ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Shades Of Noise: Konzeption und Realisierung einer mobilen Interventionsapp zur Unterstützung von Tinnituspatienten mithilfe gezielter auditorischer Stimulation

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Chris Gabler
chris.gabler@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Carsten Vogel, M. Sc.

2020

Fassung 16. Juni 2020

Satz: PDF-LaTeX 2ε

Kurzfassung

10-15% der Menschheit leiden häufig oder konstant unter Tinnitusbeschwerden. Etwa 1-2% haben starke Beschwerden, sodass sie in ihrer Lebensqualität eingeschränkt sind. Durch die demographisch alternde Gesellschaft steigert sich dieser Anteil weiter. Bis heute gibt es keine generell anwendbare Heilung für die durch Tinnitus hervorgerufene Geräuschwahrnehmung. Im Zuge dieser Arbeit wird eine mobile iOS-Anwendung zur Durchführung von Studien und der Erfassung von Datensätzen bezüglich der auditorischen Stimulation als Behandlungsmaßnahme für Tinnituserkrankte entwickelt.

Mit dem in dieser Arbeit entwicklten Prototypen soll gezielt der Tinnitus der Studienteilnehmer per auditorischer Stimulation gemildert werden. Die zugehörigen Datensätze sollen für Studien auswertbar erfasst werden. Diese Arbeit führt zunächst eine Anforderungsanalyse durch und stellt daraus abgeleitet ein Architekturkonzept zur Verfügung. In dem realisierten Prototypen stehen den Benutzern eine Vielzahl an Geräuschen zur Verfügung. Diese sind teilweise natürliche Geräusche und teilweise generierte Geräusche. Nach der auditorischen Stimulation wird der Benutzer aufgefordert, die Verbesserung bezüglich seiner Beschwerden zu bewerten. Diese Daten werden zusammen mit der Dauer der Stimulation und dem angehörten Geräusch gespeichert.

Inhaltsverzeichnis

1

Einleitung

Als subjektiver Tinnitus wird die Wahrnehmung von Geräuschen in Abwesenheit von externen Geräuschen beschrieben. Diese können als klingeln, summen, rauschen oder klicken empfunden werden [1]. 10-15% der Menschheit sind von Tinnitus betroffen und beschreiben ihre Tinnituswahrnehmung als häufig oder konstant. Etwa 1-2% haben starke Beschwerden, sodass der Tinnitus sie in ihrer Lebensqualität einschränkt. Durch eine demographisch alternde Gesellschaft steigert sich der Anteil weiter [2]. Tinnitus beeinflusst alltägliche Bereiche wie die Sprachwahrnehmung und die Geräuschlokalisierung. Darüber hinaus können Begleiterkrankungen wie Stress, Angst und Depressionen auftreten [3].

Bei einer Mehrheit der Tinnituserkrankten tritt das Geräusch als einzelner Ton mit einer definierbaren Frequenz und Lautstärke auf. Dieses Geräusch wird entweder beidseitig, beidseitig mit einer leichten Tendenz zu einer Seite oder auf einer Seite wahrgenommen [2].

Bis heute gibt es keine generell anwendbare Heilung für die Geräuschwahrnehmung durch Tinnitus. Etablierte Interventionsmöglichkeiten zielen auf das mildern des Tinnitusgeräusches oder der Begleiterscheinungen ab [3].

Eine Behandlungsmaßnahme ist die auditorische Stimulation. Bereits in den früheren 70er Jahren untersuchte Feldmann die kurzzeitige Tinnitiusunterdrückung nach der auditorischen Stimulation. Dies wird als *residual inhibition* bezeichnet [4]. Aktuelle Studien zeigen, dass die *residual inhibition* bei 60-80% der Erkrankten ausgelöst werden kann. Zudem lässt sich der Effekt verstärken, indem das abgespielte Geräusch nahe oder im Frequenzbereich des individuellen Tons liegt [5].

1.1 Problemstellung

Bisher werden Studien zur auditorischen Stimulation lokal an der interdisziplinären Tinnitusklinik der Universität Regensburg durchgeführt [2] [3] [5].

Im Rahmen des UNITI Projekts[1] werden verschiedene Studien zur Tinnitusforschung international und großflächig durchgeführt. Hierzu gehören unter anderem Studien zur auditorischen Stimulation. Deswegen wird im Rahmen dieser Arbeit eine mobile iOS-Anwendung zur Durchführung von Studien und der Erfassung von Datensätzen bezüglich auditorischer Stimulation konzipiert und realisiert.

1.2 Zielsetzung

Mit diesem Prototypen soll gezielt der Tinnitus der Studienteilnehmer per auditorischer Stimulation kurzfristig gemildert werden und entsprechende Datensätze über die kurzfristige Veränderung erfasst werden. Die mobile Anwendung soll benutzerfreundlich aufgebaut und somit einfach zu bedienen sein. Um die Datengüte zu erhöhen soll darüber hinaus ein Fragebogen mit einer Zusammenstellung von Tinnitus relevanten Fragen erfasst werden können. Durch langfristige tägliche Anwendung sollen an Tinnitus erkrankte Personen mehr Kontrolle über die Erkrankung erlangen und somit selbstständig für eine Verbesserung ihrer Symptome sorgen können.

1.3 Struktur der Arbeit

Diese Arbeit ist in acht Kapitel unterteilt. Im nachfolgenden Kapitel wird die Arbeit in den Gesamtprojektrahmen eingeordnet und der Vergleich mit ähnlichen Projekten aufgeführt. In Kapitel 3 erfolgt die Anforderungsanalyse an *Shades Of Noise*. In Kapitel 4 wird die Architektur der Anwendung beschrieben. Hierbei wird zuerst eine Übersicht gegeben, bevor unter anderem auf den Ablauf, die Datenstruktur und die lokale Datenbank eingegangen wird. Das Kapitel 5 zeigt ausgewählte Besonderheiten und Schwierigkeiten

[1] `https://uniti.tinnitusresearch.net/`, zuletzt besucht: 31. März 2020

während der Implementierung auf. Die Vorstellung von *Shades Of Noise* folgt in Kapitel 6. In Kapitel 7 werden die in Kapitel 3 erfassten Anforderungen mit dem Stand der Entwicklung abgeglichen. Das abschließende Kapitel 8 bietet eine Zusammenfassung und einen Ausblick mit möglichen Weiterentwicklungen der Anwendung.

2

Vergleichbare Projekte

In diesem Kapitel werden einige ähnliche Projekte vorgestellt. Dabei werden die Projekte jeweils kurz erläutert und ihr Zusammenhang zu *Shades Of Noise* aufgeführt.

2.1 UNITI Projektrahmen

Im Zuge einer internationalen Zusammenarbeit zur Tinnitusforschung entstand das Projekt *Unification of Treatments and Interventions for Tinnitus Patients - UNITI*[1]. Das vierte Arbeitspaket „Harmonization of Technical Solutions“ unter Leitung von Herrn Professor Rüdiger Pryss ist für die Zusammenführung von verschiedenen Anwendungen in eine harmonisierte Lösung verantwortlich. Die Apps *Track Your Tinnitus*, *Psycho Elocation* und *Shades Of Noise* sollen dafür in eine gemeinsame Anwendung überführt werden. *Match Your Tinnitus* gehört ebenfalls zu diesem Anwendungspaket, bleibt im ersten Schritt allerdings eine eigenständige Anwendung. Die Entwicklung und Harmonisierung ist für das Jahr 2020 geplant. Der Studienbeginn ist auf Anfang des Jahres 2021 angesetzt.

2.2 Track Your Tinnitus

Das Projekt *Track Your Tinnitus*[2] existiert bereits seit 2014. Da bei 60% der Tinnituspatienten die subjektive Lautstärke des Tinnitus innerhalb und zwischen den Tagen erheblich

[1] `https://uniti.tinnitusresearch.net/`, zuletzt besucht: 01. April 2020
[2] `https://www.trackyourtinnitus.org/de/`, zuletzt besucht: 01. April 2020

schwanken kann, ist die Dokumentation dieses Verlaufs sowohl für die Forschung als auch für die Betroffenen selbst wichtig. Bisher wurden Schwankungen der subjektiven Lautstärke lediglich über Tinnitustagebücher handschriftlich dokumentiert, wodurch die exakte Aufzeichnung des zeitlichen Verlaufs nicht möglich ist. Das *Track Your Tinnitus* Projekt bietet sowohl der Forschung als auch den Betroffenen die Möglichkeit diese Schwankungen exakt zu erfassen und nachzuvollziehen. Dabei können ebenfalls die Tätigkeiten und Umgebungslautstärke erfasst werden um so eine Verbindung herstellen zu können. Zudem werden dem Benutzer die Auswertung seiner Daten visuell dargestellt, wodurch dem Betroffenen ermöglicht wird seine Tinnituswahrnehmung besser zu verstehen. Im Rahmen der Diplomarbeit von Jochen Herrmann an der Universität Ulm entstand das Projekt, welches fortlaufend durch weitere Abschlussarbeiten weiterentwickelt wird. Die Apps sind im Apple AppStore[3] und im Google PlayStore [4] erhältlich[6].

2.3 Match Your Tinnitus

Match Your Tinnitus ist ein weiteres Projekt, welches an der Universität Ulm entstanden ist und sich mit der Tinnituserkrankung beschäftigt. Annika Stampf entwickelte dafür einen Prototypen in ihrer Bachelorarbeit im Jahr 2018. Mithilfe der Anwendung sollen Betroffene selbstständig ihre Tinnitusfrequenz möglichst präzise bestimmen können. Dafür absolvieren Benutzer ein Hörtraining auf mehreren Leveln, welche die Erkennung und Bewertung von differenzierten Tonhöhenunterschieden schult. Da die Frequenz ein Hauptparamter des individuellen Tinnitus ist und dadurch ein besseres Ergebnis in der auditorischen Stimulation erzielt werden kann, ist *Match Your Tinnitus* ein wichtiger Bestandteil zur Milderung von Tinnitussymptomen [2]. *Match Your Tinnitus* befindet sich aktuell[5] in der Entwicklungsphase [7].

