

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

Bachelorarbeit im Studiengang Informatik

Konzeption und Realisierung einer mobilen Anwendung zur Unterstützung Tinnitus-geschädigter Patienten

am Beispiel von iOS

vorgelegt von

Johannes Hueber

Dezember 2016

Prof. Dr. Manfred Reichert
Dipl. Inf. Rüdiger Pryss
839586
16.12.2016

Kurzfassung

Das Track Your Tinnitus Projekt wurde ins Leben gerufen, um es tinnituserkrankten Menschen zu ermöglichen, die Schwankungen der Tinnituswahrnehmung zu überwachen, sowie die daraus gewonnenen Daten für die Forschung zu verwenden. Tinnitus ist ein Symptom, bei dem der Betroffene Töne oder Geräusche wahrnimmt, obwohl es keine messbare physikalische Ursache gibt. In Deutschland leidet etwa 10 % der Bevölkerung einmal im Leben an Tinnitus. Bei 1% der deutschen Bevölkerung ist der Tinnitus so stark ausgeprägt, dass das Leben der betroffenen Menschen erheblich beeinträchtigt wird. Es gibt bei Tinnitus nicht das charakteristische Krankheitsbild, und somit auch keine eindeutige und einfache Therapie. Es gibt eine Reihe von Therapien, die bei manchen Betroffenen erfolgreich sind, bei anderen aber wiederum nicht. Dies verdeutlicht, dass es ein Wissensdefizit gibt, dem zum Wohle der Betroffenen entgegengewirkt werden muss.

Diese Arbeit zeigt, wie ein Betroffener unter Verwendung seines Smartphones Schwankungen seines Tinnitus überwachen und verfolgen kann. Zu diesem Zweck wurde die bisher verfügbare Track Your Tinnitus App (2014) überarbeitet und in Apples eigener Programmiersprache Swift neu programmiert. In der vorliegenden Arbeit wird die Entwicklung dieser App gezeigt und auf verschiedene Entwicklungsaspekte eingegangen, sowie die Umsetzung und die Funktion beschrieben.

Inhaltsverzeichnis

1	Ein.	leitung	3
	1.1	Motivation	3
	1.2	Aufbau der Arbeit	3
2	Ver	wandte Arbeiten	5
	2.1	Tinnitus HQ	5
	2.2	Tinnitracks	6
	2.3	Tinnitus Measurer	7
3	Anf	orderungsanalyse	9
	3.1	Funktionale Anforderungen	9
	3.2	Nicht-funktionale Anforderungen	11
	3.3	Konventionen und verwendete Hilfsmittel	11
	0.0	3.3.1 Verwendete Frameworks	11
		3.3.2 Konventionen	12
4	A	hitektur	13
4			
	4.1	Anwendungskontext	13
	4.2	Datenmodell	14
	4.3	Dialogstruktur	16
	4.4	Ablauf	17
5	Aus	sgewählte Implementierungsaspekte	19
	5.1	Gleichmäßige Verteilung von Notifications	19
		5.1.1 Hilfsmethoden	19
		5.1.2 Verteilungsalgorithmus	20
	5.2	Auswahl der Ergebnisse	23
6	Vor	stellung der iOS App	29
	6.1	Systembeschreibung	29
	6.2	Funktionsübersicht	29
		6.2.1 Registration und Login	29
		6.2.2 Fragebögen	31
		6.2.3 Menü	32
		6.2.4 Ergebnisdarstellung	33
		6.2.5 Einstellungen und Benachrichtigungen	34
		6.2.6 About	38
7	_	gleich der Anforderungen	41
	7.1	Abgleich der funktionalen Anforderungen	41
	7.2	Abgleich der nicht-funktionalen Anforderungen	42
8	Zus	ammenfassung und Ausblick	43

Abbildungsverzeichnis

2.1	Ausschnitt der Funktionen der Tinnitus HQ App	5
2.2	Funktionen der Tinnitracks App	6
2.3	Hauptfunktion der Tinnitus Measure App	7
4.1	Anwendungsfälle der App	13
4.2	Klassenstruktur der Fragebögen und Antworten	14
4.3	Dialogstruktur der App	16
4.4	Dialogstruktur der Einstellungen	17
4.5	Ablaufdiagramm	18
6.1	Erste View	30
6.2	Login View	30
6.3	Registrierung	31
6.4	View für ersten Login	31
6.5	Anfang des Fragebogens	32
6.6	Ende des Fragebogens	32
6.9	Sidebar Menü	32
6.7	Ausgewählte Slider	33
6.8	Beenden nach Speichern	33
6.10	Ohne Ergebnisse	34
6.11	Ergebnisse	34
	Ohne Benachrichtigungen	35
6.13	Auswahl der Klingeltöne	35
6.14	Standardeinstellungen	36
	Einstellen der Zeitspannen	36
	Benutzerderfinierte Benachrichtigungen	36
	Auswahl der Zeiten	37
6.17	Benutzerdefiniert	37
	Hinzufügen einer Benachrichtigung	38
6.20	Auswahl des Tages	38
	About	38
6.22	Senden von Feedback	36
6.23	Einer der About Texte	39

Einleitung

1.1 Motivation

Die Krankheit Tinnitus kann in den objektiven und den subjektiven Tinnitus unterteilt werden. Beim objektiven Tinnitus hört der Betroffene Töne oder Geräusche, die im Körper des Betroffenen entstehen. Diese Geräusche können auch von außen wahrgenommen und gemessen werden. Diese Form des Tinnitus tritt allerdings nur selten auf. Der subjektive Tinnitus ist eine Krankheit, bei dem ein davon Betroffener Geräusche oder Töne wahrnimmt, obwohl es keine messbare physikalische Ursache für die Geräusche oder Töne gibt. Selbst bei einem chronischen Tinnitus kann sich die Wahrnehmung des Tinnitus verändern. Für solche Schwankungen können viele Faktoren verantwortlich sein. Die Problematik bei der Krankheit des subjektiven Tinnitus ist, dass nur der Betroffene den Tinnitus wahrnehmen und diesen subjektiv beschreiben kann. Von außen lässt sich der Tinnitus nicht messen. Betroffene können meist gut die Schwankungen der letzten Tage rekonstruieren und beschreiben, aber nicht über längere Zeit. Das Track Your Tinnitus Projekt bietet eine Möglichkeit an, diese Schwankungen über längere Zeit zu verfolgen. Die erfassten Daten kann dann ein Betroffener mit seinem Arzt oder Therapeuten besprechen. Um Daten von den Betroffenen zu erfassen, sollen diese in selbst festgelegten Zeitabständen einen Fragebogen ausfüllen. Der Grundstein für den Webauftritt bzw. den mobilen Auftritt des Track Your Tinnitus Projektes wurde von Jochen Hermann 2014 gelegt [Her14]. Mit der aktuellen iOS App ist es den Benutzern möglich ihre Schwankungen der Tinnituswahrnehmung zu überwachen. Diese Arbeit soll diese Möglichkeit erneuern, Anpassungen an der bisherigen App durchführen und diese programmatisch auf den neuesten Stand der iOS App Programmierung bringen.

1.2 Aufbau der Arbeit

In diesem Abschnitt wird die Struktur der Arbeit aufgelistet. Das zweite Kapitel gibt einen kurzen Überblick über verwandte Arbeiten und stellt drei Apps vor, die zum Thema Tinnitus zu finden sind. In Kapitel 3 sind die funktionalen wie auch die nicht-funktionalen Anforderungen beschrieben und festgelegt. Die funktionalen Anforderungen beschreiben explizit welche Funktionen die App besitzen soll. Die nicht-funktionalen Anforderungen beschreiben unter anderem die Qualitätseigenschaften, die erfüllt werden sollen. Zusätzlich werden die Konventionen für die Programmierung und die verwendeten Frameworks aufgelistet. Danach schließt ein Kapitel über die Architektur an. In diesem werden alle Fälle gezeigt werden, die der Benutzer auslösen kann und das Datenmodell für die Fragen, Antworten und die Benutzer ausgeführt. Das Kapitel enthält darüber hinaus die Dialogstruktur, in der die Navigation durch die App dargestellt

wird. Anschließend folgt die Beschreibung eines typischen Ablaufs für die Benutzung der App. In Kapitel 5 werden zwei umfangreichere Algorithmen vorgestellt. Der erste Algorithmus regelt die Verteilung der Benachrichtigungszeiten. Der zweite Algorithmus ist für die Ergebnisdarstellung zuständig. Im sechsten Kapitel wird die iOS App vorgestellt. Dabei werden alle Funktionen anhand von Screenshots der App erklärt. Die Funktionsübersicht ist in Registration und Login, Fragebögen, Menü, Ergebnisdarstellung, Einstellungen und Benachrichtigungen, und den About Bereich eingeteilt. In Kapitel 7 werden die Anforderungen abgeglichen. Hier wird geprüft, ob die funktionalen und nicht-funktionalen Anforderungen erfüllt wurden. Im letzten Kapitel wird eine Zusammenfassung der Arbeit und ein Ausblick auf die weitere Entwicklung der App gegeben.

Verwandte Arbeiten

Dieses Kapitel zeigt drei weitere Arbeiten, die sich mit dem Thema Tinnitus beschäftigt haben. Es werden die drei ersten Apps vorgestellt, die unter dem Stichwort *tinnitus* zu finden sind, und die Grundfunktion der Apps beschrieben.

2.1 Tinnitus HQ

Dieser Abschnitt beleuchtet die Tinnitus HQ App, die bei der Suche nach Tinnitus unter den ersten Ergebnissen gelistet ist.

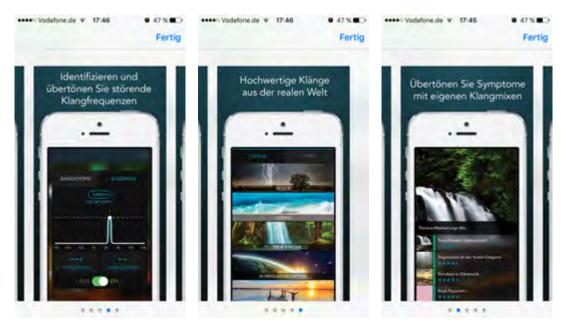


Abbildung 2.1: Ausschnitt der Funktionen der Tinnitus HQ App

Die Idee der Tinnitus HQ App [Mob16] ist es, den Tinnitus eines Betroffenen durch das Hören von Tönen zu mindern. Die gehörten Töne sollen so gefiltert werden, dass diese die Tinnitus-Frequenz des Betroffenen entfernen. Im linken Bild der Abb. 2.1 sieht man einen Bandpassund einen Bandstoppfilter. Diese Filter kann der Benutzer in der App anpassen, um seine Tinnitus-Frequenz zu ermitteln. Wenn der Benutzer seine Tinnitus-Frequenz damit ermittelt hat, kann die App angepasste Klangmasken erstellen. Solche Klangmasken sind im rechten Bild der Abb. 2.1 dargestellt. Der Benutzer kann sich selbst Klangmixe erstellen, vorgefertigte

Klangmixe anhören, oder auf Klänge aus der realen Welt, wie Regen oder Wasserrauschen des Ozeans, zugreifen. Um den vollen Umfang der Bandpass- und Bandstoppfilter und alle Klänge nutzen zu können, muss der Benutzer die Free Tinnitus HQ App aber kostenpflichtig auf die Pro Version upgraden.

