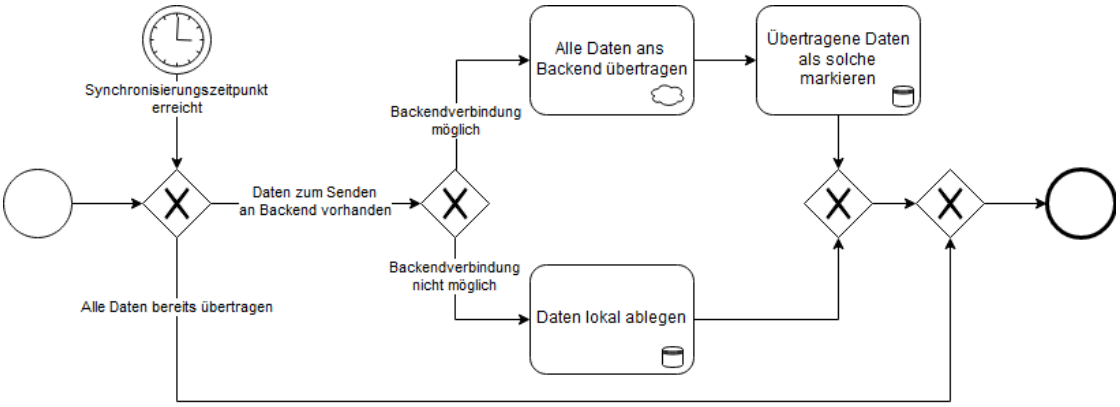


Abbildung 4.8: Subprozess: Daten übertragen

4.2 Architekturkonzept der Android Applikation

Nachfolgend wird ein Architekturkonzept zur Umsetzung der Android Applikation vorgestellt.

4.2.1 Genereller Aufbau

Grundsätzlich beinhaltet die Applikation Komponenten, die für die Realisierung der Benutzerschnittstelle verantwortlich sind (View), solche, die das Datenmodell beinhalten (Model) und Komponenten, die sich um die Umsetzung der zusätzlichen Funktionalitäten kümmern und Model und View zusammenbringen (Controller).

Der generelle technische Aufbau der Applikation wird anhand Abbildung 4.9 veranschaulicht. Das dargestellte vereinfachte Klassendiagramm legt den Fokus vor allem auf die View, zeigt also alle Klassen der Applikation, die die Benutzerschnittstelle realisieren.

Alle Activities[2] erben von einer `BaseActivity`, die bestimmte, wiederholt erforderliche, Funktionalitäten beinhaltet, zum Beispiel den Aufruf des Login-Dialogs. Die `MainActivity` beherbergt die Hauptfunktionalitäten der App, die dem Nutzer nach der Registrierung zur Verfügung stehen, realisiert durch Fragments[3]. Zusätzliche Funktionalitäten, die unabhängig von der `MainActivity` zur Verfügung stehen, wie die Anzeige der Fragebögen (`QuestionnaireStructureActivity`) oder die Registrierungsseite (`SignupActivity`) werden durch separate Activities realisiert. Dialoge werden durch dedizierte Dialogklassen, wie zum Beispiel den `EditPasswordDialog`, oder den `AccessStudyDialog`, der beim Eintreten in eine Studie angezeigt wird, umgesetzt.

Im Paket `adapters` sind alle Adapter verortet, die bei der Realisierung dynamischer Listen erforderlich sind. `Sqlite` beinhaltet alles, was mit der Datenbank zusammenhängt.

[2] Activities stellen in Android Einstiegspunkte für eine Interaktion mit dem Nutzer dar. Eine Activity repräsentiert genau einen Screen mit einer Benutzerschnittstelle. Für Details siehe [25].

[3] Ein Fragment macht ein bestimmtes Verhalten oder einen Teil der Benutzerschnittstelle einer Activity aus. Eine Activity kann mehrere Fragments mit unterschiedlichen Lebenszyklen beherbergen. Für Details siehe [26].

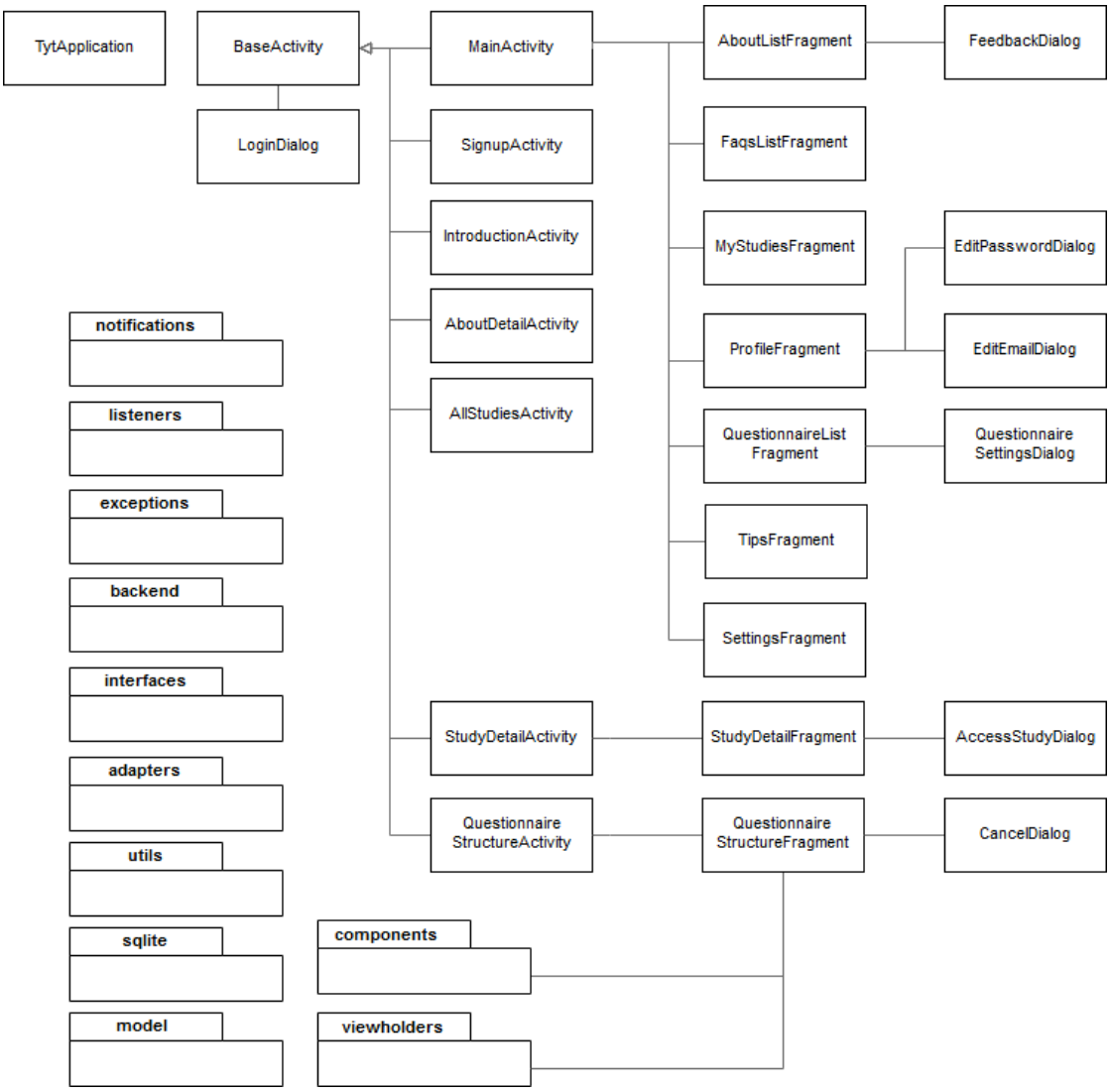


Abbildung 4.9: Vereinfachtes Klassendiagramm der Gesamtapplikation mit Fokus auf der Benutzerschnittstelle

Das Paket `listeners` enthält spezielle Listener-Klassen [4], die wiederholt verwendet werden können sollen. Alle Komponenten, die für die Kommunikation mit dem Backend erforderlich sind, sind in `backend` untergebracht und werden im nächsten Kapitel näher erläutert. Die weiterhin dargestellten Pakete dienen der Strukturierung der Komponenten und beinhalten hauptsächlich Helferklassen bzw. Datenmodelle.

4.2.2 Backendanbindung

Als elementarer Bestandteil der Android Applikation (Client) zählt die Anbindung an die REST-Schnittstelle des serverseitigen Track Your Tinnitus Backends.

Das Abrufen der Daten vom Backend wird durch Helferklassen realisiert, die im Package `synchronization` enthalten sind, wie Abbildung 4.10 zeigt. Die Helferklassen haben jeweils abgetrennte Aufgabenbereiche und sollen periodisch dazu aufgefordert werden können, Daten vom Backend abzurufen.

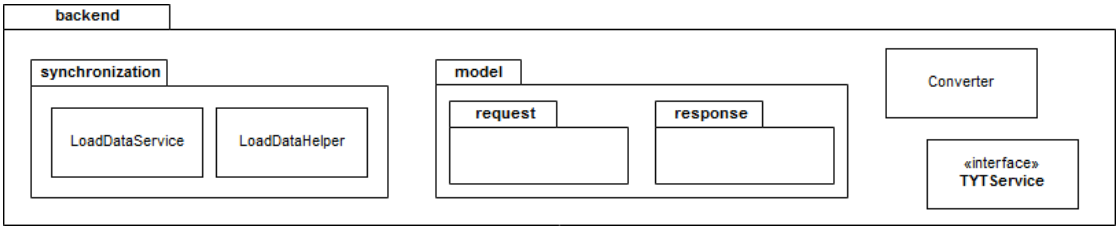


Abbildung 4.10: Vereinfachtes Klassendiagramm bzgl. der Backendanbindung

Als Austauschformat zwischen Clients und Server dient die JavaScript Object Notation (JSON). Clientseitig ist eine Abbildung des JSON Schemas auf Java Objekte hilfreich, um die Arbeit mit den Daten zu erleichtern. Die Java-Repräsentationen des Schemas befinden sich innerhalb des Packages `model`, unterteilt in Objekte für Anfragen (`requests`) und Antworten (`responses`). Das Datenmodell wird im nächsten Abschnitt 4.2.3 detailliert erläutert.

Das Interface `TYTService` repräsentiert schließlich die tatsächliche Schnittstelle zur Kommunikation mit dem Backend. Zusätzlich erforderliche Transformatoren für die

[4] Um Benutzerereignissen begegnen zu können, wird in Android das Event-Listener-Konzept angewendet. Tritt ein bestimmtes Ereignis ein, wird eine registrierte Methode aufgerufen. Siehe [27].

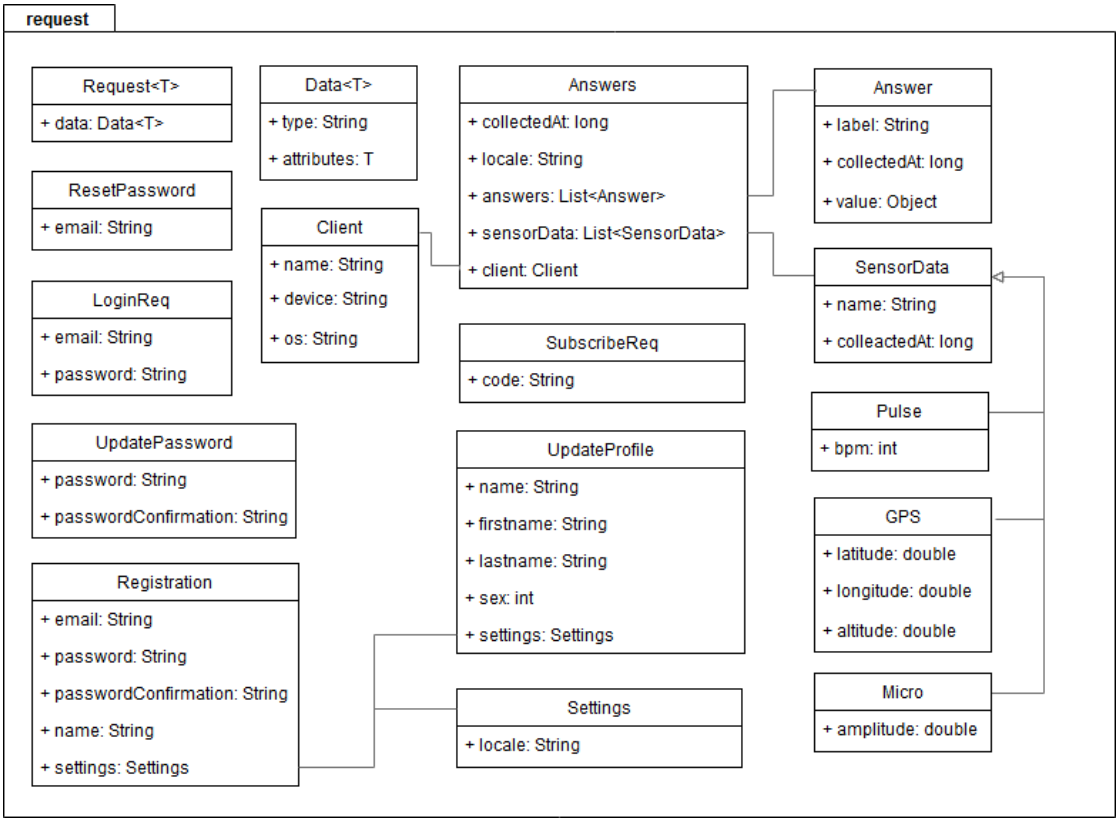


Abbildung 4.11: Klassendiagramm Request-Entitäten

Konvertierung des JSON Formats in Java Objekte werden ebenfalls innerhalb der Paketstruktur verortet.

4.2.3 Datenmodell

Alle Datenstrukturen, die zwischen der Applikation und dem REST-Backend ausgetauscht werden, wurden als Klassen realisiert. Es wurde als sinnvoll erachtet, diejenigen Objekte, die vom Backend bei Anfragen erwartet werden (Request-Objekte) und diejenigen, die vom Backend als Antwort zurückgesendet werden (Response-Objekte), zu unterteilen. Die Response-Objekte des Backends sollen der Applikation gleichzeitig als Datenmodell dienen und die Arbeit mit Inhalten erleichtern. Die grundsätzliche Struktur eines Response-Bodies des Backends besteht immer aus einem `data`, einem `meta` und einem `links` Block. Der `data` Block repräsentiert entweder eine Liste an Objekten,

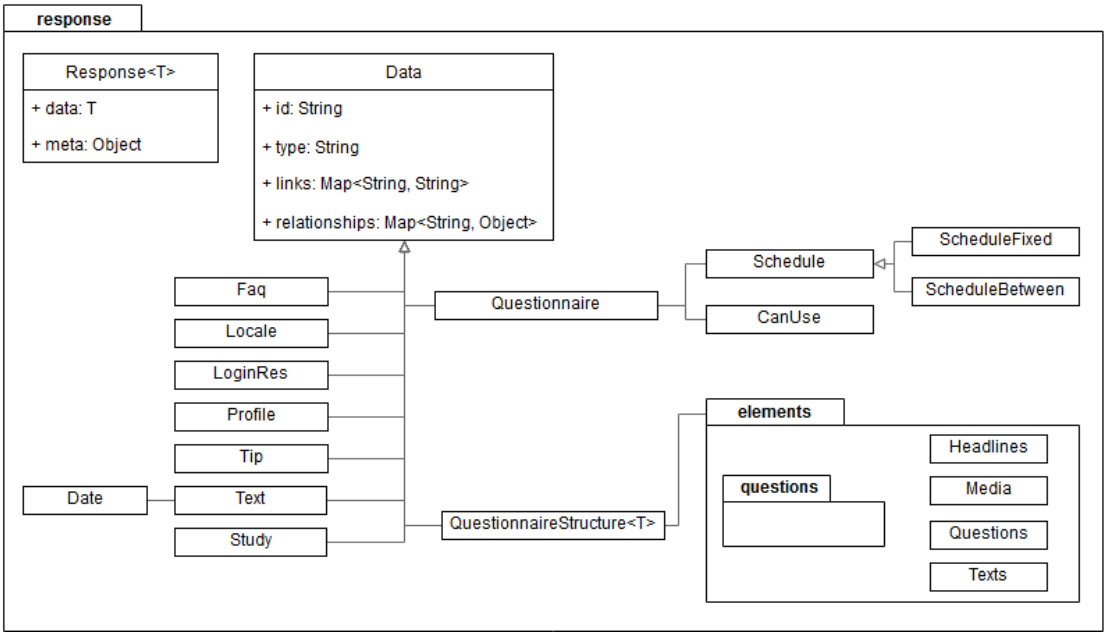


Abbildung 4.12: Vereinfachtes Klassendiagramm Response-Entitäten

oder ein tatsächliches Objekt. Jedes Objekt weist seinen Typ als Attribut (`type`) aus. Zudem enthält es eine Id (`id`), einen Link auf sich selbst (`links`), und zusätzliche Attribute (`attributes`), die für jeden Objekttyp unterschiedlich sind. Anhang A.1 zeigt die Objektstruktur als JSON für die Anfrage eines spezifischen Tipps am Backend. Auch die Request-Objekte weisen eine ähnliche Struktur auf, mit der Ausnahme, dass Links und Metadaten nicht erforderlich sind.

Aufgrund dieser einheitlichen Struktur wurde im Java Datenmodell mit generischen Typen gearbeitet[5]. Im Falle der Request-Klassen enthält die Data-Klasse einen generischen Typ, der das tatsächliche Objekt repräsentiert, z.B. `UpdateProfile`. Aus Gründen der einfacheren Verarbeitung ist bei den Response-Klassen der generische Typ direkt in der Request-Klasse enthalten, die einzelnen Typen erben dort von der Klasse Data. In Abbildung 4.11 ist das entsprechende Klassenmodell für Requests dargestellt. Das Klassenmodell für Responses ist aufgrund des Umfangs in reduzierter Form in Abbildung 4.12 abgebildet.

[5] Ein generischer Typ ist eine Klasse oder ein Interface, welches anhand von Typen parametrisiert wird. [28]

Fragebögen

Damit einfach neue Fragebögen hinzugefügt werden können, wurden bestimmte Fragebogenelemente erarbeitet, die unterschiedlichsten Anforderungen an Fragebögen gerecht werden sollen. Ein Fragebogen besteht grundsätzlich aus einer bestimmten Menge der folgenden Elemente in beliebiger Kombination:

- Überschrift (`Headlines`)
- Text (`Texts`)
- Medienobjekt (`Media`)
- Frage (`Questions`)

Dabei sind im Track Your Tinnitus Umfeld die nachfolgend genannten Fragetypen bekannt und müssen entsprechend ebenfalls modellierbar sein:

- Mehrfachauswahl (`MultipleChoice`)
- Einfachauswahl (`SingleChoice`)
- Ja-Nein-Schalter (`YesNoSwitch`)
- Textfeld einzeilig (`TextSingle`)
- Textfeld mehrzeilig (`TextBlock`)
- Textfeld Ganzzahl (`TextInt`)
- Textfeld Kommazahl (`TextFloat`)
- Schiebeauswahl (`Slider`)
- SAM Body (`SamBody`)
- SAM Face (`SamFace`)
- Datumsfeld (`Date`)
- Uhrzeitfeld (`Time`)
- Felder für Datum + Uhrzeit (`Datetime`)

Um diese Begebenheiten zu berücksichtigen, wurde, wie in Abbildung 4.12 bereits ersichtlich, die Klasse `QuestionnaireStructure<T>` als generischer Typ umgesetzt. Sie enthält das entsprechende Fragebogenelement (generischer Typ) und die zugehörige Bezeichnung des Typs als Enum-Konstante. Abbildung 4.13 zeigt den detaillierten Aufbau des Datenmodells bzgl. der Fragebogen-Entitäten. Auch die Klasse `Questions<T>`

wurde als generischer Typ implementiert, da alle Fragetypen einige ähnliche Attribute besitzen und so innerhalb der Liste einfacher verarbeitet werden können.

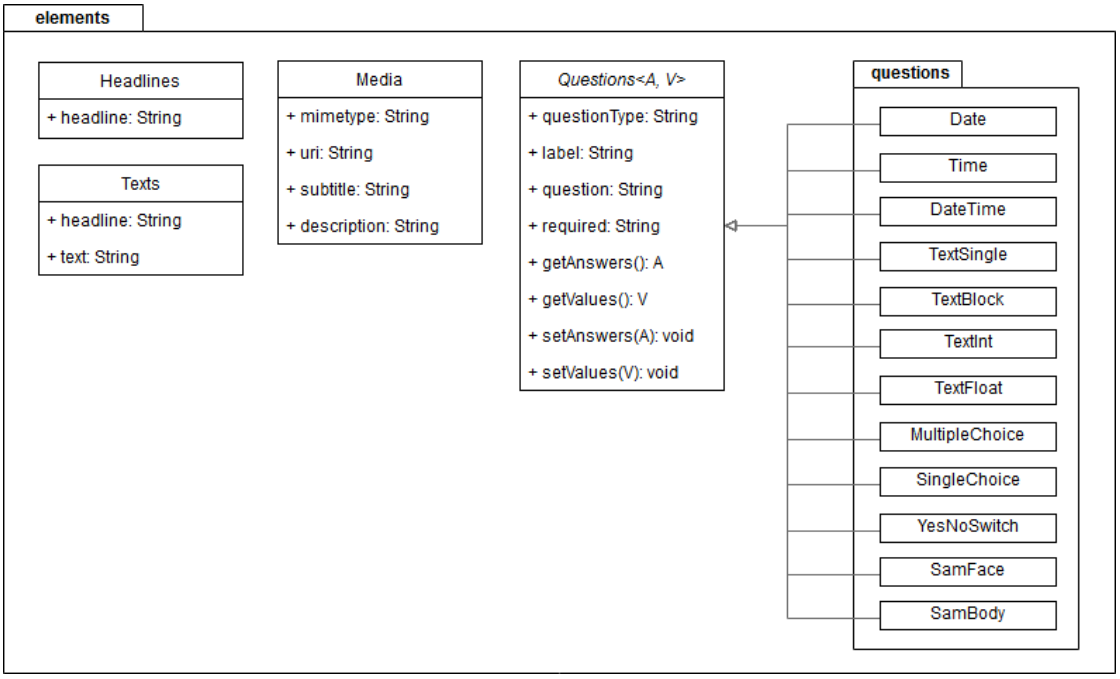


Abbildung 4.13: Vereinfachtes Klassendiagramm Fragebogen-Elemente

4.2.4 Datenbank

Die verwendete Datenbank innerhalb der Android Applikation ist eine SQLite Datenbank. Sie soll vom Backend abgerufene und vom Nutzer angegebene Daten speichern und so die Offline-Fähigkeit der App ermöglichen.