[3] `https://apps.apple.com/de/app/track-your-tinnitus/id787178122`, zuletzt besucht: 01. April 2020
[4] `https://play.google.com/store/apps/details?id=com.jochenherrmann.trackyourtinnitus`, zuletzt besucht: 01. April 2020
[5] Stand 01. April 2020

2.4 Intersession

Die Masterarbeit von Carsten Vogel beschäftigt sich mit dem aktuellen Trend, den Fokus verstärkt auf die Patientenförderung zwischen Therapiesitzungen zu legen. Dies wird *Intersession-Prozesse* genannt. Traditionell werden Patientendaten per Papierfragebögen erhoben. Daten zu Intersession Erfahrungen der Patienten, wurden herkömmlicherweise direkt vor der Therapiesitzung erhoben, wodurch diese, durch die verstrichene Zeit, oftmals verfälsch erhoben wurden. Dadurch werden verfälschte Daten erhoben. Durch die Verbreitung von Smartphones und die gegebenen technischen Möglichkeiten gewinnen die Themen eHealth und mHealth mehr an Relevanz. Im Zuge der Masterarbeit wurde in Zusammenarbeit mit der Alpen-Adria-Universität Klagenfurt eine mHealth Anwendung entwickelt, welche es erlaubt, wissenschaftliche Intersession-Datensätze direkt aus dem Alltag von Patienten in psychotherapeutischer Behandlung zu erfassen. Dabei werden die Fragebögen automatisch über die mobile Anwendung verteilt, abhängig von den Daten der Therapiesitzung und den Ergebnissen vorheriger Auswertungen. Darüberhinaus ist der Therapeut in der Lage manuelle Interventionen zuzuweisen. Dadurch kann der Therapeut deutlich einfacher und effizienter Daten erheben, auswerten und so die anstehenden Therapiesitzungen besser vorbereiten [8][9][10].

2.5 Weitere Vergleichbare Projekte

Zur Behandlung und Prävention von Tinnitus stehen im Google PlayStore und Apple App Store eine Vielzahl an Anwendungen zur Verfügung. Mehdi et al. fanden in Summe 686 Anwendungen [11]. In ihrer Studie haben sie 87 Anwendungen kategorisiert und untersucht. Die Kategorien sind Tinnitus-Erleichterung, Kognitive Verhaltenstherapie, Hörschutz, Gehörtest und -verbesserung und Smartphone-basierte Elektroenzephalogramme (EEG).

Die Tinnitus-Erleichterung wird bei bestehenden Tinnitussymptomen eingesetzt. Zu ihr gehören verschiedene Behandlungsmethoden wie die Tinnitus-Retraining-Therapie, das Tinnitus Masking oder Gehirnstimulation. 23 Anwendungen wurden von Mehdi et al. zur

Tinnitus-Erleichterung kategorisiert. Die meisten davon bieten Tinnitus Masking oder Geräuschtherapien an, welche verschiedene Techniken wie die akustische Neuromodulation, „tailor-made notched music“ oder Amplitudenmodulation einschließen. „Tailor-made notched music“ beschreibt das Filtern von Tonfrequenzen bei der Geräuschstimulation. Die Frequenz, die gefiltert wird, muss bestmöglich der subjektiven Tinnitusfrequenz entsprechen. Zur Bestimmung der Tinnitusfrequenz kann beispielsweise *Match Your Tinnitus* verwendet werden. Die Tinnitus-Erleichterung lässt sich gut auf Smartphones umsetzen, da diese in der Lage sind zuverlässige und genaue akustische Therapie zu ermöglichen. Zu den Tinnitus-Erleichterungsanwendungen gehören unter anderem *Track Your Tinnitus*, *Kalmeda mynoise*[6] und *ReSound Tinnitus Relief*[7] . *Shades Of Noise* lässt sich ebenfalls hier einordnen [11].

Auch wenn die kognitive Verhaltenstherapie die subjektive Lautstärke des Tinnitus nur geringfügig beeinflusst, so war sie doch ein zentraler Therapieansatz in der Behandlung von Tinnitussymptomen. Kognitive Verhaltenstherapie und Selbsthilfetherapie ist vorallem nützlich im Umgang mit Stress und Angstzuständen, welche durch Tinnitus ausgelöst werden können [12]. In diese Kategorie wurden 13 Anwendungen eingeordnet, darunter die *Beltone Tinnitus Calmer* App[8] und *Youper - Emotional Health*[9] [11]. Strategien im Umgang mit Stress schwanken sowohl zwischen Mann und Frau als auch zwischen verschiedenen täglichen Stresssituationen. In ihrer Studie hat O’Rourke den Einfluss verschiedener Bewältigungsstragien in täglichen Situationen untersucht. Darüberhinaus wurde die Differenz zwischen den Geschlechtern betrachtet. Hierfür wurde die TrackYourStress[10] Anwendung verwendet um von 113 Teilnehmern Datensätze zu erheben. Als Ergebnis wurde festgestellt, dass positive Effekte für Männer und Frauen auf unterschiedliche Bewältigungsstragien zurückzuführen ist. Bei Frauen führt vorallem die soziale Unterstüztung zu einer Verbesserung, wohingegen bei Männern vorallem die aktive Stressbewältigung zu einer Verbesserung führt [13].

[6] `https://www.mynoise.de/`, zuletzt besucht: 17. Mai 2020
[7] `https://www.resound.com/privacy-policy`, zuletzt besucht: 17. Mai 2020
[8] `https://www.beltone.com/en/hearing-aids/apps/tinnitus-calmer-app`, zuletzt besucht: 17. Mai 2020
[9] `https://www.youper.ai/`, zuletzt besucht: 17. Mai 2020
[10] `https://www.trackyourstress.org/home`, zuletzt besucht: 07. Juni 2020

Tinnitus ist eine häufige Begleiterscheinung von Hörverlust. Dadurch kann argumentiert werden, dass Gehörschutz das Risiko des Tinnitus verringern und Tinnituserkrankte im Umgang mit ihren Symptomen unterstützen kann. Mehdi et al. haben 15 Anwendungen zum Thema Gehörschutz betrachtet, wie *Decibel X*[11] und *NoiseScore*[12] [11].

Analog zur Begründung des Gehörschutzes kann auch für den Hörtest argumentiert werden. Der Hörtest kann beispielsweise auch für das Tinnitus Matching verwendet werden und steht deshalb in direktem Zusammenhang mit dem Tinnitus. Die Smartphone-basierte Frequenzbestimmung kann zu einem wichtigen Aspekt von zukünftigen Geräuschtherapien werden. Bei Patienten, welche durch einen Hörschaden an Tinnitus leiden, können Hörhilfen und Cochlearimplantate helfen die Symptome des Tinnitus zu lindern. Neben *Track Your Hearing*[13] und *hearWHO*[14] wurden elf weitere Anwendungen in die Kategorie Hörtest eingeordnet [11].

Anwendungen, welche eine Gehörverbesserung versprechen, können gegen die Beeinträchtigungen der Tinnitussymptome im täglichen Leben helfen, wie etwa dem Sprechen in geräuschstarken Umgebungen oder dem gezielten Hören. 15 Anwendungen beschäftigen sich mit der Hörverbesserung. Dazu gehören *Ear Booster*[15] und *AUD-1*[16] [11].

Traditionell wurde Tinnitus als reines Innerohrproblem angesehen. Hirnforschungen zeigen allerdings, dass die Komplexität von Tinnitus weit über den auditiven Cortex in nicht-auditive Hirnregionen geht. EEG erlaubt zudem die Untersucherung von Gehirnströmen während der Ruhephase und ist weit verbreitet in der Tinnitusforschung[14]. Durch die massive Weiterentwicklung von EEG-Systemen ist es möglich, EEG-Daten außerhalb von Laborbedingungen zu sammeln. Acht Anwendungen wurden in die Kategorie EEG eingeordnet, darunter *EEG 101*[17] und *MyEmotiv*[18] [11].

[11] `https://skypaw.com/decibel10.html`, zuletzt besucht: 17. Mai 2020
[12] `noisescore.com`, zuletzt besucht: 17. Mai 2020
[13] `https://www.trackyourhearing.org/`, zuletzt besucht: 17. Mai 2020
[14] `https://www.who.int/health-topics/hearing-loss/hearwho`, zuletzt besucht: 17. Mai 2020
[15] `http://www.zygneapps.com/?c=pages&a=apps`, zuletzt besucht: 17. Mai 2020
[16] `http://www.aud1.com/`, zuletzt besucht: 17. Mai 2020
[17] `https://github.com/NeuroTechX/eeg-101`, zuletzt besucht: 17. Mai 2020
[18] `https://www.emotiv.com/myemotiv/`, zuletzt besucht: 17. Mai 2020

Wertvolle Erkenntnisse über Patienten können durch das Verbinden von qualitativen *ecological momentary assesment (EMA)*-Datensätzen und Sensordaten von mobilen Anwendungen erlangt werden. Um dies geeignet in einer Softwarelösung umzusetzen sind allerdings einige Herausforderungen zu lösen. Diese sind unter anderem konzeptionelle, technische, rechtliche oder die Architektur betreffende Herausforderungen. Kraft et al. haben diese Herausforderungen identifiziert, entsprechende Empfehlungen abgegeben und eine Referenzarchitektur für eine entsprechende Plattform angegeben. Die Expertise um eine entsprechende Architektur aufzustellen wurde über langjährige Studien gesammelt, darunter eine *crowdsensing*-Studie über die Tinnituserkrankung [15].