2.2 Tinnitracks

In diesem Abschnitt wird die Tinnitracks App vorgestellt, die im Apple App Store ebenfalls unter den ersten Ergebnissen gezeigt wird.

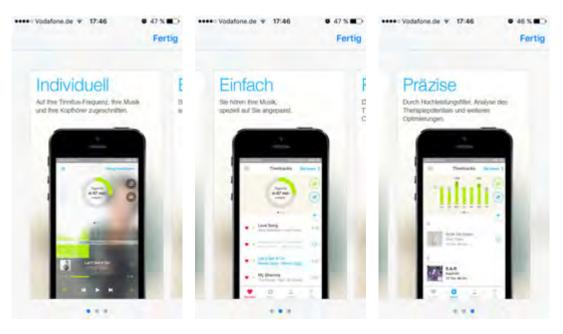


Abbildung 2.2: Funktionen der Tinnitracks App

Ähnlich der Tinnitus HQ App versucht die Tinnitracks App [Gmb16] auch den Tinnitus eines betroffenen Benutzers durch gefilterte Töne zu lindern und therapieren. Der Unterschied dabei ist, dass der Benutzer bei Tinnitracks eigene Lieder aus seiner Musikbibliothek in die App einspeist und jedes Lied analysiert und gefiltert wird. Die Analyse prüft, ob die eingespeiste Musik sich für die Tinnitus-Frequenz eignet. Allerdings wird kein Werkzeug zur Ermittlung der Tinnitus-Frequenz bereitgestellt, sondern darauf verwiesen, diese durch einen Arzt ermitteln zu lassen. Falls sich die Musik eignet, kann sie nach der Tinnitus-Frequenz des Benutzers gefiltert werden. Empfohlen wird 90 Minuten gefilterte Musik pro Tag. Zusätzlich bietet die App Statistiken an, die der Benutzer mit dem Arzt besprechen kann. Um den vollen Funktionsumfang nutzen zu können, muss der Benutzer aber kostenpflichtig auf die Vollversion upgraden. Funktionen und Design der App sind in Abb. 2.2 zu sehen. Im linken Bild der Abb. 2.2 kann man sehen, dass ein Countdown läuft, der die täglichen 90 Minuten der Musik Therapie misst. Im rechten Bild oben ist die Statistikfunktion angedeutet.

2.3 Tinnitus Measurer

Dieser Abschnitt beschreibt die Tinnitus Measurer App, welche insgesamt die mit dem kleinsten Funktionsumfang der in diesem Kapitel vorgestellten Apps ist. Bei der Stichwortsuche befindet sich die App auf dem dritten Platz.

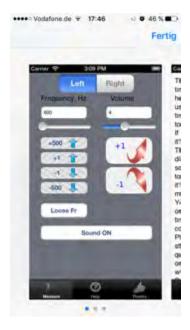


Abbildung 2.3: Hauptfunktion der Tinnitus Measure App

Die Tinnitus Measure App [Kle12] beschränkt sich darauf, die Tinnitus-Frequenz des Benutzers zu ermitteln, indem der Betroffene seinen gehörten Tinnitus Ton mit einem Testsignal vergleicht. Wie in Abb. 2.3 zu sehen, kann jeweils für das linke und für das rechte Ohr die Frequenz durch ein Testsignal ermittelt werden. Es ist auch möglich für beide Ohren gleichzeitig ein Testsignal abspielen zu lassen. Mit verschiedenen Reglern kann die Testfrequenz verändert werden. Auch die Laustärke kann eingestellt werden. In einem Hilfetext wird darauf hingewiesen, dass das Messen der Tinnitus-Frequenz mit dieser App nicht funktioniert, wenn das vom Betroffenen gehörte Geräusch, das von seinem Tinnitus ausgelöst wird, nicht als einzelner Ton auftritt. Diese App ist komplett kostenlos.

Anforderungsanalyse

Dieses Kapitel legt die Anforderungen dar, die das implementierte System später erfüllen soll, und welche Funktionen sich in der App wiederfinden sollen. Auch werden designtechnische Aspekte, die sich im System wiederfinden sollen, mit einbezogen.

3.1 Funktionale Anforderungen

In folgender Tabelle werden die Funktionen aufgestellt, die in der Track Your Tinnitus App vorhanden sein sollen. Der Benutzer soll alle in der Tabelle aufgezeigten Funktionen nutzen können.

Nr.	Beschreibung	Problemstellung
1	Registration des Benutzers	Der Benutzer soll sich in der App und auf
		der Website registrieren können.
2	Fragebogen zur Überwachung der Tinni-	Die App soll einen Fragebogen aufweisen,
	tuswahrnehmung	mit dem der Benutzer in bestimmten Ab-
		ständen seine Schwankungen der Tinnitus-
		wahrnehmung überwachen kann.
3	Einstellungen der Benachrichtigungszeiten	Der Benutzer soll die Benachrichtigungs-
		zeiten in der App setzen, die Benachrich-
		tigungszeiten aber auch wieder entfernen
		können.
4	Einstellungen des Klingeltons einer Be-	Um es zu vermeiden, dass der voreinge-
	nachrichtigung	stellte Klingelton vom Tinnitus überdeckt
		wird, soll der Benutzer die Möglichkeit ha-
		ben den Klingelton der Benachrichtigung
		zu ändern.
5	Anzeige der Ergebnisse in der App	Um die Entwicklung über die Zeit in der
		App darstellen zu können, sollen die Er-
		gebnisse aus dem Fragebogen zur Über-
		wachung der Schwankungen der Tinnitus-
		wahrnehmung visuell in einem Diagramm
		gezeigt werden.

Nr.	Beschreibung	Problemstellung
6	Messung des Geräuschpegels	Um messen zu können, ob Umgebungsgeräusche den Tinnitus überdecken oder beeinflussen, soll die App während dem Ausfüllen des Fragebogens zur Überwachung der Schwankungen der Tinnituswahrnehmung den Hintergrundgeräuschepegel messen.
7	App auch ohne Internetverbindung nutzbar	Eine funktionierende Internetverbindung auf dem Smartphone sollte keine Voraus- setzung für das Benutzen der App sein, da ein Benutzer evtl. nur schlechten oder gar keinen Empfang haben kann.
8	Ausfüllen der statistischen Fragebögen auf der Website und der App	Da das Ausfüllen der statistischen Frage- bögen Voraussetzung für die Benutzung der App ist, sollte dies auf der Webseite und in der App möglich sein
9	Synchronisierung der Antworten auf die statistischen Fragebögen	Es sollte möglich sein, auch während des Ausfüllens eines Fragebogens auf der Web- seite zur App zu wechseln, um dort die Be- arbeitung des Fragebogens zu beenden
10	Synchronisierung der Ergebnisse	Zur Visualisierung der Ergebnisse aus dem Fragebogen zur Überwachung der Tinni- tuswahrnehmung und für Forschungszwe- cke, sollten diese Ergebnisse aus der App an den Server übertragen werden
11	Fragebögen vom Server bereitgestellt	Um auch Änderungen an den Fragebögen zu Überwachung der Tinnituswahrnehmung vornehmen zu können, werden die Fragebögen auf dem Server gespeichert und, wenn eine neuere Version verfügbar ist, geupdatet.
12	Feedback geben	Um die App verbessern zu können, soll der Benutzer die Möglichkeit haben an das Projektteam eine Rückmeldung zu schreiben, Fragen zu stellen oder Anregungen zu geben.
13	Benutzerdefinierte- und Standardbenach- richtigungen	Der Benutzer soll die Möglichkeit haben, zwischen zwei verschiedenen Benachrichtigungsarten zu wählen, oder diese zu kombinieren. Er soll eine bestimmte Anzahl Benachrichtigungen in einem selbst gewählten Zeitraum setzen können oder explizit jeden Benachrichtigungszeitpunkt selbst setzen.

3.2 Nicht-funktionale Anforderungen

Die folgende Tabelle zeigt die nicht-funktionalen Anforderungen an die App. Die nicht-funktionalen Anforderungen beziehen sich auf die Performance der App.

Nr.	Beschreibung	Problemstellung
1	Ausfüllen des Fragebogens zur Überwa-	Das Ausfüllen des Fragebogens zur Über-
	chung der Schwankungen der Tinnitus-	wachung der Schwankungen der Tinnitus-
	wahrnehmung sollte in weniger als einer	wahrnehmung wird vom Benutzer mehr-
	Minute möglich sein	mals am Tag ausgefüllt. Daher sollte die-
		ser Vorgang möglichst schnell zu erledigen
		sein.
2	Benutzerfreundliche Bedienung der App	Selbst beim ersten Benutzen der App soll
		der Benutzer durch Hilfen, oder einfaches
		Design der App dazu in der Lage sein, oh-
		ne Probleme durch die App zu navigieren.
3	Zuverlässigkeit	Die App soll in bei keiner Aktion des Be-
		nutzers abstürzen oder hängen bleiben.
4	Effizienz und Erweiterbarkeit	Zwischen Aktionen des Benutzers dürfen
		kein unverhältnismäßigen Ladezeiten ent-
		stehen. Die App soll auch für andere Pro-
		grammierer leicht erweiterbar sein, ohne
		das Grundkonzept der App ändern zu
		müssen.

3.3 Konventionen und verwendete Hilfsmittel

In diesem Abschnitt werden die verwendeten Frameworks vorgestellt, wie auch die Konventionen zur Programmierung festgelegt.

3.3.1 Verwendete Frameworks

Frameworks können viel Programmieraufwand sparen. In den folgenden Stichpunkten werden die verwendeten Frameworks vorgestellt.

- SWRevealViewController von John LIuch (https://github.com/John-Lluch/ SWRevealViewController). Dieses Framework ermöglicht die Nutzung des Sidebar Menüs in der App.
- ScrollableGraphView von philackm (https://github.com/philackm/Scrollable-GraphView), welches das Design der Ergebnisdiagramme bestimmt.
- ReachabilitySwift von Ashley Mills (https://github.com/ashleymills/Reachability.swift). Durch das Reachability Framework kann einfach festgestellt werden, ob eine Internetverbindung vorhanden ist und auch welche Internetverbindung vorhanden ist.