Android sieht vor, dass eine Subklasse des `SQLiteOpenHelper` die Kommunikation mit der Datenbank erledigt. [29] Diese Klasse legt auch fest, was bei einem Up- bzw. Downgrade der Datenbankversion bzw. einer Ersterstellung geschehen soll. Hier wurde zu diesem Zweck die Klasse `TytSQLiteOpenHelper` erzeugt. Für die Realisierung des Datenbankschemas wurden dedizierte Klassen erstellt, um Tabellen- und Spaltennamen sauber getrennt als Konstanten bereitzustellen. Diese befinden sich innerhalb des Paketes `sqlite.schema`. Das Datenbankschema wird in Abbildung 4.14 dargestellt.

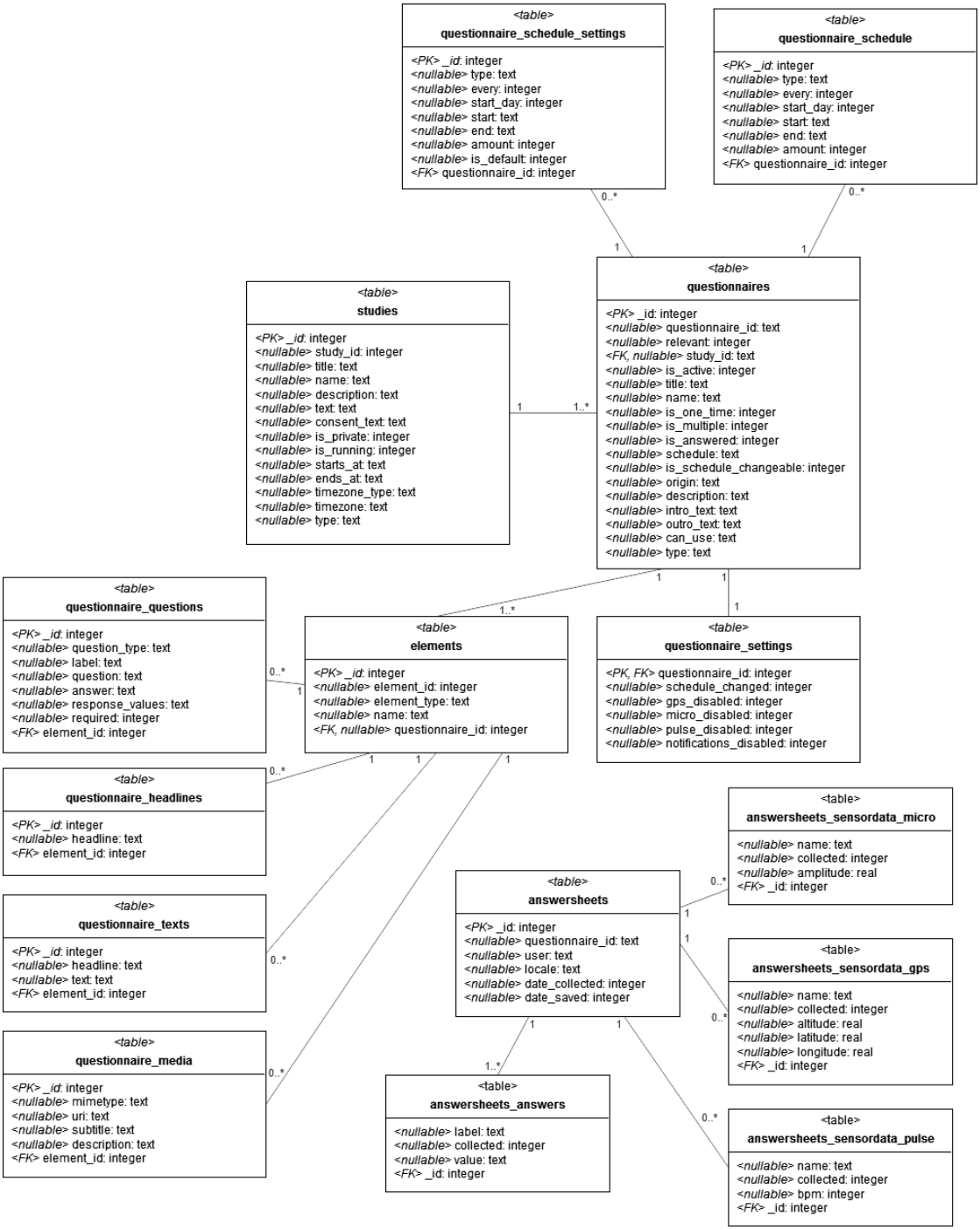


Abbildung 4.14: Datenbankschema

Schema

Vom Backend abgerufene Studien werden in der Tabelle `studies` abgelegt. Die Fragebögen befinden sich in der Tabelle `questionnaires`. Zu jedem Fagebogen werden die initial aktivierten Einstellungen, wie die Nutzung von GPS oder der Umgebungslautstärke in `questionnaire_settings` gespeichert. Jeder Fragebogen enthält Elemente (`elements`) wie Fragen, Texte, Überschriften oder Medienelemente, die in den Tabellen `questionnaire_questions`, `questionnaire_texts`, `questionnaire_headlines` und `questionnaire_media` abgelegt werden. Die einzelnen im vorigen Kapitel aufgezeigten Fragetypen werden nicht als separate Entitäten abgelegt sondern als JSON-Struktur innerhalb der `questionnaire_questions` Tabelle.

Die zu einem Fragebogen gehörenden Benachrichtigungseinstellungen werden in der Tabelle `questionnaire_schedule` abgelegt. Diese Tabelle ist nicht zu verwechseln mit der `questionnaire_schedule_settings` Tabelle, die einzig den tatsächlichen Nutzereinstellungen bzgl. der Benachrichtigungen vorbehalten ist. Die berechneten bzw. geplanten Benachrichtigungen werden in der Tabelle `schedules_notifications` gepflegt.

Nach Ausfüllen eines Fragebogens werden die Nutzer-Antworten in der Tabelle `answersheets` gespeichert. Zu jeder Entität in `answersheets` gehört eine Menge an Antworten (`answersheets_answers`) und Sensordaten, die in den Tabellen `answersheet_sensordata_micro`, `answersheet_sensordata_pulse` und `answersheet_sensordata_gps` persistiert werden.

4.2.5 Sensoren

Laut der Anforderungen sollen während des Fragebogenausfüllens drei verschiedene Messungen durchführbar sein. Da die auf dem Markt verfügbaren Smartphones großteils technisch sowohl eine Positionsbestimmung, als auch eine Tonaufnahme zulassen, wird für die Umsetzung der Anforderungen 19 und 15 auf die Möglichkeiten der Android Plattform hinsichtlich der Nutzung von Sensoren zurückgegriffen. Implementierungsdetails werden in Kapitel 5.5 näher beschrieben.

Um Anforderung 17 zu realisieren, ist jedoch ein weiteres Endgerät erforderlich, welches die Möglichkeit bietet, zuverlässig die Herzfrequenz des Nutzers zu erfassen. Die Sensoren eines Smartphones allein eignen sich hierfür nicht, da sie in keinem konstanten Kontakt zum Körper des Nutzers stehen und so keine zuverlässigen Messungen durchführen können. Besser eignen sich hier zum Beispiel Endgeräte, die am Handgelenk getragen werden können, wie Pulsuhren oder Fitnesstracker. Für die prototypische Umsetzung in der Track Your Tinnitus-App wurde der *GARMIN vívosmart HR+* Fitnesstracker verwendet. Er wurde aufgrund seiner vielversprechenden Funktionalität bezüglich der Herzfrequenz- und Positionsbestimmung zu Testzwecken von der TRI angeschafft und für diese Arbeit zur Verfügung gestellt.

Abbildung 4.15: GARMIN vívosmart HR+

Die *GARMIN vívosmart HR+* verfügt über eine *ANT+*-Schnittstelle wodurch eine einfache Verbindung zwischen Smartphone und Fitnesstracker (Device) hergestellt werden kann. ANT+ ist ein Drahtlos-Protokoll, welches die Kommunikation zwischen Endgeräten vereinheitlicht und so den Austausch von Sensordaten erlaubt. Die *Low-Power-Funktechnologie*[6] wird von der Firma *Dynastream Innovations* entwickelt und bereitgestellt, einer Tochter der *GARMIN Ltd*, und zielt schwerpunktmäßig auf die Erfassung und Übertragung von Sensordaten im Bereich Sport, Fitness, Wellness und Gesundheit ab. Alternative Low-Power-Funktechnologien sind *Bluetooth Low Energy* (BLE), *ZigBee*, *NFC* oder *Nike+*, die teilweise ähnliche Zielsetzungen verfolgen, aber im folgenden nicht näher erläutert werden sollen. Die Reichweite der ANT+-Technologie beträgt maximal 30 Meter, während die Informationssicherheit über eine 64-Bit-Verschlüsselung gewährleistet wird. Für die Datenübertragung wird das lizenzfreie 2,4 GHz-Frequenzband genutzt.

[6] Technologien, die sich durch besonders geringen Energieverbrauch auszeichnen.

Für die erfolgreiche Verbindung verschiedener Endgeräte über das ANT+-Protokoll müssen alle hardwareseitig ANT+ unterstützen. Zum Zeitpunkt der Erstellung der vorliegenden Arbeit unterstützten 715 Endgeräte das Protokoll, davon 126 Smartphones [7]. Dynastream stellt ein Software Development Kit (SDK) speziell für Android Geräte zur Verfügung, welches die Integration eines Fitnesstrackers mit der Android Applikation vereinfacht. Für die Verbindung mit einem ANT+-fähigen Device ist außerdem die Installation zweier Applikationen (*ANT Radio Service* und *ANT+ Plugins*) auf dem Android Smartphone erforderlich. In den meisten Fällen sind diese systemseitig auf dem Smartphone bereits vorinstalliert, andernfalls müssen sie über den Google Play Store installiert werden.

4.3 Gewählte Bibliotheken

Nachfolgend werden die eingesetzten Bibliotheken beschrieben, durch die bestimmte Programmieraufgaben vereinfacht bzw. abstrahiert werden.

4.3.1 Project Lombok

Als „Boilerplate“ wird Code bezeichnet, der an vielen verschiedenen Stellen wiederholt erforderlich ist, ohne dass er sich gravierend an Funktionalität unterscheidet. Automatisch generierte Getter und Setter sowie Konstruktoren sind typische Beispiele für Boilerplate Code in Java. *Project Lombok* verfolgt das Ziel, diesen Boilerplate Code durch einfache Annotations zu ersetzen und Programmierern so viele Zeilen an Sourcecode zu sparen.

```
1 @Data
2 @AllArgsConstructor
3 public class LoginReq {
4    private String email;
5    private String password;
6 }
```

Listing 4.1: Ein einfaches Java Object unter Verwendung von Lombok

[7] Details siehe `https://www.thisisant.com/directory/filter/`

Lombok erzeugt im Beispiel 4.1 die Getter und Setter der Mitglieder `email` und `password`, überschreibt die `equals`-, `hashCode`- und `toString`-Methoden und erzeugt einen zusätzlichen Konstruktor aller Mitglieder. Das kompilierte Java-Programm enthält schließlich alle relevanten Funktionalitäten.

Lombok wurde aufgrund der erheblichen Vereinfachung des Programmierprozesses und der durch Annotations erzielte Verringerung des Quellcodes in das Track Your Tinnitus Android Projekt integriert. Es steht unter der MIT Lizenz. [30]

4.3.2 Butterknife

Will man beim Entwickeln einer Android Applikation programmatisch auf Elemente wie Textfelder oder Drawables zugreifen, muss man diese mit Hilfe der Methode `findViewById` bzw. `findDrawableById` z.B. innerhalb der `onCreate`-Methode einer Activity referenzieren. Für die Android Applikation wurde das Framework Butterknife eingesetzt, um derartigen Boilerplate Code mit Hilfe einfacher Annotations zu verringern, wie Beispiel 4.2 verdeutlicht. Butterknife wird unter der Apache 2.0 Lizenz veröffentlicht. [31]

```
1 public class SignupActivity extends BaseActivity {
2     @BindView(R.id.register_title) TextView title;
3     @BindView(R.id.register_btn_signup) Button signupButton;
4 }
```

Listing 4.2: View-Binding mit Butterknife

4.3.3 Retrofit

Um die HTTP-Kommunikation mit dem Backend innerhalb der Applikation zu vereinfachen, wurde initial *Volley* in Betracht gezogen. *Volley* ist ein Open Source Projekt und wird als Teil der Android-Frameworkfamilie von Google für die HTTP-Kommunikation empfohlen [32].

Neben *Volley* erfreut sich in der Community jedoch auch das Framework *Retrofit* großer Beliebtheit. Es existieren unzählige Meinungen und Vergleiche zu *Retrofit* und *Volley* [33, 34, 35], jedoch scheint keines grundsätzlich besser zu sein.

Unter Berücksichtigung der Anforderungen wurde für diesen Anwendungsfall *Retrofit* präferiert, da es im Vergleich zu *Volley* einfacher und schneller zu konfigurieren ist, und für Standard-HTTP-Operationen in Benchmarks bessere Geschwindigkeiten erzielt. [36] Die zusätzlichen Funktionalitäten und Konfigurationsmöglichkeiten, die *Volley* bietet, sind in diesem Zusammenhang nicht erheblich. Retrofit steht ebenfalls unter der Apache 2.0 Lizenz. [31]

4.3.4 JodaTime

Um die Arbeit mit Datum und Uhrzeit zu erleichtern, wurde *JodaTime* als zusätzliche Bibliothek gewählt. *JodaTime* vereinfacht insbesondere für Android Entwickler auch die Arbeit mit Zeitzonen, die unter Verwendung der Standard-Java-API im App-Kontext Probleme verursachen kann. Eine vollständige Version der *JodaTime*-Bibliothek angepasst an die Android Plattform wird von Daniel Lew unter der Apache 2.0 Lizenz bereitgestellt. [37, 31]

4.3.5 Google Firebase

Für das Monitoring und Nutzeranalyse der Applikation wurde *Google Firebase* in die Applikation integriert. *Google Firebase* erlaubt die automatische Erkennung von Events, wie Fehlern und Nutzeraktionen [38] und ist dadurch insbesondere für den Produktivbetrieb interessant, um Einblicke in das Applikationsverhalten auf entfernten Endgeräten zu erlangen. Im Speziellen wurden die Bibliotheken `com.google.firebase:firebase-core` für Analytics und `com.google.firebase:firebase-crash` für die Meldung von Fehlerfällen und Abstürzen integriert.

4.3.6 Android Support

Um die geforderte Abwärtskompatibilität zu erreichen, wurde auf einige der Android Support Bibliotheken zurückgegriffen, die sich um die Bereitstellung äquivalenter Funktionalitäten für neue Features auf älteren Versionen kümmern. [39]
Für die Darstellung langer Listen wurde die Support-Bibliothek `Recyclerview` verwendet. Außerdem wurden die Bibliotheken `com.android.support:appcompat-v7` und `com.android.support:design` für den kompatiblen Einsatz neuerer Designelemente wie der *Toolbar* oder dem *Swipe-Refresh-Layout* integriert. Die Android Support Bibliotheken stehen unter der Apache 2.0 Lizenz. [31]

4.3.7 Apache Commons Lang

Apache Commons Lang stellt unter der Apache 2.0 Lizenz Helfer-Klassen für die Arbeit mit der `java.lang`-API bereit. [40, 31] Sie wurde vorrangig integriert, da sie auch Helfer für die Erzeugung der Standard-Methoden `equals`, `hashCode` und `toString` enthält.

4.3.8 ANT+ PluginLib

Für die Implementierung der ANT+-Schnittstelle wurde auf das Dynatrace Android ANT+ SDK zurückgegriffen, welches die *ANT+ PluginLib* beinhaltet (lizensiert unter der FIT Protocol License [41]). Diese Bibliothek enthält Interfaces und Klassen, die die Kommunikation mit anderen ANT+-fähigen Geräten abstrahieren und vereinfachen. Die Verbindung zu Pulsuhren oder Fitnesstrackern ist so einfacher realisierbar.

5

Ausgewählte Implementierungsaspekte

Nachfolgend werden ausgewählte Aspekte der Implementierung näher beschrieben und anhand von Beispielen und Schaubildern erläutert. Der Fokus wird auf die technisch aufwändigeren bzw. fachlich komplexeren Aspekte gelegt. Java- oder Androidspezifische Vorgehensmodelle und Patterns werden dabei nicht näher erläutert, da sie offiziellen Dokumentationen und zahlreichen Publikationen entnommen werden können.
Zur Verdeutlichung der Implementierung der einzelnen Aspekte wird auf Codeausschnitte zurückgegriffen, die anhand von Listings dargestellt werden. Dabei ist zu berücksichtigen, dass diese Codeausschnitte keinen Anspruch auf Lauffähigkeit bzw. Vollständigkeit erheben, da sie ausschließlich dem besseren Verständnis der Umsetzung dienen sollen. Für eine bessere Übersicht wurden daher in den Ausschnitten einzelne weniger komplexe Bestandteile wie zum Beispiel Variablendeklarationen bzw. Initialisierungsschritte entfernt.

5.1 Backendkommunikation

Das gewählte Framework *Retrofit* erlaubt die einfache Erstellung eines Interfaces, welches innerhalb der Applikation die Schnittstelle zum Backend repräsentiert. Es erfordert sehr wenig Konfiguration, um mit dem Backend zu kommunizieren. Im Interface `TYTService` wurden einmalig alle Routen der REST-API hinterlegt, die entsprechenden Parameter definiert und Request- und Response-Modelle festgelegt. Anschließend konnte die Schnittstelle zum Backend bereits fehlerfrei verwendet werden. Listing A.3 zeigt das `TYTService` Interface. Erhöhten Konfigurationsaufwand bereiteten vor allen Dingen die automatische Konvertierung von Objektmodellen und die Implementierung

des Datenabrufs und der Synchronisierung mit dem Backend, welche im nachfolgenden genauer beschrieben werden.

5.1.1 Konverter

Das Framework *Retrofit* stellt einige Standard-Konverter für die Transformation der Java-Modelle zu Request- bzw. Response-Objekten bereit. Da die REST-API mit JSON arbeitet, wurde für die Konvertierung von und zu JSON-Strings der *Retrofit gson*[1]-Konverter ausgewählt. Für die Arbeit mit den meisten Antworten der Track Your Tinnitus REST-API genügt bereits die Standardfunktionalität des gson Konverters. Für Fragebogendetails und die Fragebogenstruktur waren allerdings zusätzliche Konverter erforderlich, da das Datenmodell hier vom Inhalt der Daten abhängt und aus diesem Grund eine automatische Serialisierung der JSON-Struktur zu Java Objekten nicht möglich ist. Nachfolgend werden die beiden zusätzlich benötigten Konverter näher beschrieben.

Die Details der Fragebögen enthalten eine Liste an so genannten Schedules. Ein `schedule` ist entweder vom Typ `between` oder vom Typ `single`. Abhängig davon beinhaltet das Objekt die Attribute `amount`, `start` und `end` bzw. das Attribut `at`. Um aus dieser Struktur das entsprechende Objekt zu generieren, wurde der in Listing 5.1 abgebildete Konverter implementiert, der Antworten der REST-API unter der Route `questionnaires/{id}` automatisiert verarbeiten kann und die entsprechende Objekte der Klasse `ScheduleBetween` bzw. `ScheduleFixed` erzeugt. Er benötigt dabei nur Zusatzfunktionalität für die Konvertierung der Schedules, die in Zeile 10 ff. durchgeführt wird. Alle anderen Attribute können automatisch vom Konverter serialisiert werden (siehe Zeile 8).

```
1 public class QuestionnaireDetailsResponseConverter implements
      JsonDeserializer<Questionnaire> {
2
3     @Override
4     public Questionnaire deserialize(JsonElement json, Type typeOfT,
          JsonDeserializationContext context) throws JsonParseException {
5         JsonObject data = json.getAsJsonObject();
```

[1] google-gson ist eine Java-Bibliothek, die dafür verwendet werden kann, Java Objekte in ihre JSON Representationen und umgekehrt zu verwandeln

```
6
7         Gson gson = new Gson();
8         Questionnaire result = gson.fromJson(data, Questionnaire.class);
9
10        //  Overwrite the schedules as they cannot be serialized properly
              automatically
11        JsonElement scheduleElement = data.get(KEY_ATTRIBUTES)
              .getAsJsonObject().get(KEY_SCHEDULE);
12        JsonArray schedules = scheduleElement.getAsJsonArray();
13
14        List<Schedule> resultSchedules = new ArrayList<>(schedules.size());
15        for (JsonElement schedule : schedules) {
16           Schedule resultSchedule;
17           JsonObject scheduleObject = schedule.getAsJsonObject();
18
19           switch (scheduleObject.get(Schedule.KEY_TYPE).getAsString()) {
20              case Schedule.TYPE_BETWEEN:
21              ScheduleBetween scheduleBetween = new ScheduleBetween();
22              // Set attributes start, end and amount
23              resultSchedule = scheduleBetween;
24              break;
25              case Schedule.TYPE_SINGLE:
26              ScheduleFixed scheduleFixed = new ScheduleFixed();
27              // Set attribute at
28              resultSchedule = scheduleFixed;
29              break;
30              default:
31              resultSchedule = gson.fromJson(schedule, Schedule.class);
32              break;
33           }
34           // Set common attributes type, every and startDay
35           resultSchedules.add(resultSchedule);
36        }
37        result.getAttributes().setSchedule(resultSchedules);
38        return result;
39     }
40  }
```

Listing 5.1: QuestionnaireDetailsResponseConverter

Auch die Struktur eines Fragebogens, die unterhalb der Route `questionnaires/{id}/structure` von der REST-API ausgeliefert wird, kann verschiedene Elemente enthalten. Die Elemente Headline, Question, Media und Page sind durch das JSON-Attribut `type` in der Antwort vom Server eindeutig voneinander zu unterscheiden und folgen bestimmten vorgegebenen Strukturen. Der Typ Question kann dabei selbst wiederum unterschiedliche Ausprägungen, wie z.B. SingleChoice, MultipleChoice, SamFace oder TextSingle, haben.