3

Anforderungsanalyse

Vor der Implementierung sind die Anforderungen zu spezifieren. Diese sind in mehrere aufeinander aufbauende Abschnitte gegliedert. Zuerst werden die Anwendungsfälle in Abschnitt 3.1 beschrieben. Daraus werden die funktionalen Anforderungen in 3.2 abgeleitet. Nicht-funktionale Anforderungen ergeben sich durch Gespräche mit dem Betreuer[1] und den Verantwortlichen[2]. Diese sind in Abschnitt 3.3 beschrieben.

3.1 Anwendungsfälle

In diesem Abschnitt sind die Anwendungsfälle von *Shades Of Noise* aufgeführt. Diese sind in Abbildung 3.1 zu einem Anwendungsfalldiagramm zusammengefasst und weiter beschrieben.

Die Hauptansichten sind durch die Anwendungsfälle *Ein- und Ausloggen*, *Audio-Player nutzen*, *Statistik verwalten*, *Fragebogen verwalten* und *Einstellungen anpassen* spezifiziert.

Der Anwendungsfall *Ein- und Ausloggen* deckt die Ansicht ab, welche nach Start der App angezeigt wird. Dort kann sich der Benutzer mit vorhandenen Anmeldedaten einloggen. Der Benutzer kann sich hier alternativ auch registrieren oder die App ohne Anmeldedaten verwenden. Zusätzlich deckt dieser Anwendungsfall die Funktionalität des Ausloggens ab. Der Benutzer kann sich vom Profil aus ausloggen.

[1] Carsten Vogel, M. Sc.
[2] Prof. Dr. Rüdiger Pryss, PD Dr. Winfried Schlee und Dr. phil. Patrick Neff

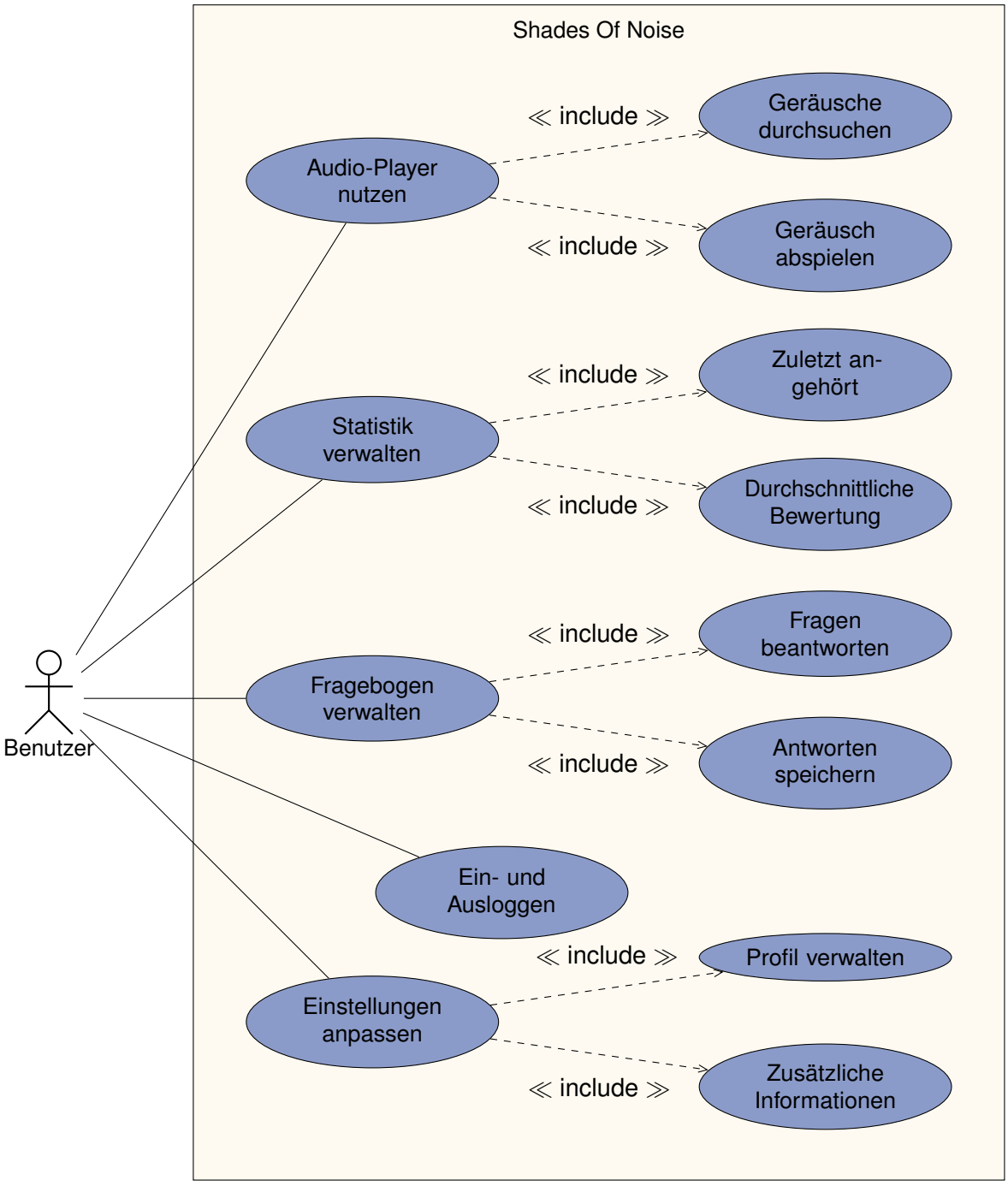

Abbildung 3.1: Anwendungsfalldiagramm von Shades Of Noise

Audio-Player nutzen inkludiert die Unteranwendungsfälle *Geräusche durchsuchen* und *Geräusch abspielen*. Der Benutzer kann hier die Hauptansicht durchsuchen, welche die verschiedenen Geräusche nach Kategorieren sortiert. Hat sich der Benutzer für eine bestimmte Kategorie entschieden, kann er diese erweitern und die einzelnen Geräusche der Kategorie einsehen. Dies wird durch den Anwendungsfall *Geräusche durchsuchen* abgedeckt. Hat sich ein Benutzer für ein Geräusch entschieden, kann er durch anklicken des Geräusches dieses abspielen. Hierbei kann der Benutzer die Wiedergabe anhalten, vor- und zurückspulen und zurücksetzen. Darüberhinaus kann der Benutzer die Wiedergabe in einer Dauerschleife mit optionaler Zeitlängenangabe abspielen. Nachdem das Geräusch abgespielt wurde und der Benutzer den Player wieder verlassen möchte, kann der Benutzer das Geräusch hinsichtlich der Untedrückung des Tinnitus über eine Skala bewerten. Diese Funktionalität wird durch den Anwendungsfall *Geräusch abspielen* abgedeckt.

Die zweite Hauptansicht bildet die Statistik. Diese ist zusammengesetzt aus dem Anwendungsfall *Statistik verwalten* mit den Unteranwendungsfällen *Zuletzt angehört* und *Durchschnittliche Bewertung*. Der Anwendungsfall *Zuletzt angehört* ermöglicht es dem Benutzer seine zuletzt angehörten Titel einzusehen. Zudem werden Informationen über die Abspiellänge und das Datum dargestellt. Es werden die fünf zuletzt angehörten Titel angezeigt. Zudem hat der Benutzer die Möglichkeit sich alle angehörten Titel anzeigen zu lassen. Die zweite Statistik *Durchschnittliche Bewertung* bildet eine Übersicht über alle Geräusche, die kummulierte Hördauer, die Anzahl der Aufrufe und die durchschnittliche Bewertung der Unterdrückung.

Der Anwendungsfall *Fragebogen verwalten* mit *Fragen beantworten* und *Antworten speichern* ermöglicht es dem Benutzer verschiedene Fragen für ein persönliches Tinnitusprofil zu beantworten und anschließend zu speichern.

Der Anwendungsfall *Einstellungen anpassen* beinhaltet *Profil verwalten* und *Zusätzliche Informationen*. *Profil verwalten* bietet dem Benutzer die Möglichkeit sein Profil einzusehen, sich auszuloggen und sein Passwort zu ändern. Der Anwendungsfall *Zusätzliche Informationen* deckt das Impressum, den Kontakt und ähnliche Informationen ab.

3.2 Funktionale Anforderungen

Ausgehend von den Anwendungsfällen des vorherigen Abschnittes ergeben sich die folgenden funktionalen Anforderungen. Diese sind unterteilt in verschiedene Prioritäten. Dabei gibt die Priorität 0 an, dass die Anforderung optional ist. Die Priorität + kennzeichnet Anforderungen, welche nicht zwingend für die Grundfunktionalität der App nötig sind und ++ entspricht Anforderungen, welche zwingend nötig sind.