- SRKUtility von Sagar R. Kothari (https://github.com/sag333ar/SRKUtility). Dieses Framework implementiert die Pop-up Fenster, mit denen der Benutzer die Benachrichtigungszeitpunkte festlegen kann.
- AVFoundation (https://developer.apple.com/reference/avfoundation). Das AVFoundation Framework macht die Auswahl und das Abspielen von Klingeltönen in der App möglich.
- UserNotifications (https://developer.apple.com/reference/usernotifications). Mithilfe dieses Frameworks lassen sich Benachrichtigungen in der App erstellen und verwalten. Sowohl Benachrichtigungen, die sich wiederholen, als auch einmalige Benachrichtigungen können implementiert werden.

3.3.2 Konventionen

Damit der Programmcode einheitlich und übersichtlich bleibt, werden Konventionen benötigt. Dieser Abschnitt zählt die festgelegten Konventionen auf.

Die standardmäßigen Swift Programmierkonventionen sollen eingehalten werden. Für Variablen sollen die Namen in der englischen Sprache sein. Jede Variable soll außerdem einen aussagekräftigen Namen besitzen, sodass auch jemand, der nicht an diesem Projekt mitentwickelt, durch den Variablennamen erahnen kann, welchen Zweck sie erfüllen könnte. Für die Übersichtlichkeit des Programmcodes muss jede Funktion, die nicht selbsterklärend ist, mit einem Kommentar beschrieben werden. In jeder Funktionsbeschreibung sollen die Rückgabewerte aufgezeigt werden. Der Aufbau der App-Struktur soll modular sein, sodass die App leicht erweiterbar ist.

Architektur

In diesem Kapitel wird die Architektur der App genauer gezeigt. Dabei zeigt dieses Kapitel alle Anwendungsfälle auf, die mit der App möglich sind, und gibt einen Überblick über das Datenmodell. Danach wird die Dialogstruktur aufgezeigt und ein typischer Ablauf im Umgang mit der App dargestellt. Die letzten zwei Aspekte zeigen die Navigation in der App auf.

4.1 Anwendungskontext

In diesem Abschnitt werden alle Anwendungsfälle gezeigt, die der Benutzer auslösen kann.

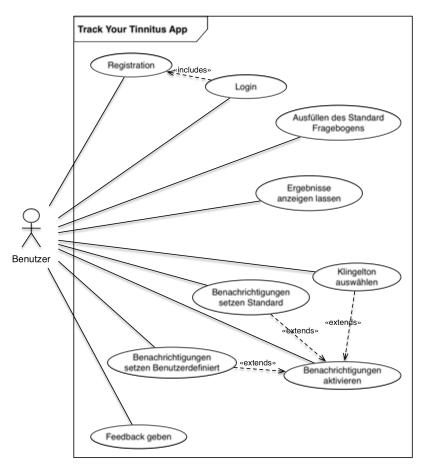


Abbildung 4.1: Anwendungsfälle der App

Abb. 4.1 zeigt alle Fälle auf, die der Benutzer in der App auslösen kann. Für die Benutzung der App muss sich der Benutzer zunächst in der App registrieren. Falls der Benutzer bereits einen Account besitzt, muss er sich nicht erst registrieren, sondern kann sich direkt einloggen. Der Hauptteil der App besteht darin, den Standardfragebogen immer wieder auszufüllen, um die Schwankungen der Tinnituswahrnehmung aufzeichnen zu können. Damit geht auch die Ergebnisanzeige einher. Die Ergebnisse der ausgefüllten Fragebögen werden hier angezeigt.

Damit der Benutzer zu bestimmten Zeitpunkten daran erinnert wird, einen Fragebogen auszufüllen, müssen die Benachrichtigungen aktiviert sein. Diese kann er als Standardeinstellungen ansetzen, wobei er nach einer gewählten Anzahl in einem bestimmten Zeitraum benachrichtigt wird, oder als benutzerdefinierte Einstellungen, wobei der Benutzer jeden einzelnen Benachrichtigungszeitpunkt explizit festlegt. Unter einem weiteren Menüpunkt ist es dem Benutzer im About Bereich unter dem Punkt Kontakt möglich, ein Feedback zu verfassen.

4.2 Datenmodell

Dieser Abschnitt zeigt das Datenmodell der Fragebögen, Fragen, Antworten, Ergebnisse und der Benutzer.

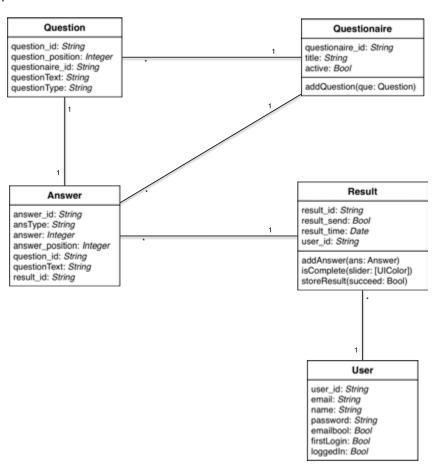


Abbildung 4.2: Klassenstruktur der Fragebögen und Antworten

In der vorhergegangenen Abb. 4.2 ist die Datenstruktur der Fragebögen, Fragen und Antworten dargestellt. Die Referenz auf den Benutzer wird zusätzlich gezeigt. Ein Fragebogen enthält eine questionaire id einen title und einen Bool (Wahrheitswert), der bestimmt, ob der Fragebogen aktiv ist. Der Fragebogen, bei dem active wahr ist, wird in der App zum ausfüllen angezeigt. Alle anderen Fragebögen, die sich in der Datenbank befinden, werden nicht geladen. Die Questionaire Klasse modelliert einen Fragebogen. Sie hat zusätzlich zu den Konstruktoren nur die Methode addQuestion, mit der dem Fragebogen eine Frage hinzugefügt werden kann. Eine Frage wird durch die questionaire id einem Fragebogen zugeordnet. Durch die question position wird der Frage auch die Position innerhalb des Fragebogens zugewiesen. Die Question Klasse beinhaltet die Fragen eines Fragebogens. Zusätzlich zur Zuordnung zu einem Fragebogen beinhaltet eine Frage noch die question id, den question Text, also die Fragestellung an sich, und den question Type. Der question Type gibt an, ob die Frage durch eine Ja/Nein Antwortmöglichkeit, einem Slider oder einem Wert zwischen 0 und 7 beantwortet werden kann. Ein Fragebogen kann mehrere Fragen enthalten. Jeder Frage ist eine Antwort zugeordnet. Eine Antwort, die in der Answer Klasse dargestellt ist, enthält ähnliche Variablen, wie eine Frage. So enthält eine Antwort eine question id, den der Frage zugeordneten question Text, einen ans Type, der die gleichen Möglichkeiten wie der question Type bietet, und eine answer position, die ihre Position in einem Ergebnis angiebt. Hinzu kommen die answer id, die result id, welche die Antwort einem Ergebnis zuordnet und die Antwort selbst als Integer. Bei einer Ja/Nein Frage steht eine null für ein Ja und eine eins für ein Nein. Die Result Klasse modelliert ein Ergebnis. Ein Ergebnis kann mehrere Antworten haben. Im Falle dieser App sind einem Fragebogen, sowie einem Ergebnis immer 8 Fragen bzw. Antworten zugeordnet. Die Result Klasse hat eine result id, und eine result time. Die result time gibt den Zeitpunkt an, an dem ein Fragebogen ausgefüllt und gespeichert wurde. Beim Speichern wird die aktuelle Zeit als Erfassungszeit genommen und für das Ergebnis eingetragen. Ein Ergebnis ist einem Benutzer immer durch die user id zugeordnet. Als Letztes gibt der result send Bool an, ob das Ergebnis schon an die externe Datenbank gesendet wurde. Die Ergebnisklasse besitzt drei Methoden. Die erste fügt, ähnlich der Methode der Fragebogenklasse, dem Ergebnis eine Antwort hinzu. Die is Complete Methode prüft, ob alle Fragen beantwortet waren. Die Antworten haben bestimmte default Werte, wenn Sie noch nicht beantwortet wurden. Zudem haben die Knöpfe für die Slider die Farbe clear Color, weswegen ein UIColor Array mit an die Methode übergeben wird. Mit der dritten Methode werden alle Antworten eines Ergebnisses in der lokalen Datenbank gespeichert. Der succeed Bool, der an die Methode übergeben wird, ist der result send Bool.

Jedes gespeicherte Ergebnis wird einem Benutzer zugeordnet. Dieser Benutzer wird in der User Klasse modelliert, erhält eine user_id und hat eine email mit zusätzlichem emailbool, durch den festgestellt wird, ob die E-Mail Adresse bereits bestätigt wurde. Weitere Attribute eines Benutzers sind der name und das password. Mit der Kombination von E-Mail oder Name und dem Passwort kann sich der Benutzer einloggen. Beim ersten Login wird das firstLogin Attribut auf false gesetzt, und das loggedIn Attribut wird auf true gesetzt. Wenn das loggedIn Attribut bei einem Benutzer in der lokalen Datenbank true ist, wird dieser nach dem erstmaligen Login bei jedem weiteren Start der App automatisch eingeloggt.

4.3 Dialogstruktur

Die Dialogstruktur zeigt die Navigation in der App, und wie der Benutzer von einem Dialog zum nächsten wechseln, und so durch die komplette App navigieren kann.

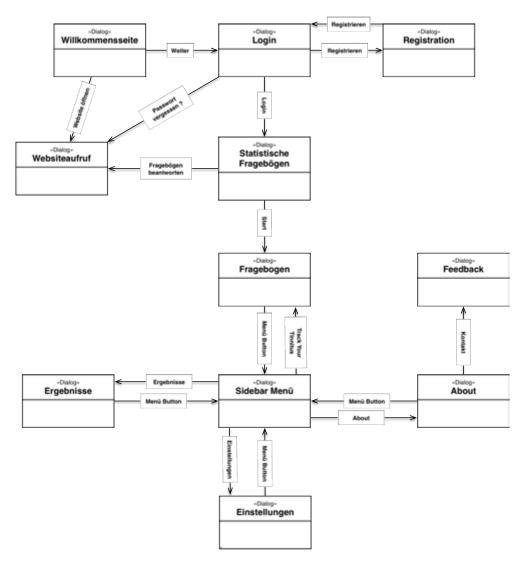


Abbildung 4.3: Dialogstruktur der App

Abb. 4.3 zeigt die Dialogstruktur der App. Ausgenommen sind hierbei die Einstellungen. Die Dialogstruktur der Einstellungen wird extra in Abb. 4.4 gezeigt. Eine Dialogstruktur zeigt alle Dialoge, welche als Rechteck dargestellt sind, des Systems und die Möglichkeit zwischen den einzelnen Dialogen zu navigieren. Die Übergangsmöglichkeiten sind mit Pfeilen markiert. Die Beschriftung der Pfeile betitelt meist die Namen von Buttons. Durch Navigation Bars in der App ist es dem Benutzer möglich, in fast jedem Fall wieder zum vorhergegangenen Dialog zurück zu navigieren. Dies ist nach erfolgreichem Login und nachdem der Start Button gedrückt wurde allerdings nicht mehr möglich. Die Dialogstruktur oberhalb des Fragebogen Dialogs kann der Benutzer nur vor dem ersten erfolgreichen Login durchlaufen.