Um auch diese JSON-Objekttypen automatisiert in die vorgesehenen Java-Objekte (siehe Kapitel 4.2.3) konvertieren zu können, wurde der `QuestionnaireStructureResponseConverter` hinzugefügt. Er deserialisiert auf Basis des `type` Attributes das eingehende JSON-Objekt, erstellt daraufhin ein entsprechendes Java Objekt und befüllt es mit den vorhandenen Attributen. So wird sichergestellt, dass das eingehende JSON-Objektarray – die Fragenliste – in ein Java Array transformiert wird, welches die tatsächlichen Strukturobjekte enthält. Dadurch wird die weitere Arbeit mit den Fragebögen in der Applikation erleichtert.

5.1.2 Datenabruf und -synchronisierung

Für die Umsetzung des Datenabruf- und Synchronisierungsprozesses wurde zunächst die nachfolgend beschriebene Unterscheidung der Daten vorgenommen. Darauf aufbauend wurde ein entsprechender Synchronisierungsprozess erarbeitet und realisiert.

Verzichtbare Daten

Verzichtbare Daten sind für die Verwendung der App nicht zwingend erforderlich. Es genügt, sie immer dann vom Server abzurufen, wenn der Nutzer eine entsprechende Anzeige aufruft. Eine lokale Ablage der Daten ist daher nicht erforderlich.
Zu den verzichtbaren Daten gehören: Alle weiteren verfügbaren Studien, in die der Nutzer nicht eingeschrieben ist; Fragebogendetails von weiteren Studien; FAQ; Tipps.

Für die verzichtbaren Daten musste kein spezieller Synchronisierungsprozess modelliert werden, da die Daten immer dann vom Backend abgerufen können, wenn der Nutzer

die entsprechende Ansicht öffnet, vorausgesetzt, zu diesem Zeitpunkt besteht eine Internetverbindung. Für den Abruf dieser Daten ihrer verzichtbaren Daten ist daher jede Activity bzw. jedes Fragment selbst verantwortlich.

Unverzichtbare Daten

Unverzichtbare Daten sind für die Verwendung der App zwingend erforderlich und müssen auch im Offline-Modus zugreifbar sein bzw. vorgehalten werden. Diese Daten müssen bei jedem erfolgreichen Synchronisierungsprozess lokal abgelegt werden und so auch ohne Internetverbindung verfügbar sein, gegebenenfalls mit einer gewissen Abweichung zum Datenstand auf dem Server. In der App erzeugte Daten, die an das Backend zurückgesendet werden sollen, müssen ebenfalls vorgehalten werden, bis eine Übertragung erfolgen kann.

Ereignis	**Aktion**
Erfolgreicher Login	• Studien des Nutzers & Studiendetails abrufen (Zeit) • Studien-Fragebögen, Fragebogendetails & -struktur abrufen (Zeit) • Ausgefüllte Fragebögen versenden • Nutzerdaten & Texte abrufen (Zeit)
Rückkehr zur App	• Studien des Nutzers & Studiendetails abrufen (Zeit) • Studien-Fragebögen, Fragebogendetails & -struktur abrufen (Zeit) • Nutzerdaten & Texte abrufen (Zeit) • Ausgefüllte Fragebögen versenden
Anzeige der Fragebogenübersicht	• Studien-Fragebögen, Fragebogendetails & -struktur abrufen (keine Daten)
Anzeige der Studienübersicht	• Studien des Nutzers & Studiendetails abrufen (keine Daten) • Studien-Fragebögen, Fragebogendetails & -struktur abrufen (keine Daten)
Anzeige des Profils	• Nutzerdaten & Texte abrufen (keine Daten)

Tabelle 5.1: Ereignisse in Verbindung mit der Datensynchronisierung unverzichtbarer Daten

Zu den unverzichtbaren Daten gehören: Studien, in die der Nutzer eingeschrieben ist; alle aktiven Fragebögen von Studien, in die der Nutzer eingeschrieben ist; Profildaten; durch den Benutzer ausgefüllte Fragebögen (Nutzerantworten).

Für die unverzichtbaren Daten wurden zunächst die Ereignisse identifiziert, zu denen der Synchronisierungsprozess starten soll. Tabelle 5.1 stellt diese Ereignisse den abzurufenden bzw. zu versendenden Daten gegenüber. Dabei ist zu berücksichtigen, dass für jeden erfolgreichen Synchronisierungsprozess eine Internetverbindung bestehen und der Nutzer eingeloggt sein muss. Die in der Tabelle hinter den Aktionen in Klammern angegebenen Informationen kennzeichnen weitere Bedingungen, die erfüllt sein müssen, damit die Synchronisation stattfinden kann. „Zeit“ bedeutet dabei, dass der Synchronisierungszeitpunkt erreicht sein muss und „keine Daten“, dass die Aktion nur durchgeführt wird, wenn auf dem Endgerät nicht bereits entsprechende Daten aus früheren Synchronisierungsprozessen existieren.

Realisierung

Die `BaseActivity` beinhaltet bestimmte Basis-Funktionalitäten, die von allen Activities verwendet werden können, indem sie von der `BaseActivity` erben. So enthält sie eine Methode `syncDataIfRequired`, die überprüft, ob die in der Tabelle 5.1 beschriebenen Voraussetzungen gegeben sind, um bestimmte Daten abzurufen bzw. zu übertragen. Sind die Voraussetzungen entsprechend erfüllt, triggert sie den `LoadDataService` bzw. fordert den `AnswersheetHelper` dazu auf, überfällige Fragebögen ans Backend zu senden. Die `syncDataIfRequired` Methode wird neben dem Aufruf durch einen erfolgreichen Login vorrangig von der `MainActivity` bei Rückkehr zur App (`onResume`) oder beim Pausieren (`onPause`) der App aufgerufen.

Die Steuerung des Datenabrufs am Backend übernimmt der `LoadDataService`. Als Android Service kann er im Hintergrund laufen ohne Behinderung des Main-Threads und daraus resultierender Verlangsamung der App. [42] Das Abrufen wird, wie in Kapitel 4.2.2 beschrieben, auf unterschiedliche Helper aufgeteilt, die der `LoadDataService` auffordert, die Daten vom Backend zu laden. Die Helper schreiben die geladenen Daten selbstständig mit Hilfe von AsyncTasks in die Datenbank. [43] Dies fördert die

Entkopplung vom Main-Thread zusätzlich. Da alle Komponenten, die unverzichtbare Daten darstellen (`QuestionnaireListFragment`, `MyStudiesFragment`, `QuestionnaireStructureFragment`, etc.), ihre Informationen aus der Datenbank beziehen, erhalten sie bei einem Abruf immer die zuletzt mit Hilfe des `LoadDataService` synchronisierten Daten. Befinden sich keine Daten in der Datenbank, triggern diese Komponenten den Datenabrufprozess direkt, ohne Verwendung des Service.

5.2 Authentifizierung und Offline-Modus

5.2.1 Nutzerauthentifizierung am Backend

Die REST-API stellt umfangreiche Funktionalitäten zum Login und zur Validierung eines Benutzers bereit. Die Authentifizierung erfolgt Token-basiert, das bedeutet, bei einmaligem Login erhält der Nutzer ein Token, welches für darauffolgende Requests an das Backend mitgeliefert wird. Es wird in den SharedPreferences [2] der Android-Applikation abgelegt. So muss seltener ein Passwort an die REST-API übertragen werden. Für die Realisierung wurde der *JSON Webtoken* (JWT) Standard eingesetzt. Das serverseitig generierte JWT lässt sich clientseitig dekodieren, um die Gültigkeit des Tokens erkennen zu können. Wird das Ablaufdatum des Tokens erreicht, kann innerhalb eines bestimmten Zeitraums an der REST-API ein neues Token erfragt werden. Kann kein neues Token generiert werden, ist der Nutzer gezwungen, sich erneut mit seinem Nutzernamen und Passwort einzuloggen.

5.2.2 Login erforderlich

Aufgrund der nichtfunktionalen Anforderung 5 wurde bei der Realisierung der App festgelegt, dass das Passwort des Nutzers nicht auf dem Endgerät gespeichert werden soll. Der Nutzer muss, sofern kein gültiges Token mehr an der REST-API beschafft werden

[2] SharedPreferences ermöglichen in Android die einfache Speicherung einer geringen Anzahl von Key-Value-Paaren im Applikationskontext. [44]

kann, seine Nutzerdaten erneut eingeben. Daraus ergibt sich für den Login-Prozess folgende Implementierung:

```
1  public abstract class BaseActivity extends AppCompatActivity {
2     private SharedPreferences prefs;
3
4     protected boolean isTokenExpired() {
5        try {
6           long expires = prefs.getLong(
                getString(R.string.preference_token_expiration), 0);
7           if (expires == 0) return true;
8
9           DateTime expirationDate = new DateTime(expires * 1000L,
                DateTimeZone.UTC);
10          return expirationDate.isBefore(new DateTime());
11       } catch (Exception e) {
12          // No expiration date for token found
13          return true;
14       }
15    }
16 }
```

Listing 5.2: Überprüfung des Ablaufdatums eines Tokens durch die BaseActivity

Vor dem Abruf von Daten an der REST-API wird überprüft, ob ein gültiges Token vorliegt. Die Funktionalität hierfür stellt die `BaseActivity` bereit, wie in Listing 5.2 dargestellt. Ist dies der Fall, können die Daten direkt mit Hilfe dieses Tokens abgerufen werden. Ist dies nicht der Fall, wird versucht, an der REST-API ein neues Token zu erzeugen, mit welchem zukünftig gearbeitet werden kann (siehe Listing 5.3). Gelingt es nicht, automatisch ein neues Token zu erzeugen, wird der Nutzer zu Eingabe seiner Nutzerdaten aufgefordert. Diesen Vorgang regelt ebenfalls die `BaseActivity` innerhalb der `showLoginDialog` Methode.

Bei einem Logout aus der App wird das aktuelle Token aus den `SharedPreferences` gelöscht, wodurch automatisch ein erneuter Login zwingend erforderlich ist.

```
1  public abstract class BaseActivity extends AppCompatActivity {
2     private SharedPreferences prefs;
3
4     private void tryRefresh(final Intent intent) {
```

```
5         final Call<Response<LoginRes>> refreshCall = tytService.refresh(
              prefs.getString(getString(R.string.preference_token), ""));
6         refreshCall.enqueue(new RefreshCallback(intent));
7     }
8 }
```

Listing 5.3: Anfrage nach einem neuen Token am Backend

5.2.3 Offline-Modus

Ein Nutzer soll laut Anforderung 2 in einen Offline-Modus wechseln können. Der Offline-Modus wird explizit durch einen Nutzer aktiviert und verhindert strikt die Verbindung mit dem Internet. Daten können in diesem Zustand nicht vom Backend abgerufen bzw. an das Backend übertragen werden. Der Offline-Modus kann erst aktiviert werden, wenn alle unverzichtbaren Daten erstmalig vom Server abgerufen und in der Datenbank hinterlegt wurden.
Neben dem Offline-Modus existiert ein weiterer Zustand, während dem ein Abrufen von Daten an der REST-API ebenfalls nicht möglich ist: Eine fehlende bzw. mangelhafte Internetverbindung aufgrund der Nichtverfügbarkeit eines Netzwerkes. Es handelt sich hierbei um einen impliziten Zustand. Bezüglich der unverzichtbaren Daten werden beide Zustände gleichermaßen behandelt. Sollen verzichtbare Daten innerhalb eines der beiden Zustände dargestellt werden, wird dem Nutzer ein entsprechender Fehlertext angezeigt.

Aktiviert ein Nutzer den Offline-Modus in den Einstellungen, wird dieser Zustand in den `SharedPreferences` der Applikation gespeichert. Die `BaseActivity` stellt eine Methode bereit, die von ihren Tochterklassen verwendet werden kann, um zu überprüfen, ob der Offline-Modus aktiviert ist (siehe Listing 5.4).

```
1 public abstract class BaseActivity extends AppCompatActivity {
2     private SharedPreferences prefs;
3
4     public boolean isOfflineModeActive() {
5         boolean active = prefs.getBoolean(
              getString(R.string.preference_offline_mode), false);
6         long lastSynced = prefs.getLong(getString(R.string.last_synced), 0);
```

```
7         if (active && DateTime.now().isAfter(lastSynced +
              Constants.OFFLINE_MODE_EXPIRES_PERIOD)) {
8             // OfflineMode is not possible anymore!
9             return false;
10        }
11        return active;
12    }
13 }
```

Listing 5.4: Überprüfung des Offline-Modus vor Verbindungsaufbau

Damit die Daten auf dem mobilen Endgerät nicht in unvertretbarem Maße von den aktuellen Daten auf dem Server abweichen, ist die maximale Dauer des Offline-Modus begrenzt (siehe Zeile 7). Nach Erreichen des Limits ist der Nutzer gezwungen, den Offline-Modus zu deaktivieren. Kann mit dem zuletzt vorliegenden Token kein neues Token an der REST-API beschafft werden, muss der Benutzer sich bei bestehender Internetverbindung erneut einloggen, damit alle unverzichtbaren Daten aktualisiert werden können. Vorher kann er die App nicht weiter bedienen.

5.3 Text und Lokalisierung

Track Your Tinnitus ist in mehreren Sprachen verfügbar. Damit Texte einheitlich gepflegt und auf unterschiedlichen Endgeräten wiederverwendet werden können, ohne dass Softwareänderungen erforderlich sind, werden sämtliche Informationstexte und die Texte der Bedienelemente vom Backend über die REST-API bereitgestellt.

Die Android Plattform sieht ein eigenes Konzept für die Pflege und Bereitstellung lokalisierter Texte vor. Allerdings erfordert dieses die statische Ablage der Texte in den Extensible Markup Language (XML)-Ressourcen der Android Applikation. [45] Im Fall einer Bereitstellung der Texte über eine REST-Ressource liegen die Texte aber zum Zeitpunkt des Kompilierens noch nicht vor, sondern können erst zur Laufzeit abgerufen und eingespielt werden. Deshalb wird ein abweichendes Konzept für das Management und die Bereitstellung der lokalisierten Texte innerhalb der App benötigt.

Zwingend erforderlich ist der einmalige Abruf der Texte im JSON-Format an der REST-Schnittstelle. Der Abruf sollte daraufhin in regelmäßigen Abständen wiederholt werden, um auch Änderungen an Texten darstellen zu können. Diese Aufgabe wird durch ein Objekt namens `LoadStringsHelper` übernommen.

Die Frage, wie die Texte innerhalb der Applikation gehalten werden, eröffnete grundsätzlich drei Möglichkeiten:

Option	**Vorteile**	**Nachteile**
1. Ablage in Dateien (Flat Files) auf dem Plattenspeicher	Ablage im JSON-Format einfach möglich, Internal App Storage als dedizierter Speicherbereich für Apps vorhanden	Filter und Suche aufwendig, Zugriff langsamer im Vergleich zu 3.
2. Ablage in der SQLite-Datenbank auf dem Plattenspeicher	Filter und Suche bei entsprechender Datenstruktur trivial	Ablageformat der Daten (JSON in SQLite nicht optimal), Zugriff langsamer im Vergleich zu 3.
3. Ablage im Java Heap (Random-Access Memory (RAM))	Schneller Zugriff	Speicherplatzprobleme bei vielen bzw. großen Texten, Inkonsistenzen durch gemanagten Heap, Neu-Abruf bei jedem Starten der App zwingend erforderlich

Tabelle 5.2: Möglichkeiten zur Persistierung geladener Texte in der App

Eine Filterung und Suche nach entsprechenden Texten war zum Zeitpunkt der Umsetzung nicht erforderlich. Um die übrigen Vorteile der Optionen zu kombinieren, wurde für die Realisierung eine Kombination aus Option eins und drei gewählt.

Ein dediziertes Objekt, als `StringsManager` bezeichnet, ist innerhalb der Applikation dafür zuständig, die durch den `LoadStringsHelper` empfangenen Texte in JSON-Dateien zu schreiben, sie beim Starten der App in den Zwischenspeicher zu laden und anfragenden Komponenten, wie Activities, Fragments oder Dialogen die entsprechenden Inhalte bereitzustellen. Der `StringsManager` behält außerdem einen Überblick über die vorhandenen Sprachen und kennt die aktuell Gewählte. Existiert ein angefragter Text in der gewählten Sprache nicht, stellt er einen Standardtext bereit, den er aus einer im Programm integrierten Datei lädt. Innerhalb des Applikationskontexts ist nur eine Instanz

dieses Objekts erforderlich, weshalb sich die Umsetzung als Singleton [3] empfiehlt. Der `StringsManager` kann so jederzeit von mehreren Applikationskomponenten gleichzeitig eingebunden und verwendet werden. Abbildung 5.1 zeigt den Ablauf anhand eines Unified Modeling Language (UML)-Sequenzdiagramms.

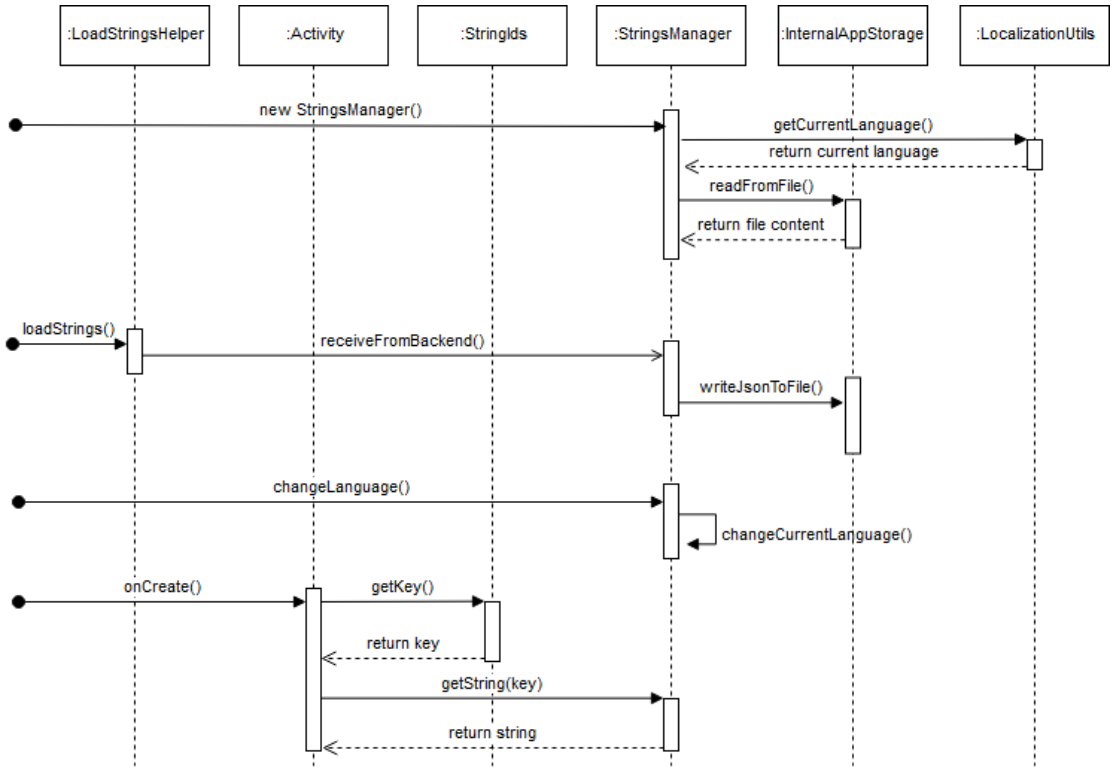


Abbildung 5.1: Sequenzdiagramm: Ablauf des Textabruf und der Textbereitstellung

5.4 Benachrichtigungen

Eine wichtige Anforderung an die Track Your Tinnitus App war bereits in der ersten Version von Jochen Herrmann die Erinnerungsfunktion an das Ausfüllen von Fragebögen. [4] Diese Anforderung wurde für die Umsetzung im Rahmen der vorliegenden Arbeit erweitert, wie in Kapitel 3.1 beschrieben (siehe Anforderungen 21 und 22).

[3] Beim Singleton-Entwurfsmuster wird sichergestellt, dass nur ein einziges Objekt einer Klasse erzeugt werden kann. Somit wird ein zentraler Zugriffspunkt geschaffen. [46]
[4] Für Implementierungsdetails der ersten Version der Android App siehe [1], Kapitel 5.2.2.2.

Wenn ein Studienleiter auf der Webseite einen neuen Fragebogen anlegt oder einen Bestehenden ändert, kann er bestimmte Zeitpläne definieren. Diese Zeitpläne (`schedules`) werden im Track Your Tinnitus Backend gespeichert und sind daraufhin in der Antwort der REST-API bei der Abfrage von Fragebögen durch Clients enthalten. Sie werden im Nachfolgenden als *vordefinierte Zeitpläne* bezeichnet.

Der grundsätzliche Aufbau eines jeden Zeitplans ist nachfolgend beschrieben.

5.4.1 Aufbau von Benachrichtigungszeitplänen

Ein Zeitplan kann eine von zwei Ausprägungen besitzen (`fixed` und `between`), die sich auf bestimmte Uhrzeitangaben beziehen. Jeder der beiden Typen besitzt einen Starttag (`startDay`), angegeben entsprechend der ISO Norm 8601 als Zahl zwischen 0 und 7, und einen Rhythmus (`every`), der angibt, in welchem Tagesrhythmus die Benachrichtigungen erfolgen sollen. Der Typ `between` ist für eine definierte Anzahl dynamisch generierter Benachrichtigungen innerhalb eines bestimmten Zeitraumes gedacht. Beim Typ `fixed` soll die Benachrichtigung zu einer fest vorgegebenen Uhrzeit erfolgen.

Mit diesen beiden Zeitplan-Typen lassen sich alle denkbaren Benachrichtigungskonstellationen abbilden. Die nachfolgende Liste zeigt einige Beispiele und deren Umsetzung:

- **Jeden Montag und Mittwoch um 9 Uhr**
  1. StartDay: 1, Every: 7, Type: Fixed, At: 9:00
  2. StartDay: 3, Every: 7, Type: Fixed, At: 9:00
- **Immer Sonntags drei Mal zwischen 10 und 18 Uhr**
  1. StartDay: 7, Every: 7, Type: Between, Start: 10:00, End: 18:00, Amount: 3
- **Wochentags einmal zwischen 16 und 20 Uhr**
  1. StartDay: 1, Every: 7, Type: Between, Start: 16:00, End: 20:00, Amount: 1
  2. StartDay: 2, Every: 7, Type: Between, Start: 16:00, End: 20:00, Amount: 1
  3. StartDay: 3, Every: 7, Type: Between, Start: 16:00, End: 20:00, Amount: 1
  4. StartDay: 4, Every: 7, Type: Between, Start: 16:00, End: 20:00, Amount: 1

5. StartDay: 5, Every: 7, Type: Between, Start: 16:00, End: 20:00, Amount: 1

- **Ab sofort jeden zweiten Tag um 8 Uhr und um 20 Uhr**

1. StartDay: 0, Every: 2, Type: Fixed, At: 08:00
2. StartDay: 0, Every: 2, Type: Fixed, At: 20:00

5.4.2 Realisierung von Benachrichtigungen in der Android App

Das Abrufen aller Fragebogendetails, darunter auch der Benachrichtigungszeitpläne, erfolgt, wie in Kapitel 5.1.2 beschrieben, periodisch zu definierten Zeitpunkten. Die abgerufenen Zeitpläne wiederkehrender Fragebögen werden in der lokalen Datenbank des Endgerätes abgelegt.