Tabelle 3.1: Anforderungsanalyse funktionale Anforderungen

ID	**FA01**
Titel	**Ansicht Player**
Beschreibung	Diese Ansicht bildet die Hauptansicht des Musikplayers. Diese Ansicht soll standardmäßig nach dem Einloggen (mit oder ohne Anmeldedaten) angezeigt werden.
Priorität	++
Anwendungsfall	Audio-Player nutzen

ID	**FA02**
Titel	**Kategorien anzeigen und durchstöbern**
Beschreibung	Die Geräusche werden in verschiedene Kategorien gruppiert. Die Kategorien werden in der Player-Ansicht dargestellt und leiten durch Auswählen der entsprechenden Kategorie weiter zu den einzelnen Geräuschen der Kategorie
Priorität	++
Anwendungsfall	Geräusche durchsuchen

ID	**FA03**
Titel	**Geräusche anzeigen und durchstöbern**
Beschreibung	Wenn eine Kategorie ausgewält wurde, werden die zugehörigen Geräusche angezeigt, welche vom Benutzer durchstöbert und ausgewählt werden können.
Priorität	++
Anwendungsfall	Geräusche durchsuchen

ID	**FA04**
Titel	**Audio-Player starten**
Beschreibung	Wird vom Benutzer ein Geräusch ausgewählt, so startet der Player und spielt das Geräusch ab. Ein Geräusch kann abgespielt, pausiert, gestoppt, gespult und in Dauerschleife abgespielt werden.
Priorität	++
Anwendungsfall	Geräusch abspielen

ID	**FA05**
Titel	**Geräusch abspielen**
Beschreibung	Ein Geräusch wird von der App an der aktuell gesetzten Sekundenmarke abgespielt.
Priorität	++
Anwendungsfall	Geräusch abspielen

ID	**FA06**
Titel	**Geräusch pausieren**
Beschreibung	Ein Geräusch kann vom Benutzer an der aktuellen Sekundenmarke pausiert werden.
Priorität	+
Anwendungsfall	Geräusch abspielen

ID	**FA07**
Titel	**Geräusch stoppen**
Beschreibung	Ein Geräusch kann vom Benutzer pausiert werden und die Sekundenmarke wird von der App auf 0 zurück gesetzt.
Priorität	+
Anwendungsfall	Geräusch abspielen

ID	**FA08**
Titel	**Geräusch spulen**
Beschreibung	Ein Geräusch kann vom Benutzer über einen visuellen Zeitstrahl vor- und zurückgespult werden.
Priorität	0
Anwendungsfall	Geräusch abspielen

ID	**FA09**
Titel	**Geräusch in Dauerschleife abspielen**
Beschreibung	Ein Geräusch kann vom Benutzer in Daurschleife abgespielt werden. Dabei soll die App den Benutzer nach einer Zeitdauer (unbegrenzt, eine Minute, zwei Minuten, fünf Minuten, zehn Minuten) fragen und das Geräusch entsprechend lange abspielen.
Priorität	+
Anwendungsfall	Geräusch abspielen

ID	FA10
Titel	**Bewertung Clinical Global Impression (CGI)**
Beschreibung	Wenn der Benutzer den Player verlassen und zurück zur Geräuschübersicht möchte, soll die App nach der Bewertung des Clinical Global Impression fragen. Hier kann der Benutzer im Intervall [1;7] $\in \mathbb{N}$ eine Auswahl treffen oder alternativ auch aktiv „Keine Bewertung“ auswählen. Der Benutzer muss keine Auswahl treffen. Das Bewertungsfenster soll nach 30 Sekunden verschwinden.
Priorität	++
Anwendungsfall	Audio-Player nutzen

ID	FA11
Titel	**Ansicht Statistiken**
Beschreibung	Diese Ansicht bildet die Hauptansicht der Statistiken. Hier werden die verschiedenen Statistiken und die Interaktionsmöglichkeiten angezeigt
Priorität	++
Anwendungsfall	Statistiken verwalten

ID	FA12
Titel	**Zuletzt gehört**
Beschreibung	Dem Benutzer sollen die letzten fünf angehörten Geräusche angezeigt werden. Der Benutzer kann diese Ansicht erweitern und wieder einklappen, sodass alle angehörten Geräusche angezeigt werden. Darüber hinaus soll das Datum, an welchem das Geräusch angehört wurde, die Dauer, wie lange das Geräusch angehört wurde und die Bewertung des CGI angezeigt werden.
Priorität	++
Anwendungsfall	Zuletzt angehört

ID	FA13
Titel	**Durchschnittliche Bewertung**
Beschreibung	Diese Statistik zeigt dem Benutzer alle vorhandenen Geräusche, die Anzahl der Aufrufe der einzelnen Geräusche und die durchschnittliche CGI-Bewertung.
Priorität	++
Anwendungsfall	Durchschnittliche Bewertung

ID	FA14
Titel	**Ansicht Fragebogen**
Beschreibung	Diese Ansicht entspricht der Hauptansicht des Fragebogens. Hier kann der Benutzer verschiedene Fragen aus einem Fragebogen beantworten, die Antworten zurücksetzen oder speichern.
Priorität	++
Anwendungsfall	Fragebogen verwalten

ID	FA15
Titel	**Fragebogen darstellen**
Beschreibung	Die App stellt dem Benutzer den Fragebogen dar. Der Benutzer kann einzelne Fragen auswählen und beantworten. Hierbei stehen Fragen mit unterschiedlicher Anzahl an Antwortmöglichkeiten zur Verfügung (Radio-Button, Multiple Choice, Freitext).
Priorität	++
Anwendungsfall	Fragen beantworten

ID	**FA16**
Titel	**Antwort zurücksetzen**
Beschreibung	Bei Fragen mit Radio-Buttons als Auswahl wird dem Benutzer zudem die Möglichkeit gegeben die Antwort zurückzusetzen. Dies entfernt den Radio-Button und markiert die Frage als unbeantwortet.
Priorität	++
Anwendungsfall	Fragen beantworten

ID	**FA17**
Titel	**Anzeige beantwortete Frage**
Beschreibung	Wenn eine Frage bereits beantwortet ist, soll die App dem Benutzer dies entsprechend darstellen. Mögliche Implementierungen sind ein Check-Haken oder ein grüner Hintergrund.
Priorität	+
Anwendungsfall	Fragen beantworten

ID	**FA18**
Titel	**Antworten speichern**
Beschreibung	Die Antworten des Nutzers müssen vom System gespeichert werden. Dies kann entweder durch eine Auto-Save-Funktion oder durch einen Speichern-Button implementiert werden.
Priorität	++
Anwendungsfall	Antworten speichern

ID	FA19
Titel	**Anzeige Vollständigkeit**
Beschreibung	Dem Benutzer soll angezeigt werden zu wie viel Prozent sein Fragebogen ausgefüllt ist.
Priorität	0
Anwendungsfall	Fragebogen verwalten.

ID	FA20
Titel	**Einloggen**
Beschreibung	Der Benutzer soll sich, falls vorhanden, mit seinen Anmeldedaten einloggen können. Alternativ funktioniert die App auch ohne Anmeldedaten. Nach der Anmeldung wird der Audio-Player angezeigt.
Priorität	++
Anwendungsfall	Ein- und Ausloggen

ID	FA21
Titel	**Profil anzeigen**
Beschreibung	Der Benutzer kann sich von jeder Ansicht aus sein Profil anzeigen lassen. Dieses beinhaltet Informationen über seinen Nutzernamen und die Möglichkeit sein Passwort zu ändern.
Priorität	+
Anwendungsfall	Profil verwalten

ID	FA22
Titel	**Ausloggen**
Beschreibung	Der Benutzer kann sich über die *Profil anzeigen*-Ansicht aus ausloggen. Nach dem Ausloggen wird die Login-Ansicht angezeigt.
Priorität	+

Anwendungsfall	Ein- und Ausloggen, Profil verwalten

ID	**FA23**
Titel	**Ansicht Einstellungen**
Beschreibung	Die App zeigt dem Benutzer zusätzliche Informationen an. Hier befindet sich das Impressum, der Kontakt und die aktuelle Versionsnummer der App.
Priorität	+
Anwendungsfall	Zusätzliche Informationen

ID	**FA24**
Titel	**Sprache ändern**
Beschreibung	Die App soll dem Benutzer die Möglichkeit bieten innerhalb der App die Sprache auf eine gewünschte, unterstützte Sprache umzustellen. Standardmäßig soll die App die Sprache des Smartphones anzeigen.
Priorität	0
Anwendungsfall	Zusätzliche Informationen

ID	**FA25**
Titel	**Navigation**
Beschreibung	Der Benutzer kann über eine Navigation zwischen den verschiedenen Hauptansichten wechseln (FA01, FA11, FA14, FA21, FA23).
Priorität	++
Anwendungsfall	Audio-Player nutzen, Statistik verwalten, Fragebogen verwalten, Einstellungen anpassen

3.3 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen ergeben sich zu Teilen aus dem Gespräch mit dem Betreuer und den Verantwortlichen und zum Teil aus vorhandener und geplanter Struktur.

Tabelle 3.2: Anforderungsanalyse nicht-funktionale Anforderungen

ID	**NFA01**
Titel	**Modularität**
Beschreibung	Die App soll modular aufgebaut werden.
Begründung	Die App wird als Standalone-App entwickelt. Allerdings wird sie als Teil von UNITI im weiteren Projektverlauf in eine gemeinsame App überführt. Hierfür ist wichtig, dass die App modular aufgebaut ist.

ID	**NFA02**
Titel	**Schnittstelle**
Beschreibung	Vorbereitungen für die vorhandene REST-Schnittstelle sollen getroffen werden.
Begründung	Das Projekt UNITI liefert eine REST-API, welche verschiedene Interaktionsmöglichkeiten bietet. Um die Daten zentral zu verwalten und auswerten zu können, sollen Vorbereitungen für diese Schnittstelle getroffen werden. Ein Framework zur Implementierung der Schnittstelle wird parallel von anderen Projektbeteiligten implementiert.

ID	**NFA03**
Titel	**Multilingual**
Beschreibung	Die App soll mehrere Sprachen unterstützen. Unter anderem Deutsch und Englisch.

Begründung	Das Projekt UNITI ist ein internationales Projekt der Tinnitusforschung. Die App soll für Studien in verschiedenen Ländern genutzt werden. Hierfür soll jeweils die Amtssprache des Landes unterstützt werden. Wenn die Amtssprache des Landes nicht vorhanden ist, soll auf Englisch zurückgegriffen werden. Die Sprache soll abhängig von den Endgeräte-Einstellungen vorgenommen werden.