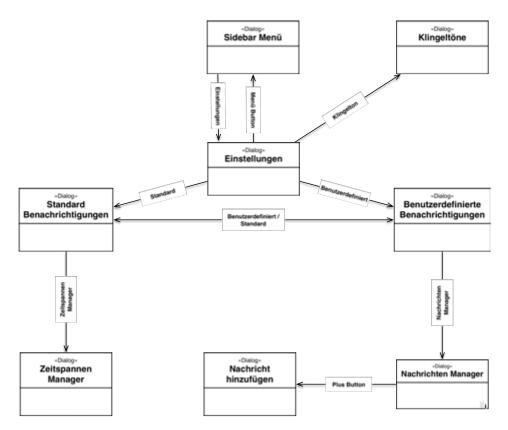


Abbildung 4.4: Dialogstruktur der Einstellungen

Die Dialogstruktur in Abb. 4.4 stellt die Einstellungen dar. Die Verbindung zum Menü ist zusätzlich noch aufgezeigt. Das Besondere an dieser Dialogstruktur ist, dass der Benutzer zwischen den benutzerdefinierten und den Standardbenachrichtigungseinstellungen direkt hin und her wechseln kann. In dieser Dialogstruktur kann der Benutzer von jedem Dialog zum vorherigen Dialog durch eine Navigation Bar zurück gelangen. Eine Ausnahme ist das Menü, da der Benutzer dorthin nur durch den Menü Button gelangt, nur durch einen Klick auf den Menüpunkt zurück oder zu einem anderen Menüpunkt navigieren kann.

4.4 Ablauf

Die Track Your Tinnitus App kann im Apple App Store heruntergeladen werden. Abb. 4.5 zeigt einen typischen Ablauf für die Benutzung der App. Falls der Benutzer bereits einen Account auf der Website erstellt hat, kann er sich direkt einloggen. Falls nicht bietet die App selbst eine Registration an. Nach dem Login hat der Benutzer die Wahl, ob er direkt einen Standardfragebogen zur Überwachung der Schwankungen der Tinnituswahrnehmung ausfüllen, oder, was von der App auch empfohlen wird, erst die statistischen Fragebogen auf der Website beantworten will. Das Beantworten der statistischen Fragebögen dauert nur etwa fünf Minuten. Bei erneutem Öffnen der App gelangt der Benutzer direkt zu dem Standardfragebogen. Füllt er diesen aus, wird die App beendet. Öffnet der Benutzer aber das Hauptmenü, kann er entweder die Ergebnisse ansehen, im About Bereich unter dem Menüpunkt Kontakt Feedback geben oder die Benachrichtigungszeiten in den Einstellungen modifizieren. Von diesen drei Punkten aus ist die

Möglichkeit gegeben, wieder zum Hauptmenü zu navigieren und von dort aus eine der anderen Aktionen auszuführen. Von allen Aktionen aus, die vom Hauptmenü erreichbar sind, kann die App beendet werden. Beim Speichern eines Fragebogens wird allerdings die App automatisch beendet.

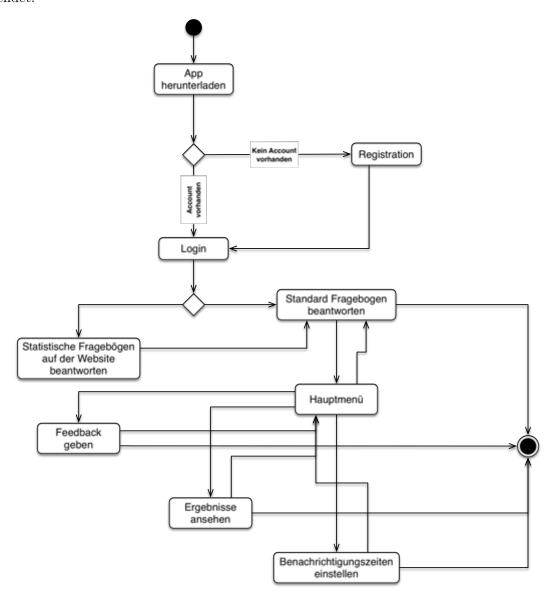


Abbildung 4.5: Ablaufdiagramm

Ausgewählte Implementierungsaspekte

Dieses Kapitel stellt zwei Algorithmen vor, die für die App wichtig sind. Der erste Algorithmus verteilt in einem gewählten Zeitintervall gleichmäßig Benachrichtigungen. Der zweite Algorithmus berechnet die Ergebnisse für die Ergebnisdarstellung, und wählt die anzuzeigenden Ergebnisse aus.

5.1 Gleichmäßige Verteilung von Notifications

In diesem Abschnitt wird der Algorithmus zur gleichmäßigen Verteilung von Benachrichtigungen vorgestellt. Dabei werden erst die Hilfsmethoden und dann der Algorithmus selbst erklärt.

5.1.1 Hilfsmethoden

Die folgenden Hilfsmethoden werden mehrfach in diesem Algorithmus und in anderen Programmteilen der App eingesetzt. Um eine erhöhte Modularität zu erhalten wurden sie von dem Hauptalgorithmus getrennt.

```
func getSplits(date: String) -> (Int,Int) {
    let split = date.components(separatedBy: ":")

let hours = Int(split[0])
    let minutes = Int(split[1])

return (hours!, minutes!)

}
```

Die getSplits Methode spaltet einen Uhrzeitstring der Form: "08:00" in Stunden und Minuten. Als Rückgabewert gibt die Funktion ein Tupel aus zwei Integer zurück. Der Uhrzeitstring wird am Doppelpunkt geteilt. Der erste Teil enthält die Stundenanzahl, der zweite Teil die Minutenanzahl. Die Teilstrings werden danach zu Integern gecastet und dann als Tupel zurückgegeben.

```
func getSeconds(date: String) -> Int {
    let split = date.components(separatedBy: ":")

let hours = Int(split[0])
    let minutes = Int(split[1])
```

Bei der getSeconds Methode läuft der Funktionsfluss ähnlich ab. Im ersten Teil wird ein Uhrzeitstring wie in der getSplits Methode in zwei Integer umgewandelt. Zusätzlich wird in Zeile 7 die Stundenanzahl und die Minutenanzahl in Sekunden umgerechnet, und beide Werte addiert. Das Ergebnis davon wird zurückgegeben.

```
func generateRandomNumber(min: Int, max: Int) -> Int {
    let randomNum = Int(arc4random_uniform(UInt32(max) - UInt32(min)) + UInt32(min))
    return randomNum
}
```

Die generateRandomNumber Funktion dient dazu, um eine Zufallszahl zwischen einem minimal und maximal Wert zu generieren. Die Zufallszahl wird zurückgegeben.

5.1.2 Verteilungsalgorithmus

In diesem Abschnitt wird die gleichmäßige Verteilung für Benachrichtigungen mithilfe des folgenden Algorithmus näher erklärt. Der Algorithmus ist deswegen so wichtig, weil er die Benachrichtigungen gleichmäßig und mit mindestens 15 Minuten Abstand verteilt.

```
func computeNotifications(startString: String, start: Int, end: Int,
         numberOfTimes: Int, day: Int) -> Bool {
2
            //Computes the Notification Times as Int
3
            let diff = end - start
4
            let canDo = diff / 900
5
            var intervalArray = [Int]()
6
            if diff > 0 {
                if numberOfTimes < canDo {</pre>
                     let interval = diff / numberOfTimes
9
                     var state = 0
10
                     for _ in 0..<numberOfTimes {</pre>
11
                         intervalArray.append(0)
                     }
14
                     for i in 0..<numberOfTimes {</pre>
15
                         let stateEnd = state + interval
16
17
                         var randomNumber = generateRandomNumber(min: state, max: stateEnd)
                         //Checking for the 15 Minutes Difference
19
                         if i > 0 {
20
                             var notificationDifference = randomNumber - intervalArray[i-1]
21
22
                             while notificationDifference < 900 {
```

```
randomNumber = generateRandomNumber(min: state,
                                         max: stateEnd)
25
                                 notificationDifference = randomNumber - intervalArray[i-1]
26
                            }
27
                            intervalArray[i] = randomNumber
28
29
                        } else {
30
                             intervalArray[i] = randomNumber
32
                        state = state + interval
33
                    }
34
35
                    //Converts the computed NotificationTimes to a Time
36
                    for r in intervalArray {
                        let plusMinutes = r / 60
39
                        let plusHours = plusMinutes / 60
40
                        let plusMinutesFinal = plusMinutes % 60
41
                        let splits = getSplits(date: startString)
                        var computedHours = (splits.0 + plusHours) % 24
                        var computedMinutes = splits.1 + plusMinutesFinal
45
46
                        if computedMinutes / 60 > 0 {
                            computedHours = (computedHours + 1) % 24
                            computedMinutes = computedMinutes % 60
                        } else {
50
                            // print("kein Überlauf")
51
                        }
52
53
                        print("Tag: \(day) Zeit \(computedHours):\(computedMinutes)")
                        //Sets the Notification
57
                        (UIApplication.shared.delegate as!
58
                        AppDelegate).setWeeklyNotification(weekDay: day, hour:
59
                        computedHours, min: computedMinutes, title: "standard")
                    }
62
63
                    return true
64
                } else {
65
                    let alertController = UIAlertController(title: "", message:
                        NSLocalizedString("TOO_MANY", comment: "Please chose a
67
                        fewer amount of Notifications in the Settings"), preferredStyle:
68
                        UIAlertControllerStyle.alert)
69
```

```
70
                    alertController.addAction(UIAlertAction(title: "Ok", style:
                    UIAlertActionStyle.default,handler: nil))
72
                    self.present(alertController, animated: true, completion: nil)
73
                    removeStandardNotifications()
74
75
                    return false
76
                }
77
            } else {
                let alertController = UIAlertController(title: "", message:
79
                    NSLocalizedString("NO_DIFF", comment: "Starttime and endtime
80
                    have to be different."), preferredStyle: UIAlertControllerStyle.alert)
81
82
                alertController.addAction(UIAlertAction(title: "Ok", style:
                UIAlertActionStyle.default,handler: nil))
84
                self.present(alertController, animated: true, completion: nil)
85
86
                return false
87
            }
88
   }
89
```

Die computeNotifications Funktion berechnet nach der Anzahl von numberOfTimes die Benachrichtigungszeitpunkte zwischen einer Startzeit und einer Endzeit. Die Benachrichtigungen sind gleichmäßig verteilt, müssen aber mindestens 15 Minuten, also 900 Sekunden auseinander liegen. Die Start und Endzeit wird durch die getSeconds Methode berechnet und die Sekundenzahl der Uhrzeit übergeben. Die Sekundenanzahl wurde gewählt, weil es so einfacher ist, den 15 Minuten Abstand zwischen den errechneten Benachrichtigungszeiten zu gewährleisten. Der Algorithmus besteht aus zwei Teilen. Der erste Teil berechnet die Benachrichtigungszeitpunkte in Sekunden. Der zweite Teil rechnet jeden Benachrichtigungszeitpunkt in eine Uhrzeit um und setzt mit dem Aufruf der setWeeklyNotification Funktion für jeden Zeitpunkt eine Benachrichtigung. Als Rückgabewert setzt die Funktion einen Bool, der true ist, wenn die Funktion ohne Fehler durchgeführt und mit möglich zu berechnenden Werten aufgerufen wurde. Andernfalls gibt der Algorithmus false zurück.