Die Aufgabe, aus vorliegenden Zeitplänen (`schedules`) tatsächliche Benachrichtigungen (`notifications`) zu generieren, übernimmt der `TytNotificationsManager`. Er pflegt alle generierten Benachrichtigungen in der Tabelle `schedules_notifications` innerhalb der SQLite-Datenbank. Der Zustand dieser Tabelle wird durch neue bzw. gelöschte Zeitpläne geändert, wie in Kapitel 5.4.5 beschrieben. Auch ändert er sich, wenn sich ein Benutzer in eine neue Studie einschreibt oder Fragebögen aufgrund des Verlassens einer Studie wegfallen. Der `TytNotificationsManager` kann anhand der Tabelle immer Auskunft über den aktuellen Stand der Benachrichtigungen geben und den Android `AlarmManager` auf dieser Basis mit dem tatsächlichen Aufsetzen (`setRepeating`) oder Beenden (`cancel`) der Benachrichtigungen beauftragen. Der Android `AlarmManager` ist dafür zuständig, dass eine App bzw. eine Aufgabe (Intent) innerhalb einer App zu einem definierten Zeitpunkt ausgeführt wird. [47]

TytNotificationManager

Innerhalb des `TytNotificationManager` ist die Methode `setupAlarms` für das Aufsetzen der Benachrichtigungen zuständig, wie in Listing 5.5 dargestellt. Sie erhält ihre Liste an Benachrichtigungen von den aufrufenden Methoden, welche diese wiederum aus der Datenbank abfragen.

```
1  public class TytNotificationManager {
2     private final AlarmManager am;
3
4     private void setupAlarms(List<Notification> notifications) {
5        for (final Notification notification : notifications) {
6           cancelAlarm(notification);
7           long interval = notification.getInterval();
8           long start = notification.getStart();
9           long postponedStart = getPostponedStart(start, interval);
10
11          Intent intent = new Intent(mContext, NotificationReceiver.class);
12          intent.putExtra("questionnaireId",
                notification.getQuestionnaireId());
13          PendingIntent pIntent = PendingIntent.getBroadcast(mContext,
                notification.getRequestCode(), intent,
                PendingIntent.FLAG_UPDATE_CURRENT)
14
15          am.setRepeating(AlarmManager.RTC_WAKEUP, postponedStart, interval,
                pIntent);
16       }
17    }
18 }
```

Listing 5.5: Aufsetzen von Erinnerungen durch TytNotificationManager

Vor dem Aufsetzen jeder Benachrichtigung ist es zunächst erforderlich, dass eventuell bereits bestehende Benachrichtigungen derselben ID gelöscht werden. Danach wird ein `PendingIntent` erzeugt, der vom `AlarmManager` bei Eintreffen des Zeitpunktes ausgeführt werden soll. Als `PendingIntent` fungiert ein Broadcast, der den `NotificationReceiver` bei Eintreffen entsprechend informiert. Die Methode `setRepeating` übergibt die Benachrichtigung schließlich an den `AlarmManager`. Sie erhält einen Modus (Wakeup-Mode), den Startzeitpunkt, das Intervall und schließlich das erzeugte PendingIntent. Der Modus `RTC_WAKEUP` (Zeile 15) sorgt dafür, dass das Gerät „aufwacht“ und den Benachrichtigungston zum vereinbarten Zeitpunkt tatsächlich direkt abspielt. [48]

NotificationReceiver

Im nachfolgenden Listing 5.6 ist die Umsetzung des `NotificationReceiver` abgebildet. Bei Eintreffen der Broadcast-Nachricht (`onReceive`) erstellt der `NotificationReceiver` die tatsächliche Benachrichtigung auf dem Gerät unter Berücksichtigung des gewählten Klingeltons. Er legt dabei auch fest, welche Ansicht geöffnet werden soll, wenn der Benutzer die Benachrichtigung auf dem Gerät öffnet (Zeile 25). Damit ein Nutzer nach dem Abschließen des entsprechenden Fragebogens die App nicht direkt wieder verlässt, sondern auf die Hauptseite zurückkehrt, wird die `MainActivity` als Intent zusätzlich in der Historie angelegt (Zeile 21).

```
1  public class NotificationReceiver extends BroadcastReceiver {
2     private final StringsManager str = StringsManager.getInstance();
3     private SharedPreferences sharedPref;
4     private QuestionnairesHelper questionnairesHelper;
5
6     @Override
7     public void onReceive(Context context, Intent intent) {
8        // Initialization omitted
9        String questionnaireId = intent.getStringExtra("questionnaireId");
10       Questionnaire questionnaire =
             questionnairesHelper.findQuestionnaireById(questionnaireId);
11
12       NotificationCompat.Builder mBuilder = new
             NotificationCompat.Builder(context)
13       .setSmallIcon(R.drawable.ic_launcher)
14       .setContentTitle(questionnaire.getAttributes().getName())
15       .setContentText(str.getString(StringIds.NOTIFICATION_TEXT))
16       .setSound(getSoundUri(context))
17       .setLights(Color.BLUE, 500, 500)
18       .setVibrate({500, 500, 500, 500});
19
20       // Create the backstack and the respective intent for
             questionnaireStructure:
21       Intent mainIntent = new Intent(context, MainActivity.class);
22       Intent resultIntent = new Intent(context,
             QuestionnaireStructureActivity.class);
```

```
23        resultIntent.putExtra(
              QuestionnaireStructureActivity.ARG_QUESTIONNAIRE_ID,
              questionnaireId);
24
25        PendingIntent resultPendingIntent =
              PendingIntent.getActivities(context, 0, new Intent[]{mainIntent,
              resultIntent}, PendingIntent.FLAG_ONE_SHOT);
26        mBuilder.setContentIntent(resultPendingIntent);
27
28        NotificationManager mNotificationManager = (NotificationManager)
              context.getSystemService(Context.NOTIFICATION_SERVICE);
29        mNotificationManager.notify(Integer.parseInt(questionnaireId),
              mBuilder.build());
30    }
31 }
```

Listing 5.6: Erstellen einer Benachrichtigung durch den NotificationReceiver

5.4.3 Algorithmus für wiederkehrende Benachrichtigungen

Der `TytNotificationManager` generiert wie bereits erwähnt anhand der vorgegebenen Zeitpläne die tatsächlichen Benachrichtigungszeitpunkte. Dazu muss er aus den Zeitplänen vom Typ Between und Fixed einen Startwert und eine Periode berechnen, da diese Werte vom Android `AlarmManager` benötigt werden, um einen wiederkehrenden Alarm zu erstellen. Der Algorithmus für die Erstellung von Benachrichtigungen für fixe Zeitpläne ist trivial. Sie besitzen bereits einen `start_day` und ein `interval` und können so eins zu eins an den `AlarmManager` übergeben werden. Wie Benachrichtigungen aus „Between“-Schedules erstellt werden, zeigt Listing 5.7.

```
1 private void saveRandomNotifications(final String questionnaireId, final
      ScheduleBetween schedule) {
2     DateTime start = getDateTimeForStartDay(schedule.getStart(),
          schedule.getStartDay());
3     DateTime end = getDateTimeForStartDay(schedule.getEnd(),
          schedule.getStartDay());
4
5     long startTime = start.getMillis();
6     long endTime = end.getMillis();
```

```
7
8     int numberOfNotifications = schedule.getAmount();
9
10    long intervalMillis = AlarmManager.INTERVAL_DAY * schedule.getEvery();
11    long scheduleId = schedule.getId();
12
13    long timeSpan = endTime - startTime;
14    long timeInterval = timeSpan / numberOfNotifications;
15
16    long lastCalculatedMillis = startTime;
17    int scheduledNotifications = 0;
18    int loopCounter = 0;
19    while (scheduledNotifications < numberOfNotifications) {
20       loopCounter++;
21       // Abort if limit is reached to prevent infinite runs:
22       if (isLoopLimitReached(numberOfNotifications, loopCounter)) break;
23
24       long randomMillis = calcRandomMillis(startTime, timeInterval,
             lastCalculatedMillis);
25
26       if (isCalculatedTimeAfterEndTime(endTime, randomMillis)) continue;
27
28       Notification notification = new Notification();
29
30       // Check if calculated time is before (now + 15min)
31       if (randomMillis <= DateTime.now().getMillis() + _15MINUTES) {
32          // Schedule the same time for tomorrow (add 24 h)
33          long tomorrow = randomMillis + _24HOURS;
34          notification.setStart(tomorrow);
35       } else {
36          notification.setStart(randomMillis);
37       }
38
39       notification.setInterval(intervalMillis);
40       notification.setScheduleId(scheduleId);
41       notification.setQuestionnaireId(questionnaireId);
42       notificationsHelper.insertNotification(notification);
43
44       // Set values for next iteration:
45       scheduledNotifications++;
46       lastCalculatedMillis = randomMillis;
```

```
47          startTime = startTime + timeInterval;
48      }
49
50      private boolean isLoopLimitReached(final int numberOfNotifications, final
            int loopCounter) {
51          if (loopCounter > numberOfNotifications * 100) return true;
52          return false;
53      }
54
55      private long calcRandomMillis(final long startTime, final long
            timeInterval, final long lastCalculatedMillis) {
56          long randomMillis = startTime + (long) (generator.nextDouble() *
                timeInterval);
57
58          // add 15 minutes if required to provide a better distance between
                scheduled times
59          if (timeInterval > _15MINUTES) {
60              // Check if calculated time is in a 15min window to the last
                    scheduled (if yes add 5min)
61              while (_15MINUTES > (randomMillis - lastCalculatedMillis)) {
62                  randomMillis = randomMillis + _5MINUTES;
63              }
64          }
65          return randomMillis;
66      }
67
68      private boolean isCalculatedTimeAfterEndTime(final long endTime, final
            long randomMillis) {
69          if (randomMillis > endTime) return true;
70          return false;
71      }
72  }
```

Listing 5.7: Algorithmus für die Berechnung einer bestimmten Anzahl an Benachrichtigungen innerhalb eines Zeitraumes

Angelehnt an den Algorithmus von Jochen Herrmann[5] wird die definierte Anzahl zufälliger Zeitpunkte innerhalb des vorgegebenen Zeitraums berechnet. Dazu wird der Zeitraum in n (n = `numberOfNotifications`) gleich große Bereiche aufgeteilt, in-

[5] Algorithmus für die zufällige Verteilung von Benachrichtigungen siehe [1], Kapitel 5.2.3.

nerhalb derer ein zufälliger Zeitpunkt gewählt wird (Zeile 24). Liegt der Zeitpunkt nicht mindestens 15 Minuten entfernt vom zuletzt berechneten bzw. zum Startzeitpunkt, werden zusätzliche Minuten addiert. Liegt der so berechnete Zeitpunkt nicht mindestens 15 Minuten in der Zukunft, wird ein ganzer Tag hinzugefügt. Dieser Vorgang wird so lange wiederholt, bis die erforderliche Menge an Benachrichtigungen geplant wurde.

5.4.4 Einschränkungen bei der Benachrichtigungsplanung

Bezüglich der Zeitpunkte, zu denen Benachrichtigungen durch den `TytNotificationManager` aufgesetzt, bzw. beendet werden, galt es aufgrund fachlicher und technischer Gegebenheiten die nachfolgend beschriebenen Aspekte zu berücksichtigen.

Verschieben des Startzeitpunktes in die Zukunft

Die Berechnung des Startzeitpunktes eines Alarms findet nur bei der Synchronisierung mit dem Server oder bei Neuerstellung eines Zeitplans statt. Danach wird die Benachrichtigung entsprechend der vorgegebenen Periode wiederholt. Befindet sich der Nutzer im Offline-Modus oder wurden Zeitpläne geändert, liegt nach einer Weile der in der Datenbank abgelegte Startzeitpunkt mit großer Wahrscheinlichkeit in der Vergangenheit. Beim Aufsetzen der Benachrichtigungen muss der Zeitpunkt um *m* Intervalle in die Zukunft verschoben werden (siehe Listing 5.8), um zu verhindern,dass Android den Alarm jedes Mal direkt nach dem Aufsetzen auslöst und die Benachrichtigung unvermittelt anzeigt, was dem Standardverhalten bei vergangenen Zeitpunkten entspricht. [47]

```
1 /** Prevents notifications to be shown directly every time setup gets called
      */
2 private long getPostponedStart(long startMillis, long interval) {
3    DateTime start = new DateTime(startMillis);
4    DateTime now = DateTime.now();
5    while (start.isBefore(now)) {
6       start = start.plus(interval);
7    }
8    return start.getMillis();
9 }
```

Listing 5.8: Neuberechnung eines Startzeitpunktes in der Vergangenheit

Keine Benachrichtigung bei Vorliegen unbeantworteter statistischer Fragebögen
Benachrichtigungen dürfen zeitlich erst dann aufgesetzt werden, wenn keine statistischen unbeantworteten Fragebögen vorliegen. Ein Nutzer, der noch nicht alle statistischen Fragebögen beantwortet hat, kann keine wiederkehrenden Fragebögen ausfüllen, da es ihm nicht möglich ist, die Ansicht zu wechseln. Daher würde eine erscheinende Benachrichtigung für einen Fragebogen, den der Nutzer noch gar nicht öffnen kann, nur verwirren und zu einer schlechten Usability beitragen. Tritt ein Benutzer einer neuen Studie bei, die statistische Fragebögen beinhaltet, müssen alle aufgesetzten Benachrichtigungen gestoppt werden. Sie werden erst wieder erzeugt, wenn alle erforderlichen Fragebögen abgeschlossen sind.

Benachrichtigungen nur für aktive Fragebögen
Verlässt ein Nutzer eine Studie, müssen die Benachrichtigungen aller zugehörigen Fragebögen gelöscht werden. Fragebögen von Studien, an denen der Nutzer nicht teilnimmt, kann er nicht ausfüllen, somit ist die Erinnerungsfunktion hinfällig. Bei Verlassen einer Studie werden daher alle zugehörigen aufgesetzten Benachrichtigungen durch den `TytNotificationManager` gelöscht.

Keine Benachrichtigungen nach Logout
Bei einem Logout aus der App müssen alle Benachrichtigungen gelöscht werden. Da zum Zeitpunkt des Ausfüllen in diesem Fall kein Nutzer eingeloggt ist, können ausgefüllte Fragebögen keinem Nutzerprofil zugeordnet werden. Bei einem Logout werden somit nicht nur die Einträge aus den Datenbanken entfernt, sondern auch die aufgesetzten Benachrichtigungen durch den `TytNotificationManager` gelöscht.

Neuerstellung der Benachrichtigungen nach Geräteneustart
Nach einem Geräteneustart muss der `TytNotificationManager` alle Benachrichtigungen unter Berücksichtigung der obigen Einschränkungen erneut an den `AlarmManager` senden, da dieser standardmäßig alle aufgesetzten Alarme beim Geräteneustart vergisst. [48] Der `TytNotificationManager` wird vom `BootReceiver` über den Neustart informiert und kann daraufhin die erforderlichen Benachrichtigungen neu aufsetzen.

5.4.5 Implikationen durch die Nutzeranpassung von Benachrichtigungen

Anforderung 22 besagt, dass Benutzer Anpassung an den Benachrichtigungen vornehmen können müssen, wenn dies vom Studienleiter nicht ausdrücklich deaktiviert wird. Bei der Umsetzung dieser Anforderung wurden zunächst die folgenden Herausforderungen identifiziert:

1. Eine Übertragung angepasster Zeitpläne zurück an das Backend ist zum aktuellen Zeitpunkt von der Schnittstelle nicht vorgesehen. Das bedeutet, dass alle Benutzeränderungen bezüglich der Benachrichtigungen nur lokal gespeichert sind und nach einem Logout bzw. einer Neuinstallation aufgrund des Verlusts des Nutzerkontexts nicht mehr zur Verfügung stehen.

2. Zeitpläne eines Fragebogens besitzen keine ID im Backend und sind deshalb nicht eindeutig zuordenbar. Da die Details eines Fragebogens, darunter auch die Benachrichtigungszeitpläne, periodisch vom Backend abgefragt und abgelegt werden, können neu synchronisierte nicht mehr zu lokal geänderten Zeitplänen zugeordnet werden. Es kann Benutzern dadurch nicht ermöglicht werden, bestehende Zeitpläne direkt anzupassen, ohne dass diese nachträglich während des Synchronisierungsprozesses wieder hinzugefügt werden würden.

Für Herausforderung eins wurden zunächst Untersuchungen angestellt, ob es sich lohnen würde, die Einstellungen unabhängig von der Applikation global innerhalb des Endgerätes abzulegen, um zu ermöglichen, dass die Daten bei einer Neuinstallation und einem Logout wieder eingespielt werden können. Dagegen sprachen allerdings Sicherheitsbedenken und die Tatsache, dass bei einem Gerätewechsel durch den Nutzer die Einstellungen trotzdem verloren gingen. Denkbar wäre daher eine Exportfunktion der Einstellungen zum Beispiel auf einen Cloud-Speicher, die aber im Rahmen dieses Prototyps nicht umgesetzt werden konnte. Die Anforderung wurde daher hier unter der Einschränkung implementiert, dass bei einem Logout die Nutzereinstellungen bezüglich der Benachrichtigungen verloren gehen und bei einem neuen Login bzw. einer Neuinstallation die vorgegebenen Zeitpläne der Fragebögen vom Backend verwendet werden, um die Benachrichtigungen aufzusetzen.

Um Herausforderung zwei zu begegnen, wurde festgelegt, dass Nutzer vorgegebene Zeitpläne nur löschen bzw. neue ergänzen können. Wird dies einmalig innerhalb eines Fragebogens getan, dürfen die vorgegebenen Zeitpläne vom Backend den lokal angepassten Stand nicht mehr überschreiben bzw. ergänzen. Dadurch wird sichergestellt, dass Nutzereinstellungen erhalten bleiben und durch den Nutzer gelöschte Zeitpläne nicht vom System automatisch bei Synchronisierung reaktiviert werden.

5.5 Messungen

In den nachfolgenden Unterkapiteln wird die Realisierung der unterschiedlichen Messungen während der Fragebogenausfüllung durch den Nutzer ausführlicher beschrieben. Welche Messungen für einen Fragebogen vorgesehen bzw. erforderlich sind, wird in den Fragebogendetails vom Backend übermittelt. Jede Messanforderung enthält einen bestimmten Wert, der angibt, wie oft die Messung durchzuführen ist. Dabei wird zwischen einer periodischen Messanforderung (`type: each`) und einer fixen Messanforderung (`type: amount`) unterschieden. Bei der periodischen Messanforderung wird die Messung alle n Sekunden gestartet. Sie wird erst beendet, wenn der Fragebogen verlassen bzw. abgeschlossen wird. Dabei werden potenziell sehr viele Messergebnisse erzeugt. Bei einer fixen Messanforderung wird eine bestimmte Anzahl an Messpunkten erzeugt. Da im Voraus nicht bekannt ist, wie lange der Nutzer für die Ausfüllung des Fragebogens benötigt, können die Messzeitpunkte nicht über die Dauer des Ausfüllens gleichmäßig verteilt werden. Für die Umsetzung der Applikation wurde daher festgelegt, dass die Messung alle zwei Sekunden stattfindet, bis die definierte Anzahl an Messpunkten erreicht ist.

Für die Umsetzung der periodischen Messungen wurde auf einen einfachen Java-Timer (`java.util.Timer`) in Verbindung mit einem TimerTask (`java.util.TimerTask`) zurückgegriffen.

5.5.1 Geräuschmessung

Für die Umsetzung der Geräuschmessung ist eine zusätzliche Berechtigung im Sinne des Android Berechtigungskonzepts erforderlich. Seit Android 6.0 (API Level 23) werden Berechtigungen für Aufgaben innerhalb von Apps erst zur Laufzeit vergeben. Zuvor mussten Nutzer bereits bei der Installation der App die Freigaben erteilen. [49] Berechtigungen werden in *normale Berechtigungen* (normal permissions) und *gefährliche Berechtigungen* (dangerous permissions) unterteilt. [50]

Der Zugriff auf das Mikrofon zur Geräuschmessung wird von Android als gefährliche Berechtigung geführt, da über das Mikrofon vertrauliche Daten abgegriffen werden könnten. Für die Erteilung einer als gefährlich eingestuften Berechtigung genügt es nicht, sie im Manifest zu deklarieren, der Nutzer muss beim Ausführen der App explizit zustimmen, dass er die Berechtigung erteilen möchte. Er kann diese auch jederzeit ablehnen oder widerrufen.
Für die Überprüfung der Berechtigung wurde auf die Methoden der `ActivityCompat` zurückgegriffen, um Abwärtskompatibilität zu erreichen, wie in Listing 5.9 dargestellt.

```
1 private void recordNoise() {
2    int microPermission = ActivityCompat.checkSelfPermission(this,
         android.Manifest.permission.RECORD_AUDIO);
3    if (microPermission == PackageManager.PERMISSION_GRANTED) {
4       new RecordNoiseTask().execute();
5    } else {
6       ActivityCompat.requestPermissions(mActivity, new
            String[]{Manifest.permission.RECORD_AUDIO},
            Constants.REQUEST_RECORD_AUDIO_PERMISSION);
7    }
8 }
```

Listing 5.9: Berechtigungsüberprüfung bzw. Berechtigungsanfrage mit Hilfe der ActivityCompat

Die tatsächliche Geräuschaufnahme wird mit Hilfe eines `AsyncTask` ausgeführt. Der `AsyncTask` triggert den `NoiseRecorder` und speichert nach dessen Ausführung das Ergebnis als Amplitude zwischen 0 und 1.