ID	**NFA04**
Titel	**Erweiterbarkeit**
Beschreibung	Die App soll, vorallem in Bezug auf Geräusche, Sprachen und Fragen, erweiterbar sein.
Begründung	Da eventuell zu Beginn nicht alle Übersetzungen der App-Texte in jede Amtssprache des Projektes UNITI vorhanden sind, sollen weitere Übersetzungen mit geringem Aufwand ($\pm$ 2h) hinzugefügt werden können. Genauso sollen die Geräusche und Fragen erweitert werden können. Im Optimalfall können Geräusche und Fragen über die REST-API (siehe NFA02) aktualisiert werden.

ID	**NFA05**
Titel	**Alternative Oberfläche**
Beschreibung	Die App soll sowohl eine Build-Option für die Standalone als auch die Integrated Version beinhalten.
Begründung	In Bezug auf NFA01 und die Integration in eine Gesamtapp sollen zwei verschiedene Build-Optionen bereitgestellt werden, über welche entweder die Standalone oder die Integrated Version erzeugt werden kann.

4

Architektur

Im folgenden Kapitel ist die Architektur von *Shades Of Noise* beschrieben. Zuerst folgt eine Übersicht mithilfe von Abbildung 4.1. In Abschnitt 4.2 wird ein typischer Ablauf aufgeführt. Darauf folgt die Datenstruktur und das lokale Datenbankmodell in Abschnitt 4.3. Abschnitt 4.4 und Abschnitt 4.5 beschreiben die ViewController und Views.

4.1 Architekturübersicht

Die Architektur entspricht dem **M**odel-**V**iew-**C**ontroller-Pattern.

Die Views werden dabei als Storyboards erstellt [16]. Storyboards sind XML-Dateien, welche die Szene beschreiben. In der IDE XCode, welche MacOS nativ ist, werden Storyboards über einen Scene Builder graphisch dargestellt und können teilweise per Drag-And-Drop bearbeitet werden.

Die Controller sind als ViewController implementiert. Jeder View sind ein oder mehrere ViewController zugewiesen. Diese verarbeiten die Nutzereingabe über die View und aktualisieren entsprechend der Eingabe die View. Die Änderung an den Datensätzen der lokalen Datenbank wird über eine Hilfsklasse realisiert. Die Controller greifen zur Manipulation dieser Daten auf die Hilfsklasse zu.

Für die Models existieren einerseits die Entitäten in der lokalen Datenbank, andererseits gibt es Transferklassen, welche zur Überführung der Entitäten in Swift-Klassenobjekte und zurück genutzt werden. Abbildung 4.1 zeigt ein vereinfachtes Modell der Architektur, welche in *Shades Of Noise* implementiert ist.

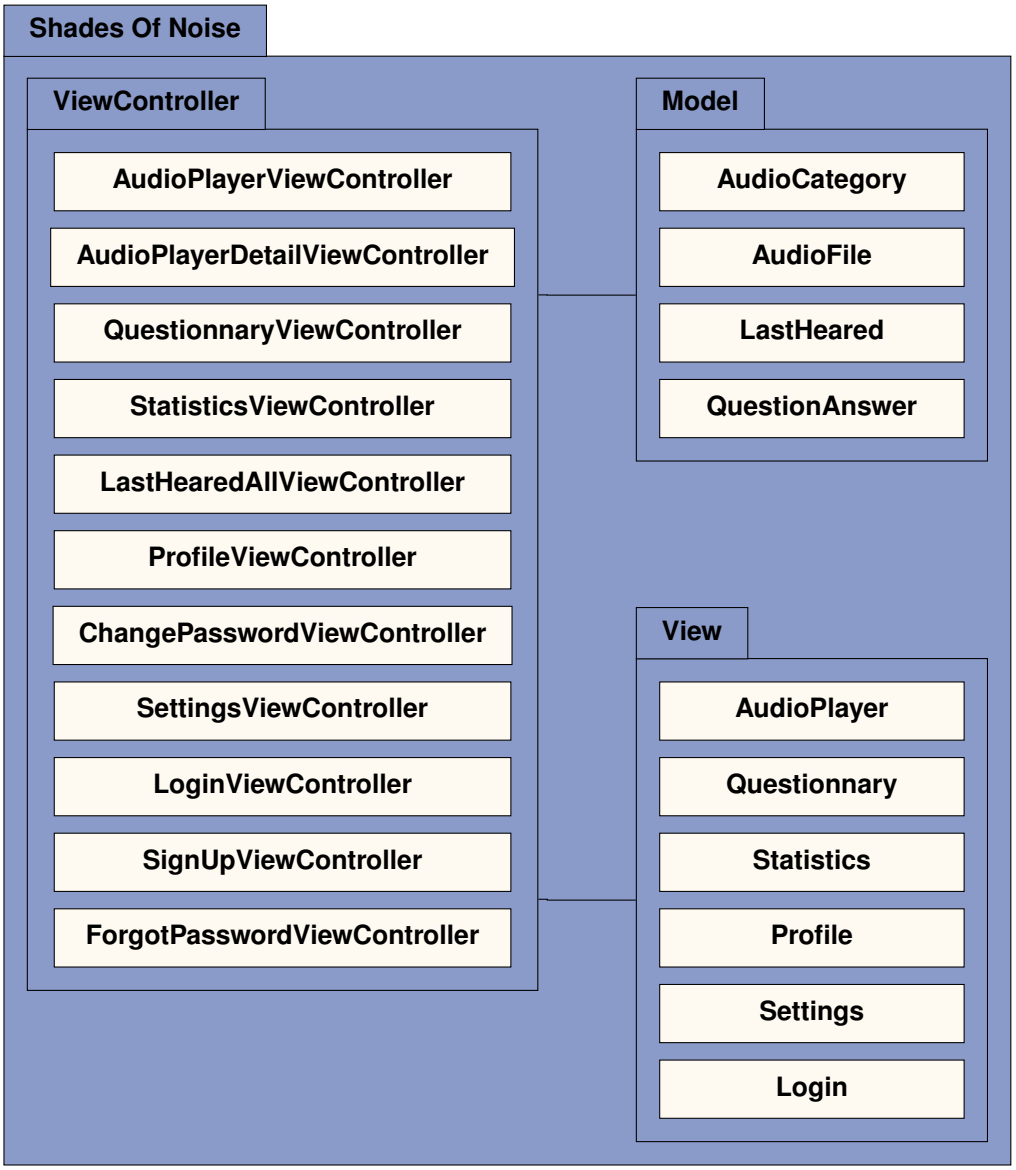

Abbildung 4.1: Architekturübersicht von Shades Of Noise

4.2 Genereller Ablauf

Das Zustandsdiagramm in Abbildung 4.2 zeigt die mögliche Navigation für die Standalone Anwendung. Alle hier aufgeführten Zustände sind in weiteren Abbildungen mit ihren Subzuständen genauer spezifiziert; da die Navigation aus jedem Subzustand möglich ist, sind diese zusammengefasst. Über die Tableiste kann der Benutzer jederzeit zu einem anderen Tab wechseln. Die möglichen Tabs sind die Audio-Übersicht, der Fragebogen, die Statistik und die Einstellungen. Das Profil lässt sich über die Navigationsleiste erreichen. Ein Endzustand ist nicht eingezeichnet. Ein Benutzer kann die Anwendung allerdings aus jedem Zustand beenden, beispielsweise durch Schließen der Anwendung. Ansonsten terminiert die Zustandsmaschine nicht.