Bevor die Benachrichtigungszeiten berechnet werden, wird in Zeile 7 und 8 geprüft, ob die benötigte Berechnungsgrundlage gegeben ist. Die Differenz muss zum einen größer als 0 sein, und zum anderen muss das Intervall für die Anzahl der Benachrichtigungen groß genug sein, sodass der 15 Minuten Abstand gegeben ist. Falls beide Bedingungen erfüllt sind, wird das Intervall gleichmäßig aufgeteilt und ein Intervallarray der Länge von numberOfTimes erstellt, sowie die Intervallänge in die Variable interval gespeichert. In das Intervallarray werden später die Zeiten in Sekundenformat eingetragen. Von Zeile 15 bis 33 werden dann die Benachrichtigungszeitpunkte berechnet und in das Intervallarray eingetragen. In Zeile 10 wird dafür schon ein Statusinteger state instanziiert. Die Benachrichtigungsgzeiten werden in diesem Abschnitt unabhängig von der Startzeit berechnet. Das Intervall in dem sich die Zeiten befinden soll geht von 0 bis zur berechneten Differenz von Start- und Endzeit. Die for-Schleife in Zeile 15 berechnet so oft einen neuen Benachrichtigungszeitpunkt, der vom alten mindestens 900 Sekunden Abstand hat, außer der Start state beträgt 0, bis die Anzahl der Schleifendurchläufe den Wert von numberOfTimes erreicht hat. Diese Bedingung wird in Zeile 20 abgefragt. Der Benach-

richtigungszeitpunkt wird durch die generateRandomNumber Funktion, die state als Startwert und endState als Endwert bekommt. Die Variable endState setzt sich aus state und der addierten Intervallänge zusammen. Nachdem die Zufallszahl generiert wurde, wird noch mit der notificationDifference Variable überprüft, ob die aktuelle Zufallszahl 900 Sekunden Abstand zur vorherig generierten Zahl hat. Falls dies nicht der Fall ist, wird von Zeile 23 bis 26 so lange eine neue Zufallszahl generiert, bis dies die Abstandsbedingung erfüllt ist. Die generierte Zahl wird in das Intervallarray gespeichert und die state Variable um die Intervallänge erhöht.

Im zweiten Schritt, der sich von Zeile 37 bis 61 erstreckt, werden die Sekunden wieder in Minuten und Stunden umgerechnet und anschließend auf die Startzeit aufaddiert. Der generierte Wert r an der jeweiligen Stelle im Intervallarray wird in Minuten umgerechnet. Da Sekunden bei der Erstellung einer Benachrichtigung nicht notwendig sind, werden die Nachkommastellen bei der Umrechnung nicht beachtet. Die Minutenanzahl wird in Stunden umgerechnet und die zusätzliche Minutenanzahl durch Division mit Rest ermittelt. Somit wird dann die Startzeit, die bei Funktionsaufruf als String zusätzlich übergeben wird, durch die getSplits Methode in Stunden- und Minutenanzahl in Form von zwei Integern umgewandelt. Auf diese werden unsere errechneten Stunden bzw. Minuten aufaddiert. Somit erhalten wir die zwei Variablen computedHours und computedMinutes. Hier muss noch auf einen Überlauf getestet werden, sodass nicht mehr als 24 Stunden und 60 Minuten errechnet werden. Falls ein Überlauf da ist, wird auf die Stunden eins aufaddiert und davon die Division mit Rest durch 24 gemacht, und die computedMinutes werden auch mit Rest durch 60 dividiert. So erhalten wir eine valide Uhrzeit, die dann in Zeile 57 der setWeeklyNotification Methode übergeben wird. Mit zusätzlichem Tag erstellt diese Methode zur errechneten Uhrzeit eine Benachrichtigung.

In den folgenden else Zweigen werden dem Benutzer Warnungen angezeigt, falls die zur Berechnung benötigten Bedingungen nicht erfüllt waren. Zusätzlich werden noch alle bereits gesetzten Benachrichtigungen im oberen else Zweig mit der removeStandardNotifications Methode entfernt.

5.2 Auswahl der Ergebnisse

Der folgende Abschnitt zeigt, wie die vorhandenen Ergebnisse für die Ergebnisdarstellung aufbereitet und dargestellt werden. Die Wichtigkeit des Algorithmus liegt darin, dass er die Ergebnisanzeige bei steigender Anzahl der Ergebnisse für den Benutzer übersichtlicher gestaltet.

```
func computeResults(_ chart: [[Double]]) -> ([[Double]], [String], Int) {
2
            //Initial Variables which are needed
3
           let count = chart[0].count
           var xLabel = [String]()
           var computedData = [[Double]]()
           var length = 320
            //The 5 Cases of how many Results are shown
9
           switch count {
10
            case 1...20:
12
                //Sets the number of xLabels, the selected results and the length
13
```

```
for i in 1...20 {
14
                     xLabel.append("\(i)")
                 }
16
                 computedData = chart
17
                 break
18
19
            case 21...40:
20
                 length = count * 16
21
                 computedData = chart
23
                 //Sets the selected results
24
                 let min = 21
25
                 let div = count - min + 1
26
                 var deletecount = div / 2
                 //How many Results will be deleted
28
                let deletefactor = 4
29
30
                 var selectedResults = [Int]()
31
32
                 for i in 0 ..< computedData[0].count {</pre>
33
                     //print("\(i)\ mit\ Wert\ \(test[i])\ zu\ löschen\ \(aktuellzuloesch)")
                     if deletecount != 0 {
35
                          if i % deletefactor == 0 && i != 0{
36
                              selectedResults.append(i)
37
                              //print("aufgerufen für \ (i)")
38
                              deletecount -= 1
                         }
40
41
                     }
42
                 }
43
                 //Deleting the chosen Results
45
                 for k in 0 ..< computedData.count {</pre>
46
                     for j in 0 ..< selectedResults.count {</pre>
47
                          //print(selectedResults[selectedResults.count-1 - j])
48
                          computedData[k].remove(at: selectedResults.count-1 - j)
49
                     }
50
                 }
52
                 //Sets the number of xLabels and the length
53
                 for i in 1...computedData[0].count {
54
                     xLabel.append("\(i)")
55
                 }
                 length = computedData[0].count * 16
57
                 break
58
59
```

```
case 41...80:
60
                 computedData = chart
62
                 //Sets the selected results
63
                 let min = 21
64
                 let div = count - min + 1
65
                 var deletecount = div / 2
66
                 //How many Results will be deleted
                 let deletefactor = 3
69
                 var selectedResults = [Int]()
70
71
                 for i in 0 ..< computedData[0].count {</pre>
                      //print("\(i)\ mit\ Wert\ \(test[i])\ zu\ l\"{o}schen\ \(aktuellzuloesch)")
                      if deletecount != 0 {
74
                          if i % deletefactor == 0 && i != 0{
75
                              selectedResults.append(i)
76
                              //print("aufgerufen für \(i)")
                              deletecount -= 1
                          }
                     }
81
                 }
82
83
                 //Deleting the chosen Results
                 for k in 0 ..< computedData.count {</pre>
                     for j in 0 ..< selectedResults.count {</pre>
86
                          //print(selectedResults[selectedResults.count-1 - j])
87
                          computedData[k].remove(at: selectedResults.count-1 - j)
88
                     }
89
                 }
91
                 //Sets the number of xLabels and the length
                 for i in 1...computedData[0].count {
93
                     xLabel.append("\(i)")
94
                 }
95
                 length = computedData[0].count * 16
                 break
98
             case 81...160:
99
                 computedData = chart
100
101
                 //Sets the selected results
                 let min = 21
103
                 let div = count - min + 1
104
                 var deletecount = div / 2
105
```

```
//How many Results will be deleted
106
                  let deletefactor = 2
108
                 var selectedResults = [Int]()
109
110
                 for i in 0 ..< computedData[0].count {</pre>
111
                      //print("\(i)\ mit\ Wert\ \(test[i])\ zu\ löschen\ \(aktuellzuloesch)")
112
                      if deletecount != 0 {
113
                           if i % deletefactor == 0 && i != 0{
                               selectedResults.append(i)
115
                               //print("aufgerufen für \(i)")
116
                               deletecount -= 1
117
                          }
118
                      }
120
                  }
121
122
                  //Deleting the chosen Results
123
                  for k in 0 ..< computedData.count {</pre>
                      for j in 0 ..< selectedResults.count {</pre>
125
                           //print(selectedResults[selectedResults.count-1 - j])
126
                           computedData[k].remove(at: selectedResults.count-1 - j)
127
                      }
128
                  }
129
130
                  //Sets the number of xLabels and the length
                 for i in 1...computedData[0].count {
132
                      xLabel.append("\(i)")
133
134
                  length = computedData[0].count * 16
135
                 break
             default:
137
                  //Sets the number of xLabels, the selected results and the length
138
                  for i in 1...160 {
139
                      xLabel.append("\(i)")
140
                  }
141
                  length = 90 * 16
142
                  computedData = chart
143
144
                  //Filtering the last 160 Results
145
                  let deletes = count - 160
146
                  for 1 in 0 ..< computedData.count {</pre>
147
                      for _ in 0 ..< deletes {</pre>
                               computedData[1].removeFirst()
149
                      }
150
                  }
151
```

```
152
                   //Sets the selected results
153
                   let min = 21
154
                   let div = 160 - min + 1
155
                   var deletecount = div /
156
                   //How many Results will be deleted
157
                   let deletefactor = 2
158
159
                   var selectedResults = [Int]()
161
                   for i in 0 ..< computedData[0].count {</pre>
162
                        //print("\setminus(i) \ mit \ Wert \ \setminus(test[i]) \ zu \ l\"{o}schen \ \setminus(aktuellzuloesch)")
163
                        if deletecount != 0 {
164
                            if i % deletefactor == 0 && i != 0{
165
                                 selectedResults.append(i)
166
                                 //print("aufgerufen für \(i)")
167
                                 deletecount -= 1
168
                            }
169
170
                       }
171
                   }
172
173
                   //Deleting the chosen Results
174
                   for k in 0 ..< computedData.count {</pre>
175
                       for j in 0 ..< selectedResults.count {</pre>
176
                            //print(selectedResults[selectedResults.count-1 - j])
                            computedData[k].remove(at: selectedResults.count-1 - j)
178
                       }
179
                   }
180
                   break
181
              return (computedData, xLabel, length)
183
    }
184
```