Innerhalb des `NoiseRecorder` sorgt der Android `AudioRecord` dafür, die Aufnahme über das Mikrofon entgegenzunehmen und bereitzustellen. Der `AudioRecord` benötigt eine auf das Endgerät abgestimmte Konfiguration, um die Aufnahme durchführen zu können. Listing 5.10 zeigt die aufwendige Instanziierung eines funktionierenden `AudioRecord`.

```
1  // max amplitude is between 0 and 32767 for 16 bit encoding (abs)
2  private static final float _16_BIT_RANGE = 32767f;
3
4  private static final int[] mSampleRates = new int[]{8000, 11025, 22050,
       44100};
5  private static final short[] mAudioFormats = new
       short[]{AudioFormat.ENCODING_PCM_16BIT};
6  private static final short[] mChannels = new
       short[]{AudioFormat.CHANNEL_IN_STEREO, AudioFormat.CHANNEL_IN_MONO};
7
8  public static double getNoiseLevel(Activity activity) throws
       InvalidNoiseLevelException {
9     AudioRecord recorder = null;
10    int bufferSize = 0;
11
12    sampleRateLoop:
13    for (int rate : mSampleRates) {
14       for (short audioFormat : mAudioFormats) {
15          for (short channelConfig : mChannels) {
16             try {
17                bufferSize = AudioRecord.getMinBufferSize(rate,
                      channelConfig, audioFormat);
18
19                if (bufferSize != AudioRecord.ERROR_BAD_VALUE) {
20                   //making the buffer bigger...
21                   bufferSize = bufferSize * RECORDING_PERIOD;
22                   // check if instantiation is possible
23                   recorder = new
                         AudioRecord(MediaRecorder.AudioSource.DEFAULT, rate,
                         channelConfig, audioFormat, bufferSize);
24
25                   if (recorder.getState() == AudioRecord.STATE_INITIALIZED) {
26                      // This config is working, finish!
27                      break sampleRateLoop;
28                   }
```

```
29                  }
30              } catch (Exception e) {
31                  // This config is not working, continue
32              }
33          }
34      }
35  }
36  // ....
37 }
38 }
```

Listing 5.10: Bestimmung eines funktionierenden AudioRecord

Konnte ein `AudioRecord` erfolgreich instanziiert werden, beginnt daraufhin die Aufnahme. Er liest dazu Daten in Größe des zuvor gefundenen Pufferspeichers (Buffer) ein, er führt also die Tonaufnahme durch, bis der Speicher voll ist.

Um auf Basis der eingelesenen Töne nun den Geräuschpegel zu messen, wird der in Listing 5.11 dargestellte Algorithmus angewendet:

```
1  // ...
2  short data[] = new short[bufferSize];
3  recorder.startRecording();
4  recorder.read(data, 0, data.length);
5  recorder.stop();
6
7  double total = 0.0;
8  for (short s : data) {
9     if (s > 0) {
10       total += Math.abs(s);
11    } else {
12       bufferSize--;
13    }
14 }
15
16 double averageAmplitude = total / bufferSize;
17 recorder.release();
18
19 if (averageAmplitude == 0) {
20    throw new InvalidNoiseLevelException(averageAmplitude);
21 }
```

```
22
23 // Normalize the amplitude to a value between 0 and 1
24 return averageAmplitude / _16_BIT_RANGE;
```

Listing 5.11: Algorithmus für die Identifizierung einer normierten Amplitude zur Lärmpegelmessung

Zunächst werden alle ungültigen Werte innerhalb des Pufferspeichers eliminiert (negative Werte) (Zeile 7) Daraufhin wird der Durchschnitt der verbleibenden Werte des Speichers berechnet. Das Ergebnis ist die durchschnittliche Amplitude der Geräuschmessung (Zeile 16). Um die Werte später am Backend vergleichbar zu machen, wird diese Amplitude normiert, unter Verwendung der für die Aufnahme verwendeten Bit-Rate (16 Bit). So entsteht eine Amplitude zwischen 0 und 1 (Zeile 24).

5.5.2 Standortmessung

Auch die Identifikation des Standortes wird als gefährliche Berechtigung geführt. Für die Berechtigungsanfrage wurde eine synonyme Implementierung wie bei der Geräuschmessung verwendet. Zur Messung des Standortes wurde auf das Android Interface `LocationListener`[6] zurückgegriffen. Es sieht die Implementierung von vier Methoden (`onProviderDisabled`, `onProviderEnabled`, `onStatusChanged` und `onLocationChanged`) vor, von denen nur letztere benötigt wurde.
Die `onLocationChanged`-Methode wird immer dann aufgerufen, wenn ein neuer Standort vom Gerät erkannt wurde. Ihre Implementierung ist in Listing 5.12 ersichtlich.

```
1 /** Listens to location change events and adds them to the result (answers)
      based on specific rules */
2 private class LocationChangeListener implements LocationListener {
3
4    private int measures = 0;
5    private int maxMeasures, period;
6    private DateTime lastMeasure;
7
8    // Initialization omitted
```

[6] Für Details siehe [51]

```
9
10    @Override
11    public void onLocationChanged(Location loc) {
12       DateTime now = DateTime.now();
13       if (maxMeasures == 0) {
14          // Infinitely measure the position as no amount is specified
15          if (lastMeasure.plusSeconds(period).isAfter(now)) {
16             addLocation(loc, now);
17             lastMeasure = now;
18          }
19       } else if (measures <= maxMeasures) {
20          if (lastMeasure
21          .plusSeconds(Constants.MIN_SECONDS_BETWEEN_GPS_MEASURES)
22          .isBefore(now)) {
23             // Measure received too early, wait for next one
24             return;
25          } else {
26             addLocation(loc, now);
27             lastMeasure = now;
28             measures++;
29          }
30       } else {
31          stop();
32       }
33    }
```

Listing 5.12: Positionsbestimmung durch den MyLocationListener

5.5.3 Herzfrequenzmessung

Um den Verbindungsaufbau zum Fitnessarmband und das Warten auf Herzfrequenzereignisse zu realisieren, wurde die Klasse `HeartRateRecorder` erstellt. Sie wurde als Singleton umgesetzt, hält Objekte der ANT+-Schnittstellenbibliothek (`PccReleaseHandle<AntPlusHeartRatePcc>` und `AntPlusHeartRatePcc`) und besitzt zwei öffentliche Methoden.
Mit Hilfe der Methode `searchAndConnect` kann ein Verbindungsversuch gestartet werden. Die Methode erwartet einen Context und einen Listener vom Typ `IHeartRateDeviceConnector`, dessen Methoden im Fehler- bzw. Erfolgsfall eines

Verbindungsaufbaus aufgerufen werden. Listing 5.13 zeigt das Interface `IHeartRateDeviceConnector`.

```
1 public interface IHeartRateDeviceConnector {
2    void onDeviceConnected();
3    void onNoDeviceConnectedError(final RequestAccessResult resultCode, final
         DeviceState initialDeviceState);
4 }
```

Listing 5.13: Listener-Interface IHeartRateDeviceConnector

Innerhalb der `searchAndConnect` Methode wird die statische Methode der ANT+ Bibliothek `AntPlusHeartRatePcc.requestAccess` aufgerufen, die zwei Interfaces erfordert. Ein Interface (`IPluginAccessResultReceiver<AntPlusHeartRatePcc>`) wird verwendet, um Verbindungsereignisse zu behandeln. Das andere Interface (`IDeviceStateChangeReceiver`) behandelt das Eintreffen von Zustandsänderungen vorhandener Geräte. Kann eine Verbindung aufgebaut werden, speichert der `HeartRateRecorder` das soeben erzeugte `AntPlusHeartRatePcc` Objekt, sendet ein `onDeviceConnected` Event und stellt fortan mit Hilfe der Methode `subscribeToHrEvents` die Möglichkeit bereit, Herzfrequenzereignisse zu erhalten.

Die Herzfrequenzmessung soll während des Ausfüllens eines Fragebogens erfolgen, sofern der Benutzer diese Funktionalität nicht deaktiviert hat und sie für den entsprechenden Fragebogen vorgesehen ist. Um die `QuestionnaireStructureActivity` übersichtlich zu halten, wurde die Klasse `PulseMeasurementsHelper` erstellt, die sich um die
Vorbereitung und Abwicklung der Messung kümmert und von der `QuestionnaireStructureActivity` verwendet werden kann. Sie interagiert mit dem zuvor beschriebenen `HeartRateRecorder` und übernimmt die entsprechende Anzeige von Dialogen bzw. Statusmeldungen auf der Nutzeroberfläche. Kann keine automatische Verbindung aufgebaut werden (`onNoDeviceConnectedError`), zeigt sie einen entsprechenden `AlertDialog` an, der Hinweise zur Fehlerursache und Problembehebung liefert. Der Nutzer kann hier den Verbindungsaufbau abbrechen oder neu starten. Während des erneuten Verbindungsversuchs wird dem Nutzer eine Statusleiste angezeigt, die darüber informiert, wie lange der Verbindungsversuch noch

andauert.

Kann eine Verbindung erfolgreich aufgebaut werden (`onDeviceConnected`), übernimmt die innere Klasse `HeartRateListener` (siehe Listing 5.14) die Behandlung daraufhin eintreffender Herzfrequenzwerte, ähnlich dem in Listing 5.12 dargestellten `LocationListener`, und fügt die Werte den Fragebogenergebnissen hinzu.

```
1  /** Listens to heartRate events arriving from a connected device and saves
       them to the result data (answers) */
2  private class HeartRateListener implements
       AntPlusHeartRatePcc.IHeartRateDataReceiver {
3     private int measures = 0;
4     private int maxMeasures, period;
5
6     private DateTime lastMeasure;
7     private int lastHeartRate;
8
9     // Initialization omitted
10
11    @Override
12    public void onNewHeartRateData(final long estTimestamp,
          EnumSet<EventFlag> eventFlags, final int computedHeartRate, final
          long heartBeatCount, final BigDecimal heartBeatEventTime, final
          AntPlusHeartRatePcc.DataState dataState) {
13       if (computedHeartRate == 0)
14       return; // Don't consider events from tracker with no actual heart rate
15
16       if (lastHeartRate == computedHeartRate)
17       return; // Don't consider same heart rates twice
18
19       DateTime now = DateTime.now();
20       if (maxMeasures == 0) {
21          // Infinitely add the bpm as no amount is specified
22          if (lastMeasure.plusSeconds(period).isAfter(now)) {
23             addBpm(computedHeartRate, now);
24             lastMeasure = now;
25             lastHeartRate = computedHeartRate;
26          }
27       } else if (measures <= maxMeasures) {
28          if (lastMeasure.plusSeconds(Constants.
29          MIN_SECONDS_BETWEEN_BPM_MEASURES).isBefore(now)) {
30             // Measure received too early, wait for next one
```

```
31                return;
32            } else {
33                addBpm(computedHeartRate, now);
34                lastMeasure = now;
35                lastHeartRate = computedHeartRate;
36                measures++;
37            }
38        } else {
39            stop();
40        }
41    }
42 }
```

Listing 5.14: Herzfrequenzbestimmung durch den HeartRateListener

Nutzer können den Einsatz eines Herzfrequenzmessers global in den Einstellungen deaktivieren. Aus diesem Grund interagiert der `PulseMeasurementsHelper` nur, wenn kein entsprechendes Verbot existiert. Der `PulseMeasurementsHelper` erkennt außerdem mit Hilfe der `SharedPreferences`, ob zuvor irgendwann ein Herzfrequenzmesser erfolgreich angebunden werden konnte. Wenn nicht, zeigt er dem Nutzer Informationen zur Ersteinrichtung an.

Einschränkungen

Bei der Umsetzung der Herzfrequenzmessung wurde ein potenzielles Problem hinsichtlich der Nutzerfreundlichkeit identifiziert: Ein Nutzer muss auf seiner GARMIN vívosmart HR/HR+ explizit freigeben, dass seine Herzfrequenz gesendet wird (*Broadcast-Modus*). Diese Einstellung ist nicht speicherbar. Ist sie aktiv, kann der Nutzer ausschließlich die Herzfrequenzseite auf dem Herzfrequenzmesser anzeigen. [52] Eine Kopplung mit dem Smartphone ist erst möglich, wenn der Nutzer diesen Sendemodus aktiviert hat.

Daraus wurden für die Kopplung mit ANT+ fähigen Gerät folgende Implikation abgeleitet: Bei Anzeige eines Fragebogens wird innerhalb eines bestimmten Zeitraumes automatisch versucht, eine Verbindung mit einem ANT+ fähigen Gerät herzustellen. Wird in diesem Zeitraum kein Gerät gefunden, d.h. ist keine automatische Kopplung möglich, soll dem Benutzer ein Hinweis angezeigt werden, der ihm mitteilt, den Sendemodus

auf seinem Gerät zu aktivieren. Er hat daraufhin die Möglichkeit, die Herzfrequenzmessung abzubrechen oder einen erneuten Verbindungsversuch vorzunehmen. Wird bei einem erneuten Verbindungsversuch kein Gerät gefunden, wird der Dialog mit der entsprechenden Fehlermeldung erneut angezeigt.

Grundsätzlich kann die Kopplung der Track Your Tinnitus App durch durch die oben beschriebene Implementierung mit jedem ANT+ fähigen Gerät erfolgen, welches das HeartRate-Profil implementiert hat. Es besteht die Annahme, dass ein Benutzer jeweils nur einen Fitnesstracker zur selben Zeit im Einsatz hat. Die Kopplung erfolgt deshalb direkt mit dem ersten gefundenen ANT+ fähigen Gerät. Dies könnte zu Problemen führen, wenn der Nutzer zwei unterschiedliche ANT+ fähige Geräte gleichzeitig verwendet, aber nur eines direkt am Körper trägt, die automatische Kopplung sich aber mit dem anderen verbindet. Außerdem können erforderliche Einstellungen zum Senden der Herzfrequenzdaten von Fitnesstracker zu Fitnesstracker unterschiedlich sein. Aus diesem Grund gilt für die Umsetzung des Prototyps die Einschränkung, dass die Herzfrequenzmessung insbesondere in Hinsicht auf die Nutzerfreundlichkeit ausschließlich auf eine GARMIN vívosmart HR+ ausgelegt ist.

Erweiterungen hinsichtlich der Suche und der expliziten Auswahl eines Fitnesstrackers durch den Nutzer sind jederzeit möglich. Ebenso könnten Hinweise und Anleitungen auf mehrere verschiedene unterstützte Endgeräte ausgelegt werden bzw. verallgemeinert werden.

6

Vorstellung der Applikation

In diesem Kapitel wird der im Rahmen dieser Arbeit realisierte Android Prototyp aus Sicht des Benutzers dargestellt. Das Kapitel ist in die unterschiedlichen Hauptfunktionalitäten der Applikation unterteilt.

6.1 Erstmalige Nutzung

Die initiale Ansicht, die dem Benutzer bei der ersten Verwendung der Applikation nach der Installation angezeigt wird, ist die Einführung, dargestellt in Abbildung 6.1. Hier erhält der Nutzer erste Informationen zur Applikation. Mit einem Klick auf „Weiter“ gelangt er zur Login-Ansicht. Er kann auch vom mobilen Endgerät aus die Website öffnen, indem er auf den entsprechenden Button klickt.

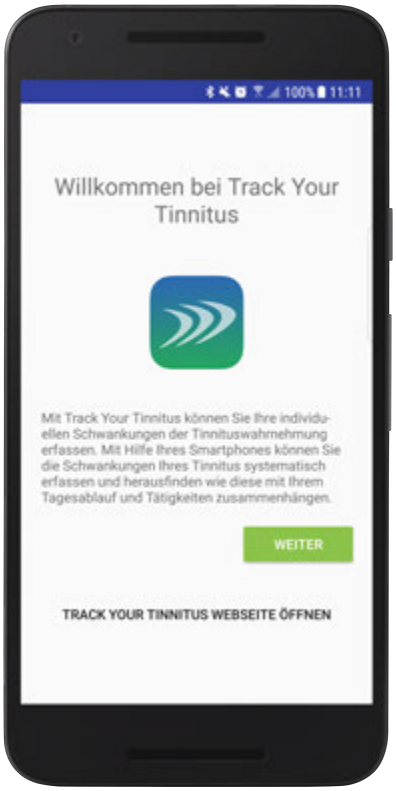


Abbildung 6.1: Einführung

6.1.1 Login

Auf der Ansicht in Abbildung 6.2 hat der Nutzer, sofern er bereits ein Konto besitzt, die Möglichkeit, sich einzuloggen. Besteht keine Internetverbindung oder werden falsche Authentifizierungsdaten eingegeben, erscheint eine entsprechende Fehlermeldung. Kann er sich nicht mehr an sein Passwort erinnern,

gelangt er mit einem Klick auf „Passwort vergessen?“ zur Webseite, die es ihm ermöglicht, sein Passwort zurückzusetzen.
Besitzt der Nutzer noch kein Konto, gelangt er über den Registrieren-Button zur Registrierungsansicht in Abbildung 6.3, die im nächsten Kapitel ausführlich vorgestellt wird.

Nach erfolgreicher erstmaliger Authentifizierung in der App werden dem Benutzer automatisch die statistischen Fragebögen der Track Your Tinnitus Studie, in die jeder Nutzer standardmäßig eingeschrieben ist, angezeigt. Das Ausfüllen statistischer Fragebögen ist im Kapitel 6.4.1 beschrieben. Nach Abschluss der statistischen Fragebögen gelangt der Nutzer zur Track Your Tinnitus Hauptseite, die in Kapitel 6.2 ff. vorgestellt wird.

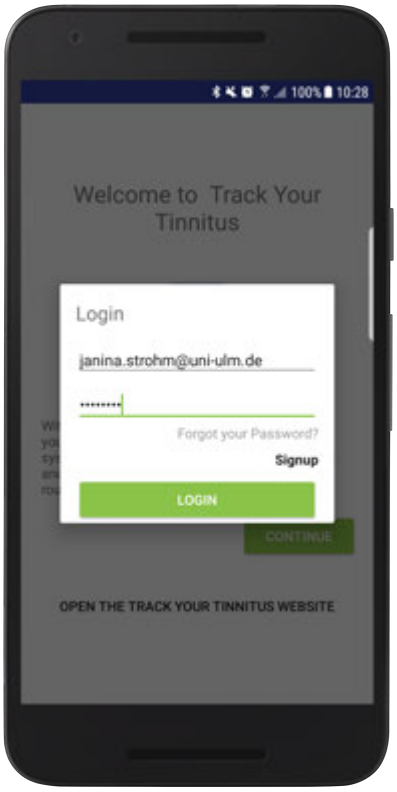


Abbildung 6.2: Login-Ansicht

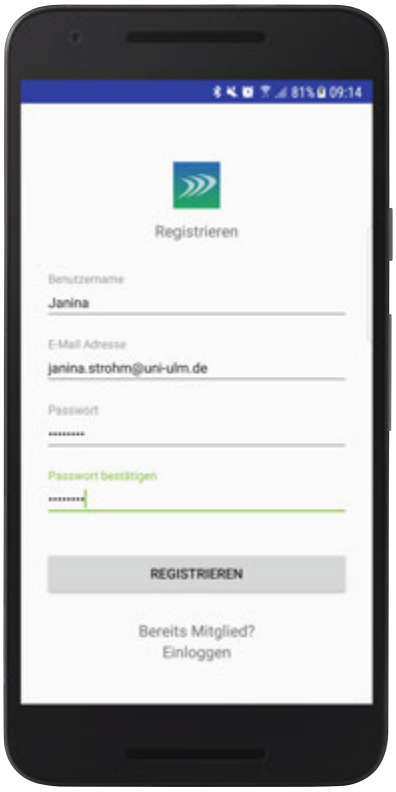


Abbildung 6.3: Registrierungsansicht

6.1.2 Registrierung

Um ein neues Nutzerkonto anzulegen, ist der Benutzer aufgefordert, auf der Registrierungsansicht (Abbildung 6.3) die Registrierungsdaten, wie seinen gewünschten Benutzernamen, seine E-Mail-Adresse und das gewünschte Passwort, anzugeben.
Schlägt die Validierung der Daten, die vor dem Absenden durchgeführt wird, fehl, erhält der Nutzer Hinweise auf die fehlerhaften Angaben und kann die eingegebenen Werte korrigieren. Besteht keine Internetverbindung wird ebenfalls ein entsprechender Hinweistext angezeigt.

Bei der Registrierung wird automatisch die aktuell eingestellte Gerätesprache mit an das Backend gesendet. Dadurch wird sichergestellt, dass der Nutzer die Track Your Tinnitus Daten von Beginn an in der richtigen Sprache erhält, sofern verfügbar. Andernfalls wird auf die Standardsprache Englisch zurückgegriffen.
Bei erfolgreicher Registrierung am Backend wird dem Nutzer ein Hinweis angezeigt, zum Abschluss seiner Registrierung seine E-Mail-Adresse zu verifizieren. Dazu soll er sein E-Mail-Postfach öffnen und den Link in der empfangenen Registrierungs-E-Mail öffnen.
Im Anschluss kann sich der Nutzer schließlich wie in Kapitel 6.1.1 beschrieben, in der App mit seinen angegebenen Registrierungsdaten einloggen.

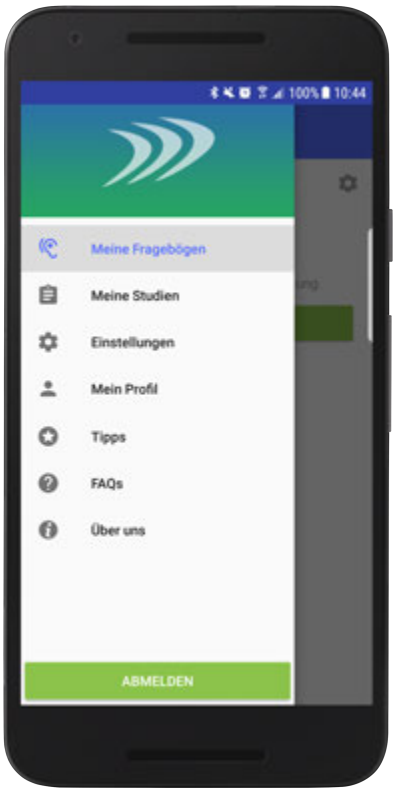


Abbildung 6.4: Hauptmenü

6.2 Hauptmenü

Hat ein Benutzer alle initial statistischen Fragebögen erfolgreich abgeschlossen, werden ihm auf der Hauptseite der Track Your Tinnitus Applikation alle momentan verfügbaren wiederkehrenden Fragebögen angezeigt. Mit

einem Klick auf den Button im oberen linken Eck der Applikation kann der Nutzer das Hauptmenü öffnen, welches in Abbildung 6.4 dargestellt ist. Hier hat er die Wahl, zwischen unterschiedlichen Ansichten zu wechseln, die in den folgenden Kapiteln näher beschrieben werden.