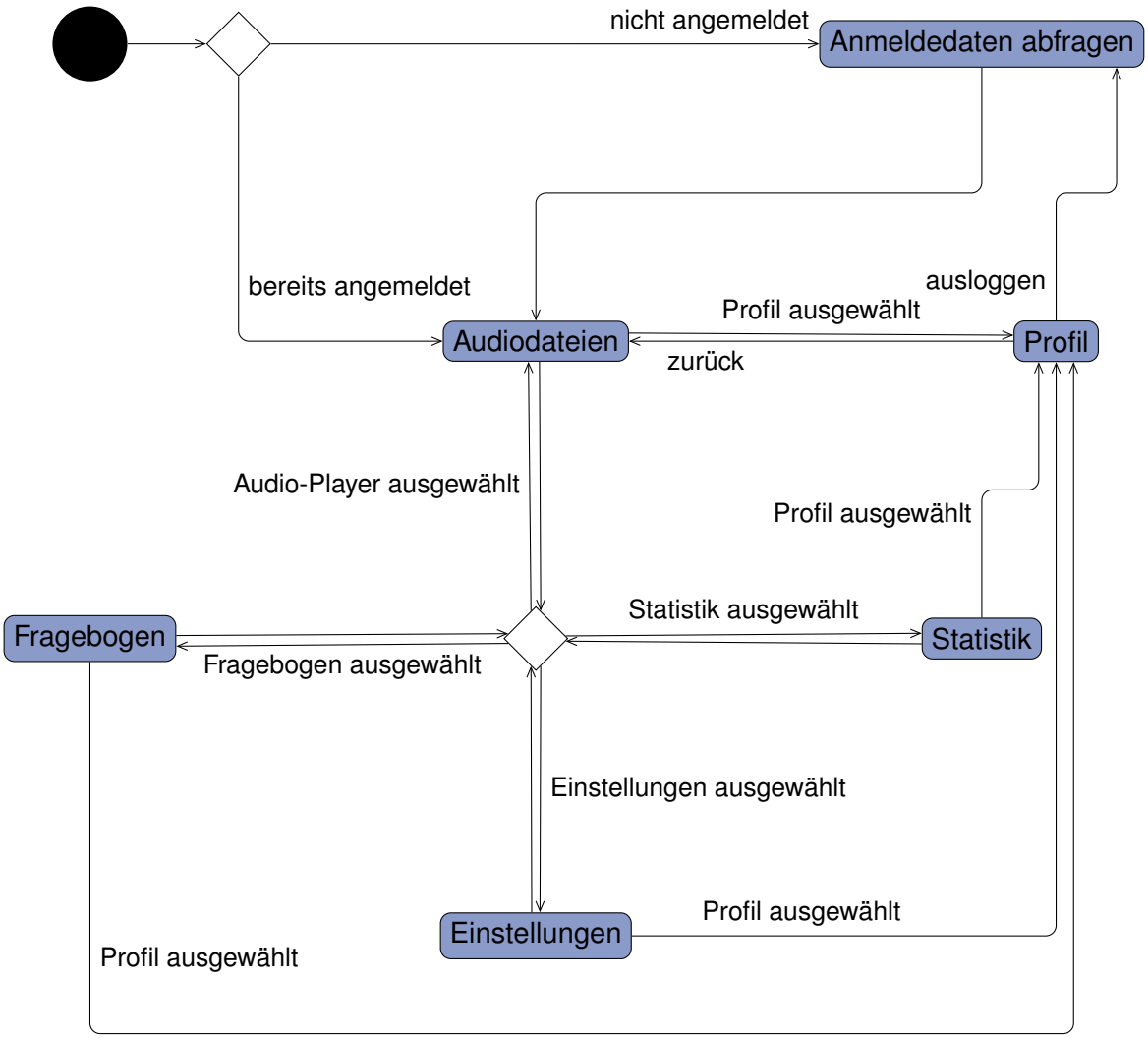

Abbildung 4.2: Zustandsdiagramm Navigation

Wenn ein Benutzer die Anwendung startet und nicht angemeldet ist, findet er die Login-Ansicht vor. Diese ist beschrieben durch den Zustand *Anmeldedaten abfragen*. Abbildung 4.3 beschreibt das Subzustandsdiagramm. Der Benutzer kann sich registrieren oder sein Passwort zurücksetzen. Dadurch befindet der Benutzer sich immer noch in der Login-Ansicht. Hier kann er sich entscheiden, ob er sich mit seinen Anmeldedaten anmelden möchte oder die Anwendung ohne Anmeldedaten nutzen möchte.

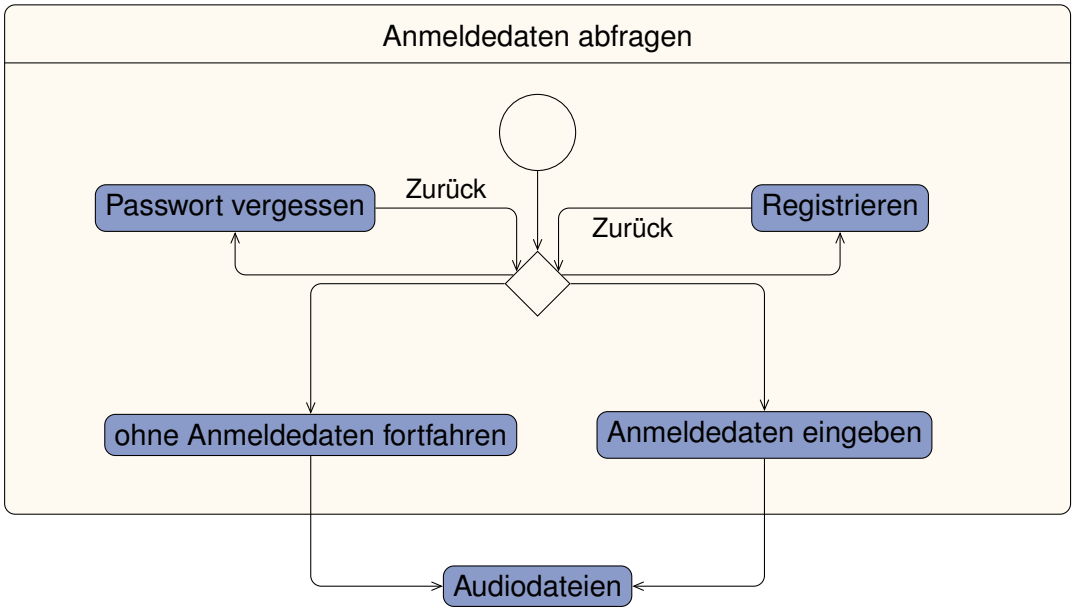

Abbildung 4.3: Anmeldedaten abfragen

Ist der Benutzer bereits angemeldet und startet die Anwendung, befindet er sich in der Übersicht der Audiodateien. Hier werden die Audiodateien in Kategorien unterteilt dargestellt. Zu Beginn sind alle Kategorien eingeklappt. Wie in Abbildung 4.4 zeigt, kann der Benutzer diese Kategorien ein- und ausklappen. Hat er sich für eine Audiodatei entschieden, wird der Audio-Player gestartet. Der Player kann direkt wieder verlassen oder genutzt werden. Wird der Audio-Player genutzt, so kann die Audiodatei vor- und zurückgespult und gestoppt werden. Möchte der Benutzer die Audiodatei in Dauerschleife hören, so kann er den Loop aktivieren und bei Bedarf auch wieder deaktivieren. Wenn die Audiodatei nicht abgespielt wird, kann der Benutzer das Abspielen starten. Ansonsten kann er das Abspielen pausieren. Verlässt der Benutzer den Audio-Player,

hängt es davon ab, ob die Audiodatei abgespielt wurde oder nicht. Wurde sie nicht abgespielt, wird der Audio-Player verlassen und der Benutzer findet sich wieder in der Kategorienübersicht vor. Wurde die Audiodatei hingegen abgespielt, so wird der Benutzer zusätzlich aufgefordert eine Bewertung abzugeben.

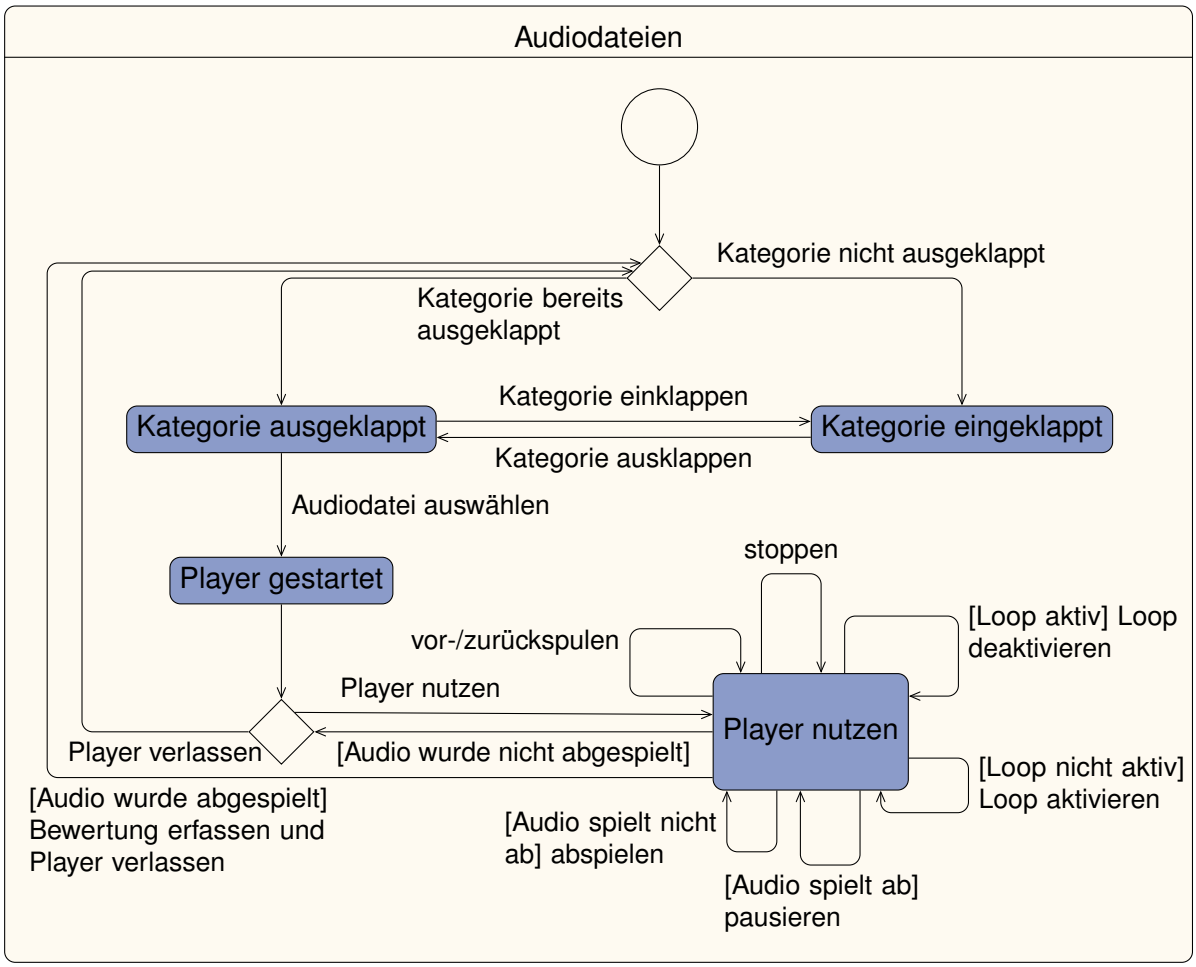

Abbildung 4.4: Audiodateien

Über das Profil kann der Benutzer sein Passwort ändern, sich ausloggen oder zurück zur Audio Übersicht gelangen. Im Profil selbst wird dem Benutzer sein Nutzername und seine E-Mail-Adresse angezeigt. Das Zustandsdiagramm in Abbildung 4.5 beschreibt diese Möglichkeiten.