Die computeResults Methode wählt aus allen vom Benutzer eingespeisten Werten eine berechnete Auswahl an Ergebnissen aus, sodass der Benutzer auch bei sehr vielen ausgefüllten Fragebögen noch einen übersichtlichen Ergebnisverlauf dargestellt bekommt. Als Eingabe bekommt die Funktion ein Zahlenarray der Ergebnisse. Die Ergebnisse werden als Double übergeben, weil das Diagrammframework, das zur Ergebnisdarstellung benutzt wird, Double benötigt. Dabei enthält das erste Array des chart Arrays die Ergebnisse auf die erste Frage. Die Ergebnisse werden vor Aufruf der computeResults Funktion aus der lokalen Datenbank gelesen und in ein Ergebnisarray der gewünschten Form gebracht. Als Rückgabewert hat die Funktion die Werte, die in Zeile 5,6,7 initialisiert werden. Die Rückgabewerte sind in einem Tripel gebündelt, und können später mit einem .0, .1, .2 Aufruf auf der Rückgabevariablen daraus extrahiert werden. Der erste Wert ist das Ergebnisarray, das aber nur die Ergebnisse enthält, die durch den Algorithmus ausgewählt wurden. Der zweite Wert enthält die Beschriftung der x-Achse für das Ergebnisdiagramm. Der letzte Wert enthält die Länge der x-Achse, die das Ergebnisdiagramm

haben soll. Der Algorithmus behandelt 5 Fälle, die durch ein Switch Statement umgesetzt sind. Die Fälle sind nach Anzahl der in der Eingabe enthaltenen Ergebnisse unterteilt. Je nach Fall werden mehr oder weniger Ergebnisse für die Anzeige ausgesiebt. Um über die Anzahl der Ergebnisse zu "switchen", bekommt die Variable *count* die Anzahl der enthaltenen Werte des *chart* Arrays.

Der erste Fall tritt ein, wenn der Benutzer erst zwischen 1 und 20 Fragebögen ausgefüllt hat. In diesem Fall werden alle Ergebnisse für die Ergebnisdarstellung genommen und in *computed-Chart* geschrieben. Das Array, das zurückgegeben wird, ist in jedem Fall *computedChart*. Auch die x-Achse bekommt 20 Stellen zugewiesen, und die länge des Diagramms beläuft sich auf die default Länge von 320.

Der zweite Fall deckt 21 bis 40 Ergebnisse ab. Die Vorgehensweise bei diesem, wie auch in Fall drei und vier ist gleich. Das einzige, was sich ändert, ist die deletefactor Variable. Als erstes wird die Differenz der aktuellen Ergebnisse zum Minumum des Falles berechnet. Die Anzahl der zu löschenden Ergebnisse ist die Hälfte der Differenz und wird in der Variable deletecount gespeichert. In Fall zwei liegt der deletefactor bei vier, das heißt es wird so lange jedes vierte Ergebnis gelöscht, bis die Anzahl zu löschender Ergebnisse erreicht ist. Somit folgt, dass innerhalb eines Falles mehr Ergebnisse gelöscht werden, wenn mehr vorhanden sind. Um die angesprochene Filterung oder Löschung der Ergebnisse durchzuführen, wird noch ein Integer Array selectedresults initialisiert und die vom Benutzer erhaltenen Ergebnisse in die computedData Variable gespeichert. Danach iteriert der Algorithmus über eine for-Schleife über das erste Array von computedData, um die zu löschenden Stellen in das selectedResults Array zu speichern. Dabei wird geprüft, ob deletecount bereits null ist. Wenn dem nicht so ist, wird geprüft, ob der Zähler der betrachteten Stelle des ersten computedData Arrays ohne Rest durch den deletefactor teilbar ist, und ob der Zähler nicht gleich null ist. Wenn beide Bedingungen erfüllt sind, wird der betreffende Zähler dem selectedresults Array hinzugefügt, und der deletecount um eins erniedrigt. In den folgenden beiden for-Schleifen werden für jedes Ergebnisarray von computedData diejenigen Ergebnisse gelöscht, deren Zähler sich in dem selectedResults Array befindet. Dabei werden die Ergebnisse mit dem höchsten Zähler zuerst gelöscht, um eine Index-out-of-Bounds Exception zu vermeiden. Für die Beschreibung der x-Achse wird nun die Länge des ersten Arrays von computedData hergenommen, welche in Zeile 55 bis 57 ermittelt wird. Das Löschen der ausgewählten Ergebnisse wird in Zeile 47 bis 50 durchgeführt. Durch Ausprobieren in den Views hat sich herausgestellt, dass pro Ergebnis eine Breite von 16 Punkten optimal ist, und die Ergebnisse dann nicht zu weit auseinander oder zu nahe beieinander sind. Deswegen wird in Zeile 58 die Anzahl der Ergebnisse mal 16 genommen, um die für das Diagramm benötigte Länge zu ermitteln.

Der dritte Fall erstreckt sich von 41 bis 80 Ergebnissen. Die Ausprogrammierung dieses Falles unterscheidet sich nur durch die Wahl des deletefactors. Dieser beläuft sich hier auf 3. In Fall vier, der die Ergebnisanzahl 81 bis 160 abdeckt, wurde der deletefactor mit zwei gewählt. Im letzten Fall nun, der alle Anzahlen an Ergebnissen über 160 abdeckt, wird der deletefactor von zwei zwar beibehalten, aber die Ergebnisanzahl auf 160 Ergebnisse gestutzt. In Zeile 147 wird die Differenz zu 160 Ergebnissen ermittelt. Im Folgenden Code von Zeile 148 bis 150 wird so oft das erste Element eines jeden Ergebnisarray aus computedData entfernt, bis nur noch die letzten 160 Ergebnisse übrig sind. Danach wird wie in den anderen Fällen mit der Ergebnisfilterung verfahren. Die Länge beträgt 90 mal 16 Punkte, da bei jeder Ergebnisanzahl nach der Filterung 90 Ergebnisse bleiben.

Vorstellung der iOS App

Dieses Kapitel beleuchtet die Track Your Tinnitus App näher und zeigt die Bedienung und den Inhalt der App. Mithilfe von Screenshots werden die verschiedenen Funktionen direkt abgebildet und erklärt.

6.1 Systembeschreibung

Die Track Your Tinnitus App soll es einem Benutzer erlauben, die Schwankungen seiner Tinnituswahrnehmung überwachen zu können. Dies geschieht in der App durch regelmäßiges Ausfüllen des Fragebogens zur Überwachung der Tinnituswahrnehmung. Mithilfe dieses Fragebogens ergibt sich mit der Zeit ein Ergebnisverlauf, der in der App visualisiert werden kann. Diesen Verlauf kann der Benutzer dazu verwenden, die Schwankungen seines Tinnitus über die Zeit zu verfolgen.

6.2 Funktionsübersicht

Dieser Abschnitt zeigt alle Funktionen der App anhand von Screenshots eines Smartphones. Die Funktionen sind in Login, Fragebögen, Menü, Einstellungen und About unterteilt.

6.2.1 Registration und Login

Beim ersten Start der App sieht man Abb. 6.1 als erstes Fenster angezeigt. Es wird in dem zentralen Text beschrieben, zu welchem Zweck die App entwickelt wurde, und mit dem Button links unten auch gleichzeitig auf den Webauftritt des Projektes hingewiesen. Bei Auswahl dieses Buttons wird die Website in Safari geöffnet. Mit dem Weiter Button gelangt man direkt zu Abb. 6.2. Wenn man sich bereits auf der Website einen Account erstellt hat, kann man sich direkt einloggen. Falls das Passwort vergessen wurde, kann dieses durch Drücken des Passwort vergessen? Buttons auf der Homepage zurücksetzt werden. Das Erstellen eines Accounts ist aber auch direkt in der App durch den Registrieren Button möglich. Dieser leitet direkt zu Abb. 6.3 weiter.



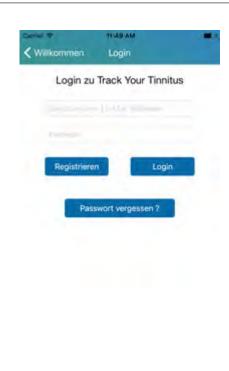


Abbildung 6.1: Erste View

Abbildung 6.2: Login View

Für die Registration müssen alle Felder des Formulars, und die Wiederholungsfelder zu ihren darüberliegenden Feldern identisch ausgefüllt werden. Zusätzlich muss ein Benutzername, sowie die E-Mail Adresse eingetragen werden. Diese dürfen noch nicht im System vorhanden sein. Nach Auswahl des **Registrieren** Buttons unten wird dem Benutzer durch einen Dialog mitgeteilt, ob die Registration erfolgreich war. Bei nicht bestehender Internetverbindung kann kein Account angelegt werden. Nachdem der Account vom Benutzer angelegt wurde, kann sich dieser über die Login View einloggen.

Es erscheint bei erfolgreichem Login Abb. 6.4 als Fenster. Hier kann der Benutzer entscheiden, ob er gleich mit dem **Start** Button fortfahren möchte, oder auf der Website die drei Statistischen Fragebögen beantworten, und danach fortfahren will.





Abbildung 6.3: Registrierung

Abbildung 6.4: View für ersten Login

6.2.2 Fragebögen

Der Benutzer loggt sich nur einmalig in die App ein. Bei jedem weiteren Start wird der Benutzer automatisch eingeloggt, und sieht als Erstes Abb. 6.5. Diese Ansicht ist das Herzstück der App. Am Ende des Fragebogens steht der **Speichern** Button, um die Ergebnisse des Fragebogens zu speichern. Ein Fragebogen enthält drei verschiedene Fragetypen. Der erste Typ ist die Ja/Nein Frage, die durch ein Segmented Control mit zwei Möglichkeiten realisiert wird. Der zweite, wie auch der dritte Fragetyp, ist eine Ermessensfrage. Hierbei kann der Benutzer die Antwort für das Gefragte zwischen einem Minimal- und Maximalwert einordnen. Die Antwortmöglichkeit wird beim zweiten Fragetyp durch einen Slider ohne initialen Wert realisiert. Es wird vom Benutzer ein Wert zwischen dem minimalen und dem maximalen Wert gewählt. Da der Slider keinen initialen Wert besitzt, wird der Benutzer somit nicht beeinflusst. Der dritte Fragetyp wird durch ein acht gliedriges Segmented Control dargestellt. Der Sinn ist wieder, einen Wert zwischen Minimum und Maximum festzulegen, hier aber nur eine begrenzte Anzahl an Werten (8 Werte) zuzulassen.