6.3 Meine Studien

Abbildung 6.5: Meine Studien

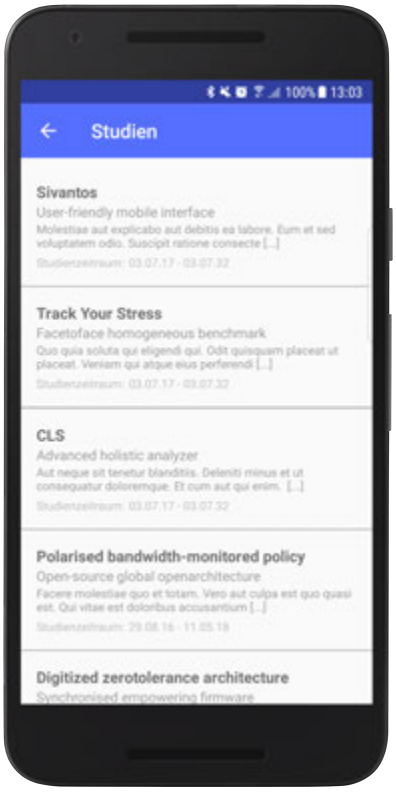


Abbildung 6.6: Weitere verfügbare Studien

Öffnet der Nutzer den Menüpunkt „Meine Studien“, sieht er eine Übersicht aller Studien, in die er zum aktuellen Zeitpunkt eingeschrieben ist (Abbildung 6.5). Grundsätzlich können Benutzer nur Fragebögen von Studien ausfüllen, in die sie eingeschrieben sind. Mit der Registrierung wird jeder Benutzer automatisch in die Track Your Tinnitus Studie eingeschrieben, d.h. initial wird immer mindestens eine Studie angezeigt, wie in Abbildung 6.5 ersichtlich. Bei bestehender Internetverbindung erscheint am unteren

Bildschirmrand auf der rechten Seite ein „+“-Button, über den der Nutzer neue Studien hinzufügen kann. Dazu gelangt er auf die Ansicht über alle weiteren verfügbaren Studien, die in Abbildung 6.6 dargestellt ist. Besteht keine Internetverbindung oder befindet sich der Nutzer im Offline-Modus (siehe 6.5.1), hat er keine Möglichkeit, sich in neue Studien einzuschreiben, er kann nur seine aktuellen Studien einsehen. Eine Studie hat einen Studienzeitraum und einen bestimmten Status. Es wird zwischen öffentlichen Studien, an denen jeder teilnehmen kann, passwortgeschützten und privaten Studien unterschieden. Für die Teilnahme an einer passwortgeschützten Studie benötigt der Benutzer ein Passwort, welches er von seinem Studienleiter erhält. Zur Teilnahme an einer privaten Studie ist eine Einladung durch den Studienleiter erforderlich. Diese kann der Benutzer anfordern, sofern sie nicht bereits vorliegt. Vorliegende Einladungen werden in der Übersicht mit Hilfe eines Symbols besonders gekennzeichnet.

6.3.1 Studiendetails

Zur Detailansicht einer Studie gelangt der Benutzer, wenn er eine bestimmte Studie auswählt. In Abbildung 6.7 ist die Detailansicht einer Studie abgebildet, an der der Nutzer noch nicht teilnimmt.

An Studie teilnehmen

Um einer neuen Studie beizutreten, wählt der Nutzer den Button „STUDIE BEITRETEN“ in der Detailansicht der Studie. Handelt es sich um eine passwortgeschützte Studie, wird der Benutzer daraufhin aufgefordert, sein Passwort einzugeben.

Fragebögen einer Studie

Sofern der Nutzer in eine Studie eingeschrieben ist, kann er in der Detailansicht zum Tab „Fragebögen“ wechseln. Hier sieht er eine Liste aller Fragebögen, die zu der Studie gehören, wie in Abbildung 6.8 dargestellt. Auch Fragebögen, die momentan nicht ausgefüllt werden können, werden zur Information angezeigt. Aktive Fragebögen kann der

Nutzer direkt über diese Ansicht öffnen und ausfüllen. Das Ausfüllen von Fragebögen ist in Kapitel 6.4 beschrieben.

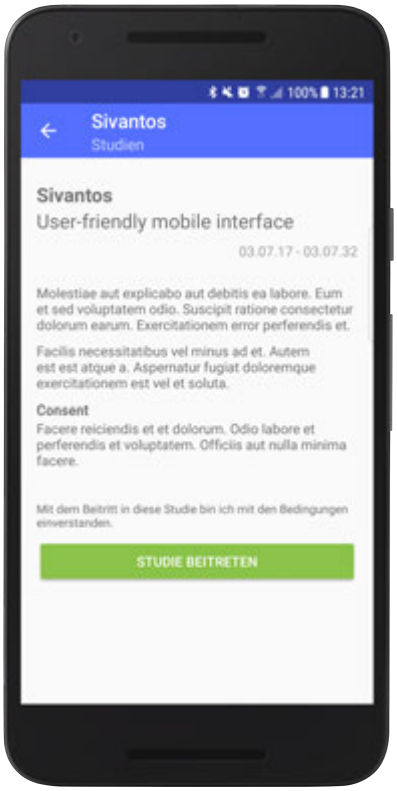


Abbildung 6.7: Studiendetails

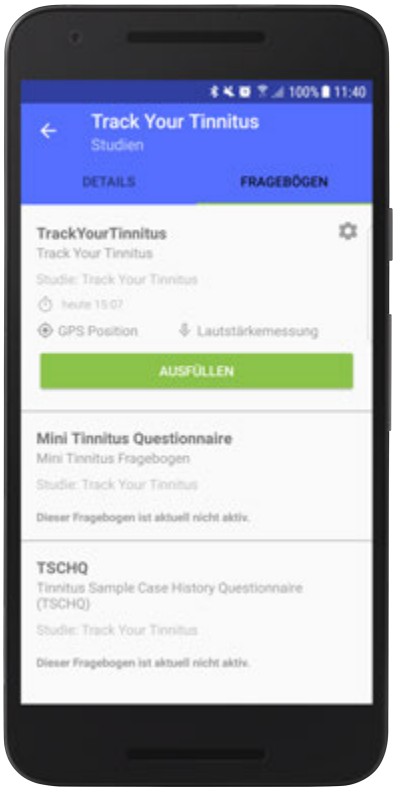


Abbildung 6.8: Fragebögen einer Studie

Studie verlassen

Ein Nutzer kann jederzeit entscheiden, an welchen Studien er teilnehmen möchte. Die einzige Studie, aus der ein Nutzer nicht austreten kann, ist die TrackYourTinnitus Studie. Jede andere Studie kann mit einem Klick auf den Button „Studie verlassen“ in der Studiendetailansicht verlassen werden.

6.4 Meine Fragebögen

Durch Auswahl des Menüpunkts „Fragebögen“ gelangt der Benutzer zur Übersicht über alle wiederkehrenden Fragebogen, dargestellt in Abbildung 6.9. Das Ausfüllen wiederkehrender Fragebögen wird in Kapitel 6.4.2 beschrieben. Abweichend dazu existieren statistische Fragebögen, die im nächsten Unterkapitel näher erläutert werden.

6.4.1 Statistische Fragebögen

Statistische Fragebögen sind spezielle Fragebögen, die Teilnehmer einer Studie einmalig ausfüllen müssen, da sie statistische und demographische Informationen erfragen. Diese Fragebögen werden automatisch angezeigt und können nicht vom Benutzer abgebrochen werden, es sei denn, er beendet die Teilnahme an der zugehörigen Studie. Nach erfolgreichem Abschluss eines statistischen Fragebogens wird der nächste statistische Fragebogen angezeigt, sofern ein solcher vorhanden ist. Das Ausfüllen statistischer Fragebögen unterscheidet sich ansonsten nicht weiter vom Ausfüllen wiederkehrender Fragebögen, welches im nächsten Kapitel genauer erläutert wird.

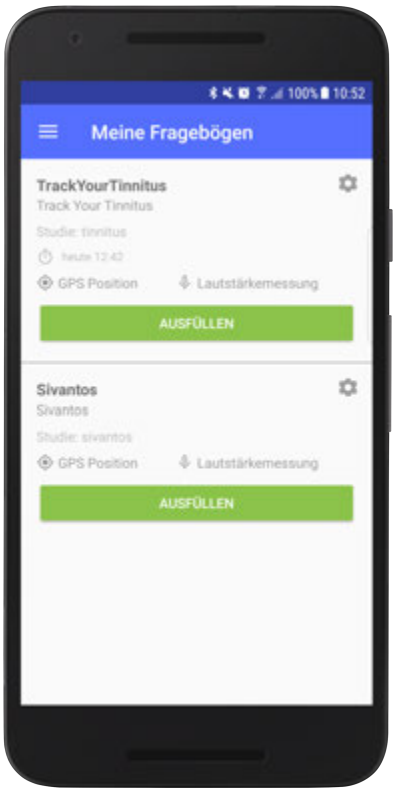


Abbildung 6.9: Fragebogen-Übersicht

6.4.2 Wiederkehrende Fragebögen

Wiederkehrende Fragebögen können beliebig oft und jederzeit vom Benutzer ausgefüllt werden, indem er in der Fragebogenübersicht auf den Button „AUSFÜLLEN“ drückt. Typischerweise besitzen wiederkehrende Fragebögen einen definierten Zeitplan, nach dem der Benutzer an das Ausfüllen des Fragebogens erinnert wird, sofern er die Benach-

richtigungseinstellungen nicht verändert hat. Jeder Fragebogen enthält eine bestimmte Menge an Fragen und Informationen, wie in Abbildung 6.10 und 6.11 beispielhaft gezeigt.

Abbildung 6.10: Statistischer Fragebogen

Abbildung 6.11: Wiederkehrender Fragebogen

Es existieren verschiedene Fragetypen, die im Nachfolgenden näher erläutert werden:

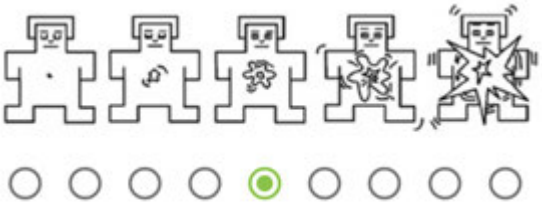
- Eine **Mehrfachauswahl** ermöglicht die Selektion mehrerer vorgegebener Antwortmöglichkeiten gleichzeitig.
- Bei einer **einfachen Auswahl** hat der Nutzer die Möglichkeit, eine aus verschiedenen vorgegebenen Antwortmöglichkeiten zu wählen.

- Die **Ja-Nein-Auswahl** ermöglicht die Beantwortung einer einfachen Ja-Nein Frage.

- Ein **einzeiliger Freitext** lässt den Benutzer die Antwort in einer Zeile frei in eigenen Worten formulieren.
- Der **mehrzeilige Freitext** erlaubt auch die Eingabe längerer Texte durch den Nutzer.
- Ein Freitextfeld kann auf die Eingabe einer **Fließkommazahl** beschränkt werden.
- Ebenso können Fragen existieren, die die Eingabe einer **ganzen Zahl** erfordern.
- Bei einem **Schieberegler** kann der Nutzer einen Wert innerhalb eines vorgegebenen Bereichs durch Verschieben des Reglers selektieren.

- Die **Datumsauswahl** ist für die Angabe eines spezifischen Datums vorgesehen.
- Genauso können Uhrzeiten mit Hilfe der **Uhrzeitauswahl** genau angegeben werden.
- Ist ein spezifischer Zeitpunkt gefragt, wird ein **Datums- und Uhrzeitauswahlfeld** angezeigt.
- Um das **Stimmungsbild** des Nutzers abzufragen, werden bestimmte Grafiken dargestellt, aus denen der Nutzer eine auswählen kann, die seiner aktuellen Stimmung am ehesten entspricht.

- Für die Messung des momentanen **Gemütszustands** werden ebenfalls bestimmte Grafiken dargestellt, anhand derer der Nutzer diesen möglichst einfach beschreiben kann.



Einige Fragen innerhalb eines Fragebogens müssen zwingend beantwortet werden, um den Fragebogen erfolgreich abschließen zu können. Diese sind mit dem Hinweis

„* Erforderlich “ gekennzeichnet. Überspringt oder vergisst ein Benutzer eine erforderliche Frage, wird ihm ein entsprechender Hinweistext angezeigt, wie in Abbildung 6.12 ersichtlich.

Abbildung 6.12: Unvollständigkeitshinweis

Abbildung 6.13: Pulsmesser verbinden

Fragebogeneinstellungen

Ein Benutzer kann die Einstellungen eines wiederkehrenden Fragebogens anpassen, wie in Abbildung 6.14 dargestellt. Dazu wählt er in der Fragebogenübersicht 6.9 das Zahnrad im oberen rechten Eck des entsprechenden Fragebogens aus. Unterschieden wird dabei in Benachrichtigungseinstellungen und Sensor-Berechtigungen.

Möchte der Benutzer die Benachrichtigungsfunktion erweitern, ändern oder für einen Fragebogen vollständig deaktivieren, kann er dies, wie in Abbildung 6.15 dargestellt, tun. Ein Fragebogen kann beliebig viele Benachrichtigungszyklen aufweisen. Benachrichti-

gungen können zu definierten Uhrzeiten an festgelegten Wochentagen, oder für einen bestimmten Zeitraum in einer definierten Anzahl erstellt werden.
So kann der Benutzer beispielsweise einstellen, dass er immer Sonntags um 14 Uhr und zusätzlich an jedem Wochentag (Montag bis Freitag) zwischen 8 und 10 Uhr zwei Mal eine Benachrichtigung erhält. Er kann die Benachrichtigungsfunktion auch vollständig deaktivieren. Ändert der Benutzer die Benachrichtigungseinstellungen nicht, werden die Standardeinstellungen des Fragebogens verwendet.

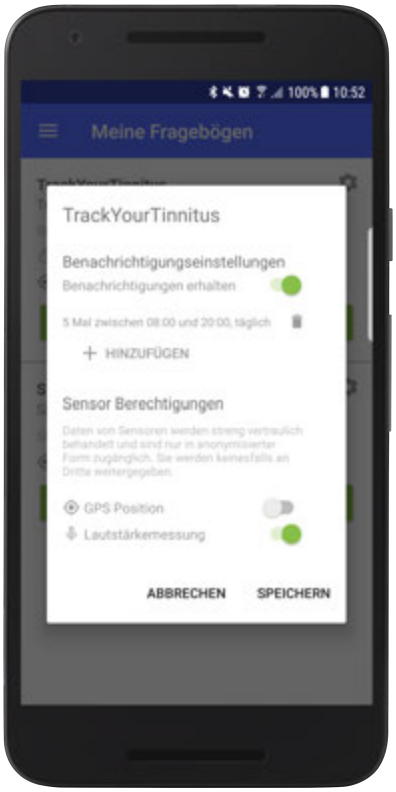


Abbildung 6.14: Fragebogeneinstellungen

Abbildung 6.15: Benachrichtigung anpassen

Neben der Einstellung von Benachrichtigungen ist es Nutzern möglich, die Sensor-Berechtigungen anzupassen. Fragebögen können grundsätzlich um die Messung der Umgebungslautstärke (Noise), der Herzfrequenz (Pulse) und des Gerätestandorts (GPS) bitten, falls diese Metadaten vom Studienleiter als sinnvoll für die weitere Analyse der Fragebögen erachtet werden. Die Messungen finden im Hintergrund während des Aus-

füllens eines Fragebogens statt und informieren den Nutzer während der Durchführung über die Android Aktivitätsleiste.

Ein Benutzer kann die Messungen eines Fragebogens jederzeit individuell anpassen und gegebenenfalls deaktivieren. Hierzu befinden sich in den Fragebogeneinstellungen für jeden vorgesehenen Sensor An-Aus-Schalter, mit dem der Nutzer die Sensoren individuell steuern kann. Am Beispiel in Abbildung 6.14 wurde die Lautstärkemessung deaktiviert. Die Pulsmessung ist für diesen Fragebogen nicht vorgesehen, deshalb erscheint sie nicht in der Liste.

6.4.3 Pulsmessung

Anders als für die Standortbestimmung oder die Messung der Umgebungslautstärke, ist für eine erfolgreiche Pulsmessung ein zusätzliches Endgerät erforderlich. Der Benutzer benötigt ein ANT+-fähiges Gerät, welches das „Heart-Rate“-Profil implementiert. [1] Erfolgreich getestet wurde der aktuelle Stand der Applikation mit einer GARMIN vívosmart HR+.
Ist die Pulsmessung für den Fragebogen vorgesehen und durch den Nutzer nicht explizit in den Einstellungen deaktiviert, versucht die Applikation automatisch eine Verbindung zu einem in der Nähe befindlichen Pulsmesser herzustellen. Gelingt dies nicht, wird der Benutzer innerhalb der Fragebogenansicht aufgefordert, den Sendemodus auf seinem Pulsmesser zu aktivieren. Er kann daraufhin den Verbindungsversuch erneut starten oder die Pulsmessung für diesen Fragebogen abbrechen (siehe Abbildung 6.13).

[1] Eine vollständige Liste der ANT+ fähigen Geräte ist unter `https://www.thisisant.com/directory/filter/` zu finden.

6.5 Einstellungen

Im Menüpunkt „Einstellungen“ kann der Benutzer globale Einstellungen für die Track Your Tinnitus App vornehmen. Abbildung 6.16 zeigt den Aufbau der Einstellungsanzeige.

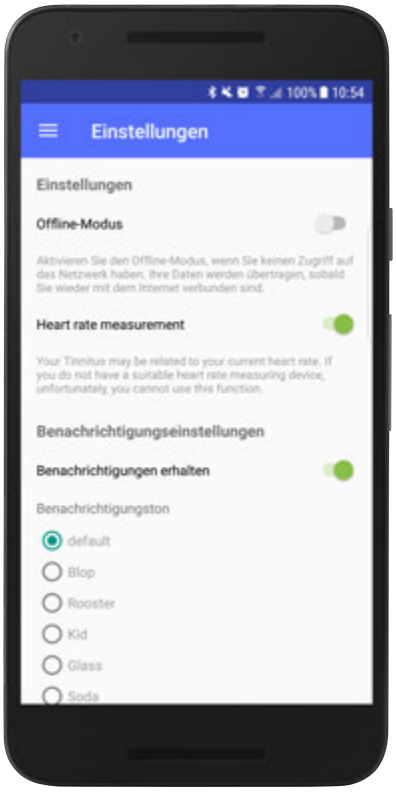


Abbildung 6.16: Allgemeine Einstellungen

6.5.1 Offline-Modus

Ist der Offline-Modus aktiviert, werden keine Hintergrundinformationen über das Internet übertragen. Es ist zu beachten, dass der Offline-Modus nur für einen bestimmten Zeitraum aktivierbar ist. Nach Ablauf dieses Zeitraums ist zwingend ein Login durch den Benutzer erforderlich, um sicherzustellen, dass die Daten auf dem mobilen Endgerät nicht veralten. Während des Offline-Modus sind nur die Hauptfunktionalitäten der App verfügbar. Fragebögen können zum Beispiel ohne Weiteres im Offline-Modus ausgefüllt werden. Ebenso sind alle Informationen zu den Studien, in die der Nutzer eingeschrieben ist, verfügbar. Jedoch ist es nicht möglich, sich während des Offline-Modus von Studien abzumelden oder sich in Neue einzuschreiben. Die Zusatzfunktionalitäten wie Tipps und FAQ sind ebenfalls im Offline-Modus nicht verfügbar.

6.5.2 Benachrichtigungseinstellungen

Innerhalb der Benachrichtigungseinstellungen hat der Nutzer die Möglichkeit, einen individuellen Ton für die Benachrichtigungen zu selektieren oder diese für die gesamte

Track Your Tinnitus App zu deaktivieren. In diesem Fall erhält er keine Erinnerungen an auszufüllende Fragebogen mehr. Aktiviert er die Benachrichtigungen zu einem beliebigen Zeitpunkt wieder, werden alle vorherigen individuellen Anpassungen bzgl. der Benachrichtigungen berücksichtigt und die Erinnerungsfunktion entsprechend reaktiviert.

6.6 Profil

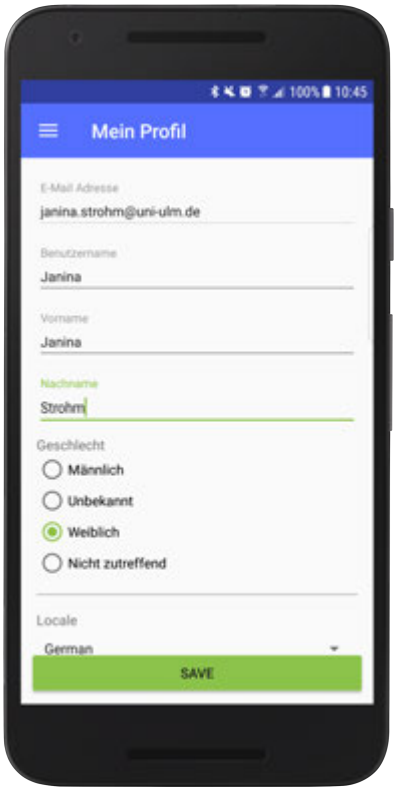


Abbildung 6.17: Profileinstellungen

Abbildung 6.18: Passwort ändern

Mit einem Klick auf „Profil“ kann der Benutzer seine Profildaten einsehen und gegebenenfalls ändern. Zu den Profildaten gehören der Benutzername, der Vor- und Nachname des Nutzers, die verwendete Sprache und das Geschlecht. Auch die E-Mail-Adresse ist Teil des Benutzerprofils. Diese kann jedoch zum aktuellen Zeitpunkt nicht über die App geändert werden.
Der Nutzer kann innerhalb der App sein Passwort zu ändern. Hierzu klickt er auf „Pass-

wort ändern“ und gibt sein altes und das neue Wunschpasswort ein. Beim Speichern der Angaben werden die Daten zurück an das Backend gesendet. Die Änderung des Profils ist somit im Offline-Modus nicht verfügbar.

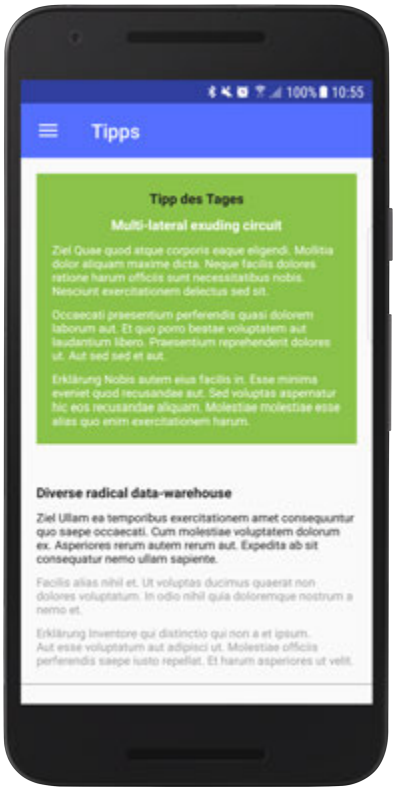


Abbildung 6.19: Tipps

Abbildung 6.20: FAQ

6.7 Tipps

Eine weitere Funktionalität der Track Your Tinnitus App sind Tipps. Die Tipps bieten Nutzern die Möglichkeit, weitere Hinweise zu ihren Symptomen und zum Umgang mit ihrem Tinnitus zu erlangen. Der Nutzer kann sich mit einem Klick auf „Tipps“ zum einen den Tipp des Tages, als auch alle weiteren Tipps anzeigen lassen. Für die Darstellung der Tipps ist eine bestehende Internetverbindung erforderlich. Das bedeutet, dass diese Funktionalität im Offline-Modus nicht verfügbar ist.