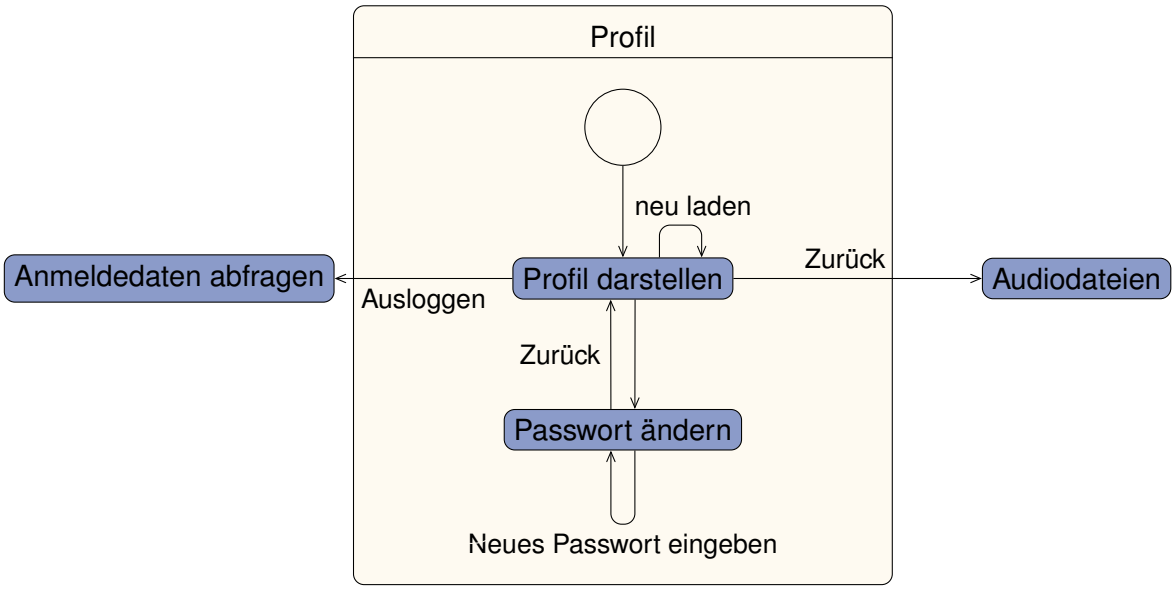

Abbildung 4.5: Profil

Im Fragebogen-Tab wird dem Benutzer ein Fragebogen dargestellt, welcher demographische und Tinnitus-relevante Fragen beinhaltet. Der Benutzer kann hier seine Antworten ändern. Ändern beinhaltet in diesem Sinne sowohl das erstmalige Beantworten von Fragen, sowie das Abändern und das Löschen einer bereits gegebenen Antwort. Veranschaulicht wird das durch Abbildung 4.6.

Wechselt ein Benutzer zur Statistikseite, so werden dort beide Statistiken dargestellt. Standardmäßig ist die Bewertungsstatistik nach Namen sortiert, die Sortierung lässt sich durch den Benutzer ändern. Um alle zuletzt angehörten Dateien zu sehen, muss der Benutzer zu einer weiteren Ansicht navigieren. Diese ist durch *Alle Einträge zuletzt gehört darstellen* in Abbildung 4.7 beschrieben. Von hier aus kann der Benutzer über die Navigationsleiste wieder zurück zur geteilten Statistikseite wechseln.

In den Einstellungen findet der Benutzer zunächst ein weiteres Menü vor. Hier kann er sich das Impressum oder den weiteren Kontakt darstellen lassen. Zusätzlich lässt sich hier die Sprache der App ändern. Sollte der Benutzer eine neue Sprache einstellen, so muss die Anwendung neugestartet werden um die Änderung zu übernehmen. Abbildung 4.8 zeigt die Möglichkeiten in den Einstellungen.

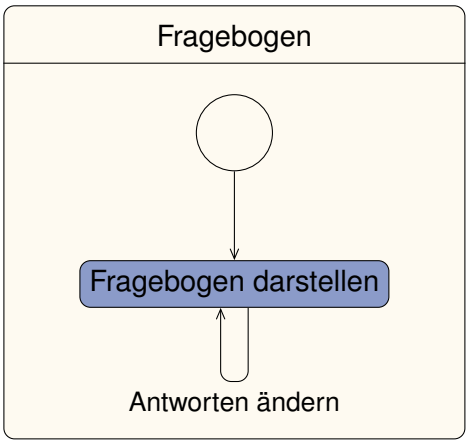

Abbildung 4.6: Fragebogen

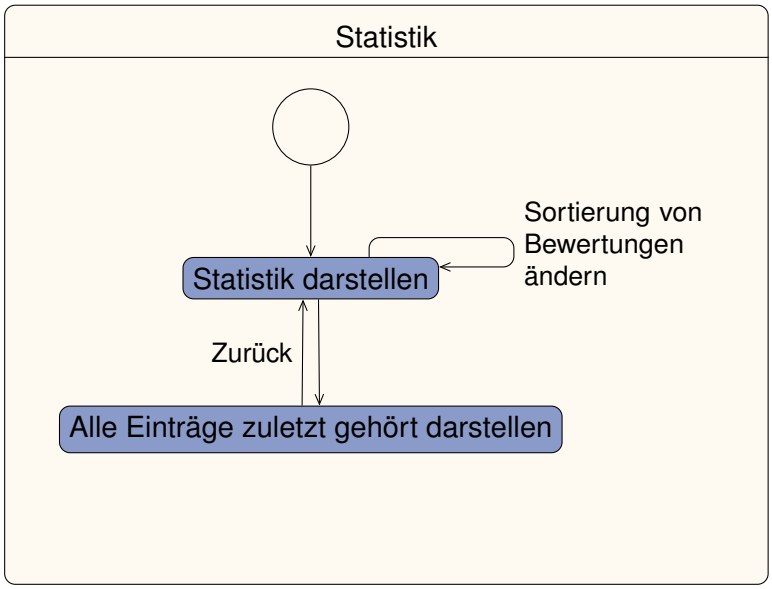

Abbildung 4.7: Statistik

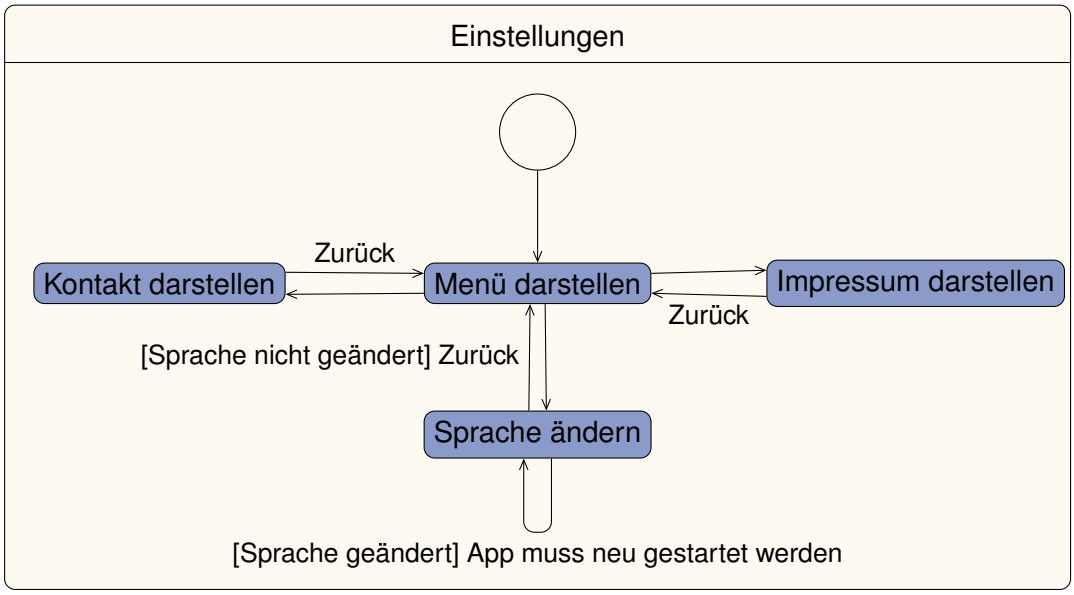

Abbildung 4.8: Einstellungen

4.3 Datenbankschema

Abbildung 4.9 beschreibt das Datenbankmodell der lokalen Datenbank. Die lokale Datenbank ist mit Swifts Core Data implementiert [17]. Core Data nutzt eine zu Grunde liegende SQLite Datenbank [18]. SQLite ist ein relationales Datenbanksystem.

Die Entität *AudioFile* beschreibt die Audiodateien, welche mit dem Player angehört werden können. *AudioCategory* bildet die Referenz zu der entsprechenden Kategorie und speichert den aktuellen Zustand mit, ob die Kategorie ein- oder ausgeklappt ist. Die *LastHeared* werden für die Statistik benötigt und speichern sowohl den Zeitpunkt als auch die Bewertung der auditiven Stimulation. Für den Fragebogen werden die Entitäten *Question* und *Answer* benötigt. *Question* beinhaltet die Frage und *Answer* beinhaltet die Antworttexte, den Typ der Antwort, also ob es eine Single-Choice, Multiple-Choice oder Texteingabe ist, und ob die Antwort ausgefüllt ist.