Ausgefüllt sieht ein Fragebogen wie in Abb. 6.7 aus. Wenn der Benutzer jede Frage beantwortet hat, kann der **Speichern** Button aus Abb. 6.6 betätigt werden, und der Benutzer wird zur Ansicht aus Abb. 6.8 weitergeleitet. In dieser Ansicht sieht der Benutzer einen drei Sekunden Countdown ablaufen. Nachdem der Countdown abgelaufen ist, wird die App geschlossen. Wenn der Benutzer die App wieder öffnet, sieht er wieder die Ansicht aus Abb. 6.5, also den Fragebogen.



Abbildung 6.5: Anfang des Fragebogens



Abbildung 6.6: Ende des Fragebogens

6.2.3 Menü



Abbildung 6.9: Sidebar Menü

Auf fast jeder Ansicht in der App ist links oben das Symbol mit den drei waagerechten Strichen zu finden. Durch diesen Button kann der Benutzer zum Menü der App navigieren. Dieses ist durch ein Sidebar Menü realisiert, wie es auch in der Android App zu finden ist. In Abb. 6.9 ist dieses Menü zu sehen. Durch Track Your Tinnitus kommt der Benutzer zu dem Fragebogen. Die zweite Säule der App ist die Ergebnisdarstellung, die auch durch das Menü zu erreichen ist. Zusätzlich kann der Benutzer noch zu den Einstellungen und dem About Bereich navigieren.

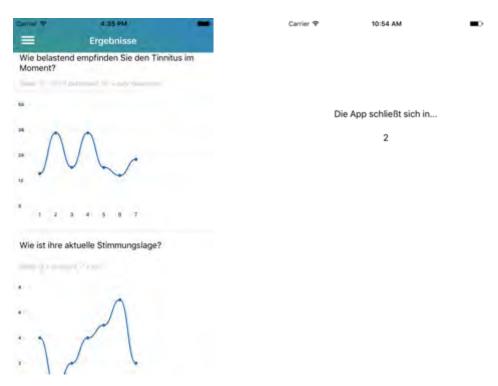


Abbildung 6.7: Ausgewählte Slider

Abbildung 6.8: Beenden nach Speichern

6.2.4 Ergebnisdarstellung

Bei den Ergebnissen werden die Antworten der vom Benutzer ausgefüllten Fragebögen angezeigt. In Abb. 6.10 hat der Benutzer bisher noch keinen Fragebogen ausgefüllt. Deswegen wird der Benutzer auch an den Stellen, wo die Ergebnisse angezeigt werden sollten, informiert, dass noch keine Ergebnisse vorhanden sind. Wenn der Benutzer aber schon einige Fragebögen ausgefüllt hat, sieht die Ergebnisdarstellung wie in Abb. 6.11 aus. Die Diagramme passen sich dynamisch der Anzahl der Ergebnisse an.



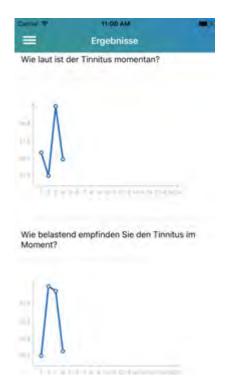


Abbildung 6.10: Ohne Ergebnisse

Abbildung 6.11: Ergebnisse

6.2.5 Einstellungen und Benachrichtigungen

In den Einstellungen kann der Benutzer den Klingelton für eine Benachrichtigung auswählen, die Geräuschmessung aktivieren, mit der während dem Ausfüllen des Fragebogens der Umgebungsgeräuschpegel gemessen wird, und die Benachrichtigungen verwalten. Wenn die Benachrichtigungen deaktiviert sind, hat der Benutzer die Ansicht, wie in Abb. 6.12. Die Abb. 6.13 zeigt die Klingeltöne, die der Benutzer auswählen kann. Bei Auswahl eines Klingeltons, wird dieser kurz vorgespielt, sodass der Benutzer auch den Ton mit der Beschreibung verbinden kann. Mit der Navigation Bar am oberen Bildschirmrand kann man zu den Einstellungen zurück navigieren.

Wenn der Benutzer die Benachrichtigungen aktiviert kann er nun zwischen zwei Benachrichtigungsarten wählen. Zum einen gibt es die Standardeinstellungen, und zum anderen die benutzerdefinierten Einstellungen.

Bei den Standardeinstellungen wählt der Benutzer die Anzahl der Benachrichtigungen, die er pro Tag bekommen will. In Abb. 6.14 zeigt, dass die Anzahl durch Klicken auf das + erhöht und durch das - verkleinert wird. Nachdem diese Einstellung festgelegt wurde, gelangt der Benutzer nun durch Auswählen des **Zeitspannen Manager** zu der Ansicht von Abb. 6.15. In dieser Ansicht kann die Zeitspanne für jeden Tag eintragen werden. Um alle Zeitspannen zu speichern, und die Benachrichtigungen zu setzen, muss der Benutzer unten rechts auf den **Speichern** Button drücken. Bei erfolgreichem Setzen der Benachrichtigungen, wird der Benutzer auf die Ansicht Einstellungen zurück geleitet. Wenn der Benutzer alle gesetzten Benachrichtigungen löschen will, kann er dies durch Auswählen des **Alle löschen** Buttons erreichen. Dabei werden

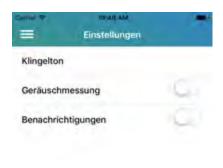




Abbildung 6.12: Ohne Benachrichtigungen Abbildung 6.13: Auswahl der Klingeltöne

auch alle Zeitspannen auf die Ausgangswerte mit 8:00 Uhr Startzeit und 22:00 Uhr Endzeit zurückgesetzt.

In Abb. 6.16 ist das Pop-up Fenster dargestellt, das erscheint, wenn der Benutzer ein Uhrzeitfeld auswählt. Bei Bestätigung der ausgewählten Uhrzeit wird diese Übernommen. Bei Abbruch wird das Uhrzeitfeld auf 8:00 Uhr bei einem Startfeld oder 22:00 Uhr bei einem Endfeld gesetzt. Ist die Endzeit zeitlich früher als die Startzeit, werden die Werte der beiden Felder getauscht, sodass die Endzeit dann die Startzeit ist und die Startzeit die Endzeit.

Bei den benutzerdefinierten Einstellungen kann der Benutzer über den **Nachrichten Manager** einzelne Benachrichtigungen hinzufügen. Dadurch kann der Benutzer bei jeder Benachrichtigung festlegen, wann diese zugestellt werden sollen. Bei den Standardeinstellungen werden die Benachrichtigungen gleichmäßig verteilt, wobei der Benutzer jedoch nur Einfluss auf die Zeitspanne, in der die Benachrichtigungen verteilt werden, hat.



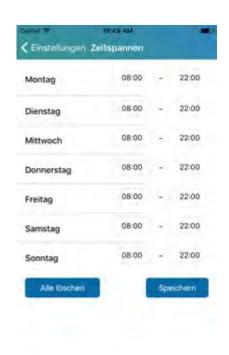


Abbildung 6.14: Standardeinstellungen

Abbildung 6.15: Einstellen der Zeitspannen



Abb. 6.18 ist die Ansicht, die der Benutzer bekommt, wenn er den Time Manager aufruft. Hier sind alle Benachrichtigungszeitpunkte aufgelistet, die der Benutzer hinzugefügt hat. Eine Benachrichtigung kann durch das weiße + rechts oben in der Navigation Bar hinzugefügt werden. Um eine Benachrichtigung zu löschen wählt man einen Benachrichtigungszeitpunkt aus und zieht mit dem Finger von rechts nach links über den Bildschirm. In Abb. 6.18 wurde diese Aktion bei der dritten Benachrichtigung gemacht. Mit anschließendem Auswählen des erscheinten Löschen Buttons wird die Benachrichtigung gelöscht.

Abbildung 6.18: Benutzerderfinierte Benachrichtigungen



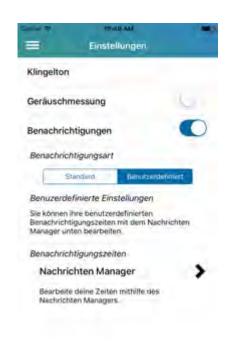


Abbildung 6.16: Auswahl der Zeiten

Abbildung 6.17: Benutzerdefiniert

Nachdem der Benutzer auf das + in Abb. 6.18 gedrückt hat, wird er zur Ansicht aus Abb. 6.19 weitergeleitet. In dieser Ansicht ist es dem Benutzer möglich eine Benachrichtigung zu erstellen. Bei Auswählen des Tagesfeldes oder des Zeitfeldes erscheint wieder ein Pop-up Fenster, um Benachrichtigungstag oder Benachrichtigungszeit festzulegen. In Abb. 6.20 sieht man das Pop-up Fenster für den Tag. Das Auswahlfenster für die Uhrzeit ist gleich, wie in Abb. 6.16 gezeigt. Der Benutzer kann die Benachrichtigung speichern und wird zur Ansicht aus Abb. 6.18 zurück geleitet. Falls aber der Benachrichtigungszeitpunkt schon besteht, wird dies dem Benutzer mitgeteilt, und er muss den Zeitpunkt neu setzen.



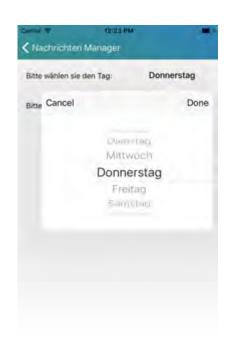


Abbildung 6.19: Hinzufügen einer Benachrichtigung

Abbildung 6.20: Auswahl des Tages

6.2.6 About



Abbildung 6.21: About

Über das Menü gelangt man in den About Bereich. Der About Bereich ist hauptsächlich dafür da, um mehr über das Projekt und die rechtlichen Grundlagen zu erfahren und die Möglichkeit zu bieten, dem Projektteam ein Feedback zu geben. Der Benutzer kann unter dem Menüpunkt Kontakt ein Feedback geben, und wird danach zu Abb. 6.22 weitergeleitet. Informationen gibt es über das Team, über das Projekt, über die Datenschutzbestimmungen, die Lizenzen, und das Impressum.





Abbildung 6.22: Senden von Feedback

Abbildung 6.23: Einer der About Texte

In der Kontakt Ansicht kann der Benutzer über ein Textfeld sein Feedback, Fragen oder Anregungen eingeben. Sofern er auch eine Antwort auf beispielsweise eine Frage haben möchte kann er unter dem Textfeld seine E-Mail Adresse in ein eigenes Textfeld eintragen. Durch den **Sende Feedback** Button wird der Inhalt das geschriebene Feedback an das Projektteam gesendet. Auf Abb. 6.23 ist dargestellt, wie einer der About Texte aufgebaut ist. Die Abbildung zeigt aber hier nur den obersten Teil des Impressums.

Abgleich der Anforderungen

In diesem Kapitel wird geprüft, ob die funktionalen Anforderungen an die App erfüllt, nur teilweise erfüllt, oder gar nicht implementiert wurden. Dabei werden die Anforderungen aus dem zweiten Kapitel genommen, und mit dem Stand der App verglichen.