6.8 FAQ

FAQ ermöglichen es dem Benutzer, potenziell Antworten auf Fragen zu erhalten, die ihn beschäftigen. Es werden Zusatzinformationen zur Applikation und dem Tinnitus als solches vermittelt. Die FAQ werden regelmäßig aktualisiert und um neue Fragen, die häufig gestellt werden, ergänzt. Sie sind ebenfalls nur bei bestehender Internetverbindung verwendbar.

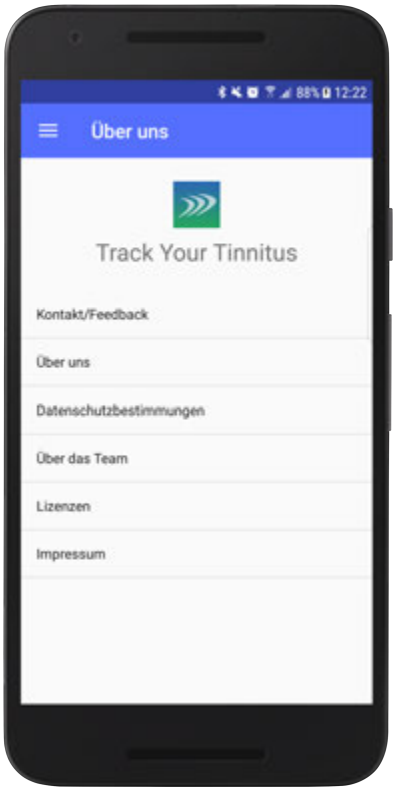


Abbildung 6.21: Über uns

6.9 Über uns

Unterhalb des Menüpunkts „Über uns“ finden Benutzer Hilfestellungen und Informationen rund um das Track Your Tinnitus Team, die Applikation und rechtliche Hinweise. Unter „Über uns“ erfahren Benutzer mehr über das Track Your Tinnitus Projekt. Die „Datenschutzbestimmungen“ definieren den Umgang mit den Nutzerdaten und deren Verwendung. Der Menüpunkt „Über das Team“ gibt den Nutzern Auskunft über das Team hinter dem Track Your Tinnitus Projekt und der zugehörigen Anwendungen. Rechtliche Hinweise lassen sich unterhalb der Menüpunkte „Impressum“ und „Lizenzen“ einsehen.

6.10 Logout

Natürlich haben Benutzer auch die Möglichkeit, sich aus der Applikation auszuloggen. Dies geschieht über den Menübutton „ABMELDEN“. Ein Logout zieht die Löschung

der nutzerspezifischen Daten auf dem Endgerät nach sich. Das bedeutet, dass nach einem Logout erst ein Login durch einen anderen oder denselben Nutzer erfolgen muss, bevor die Applikation erneut alle Daten bereitstellen und so z.B. im Offline-Modus verwendet werden kann. Es ist zu beachten, dass hierbei gerätespezifische Einstellungen, wie beispielsweise die Häufigkeit der Benachrichtigungen, nach einem Logout nicht wiederherstellbar sind, da diese nicht mit dem Backend synchronisiert werden können.

6.11 Login erforderlich

Bei jedem erfolgreichen Login wird ein Benutzertoken erzeugt, welches eine definierte Lebensdauer besitzt und mithilfe dessen sich die Applikation fortan am Backend authentifizieren kann. Das Token wird innerhalb eines bestimmten Zeitraumes automatisch verlängert. Wird dieser Zeitraum überschritten, z.B. durch andauernde Verwendung des Offline-Modus, kann ein erneuter Login erforderlich werden. In diesem Fall erscheint automatisch der Login-Dialog mit einer entsprechenden Hinweismeldung. Die App kann dann erst weiter verwendet werden, wenn sich ein Benutzer mit gültigen Daten und bestehender Internetverbindung einloggt. Dadurch wird sichergestellt, dass die Daten auf dem mobilen Endgerät nicht veralten und ein regelmäßiger Austausch mit dem Backend stattfinden kann.

7

Anforderungsabgleich

Nachfolgend werden alle Anforderungen aus Kapitel 3 erneut aufgegriffen und auf ihre Umsetzung untersucht, um so den Ist-Zustand der umgesetzten Applikation bewerten zu können.

7.1 Funktionale Anforderungen

Anhand der funktionalen Anforderungen kann überprüft werden, ob der realisierte Prototyp den gewünschten Vorstellungen entspricht und alle erwarteten Funktionen zulässt.

Allgemeine funktionale Anforderungen

Nr.	Titel und Bewertung
(1)	**Nutzer können sich in der App registrieren** *Anforderung erfüllt.* Benutzer haben die Möglichkeit, direkt über die App ein Benutzerkonto anzulegen, was im Anschluss durch eine E-Mail verifiziert werden muss (siehe Kapitel 6.1.2).
(2)	**Nutzer können die App auch ohne Internetverbindung verwenden** *Anforderung erfüllt.* Fragebögen können während des Offline-Modus wie gewohnt ausgefüllt werden. Entsprechende Antworten werden bis zur erneuten Verfügbarkeit der Internetverbindung vorgehalten und nachgelagert übertragen. Wichtige Details zu Fragebögen und Studien sind ebenfalls einsehbar (siehe Kapitel 6.5.1).

(3)	**Nutzer können ihre Sprache auswählen** *Anforderung erfüllt.* Zum Zeitpunkt der Erstellung dieser Arbeit sind nur zwei Sprachen verfügbar. Die Applikation unterstützt weitere Sprachen automatisch, sofern sie serverseitig bereitgestellt werden. Selektiert ein Benutzer in der App eine verfügbare Sprache, wird diese Einstellung an das Backend übertragen, um Konsistenz zwischen den Endgeräten zu ermöglichen.

Tabelle 7.1: Abgleich allgemeiner funktionaler Anforderungen

Funktionale Anforderungen an die Studienteilnahme

Nr.	Titel und Bewertung
(4)	**Nutzer können an verschiedenen Studien gleichzeitig teilnehmen** *Anforderung erfüllt*, siehe Kapitel 6.3.1.
(5)	**Nutzer können alle verfügbaren Studien sehen** *Anforderung erfüllt*. Benutzer sehen zum einen alle Studien, in denen sie eingeschrieben sind, können aber bei bestehender Internetverbindung auch alle weiteren verfügbaren Studien einsehen (siehe Kapitel 6.3).
(6)	**Nutzer können Studieneinladungen annehmen** *Anforderung erfüllt*. Wenn Nutzer von einem Studienleiter eingeladen werden, werden sie zum einen per E-Mail informiert, sehen aber auch in der Gesamtübersicht der Studien in der App die vorliegenden Einladungen und können sie annehmen.
(7)	**Nutzer können an einer privaten Studie teilnehmen, sofern sie das Passwort kennen** *Anforderung erfüllt*. Die privaten Studien werden in der Gesamtübersicht mit einem entsprechenden Symbol zur Kennzeichnung angezeigt.

Tabelle 7.2: Abgleich funktionaler Anforderungen an die Studienteilnahme

Funktionale Anforderungen an die Fragebögen

Nr.	Titel und Bewertung
(8)	**Nutzer werden zum Ausfüllen statistischer Fragebögen nach dem Einschreiben in eine Studie aufgefordert** *Anforderung erfüllt*, siehe Kapitel 6.4.1.
(9)	**Nutzer erhalten Aktualisierungen bzw. Änderungen von Fragebögen** *Anforderung erfüllt.* Die Synchronisierung erfolgt in regelmäßigen Abständen im Hintergrund, unbemerkt vom Nutzer.
(10)	**Nutzer werden zum Ausfüllen neuer statistische Fragebögen aufgefordert** *Anforderung erfüllt*, siehe Kapitel 6.4.1.
11)	**Nutzer können nur aktive Fragebögen ausfüllen** *Anforderung erfüllt.* Deaktivierte Fragebögen werden in der Hauptansicht nicht dargestellt. In der Übersicht für die Fragebögen einer Studie erscheinen sie, sind aber als inaktiv gekennzeichnet.
(12)	**Nutzer können die Schwankungen ihres Tinnitus und entsprechende Faktoren regelmäßig erfassen** *Anforderung erfüllt.* Ein entsprechender Fragebogen ist in der Studie „Track Your Tinnitus“ backendseitig hinterlegt. Die App zeigt stets alle aktiven Fragebögen der Studie, der Benutzer kann diese wie in Kapitel 6.4.2 beschrieben, zu beliebigen Zeitpunkten ausfüllen. Der Benutzer kann sich aus der Track Your Tinnitus Studie nicht ausschreiben.
(13)	**Gespeicherte Antworten werden sofort oder nachgelagert an das Backend gesendet** *Anforderung erfüllt.* Besteht keine Internetverbindung werden die Antworten gespeichert und unbemerkt vom Benutzer nachträglich gesendet.

(14)	**Nutzer werden nicht von vorgegebenen Werten beeinflusst** *Anforderung erfüllt.* Alle Fragetypen, wie z.B. der Schieberegler, besitzen keinen applikationsseitig initialen Wert. Ist vom Studienleiter jedoch ein Standardwert vorgesehen, wird dieser angezeigt. Der Studienleiter kann also selbst bestimmen, ob vorgegebene Werte in manchen Fällen sinnvoll sind und diese konfigurieren.
(15)	**Der Pegel der Umgebungsgeräusche eines Nutzers kann erfasst werden** *Anforderung erfüllt.* Die Lautstärkemessung findet im Hintergrund statt, der Benutzer wird über ein Symbol in der Statusbar über die Tonaufnahme informiert.
(16)	**Nutzer können die Geräuschpegelmessung deaktivieren** *Anforderung erfüllt*, siehe Kapitel 6.4.2.
(17)	**Die Herzfrequenz eines Nutzers kann erfasst werden** *Anforderung erfüllt.* Der Nutzer benötigt zur Herzfrequenzmessung eine GARMIN vívosmart HR+ oder eine andere ANT+-fähige Pulsuhr. Er wird über ein Symbol in der Statusbar über die Messung informiert.
18)	**Nutzer können die Herzfrequenzmessung deaktivieren** *Anforderung erfüllt*, siehe Kapitel 6.4.2.
(19)	**Der Standort bzw. der Bewegungszustand des Nutzers kann erfasst werden** *Anforderung erfüllt.* Der Benutzer wird über ein Symbol in der Statusbar über die Standortbestimmung informiert.
(20)	**Nutzer können die Standortbestimmung deaktivieren** *Anforderung erfüllt*, siehe Kapitel 6.4.2.
(21)	**Der Nutzer wird anhand bestimmter Zeitpläne, an das Ausfüllen von Fragebögen erinnert** *Anforderung erfüllt*, siehe Kapitel 5.4.
(22)	**Nutzer können die Benachrichtigungsfunktion individuell anpassen** *Anforderung erfüllt*, siehe Kapitel 6.4.2.
(23)	**Nutzer können den Klingelton einer Benachrichtigung anpassen** *Anforderung erfüllt*, siehe Kapitel 6.5.2.

(24)	**Nutzer können ihre Ergebnisse in der App sehen** *Anforderung nicht erfüllt*. Ergebnisse werden backendseitig noch nicht ausreichend bereitgestellt. Dadurch war es nicht möglich, die kumulierten Ergebnisse der Fragebögen eines Nutzers in der Applikation darzustellen.
(25)	**Nutzer können ihre Daten exportieren** *Anforderung nicht erfüllt*. Aufgrund der Nichtverfügbarkeit der Ergebnisse ist eine Exportfunktion bislang nicht sinnvoll, da keine Ergebnisse in der Applikation vorhanden sind, die exportiert werden könnten.

Tabelle 7.3: Abgleich funktionaler Anforderungen an die Fragebögen

Hinsichtlich der funktionalen Anforderungen kann abschließend festgestellt werden, dass bis auf zwei Anforderungen, alle vorgesehenen Funktionalitäten im Prototyp realisiert wurden. Die Anforderungen 24 und 25 konnten aufgrund des momentanen Zustandes der REST-API nicht umgesetzt werden. Wie der Vorstellung in Kapitel 6 zu entnehmen ist, wurden aber weitere Funktionalitäten wie FAQ und Tipps in den Prototyp integriert, da sie zum Zeitpunkt der Umsetzung bereits vom Backend unterstützt wurden.

7.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen lassen sich schwerer messen, als die zuvor bewerteten funktionalen Anforderungen. Nachfolgend wird deshalb zu jeder Anforderung eine kurze Einschätzung bezüglich ihres Erfüllungsgrads gegeben.

Nr.	Titel und Bewertung
(1)	**Technische Anforderungen** Die Applikation ist unter Android 4.1 oder neuer lauffähig. Getestet wurde die Applikation mit einem Samsung Galaxy S7 edge (Android 7.0), einem Samsung Galaxy S3 (Android 4.3), einem Samsung Galaxy Note 10.1 (Android 4.4.2) und dem Android Studio Emulator.

(2)	**Bedienbarkeit** Bei der Umsetzung wurde darauf geachtet, dass vorgesehene Benutzerinteraktionen einfach verständlich sind. Die Bedienung der App soll einfach erlernbar sein. Entsprechende Hinweistexte sorgen dafür, dass Nutzer die Applikation leichter verstehen.
(3)	**Aussehen** Die App entspricht, soweit der Aufwand für die Umsetzung vertretbar war, den Gestaltungsprinzipen des *Material Design* der Android Plattform. Es wurden viele Standard-Komponenten des Android SDK verwendet, wodurch eine einheitliche Optik zustande kommt.
(4)	**Änderbarkeit** Es wurde großer Wert darauf gelegt, durch die Einhaltung bestimmter Patterns und Prinzipien während des Entwicklungsprozesses, die App möglichst gut anpassbar und wartbar zu machen. Alle Komponenten haben eine bestimmte Hauptaufgabe und sind, soweit möglich, lose gekoppelt.
(5)	**Sicherheit** Bei der Architektur und Implementierung wurde darauf geachtet, dass vertrauliche Daten wie Passwörter nirgends auf dem Gerät abgelegt werden. Andere Daten werden in der vorgesehenen App-internen SQLite Datenbank, bzw. in den internen Einstellungen (SharedPreferences) einer Applikation abgelegt. Die Kommunikation mit dem Server erfolgt über das Protokoll Hypertext Transfer Protocol Secure (https).
(6)	**Fehlertoleranz** Es wurde darauf geachtet, dass Fehler, die vom Nutzer nicht behoben werden können, keine Beeinträchtigung der Bedienbarkeit nach sich ziehen. Benutzer werden auf Fehler, die sie verursacht haben bzw. die sie selbst beheben können, hingewiesen. Dabei wurde darauf geachtet, dass nicht nur auf Fehler hingewiesen, sondern auch mögliche Lösungsschritte präsentiert werden.

Tabelle 7.4: Abgleich nichtfunktionaler Anforderungen

8

Fazit

8.1 Zusammenfassung

Das Ziel dieser Arbeit bestand darin, eine bereits vorhandene mobile Applikation für die Durchführung Klinischer Studien zur Messung der individuellen Tinnitus Wahrnehmung unter Berücksichtigung erweiterter Anforderungen von Grund auf neu zu konzipieren und prototypisch auf Basis der Android Plattform umzusetzen.

Dazu wurden zunächst anhand bereits vorliegender Projekte und Arbeiten alle Anforderungen an die Applikation identifiziert und beschrieben. Darauf aufbauend wurde ein Architekturkonzept ausgearbeitet. Um die Anforderungen weitestgehend abzudecken, wurden alle Abläufe innerhalb der Applikation modelliert, einzelne Kernaspekte der Applikation genauer aufgegriffen und daraufhin hilfreiche Bibliotheken identifiziert. Im darauffolgenden Kapitel wurden einzelne Implementierungsaspekte genauer erläutert und Umsetzungshürden gekennzeichnet.

Anschließend folgte eine Vorstellung der gesamten Applikation aus Nutzersicht. Dieses Kapitel soll Benutzern auch zukünftig den Einstieg in die Applikation erleichtern, vorhandene Funktionalitäten genauer erläutern und dadurch eine Art Benutzerdokumentation darstellen.

Ein abschließender Anforderungsabgleich zeigte vorhandene Abweichungen des Soll- und Ist-Standes der Applikation. Diese beschränkten sich auf Anforderungen bezüglich der Ergebnisdarstellung, die um Zeitpunkt der Umsetzung noch nicht vom Backend unterstützt wurde. Dazu zählen die Darstellung der Ergebnisse in der App (Anforderung 24)

und die darauf aufbauende Exportfunktion der Daten aus der App (Anforderung 25). Abgesehen davon wurden alle weiteren Anforderungen erfüllt.

Insgesamt wurde im Rahmen dieses Projektes ein funktionaler Prototyp geschaffen, der bis auf wenige Ausnahmen alle geforderten Funktionalitäten zur Erfassung von Tinnitusschwankungen und zur Durchführung Klinischer Studien beinhaltet. Der Prototyp wurde auf unterschiedlichen Endgeräten getestet, eine Anbindung an die neue Version des Backends ist ebenfalls testweise erfolgt. Dadurch befindet sich der Prototyp bereits in einem Beta-Zustand und könnte im nächsten Schritt anhand von Nutzerstudien evaluiert, bezüglich Fehlerfällen gehärtet und weiter verfeinert werden.

8.2 Ausblick

Das Projekt Track Your Tinnitus und alle damit zusammenhängenden Applikationen sind sicherlich noch nicht in ihrem Endstadium angekommen. Zum aktuellen Zeitpunkt befindet sich das gesamte Projekt in der Weiterentwicklung. Während der Umsetzung des Prototyps wurde auch ein komplett neues Backend erarbeitet. Es stellt die Basis für viele weitere neue Funktionalitäten dar, wie die verbesserte Interaktion mit Studienleitern oder den Austausch mit anderen Betroffenen.

Auf der Grundlage dieses erweiterten Backends sind auch für die Android Applikation weitere Funktionalitäten denkbar. Zunächst einmal sollten bei Verfügbarkeit die Ergebnis- und Exportfunktion in die Applikation integriert werden. Denkbar ist hier eine umfangreiche Ergebnisdarstellung, die dem Nutzer aggregierte Werte anzeigt. Davon abgesehen wäre eine Feedbackfunktion sinnvoll, über die Studienteilnehmer individuelles Feedback von ihrem Studienleiter erhalten können. Auch der direkte Austausch mit den Studienleitern wird als sinnvolle Erweiterung erachtet.

Viele Tinnitus Betroffene haben in ihrem persönlichen Umfeld keinen Ansprechpartner. Menschen, die nicht unter Tinnitus leiden, können die Probleme der Betroffenen selten nachvollziehen. Aus diesem Grund sollte auch die soziale Komponente nicht vernachlässigt werden. Betroffene könnten sich über die Track Your Tinnitus Applikationen mit anderen austauschen und Erfahrungswerte mitteilen. Oft hilft es Patienten schon, ver-

standen zu werden und zu wissen, dass sie mit ihren Beschwerden nicht alleine da stehen.

Backendseitig ist außerdem bereits eine Like-Funktionalität vorgesehen. Diese könnte auch in die Applikation integriert werden, mit dem Ziel, dass Nutzer zum Beispiel Studien oder Tipps favorisieren können. Darauf aufbauend könnte ein Sortieralgorithmus dafür sorgen, dass besonders beliebte Inhalte weiter oben dargestellt werden.

Hinsichtlich der Anbindung von „Wearables“ wie Gesundheits- und Fitnesstrackern besteht großes Erweiterungspotenzial. Es ist wahrscheinlich, dass nicht nur die Herzfrequenz eines Tinnitus-Betroffenen, sondern auch andere Werte wie Blutdruck, Stresslevel oder Atmung Auswirkungen auf den Tinnitus haben. Auch könnten Angewohnheiten bezüglich Schlafrhythmus oder Sport potenzielle Einflussfaktoren darstellen. Die Messung dieser Werte rückt mit der technologischen Entwicklung in greifbare Nähe und sollte auf keinen Fall unberücksichtigt bleiben.
Auch sollten Benutzer die Möglichkeit haben, zusätzliche Informationen, wie Medikation, Urlaub oder Essgewohnheiten auf einer täglichen Basis zu erfassen und so über einen Zeitraum gewisse Faktoren mit ihren Tinnitusschwankungen in Verbindung bringen.

Wenn die Track Your Tinnitus Plattform weiter neue Technologien zu ihrem Zweck nutzt und Anwender durch hohe Bedienbarkeit und Fehlerfreiheit der Applikationen einen persönlichen Nutzen erkennen lässt, besteht großes Potenzial, dass sie ein fester Bestandteil der täglichen Routine Tinnitus geschädigter Patienten werden. Dadurch können weiterhin umfangreiche Informationen über den Tinnitus und mit ihm in Zusammenhang stehende Faktoren gesammelt werden, die schließlich dabei helfen können, über die Auslöser des Tinnitus weiter aufzuklären und so Patienten eine optimale Behandlung zukommen zu lassen.

Literaturverzeichnis

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

[2] Pilgramm, M., Rychlik, R., Lebisch, H., et al.: Tinnitus in der Bundesrepublik Deutschland – eine repräsentative epidemiologische Studie. HNO aktuell **7** (1999) 261–265

[3] Greimel, K.V.: Psychologische Faktoren oder Verhaltensfaktoren bei andernorts klassifizierten Erkrankungen - HNO Heilkunde: Tinnitus. In: Handbuch der klinisch-psychologischen Behandlung. 2 edn. Springer Science & Business Media, Wien (2006) 513–523

[4] Tönnies, S.: Tinnitus. In: Verhaltensmedizin: Psychobiologie, Psychopathologie und klinische Anwendung. W. Kohlhammer Verlag, Stuttgart (2008) 266–279

[5] Deutsche Gesellschaft für Hals-, Nasen-, Ohrenheilkunde, Kopf- und Halschirurgie e.V: S3-Leitlinie 017/064: Chronischer Tinnitus. Sz-leitlinie, Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften e.V. (AWMF), Berlin (2015) `http://www.awmf.org/uploads/tx_szleitlinien/017-064l_S3_Chronischer_Tinnitus_2015-02.pdf`. Abruf: 28.02.2015.