7.1 Abgleich der funktionalen Anforderungen

Dieser Abschnitt gleicht die funktionalen Anforderungen mit den funktionalen Anforderungen aus dem dritten Kapitel ab.

Nr.	Beschreibung	Abgleich
1	Registration des Benutzers	Die Registration ist möglich. Nur die Be-
		stätigung der E-Mail ist noch nicht imple-
		mentiert
2	Fragebogen zur Überwachung der Tinni-	Nach Anforderung implementiert
	tuswahrnehmung	
3	Einstellungen der Benachrichtigungszeiten	Nach Anforderung implementiert mit klei-
		ner Umstrukturierung zu alten App
4	Einstellungen des Klingeltons einer Be-	Nach Anforderung implementiert
	nachrichtigung	
5	Anzeige der Ergebnisse in der App	Nach Anforderung implementiert mit Um-
		strukturierung in der Ergebnisanzeige
6	Messung des Geräuschpegels	Noch nicht Implemeniert
7	App auch ohne Internetverbindung nutz-	Nach Anforderung implementiert. Ledig-
	bar	lich die Registration erfordert eine Inter-
		netverbindung
8	Ausfüllen der statistischen Fragebögen auf	Ausfüllen der statistischen Fragebögen
	der Website und der App	derzeit nur auf der Website möglich
9	Synchronisierung der Antworten auf die	Da die statistischen Fragebögen noch nicht
	statistischen Fragebögen	in der App implementiert sind, ist auch die
		Synchronisierung noch nicht möglich
10	Synchronisierung der Ergebnisse	Noch keine Anbindung an die Datenbank
		der App. Somit noch nicht implementiert

Nr.	Beschreibung	Abgleich
11	Fragebögen vom Server bereitgestellt	Da die App noch nicht an die Datenbank
		angebunden ist, ist dies noch nicht mög-
		lich. Die Datenstruktur in der App ist aber
		darauf ausgelegt die Fragebögen generisch
		vom Server laden zu können. Derzeit sind
		noch Default Fragebögen in der lokalen
		Datenbank der App gespeichert.
12	Feedback geben	Nach Anforderung implementiert, wird
		aus fehlender Anbindung an die externe
		Datenbank aber noch nicht gesendet
13	Benutzerdefinierte- und Standardbenach-	Nach Anforderung implementiert
	richtigungen	

7.2 Abgleich der nicht-funktionalen Anforderungen

In diesem Abschnitt werden die nicht-funktionalen Anforderungen mit den nicht-funktionalen Anforderungen aus dem zweiten Kapitel abgeglichen.

Nr.	Beschreibung	Abgleich
1	Ausfüllen des Fragebogens zur Überwa-	Das Ausfüllen des Fragebogens wurde ein-
	chung der Schwankungen der Tinnitus-	fach und schnell gestaltet. Somit ist die
	wahrnehmung sollte in weniger als einer	Anforderung erfüllt
	Minute möglich sein	
2	Benutzerfreundliche Bedienung der App	Aufbau der App wurde einfach und über-
		sichtlich gehalten. Somit ist die Anforde-
		rung erfüllt.
3	Zuverlässigkeit	Die App stürzt bisher in keinem Testfall
		ab.
4	Effizienz und Erweiterbarkeit	Bei Tests keine Verzögerungen festgestellt.
		Die Appstruktur lässt sich ohne Probleme
		erweitern.

Zusammenfassung und Ausblick

Im Rahmen dieser Bachelorarbeit ist eine iOS App entstanden, die auf Basis von Swift 3 und für iOS 10 entwickelt wurde. Diese App soll die alte, bereits vorhandene Track Your Tinnitus App im Apple App Store ersetzen. Das Design der App wurde überarbeitet und anstelle von Objective C als Programmiersprache Swift verwendet. Der derzeitige Stand der App ist allerdings noch nicht so weit, dass die alte App sofort ersetzt werden kann. Die Anbindung an die externe Datenbank und die Website ist noch nicht realisiert. Bei der Entwicklung der App wurde die Anbindung an eine externe Datenbank bereits berücksichtigt. Das gesamte Interface wurde überarbeitet und die Benutzerfreundlichkeit der App optimiert.

Nach Abgabe dieser Arbeit werde ich die App fertig programmieren, sodass die bisherige App durch diese im App Store ersetzt werden kann. Das User Interface wurde für ein generisches Gerät programmiert. Das heißt, die App wird nicht nur für iPhone, sondern auch für Tablets verfügbar sein. Ich hoffe, dass diese App mehr Benutzer anspricht, und mehr genutzt wird als die bisherige App, denn die von der App erfassten Daten sind ein wichtiger Bestandteil des Track Your Tinnitus Projektes. Als ich dieses Thema als Bachelorarbeit gewählt habe, wollte ich zunächst nur eine App schreiben. Doch das Thema Tinnitus interessierte mich zunehmend. Deswegen beschäftigte ich mich intensiver mit dem Thema Tinnitus und gewann einige neue Erkenntnisse über diese Erkrankung. Für die App Entwicklung musste ich mir im Selbststudium die Programmiersprache Swift beibringen, was mich auch in meinem Informatikstudium weitergebracht hat. Während der Entwicklung hatte ich Freude an diesem Projekt mitarbeiten zu können und etwas zu programmieren, was anderen Menschen in Zukunft helfen könnte.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sinngemäße Übernahmen aus anderen Werken sind als solche kenntlich gemacht und mit genauer Quellenangabe (auch aus elektronischen Medien) versehen.

Ulm, den 16.12.2016

Johannes Hueber

Literaturverzeichnis

- [Gmb16] Sonormed GmbH. Tinnitracks. http://www.tinnitracks.com/de, Nov 2016.
- [Her14] Jochen Herrmann. Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten. March 2014.
- [Kle12] Oleg Klemin. Tinnitus measurer. https://itunes.apple.com/de/app/tinnitus-measurer/id464653412?mt=8, Okt 2012.
- [Mob16] Phase4 Mobile. Tinnitus hq: Natural sound masking and filters to relieve ringing in the ears. https://itunes.apple.com/app/id891628361, Aug 2016.
- [PPLS16a] Thomas Probst, Rüdiger Pryss, Berthold Langguth, and Winfried Schlee. Emotion dynamics and tinnitus: Daily life data from the trackyourtinnitus application. *Scientific Reports*, 6, 2016.
- [PPLS16b] Thomas Probst, Rüdiger Pryss, Berthold Langguth, and Winfried Schlee. Emotional states as mediators between tinnitus loudness and tinnitus distress in daily life: Results from the trackyourtinnitus application. Scientific Reports, 6, February 2016.
- [PRH+15] Rüdiger Pryss, Manfred Reichert, Jochen Herrmann, Berthold Langguth, and Winfried Schlee. Mobile crowd sensing in clinical and psychological trials a case study. In 28th IEEE Int'l Symposium on Computer-Based Medical Systems, pages 23–24. IEEE Computer Society Press, June 2015.
- [PRLS15] Rüdiger Pryss, Manfred Reichert, Berthold Langguth, and Winfried Schlee.

 Mobile crowd sensing services for tinnitus assessment, therapy and research. In

 IEEE 4th International Conference on Mobile Services (MS 2015), pages 352–359.

 IEEE Computer Society Press, June 2015.
- [SPR15] Johannes Schobel, Rüdiger Pryss, and Manfred Reichert. Using smart mobile devices for collecting structured data in clinical trials: Results from a large-scale case study. In 28th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2015), pages 13–18. IEEE Computer Society Press, June 2015.
- [SPR⁺16a] Marc Schickler, Rüdiger Pryss, Manfred Reichert, Martin Heinzelmann, Johannes Schobel, Berthold Langguth, Thomas Probst, and Winfried Schlee. Using wearables in the context of chronic disorders results of a pre-study. In 29th IEEE Int'l Symposium on Computer-Based Medical Systems, pages 68–69, June 2016.
- [SPR⁺16b] Marc Schickler, Rüdiger Pryss, Manfred Reichert, Johannes Schobel, Berthold Langguth, and Winfried Schlee. Using mobile serious games in the context of chronic disorders a mobile game concept for the treatment of tinnitus. In 29th IEEE Int'l Symposium on Computer-Based Medical Systems (CBMS 2016), pages 343–348, June 2016.

- [SPS⁺16] Johannes Schobel, Rüdiger Pryss, Marc Schickler, Martina Ruf-Leuschner, Thomas Elbert, and Manfred Reichert. End-user programming of mobile services: Empowering domain experts to implement mobile data collection applications. In 5th IEEE International Conference on Mobile Services (MS 2016), pages 1–8. IEEE Computer Society Press, May 2016.
- [SPSR16a] Johannes Schobel, Rüdiger Pryss, Marc Schickler, and Manfred Reichert. A configurator component for end-user defined mobile data collection processes. In Demo Track of the 14th International Conference on Service Oriented Computing (ICSOC 2016), October 2016.
- [SPSR16b] Johannes Schobel, Rüdiger Pryss, Marc Schickler, and Manfred Reichert. A lightweight process engine for enabling advanced mobile applications. In 24th International Conference on Cooperative Information Systems (CoopIS 2016), number 10033 in LNCS, pages 552–569. Springer, October 2016.
- [SPSR16c] Johannes Schobel, Rüdiger Pryss, Marc Schickler, and Manfred Reichert. Towards flexible mobile data collection in healthcare. In 29th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2016), pages 181–182, June 2016.
- [SPW+16] Johannes Schobel, Rüdiger Pryss, Wolfgang Wipp, Marc Schickler, and Manfred Reichert. A mobile service engine enabling complex data collection applications. In 14th International Conference on Service Oriented Computing (ICSOC 2016), number 9936 in LNCS, pages 626-633, October 2016.
- [SRP+15] Marc Schickler, Manfred Reichert, Rüdiger Pryss, Johannes Schobel, Winfried Schlee, and Berthold Langguth. *Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health.* eXamen.press. Springer Vieweg, October 2015.
- [SSP⁺15] Jaime Serquera, Winfried Schlee, Rüdiger Pryss, Patrick Neff, and Berthold Langguth. Music technology for tinnitus treatment within tinnet. In Audio Engineering Society Conference: 58th International Conference: Music Induced Hearing Disorders. Audio Engineering Society, 2015.
- [SSPR15a] Marc Schickler, Johannes Schobel, Rüdiger Pryss, and Manfred Reichert. Mobile crowd sensing a new way of collecting data from trauma samples. In XIV Conference of European Society for Traumatic Stress Studies (ESTSS)

 Conference, page 244, June 2015.
- [SSPR15b] Johannes Schobel, Marc Schickler, Rüdiger Pryss, and Manfred Reichert. Process-driven data collection with smart mobile devices. In 10th International Conference on Web Information Systems and Technologies (Revised Selected Papers), number 226 in LNBIP, pages 347–362. Springer, 2015.