[6] Ametsreiter, H.: Smartphone-Markt: Konjunktur und Trends (2017) `https://www.bitkom.org/Presse/Anhaenge-an-PIs/2017/02-Februar/Bitkom-Pressekonferenz-Smartphone-Markt-Konjunktur-und-Trends-22-02-2017-Praesentation.pdf`. Abruf: 06.08.2017.

[7] Bitkom: Am Puls der Zeit: Smartwatches haben großes Potenzial (2016) `https://www.bitkom.org/Presse/Presseinformation/Am-Puls-der-Zeit-Smartwatches-haben-grosses-Potenzial.html`. Abruf: 06.08.2017.

[8] Sonormed GmbH: Tinnitracks: Wirkung der Therapie (o.J.) `http://www.tinnitracks.com/de/therapiewirkung`. Abruf: 15.08.2017.

[9] Stracke, H., Okamoto, H., Pantev, C.: Customized notched music training reduces tinnitus loudness. Communicative integrative biology **3(3)** (2010) 274–277

[10] Teismann, H., Okamoto, H., Pantev, C.: Short and intense tailor-made notched music training against tinnitus: the tinnitus frequency matters. PloS one **6(9)** (2011)

[11] Stracke, H., Okamoto, H., Pantev, C.: Playing and listening to tailor-made notched music: cortical plasticity induced by unimodal and multimodal training in tinnitus patients. Neural Plasticity **516163** (2014)

[12] Teismann, H., Wollbrink, A., Okamoto, H., et al.: Combining transcranial direct current stimulation and tailor-made notched music training to decrease tinnitus-related distress – a pilot study. PloS one **9(2)** (2014)

[13] Stein, A., Engell, A., Junghoefer, M., et al.: Inhibition-induced plasticity in tinnitus patients after repetitive exposure to tailor-made notched music. Clinical Neurophysiology **126(5):1007-15** (2015)

[14] Stein, A., Wunderlich, R., Lau, P., et al.: Clinical trial on tonal tinnitus with tailor-made notched music training. BMC Neurology **16:38** (2016)

[15] Healint Pte Ltd: Migraine Buddy - für Migräne und Kopfschmerzen (2017) `https://play.google.com/store/apps/details?id=com.healint.migraineapp`. Version: 23.8.2. Abruf: 04.08.2017.

[16] Healint Pte Ltd: Upcoming Projects (o.J.) `https://www.healint.com/upcoming-projects/`, Abruf: 04.08.2017.

[17] AsthmaMD: AsthmaMD (2016) `https://play.google.com/store/apps/details?id=com.mobilebreeze.AsthmaMD`. Version: 1.7.10.5. Abruf: 04.08.2017.

[18] Japps Medical : Depressionstest (2016) `https://play.google.com/store/apps/details?id=nl.japps.android.depressiontest`. Version: 1.24. Abruf: 04.08.2017.

[19] axovis GmbH: RheumaTrack RA (2016) `https://play.google.com/store/apps/details?id=com.rheumatrack`. Version: 2.1.1. Abruf: 04.08.2017.

[20] Hazell, J.W.P.: Measurement of tinnitus in humans. In Evered, D., Lawrenson, G., eds.: Tinnitus. Volume 85 of CIBA Foundation Symposium., London, Pitman Books (1981) 35–53

[21] Newman, C.W., Sandridge, S.A.: Tinnitus Questionnaires. In: Tinnitus Theory and Management. BC Decker, Hamilton, Ontario (2004) 237–254

[22] Ryan, D., Cobern, W., Wheeler, J., et al.: Mobile phone technology in the management of asthma. Journal of Telemedicine and Telecare **11** (2005) 43–46

[23] Google Inc., Open Handset Alliance: Up and running with material design (o. J.) `https://developer.android.com/design/index.html`. Abruf: 28.08.2017.

[24] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. dissertation, University of California, Irvine (2000) `https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`. Abruf: 28.08.2017.

[25] Google Inc., Open Handset Alliance: Activities (o. J.) `https://developer.android.com/guide/components/activities/index.html`. Abruf: 28.08.2017.

[26] Google Inc., Open Handset Alliance: Fragments (o. J.) `https://developer.android.com/guide/components/fragments.html`. Abruf: 28.08.2017.

[27] Google Inc., Open Handset Alliance: Input Events (o. J.) `https://developer.android.com/guide/topics/ui/ui-events.html`. Abruf: 28.08.2017.

[28] Oracle: The Java Tutorials: Generic Types (2013) `https://docs.oracle.com/javase/tutorial/java/generics/types.html`. Abruf: 15.07.2017.

[29] Google Inc., Open Handset Alliance: SQLiteOpenHelper (o. J.) `https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html`. Abruf: 28.08.2017.

[30] The Open Source Initiative: The MIT License (o. J.) `https://opensource.org/licenses/mit-license.php`. Abruf: 28.08.2017.

[31] The Apache Software Foundation: Apache License, Version 2.0 (2004) http://www.apache.org/licenses/LICENSE-2.0. Abruf: 28.08.2017.

[32] Google Inc., Open Handset Alliance: Transmitting Network Data Using Volley (o. J.) https://developer.android.com/training/volley, Abruf: 30.07.2017.

[33] Teegarden, C.: Volley vs. Retrofit - A Battle Royale (2014) https://www.captechconsulting.com/blogs/volley-vs-retrofit---a-battle-royale. Abruf: 07.08.2017.

[34] Jawaid, A.: Comparison of Android networking libraries: OkHTTP, Retrofit, and Volley (2013) https://stackoverflow.com/questions/16902716/comparison-of-android-networking-libraries-okhttp-retrofit-and-volley. Abruf: 03.07.2017.

[35] Ruesch, J.: Android Async HTTP Clients: Volley vs Retrofit (2013) http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/. Abruf: 03.07.2017.

[36] Muzaffar, A.: Is Retrofit faster than Volley? The answer may surprise you! (2016) https://medium.com/@ali.muzaffar/is-retrofit-faster-than-volley-the-answer-may-surprise-you-4379bc589d7c. Abruf: 03.07.2017.

[37] Lew, D.: joda-time-android (2017) https://github.com/dlew/joda-time-android/releases/tag/v2.9.9. Abruf: 30.07.2017.

[38] Google Inc.: Get Started with Analytics for Android (2017) https://firebase.google.com/docs/analytics/android/start/. Abruf: 15.07.2017.

[39] Google Inc., Open Handset Alliance: Android Developers: Support Library (o. J.) https://developer.android.com/topic/libraries/support-library/index.html. Abruf: 15.07.2017.

[40] Apache Software Foundation: Apache Commons, Commons Lang (2017) https://commons.apache.org/proper/commons-lang/. Version: 3.6. Abruf 17.07.2017.

[41] Dynastream Innovations: Flexible and Interoperable Data Transfer (FIT) Protocol License (o.J.) https://www.thisisant.com/developer/ant/licensing/flexible-and-interoperable-data-transfer-fit-protocol-license. Abruf: 28.08.2017.

[42] Google Inc., Open Handset Alliance: Services - Extending the Service class (o. J.) https://developer.android.com/guide/components/services.html#ExtendingService, Abruf: 30.07.2017.

[43] Google Inc., Open Handset Alliance: AsyncTask Class Reference (o. J.) https://developer.android.com/reference/android/os/AsyncTask.html. Abruf: 30.07.2017.

[44] Google Inc., Open Handset Alliance: Saving Key-Value Sets (o.J.) https://developer.android.com/training/basics/data-storage/shared-preferences.html. Abruf: 28.08.2017.

[45] Google Inc., Open Handset Alliance: Localizing with Resources (o. J.) https://developer.android.com/guide/topics/resources/localization.html#managing-strings. Abruf: 07.07.2017.

[46] Dunkel, J., Holitschke, A.: Softwarearchitektur für die Praxis. Springer-Verlag (2013)

[47] Google Inc., Open Handset Alliance: AlarmManager Class Reference (o. J.) https://developer.android.com/reference/android/app/AlarmManager.html, Abruf: 30.07.2017.

[48] Google Inc., Open Handset Alliance: Scheduling Repeating Alarms - Start an Alarm When the Device Boots (o. J.) https://developer.android.com/training/scheduling/alarms.html#boot, Abruf: 30.07.2017.

[49] Google Inc., Open Handset Alliance: Requesting Permissions at Run Time (o. J.) https://developer.android.com/training/permissions/requesting.html. Abruf: 30.07.2017.

[50] Google Inc., Open Handset Alliance: Requesting Permissions (o. J.) https://developer.android.com/guide/topics/permissions/requesting.html#normal-dangerous. Abruf: 30.07.2017.

[51] Google Inc., Open Handset Alliance: Receiving Location Updates (o. J.) https://developer.android.com/training/location/receive-location-updates.html, Abruf: 30.07.2017.

[52] Garmin: vívosmart HR/HR+ – Broadcasting Heart Rate Data to Garmin® Devices (o.J.) http://www8.garmin.com/manuals/webhelp/vivosmarthr/EN-US/GUID-AC0E8CD6-FD71-4845-ADB1-2E97F785EFAC.html. Abruf: 15.08.2017.

[53] Schobel, J.: dingodocs (2017) https://packagist.org/packages/johannesschobel/dingodocs, Abruf: 29.08.2017.

A

Track Your Tinnitus REST Schnittstelle

Die vorhandene TYT-REST Schnittstelle, auf der der umgesetzte Prototyp aufsetzt, wird nachfolgend näher erläutert. Zum Zeitpunkt der Umsetzung der Arbeit ist eine Testversion der Schnittstelle bereits verfügbar. Als Schnittstellenbeschreibung (siehe Abbildungen A.1 und A.2) dient eine mit Hilfe von dingodocs[1] bereitgestellte Website, die Informationen über alle möglichen Operationen und Parameter liefert. Auf eine vollständige Beschreibung aller Operationen soll hier verzichtet werden, da sich die Schnittstelle zum Umsetzungszeitpunkt im Aufbau befindet und Änderungen wahrscheinlich sind.

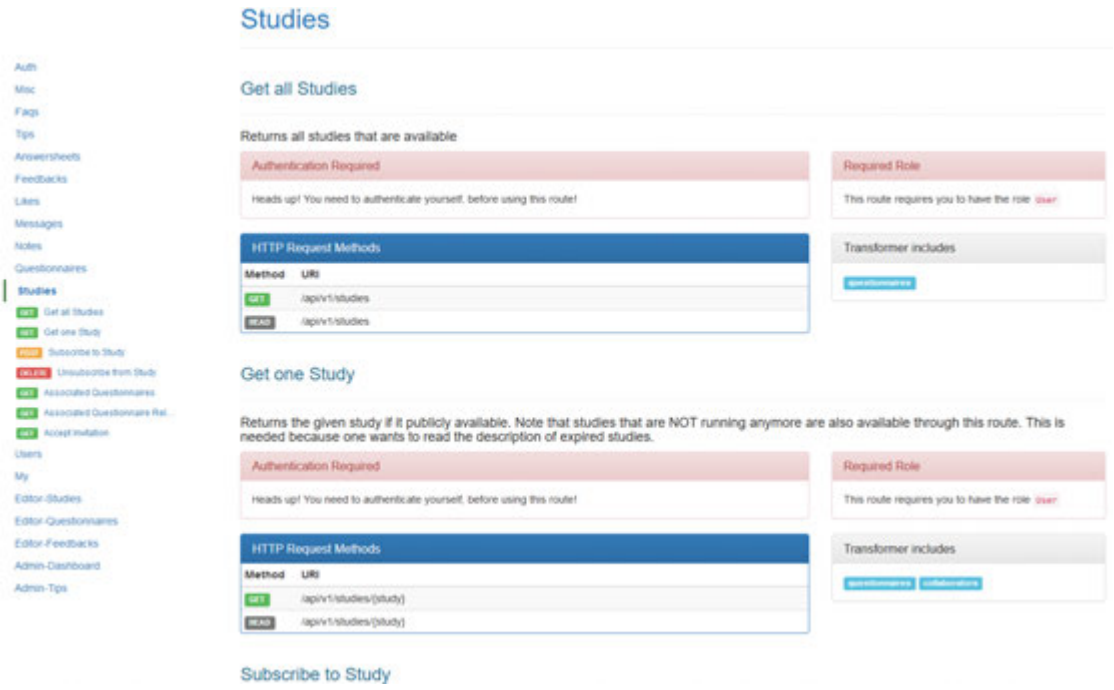


Abbildung A.1: Ausschnitt 1 Schnittstellendokumentation

[1] Tool zur automatischen Generierung einer API Dokumentation für eine Laravel/Dingo API, siehe [53]

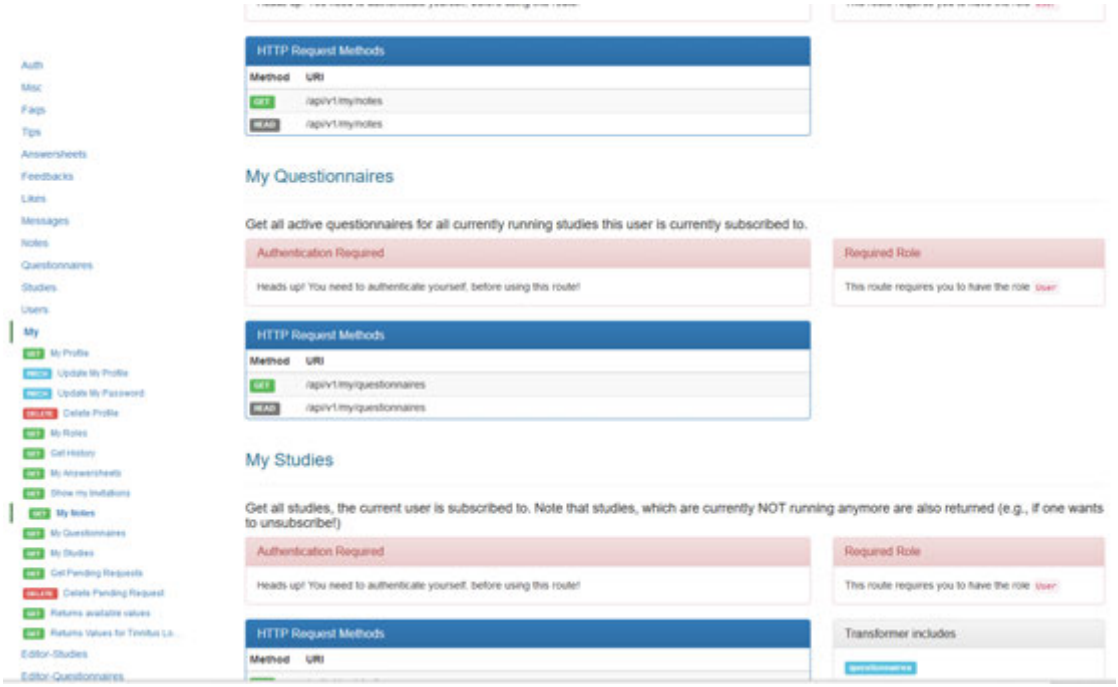


Abbildung A.2: Ausschnitt 2 Schnittstellendokumentation

Im Folgenden werden Beispielantworten der REST-API aufgelistet, um die Struktur der Antworten von der REST Schnittstelle im JSON Format zu veranschaulichen (siehe Listing A.1 und Listing A.2)

```
{
   "data": {
      "type": "tips",
      "id": "1",
      "attributes": {
         "name": "Reverse-engineered solution-oriented installation",
         "is_active": 1,
         "title": "Object-based object-oriented installation",
         "text": "Ad qui quia a deleniti sed rerum ea aliquid. Est quia
         quidem et. Et qui cum aperiam nihil. Temporibus nemo ratione
         aut.\nPariatur et dolorum sunt et. Consequatur atque reiciendis
         nihil ea."
      },
      "links": {
         "self": "tyt.johannesschobel.com/api/v1/tips/1"
      }
```

```
    },
    "links": {
        "next": "https://tyt.johannesschobel.com/api/v1/tips/2"
    },
    "meta": {
        "count": 1,
        "include": []
    }
}
```

Listing A.1: JSON Response der TYT-REST Schnittstelle unter /api/v1/tips/1

```
{
    "data": [
    {
        "type": "studies",
        "id": "1",
        "attributes": {
            "name": "tinnitus",
            "title": "Synergistic needs-based GraphicInterface",
            "description": "Distinctio deleniti numquam impedit. Tenetur libero
                quia molestiae aut. Repudiandae ea non repudiandae consequatur.
                Eum rerum possimus inventore. Eum nostrum unde nobis distinctio
                sed odit.",
            "picture": "",
            "accesstype": null,
            "starts_at": 1503575110,
            "ends_at": 1976960710,
            "is_private": 0,
            "is_running": true
        },
        "meta": {
            "likes": 0,
            "avg": 0
        },
        "links": {
            "self": "tyt.johannesschobel.com/api/v1/studies/1"
        }
    }
    ],
    "meta": {
        "count": 1,
```

```
28          "include": [
29              "questionnaires"
30          ]
31      },
32      "links": {
33          "self": "gekuerzt",
34          "first": "gekuerzt",
35          "last": "gekuerzt"
36      }
37  }
```

Listing A.2: JSON Response der TYT-REST Schnittstelle unter /api/v1/my/studies

Listing A.3 zeigt schließlich einen Ausschnitt aus dem applikationsseitig hinterlegten Interface, das die Schnittstelle zur REST API abbildet.

```
1  import okhttp3.ResponseBody;
2  import retrofit2.Call;
3  import retrofit2.Retrofit;
4  import retrofit2.http.Body;
5  [...]
6
7  public interface TYTService {
8
9      String BASE_URL = "https://tyt.test-system-host.com/api/v1/";
10
11     Retrofit retrofit = new Retrofit.Builder()
12     .baseUrl(BASE_URL)
13     .addConverterFactory(TYTServiceHelper.gsonConverter()).build();
14
15     @GET("misc/locales")
16     Call<Response<List<Locale>>> getLocales();
17
18     @GET("misc/texts/_?limit=2000")
19     Call<Response<List<Text>>> getTexts(@Query("token") String token,
           @Query("updated_at") String updatedAt);
20
21     @GET("misc/feedback")
22     Call<ResponseBody> sendFeedback(@Body Request<String> data);
23
24     @GET("user/{id}")
```

```
25      Call<Response<User>> getUser(@Path("id") String id, @Query("token")
            String token);
26
27      /*  auth */
28      @POST("auth/register")
29      Call<ResponseBody> register(@Body Request<Registration> data);
30
31      @POST("auth/verify/resend")
32      Call<ResponseBody> resendVerification(@Body Request<Email> data);
33
34      @POST("auth/password/reset/instructions")
35      Call<ResponseBody> initResetPassword(@Body Request<Email> data);
36
37      @POST("auth/password/reset")
38      Call<ResponseBody> resetPassword(@Body Request<ResetPassword> data);
39
40      @POST("auth/login")
41      Call<Response<LoginRes>> login(@Body Request<LoginReq> data);
42
43      @POST("auth/refresh")
44      Call<Response<LoginRes>> refresh(@Query("token") String token);
45
46      /*  My  */
47      @GET("my/profile")
48      Call<Response<Profile>> myProfile(@Query("token") String token);
49
50      @PATCH("my/profile")
51      Call<Response<Profile>> updateProfile(@Query("token") String token, @Body
            Request<UpdateProfile> data);
52
53      @PATCH("my/profile/password")
54      Call<ResponseBody> updatePassword(@Query("token") String token, @Body
            Request<UpdatePassword> data);
55
56      @GET("my/studies?include=questionnaires")
57      Call<Response<List<Study>>> myStudies(@Query("token") String token);
58
59      @GET("my/studies/pending")
60      Call<Response<List<Study>>> myPendingStudies(@Query("token") String
            token);
61
```

```
62      @GET("my/messages")
63      Call<ResponseBody> myMessages(@Query("token") String token);
64
65      @GET("my/invitations")
66      Call<Response<List<Invitation>>> myInvitations(@Query("token") String
            token);
67
68      @GET // Used for individually returned invitation urls
69      Call<ResponseBody> acceptInvitation(@Url String url, @Query("token")
            String token);
70
71      @GET("my/questionnaires")
72      Call<Response<List<Questionnaire>>> myQuestionnaires(@Query("token")
            String token);
73
74      /*  Questionnaires  */
75      @GET("questionnaires/{id}")
76      Call<Response<Questionnaire>> findQuestionnaire(@Path("id") String id,
            @Query("token") String token);
77
78      @GET("questionnaires/{id}/structure")
79      Call<List<QuestionnaireStructure>> questionnaireElements(@Path("id")
            String id, @Query("token") String token);
80
81      @POST("questionnaires/{id}/answersheets")
82      Call<ResponseBody> questionnaireAnswers(@Path("id") String id,
            @Query("token") String token, @Body Request<Answers> data);
83
84      /*  Studies  */
85      @GET("studies?is_running=true")
86      Call<Response<List<Study>>> allStudies(@Query("token") String token);
87
88      @GET("studies/{id}")
89      Call<Response<Study>> studyDetails(@Path("id") String id, @Query("token")
            String token);
90
91      @POST("studies/{id}/subscribe")
92      Call<ResponseBody> subscribeToStudy(@Path("id") String id,
            @Query("token") String token, @Body Request<SubscribeReq> data);
93
```

```
94      @Headers("Content-Length: 0") // DELETE should actually never have a
            body, but retrofit wants it this way...
95      @DELETE("studies/{id}/unsubscribe")
96      Call<ResponseBody> unsubscribeFromStudy(@Path("id") String id,
            @Query("token") String token);
97
98      @GET("studies/{id}/questionnaires")
99      Call<Response<List<Questionnaire>>> questionnairesOfStudy(@Path("id")
            String id, @Query("token") String token);
100
101     /*  FAQs   */
102     @GET("faqs")
103     Call<Response<List<Faq>>> allFaqs();
104
105     @GET("faqs/{id}")
106     Call<Response<Faq>> getFaq(@Path("id") String id);
107
108     /*  Tipps  */
109     @GET("tips")
110     Call<Response<List<Tip>>> allTips();
111
112     @GET("tips/random")
113     Call<Response<Tip>> randomTip();
114
115     @GET("tips/{id}")
116     Call<Response<Tip>> getTip(@Path("id") String id);
117 }
```

Listing A.3: TYTService Interface

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Name: Janina Strohm Matrikelnummer: 925253

Erklärung

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

Ulm, den .....

Janina Strohm