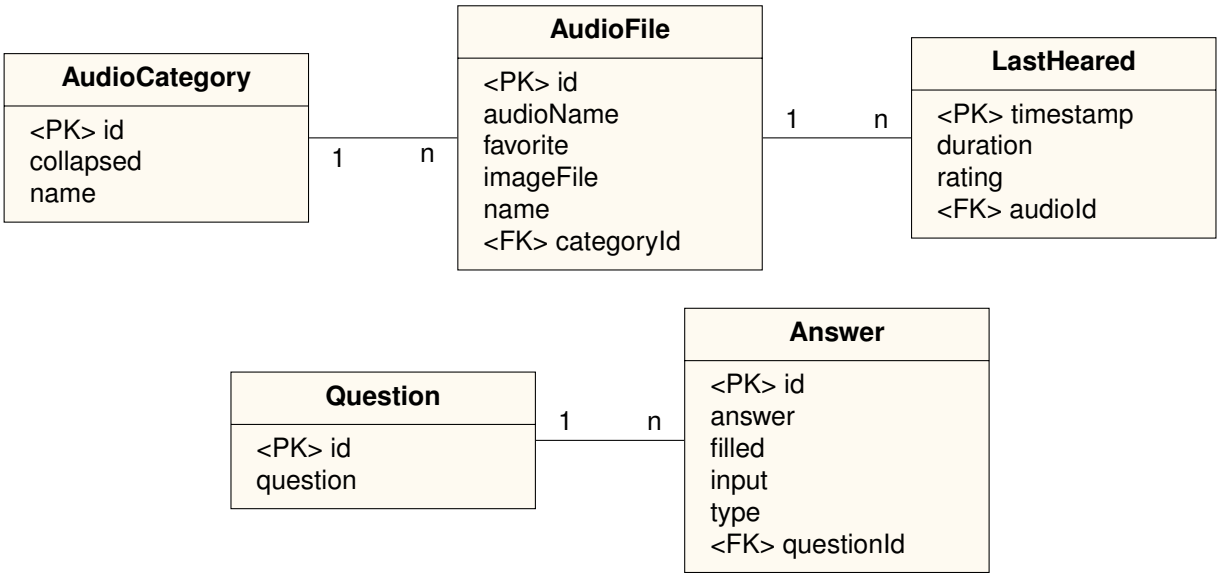

Abbildung 4.9: Datenmodell von Shades Of Noise

4.4 ViewController

Wie bereits in der Architekturübersicht beschrieben, sind die ViewController die Bindeglieder zwischen den Views und dem Model. Der Quelltext des *AudioPlayerDetail-ViewController*, also dem ViewController, der für die Darstellung und Verarbeitung des Audio-Players verantwortlich ist, ist unter Quelltext A.1 aufgeführt. Dieser Quelltext steht exemplarisch für die Architektur der gesamten Anwendung. Hier wird deutlich, wie ein ViewController das Model mit der View verknüpft und die Aktionen der View verarbeitet werden. Zeile 33 bis einschließlich Zeile 78 beschreiben beispielsweise die Verbindung zwischen der View und dem Controller. Hier werden die Aktionen der View verarbeitet und die Wiedergabe gestartet, pausiert, gestoppt und gespult. Die Zeilen 130 bis 164 zeigen eine Verknüpfung des Controllers mit dem Model. Hier wird zuerst versucht, die Audiodatei über den übergebenen Parameter *audio.audioName* zu laden. Ist dies erfolgreich, so wird eine Instanz des Audio-Players gestartet und einige Felder der View mit Daten des Models gefüllt.

4.5 Views

Wie bereits beschrieben werden die Views als Storyboard implementiert. Hierfür stellt Apple in der Entwicklungsumgebung XCode einen eigenen Interface Builder bereit [19]. In Abbildung 4.10 ist ein Screenshot des Interface Builders zu sehen. In der Hauptansicht des Builders lassen sich verschiedene Elemente, wie ein NavigationbarController oder eine TableView, per Drag and Drop erstellen. Die Erscheinung kann über die rechte Kontrollleiste bearbeitet werden. Um geräteübergreifend eine ansprechende Darstellung zu gewährleisten und ein responsives Design zu implementieren, können Beschränkungen an das Interface gestellt werden. Diese können abhängig von diversen Punkten eingestellt werden. Beispielsweise ob sich das Gerät im horizontalen oder vertikalen Modus befindet oder wie hoch oder breit der sichtbare Bildschirm ist. Zusätzlich lassen sich die Beschränkungen in Abhängigkeit von anderen Elementen einstellen. Durch diese beschränkungsabhängige Darstellung werden sowohl iPhones als auch iPads, ab 4 Zoll, unterstützt. Visuell unterscheiden sich die verschiedenen Geräte und Ausrichtungen in geeigneter Weise.

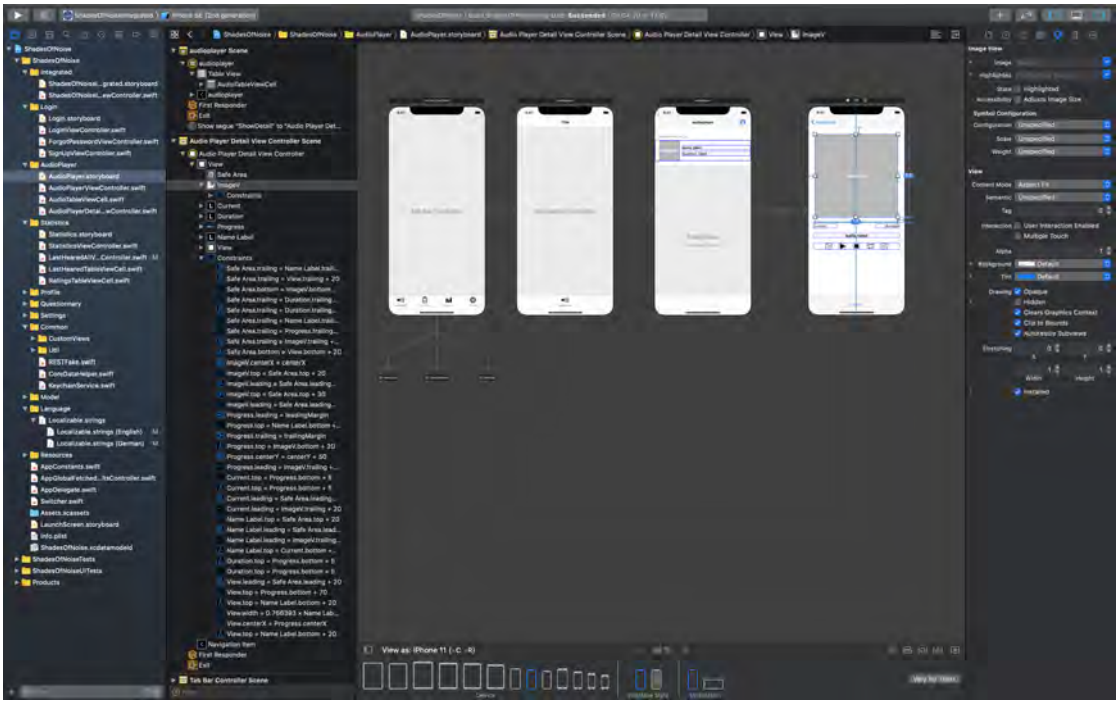

Abbildung 4.10: Interface Builder

5

Implementierung und technische Aspekte

Dieses Kapitel beschäftigt sich mit einigen Implementierungsdetails und technischen Aspekten der Entwicklung der *Shades Of Noise* Anwendung. Dabei wird insbesondere auf den Audio-Player eingegangen, da dieser die Hauptfunktion von *Shades Of Noise* darstellt.

5.1 Audio-Player

Zunächst beinhaltet der Audio-Player eine Standardinstanz des *AVAudioPlayer*[1]. Dieser wird beim Öffnen des Audio-Players, wie Zeile 130 bis Zeile 152 des Quelltextes A.1 zu entnehmen ist, mit der ausgewählten Audiodatei instanziiert. Um die Abspieldauer zu erfassen, werden die Variablen *timestampStart*, *timestampStop* und *timestampDifference* genutzt, siehe Zeile 22ff. Da die Wiedergabe pausiert werden kann und die Pause nicht mit in die Gesamtdauer einfließen soll, wird bei jedem Start des Abspielens der aktuelle Zeitstempel in *timestampStart* gespeichert. Bei Pausieren oder Stoppen der Wiedergabe wird der aktuelle Zeitstempel in *timestampStop* gespeichert. Durch die Differenz wird dann die Abspieldauer bestimmt. Bei erneuter Wiedergabe wird somit die Gesamtdauer in *timestampDifference* aufsummiert.

Für die Dauerschleife stehen dem Benutzer verschiedene Zeitintervalle zur Verfügung[2]. Wählt der Benutzer ein Zeitintervall aus, so muss die korrekte Anzahl an Schleifen und die restliche Abspieldauer in der letzten Schleifeniteration berechnet werden. Dies geschieht

[1] `https://developer.apple.com/documentation/avfoundation/avaudioplayer`, zuletzt besucht: 23. April 2020

[2] Stand 24. April 2020: 2 Minuten, 5 Minuten, 10 Minuten und endlos

in der Funktion *caluclateLoopsAndRemainder(minutes: Double)*, welche bei Zeile 205 im Quelltext beginnt. In der *if*-Bedingung wird überprüft, ob die restliche Laufzeit der Audiodatei, bestimmt durch die Gesamtdauer abzüglich des aktuellen Abspielpunktes, größer als die gewählte Schleifendauer ist. Ist dies der Fall, so wird lediglich der Abspielpunkt bestimmt an dem die Wiedergabe stoppen soll. Andernfalls wird zuerst von der Schleifenzeit der Rest der aktuellen Iteration abgezogen. Die restliche Schleifenzeit wird dann durch die Länge der Audiodatei geteilt um die Anzahl der Schleifeniterationen zu berechnen. Der Endzeitpunkt der Wiedergabe wird dann bestimmt durch die Schleifenzeit abzüglich der Länge der Audiodatei mal die Anzahl der Iterationen.

Für die Bewertung der kurzfristigen Verbesserung des Tinnitus wird eine *UIAlertView*[3] genutzt. Diese wird erstellt und angezeigt, wenn der Benutzer den Audio-Player verlässt und die Datei wiedergegeben hat. Diese Bedingung ist in Zeile 175 des Quellcodes zu sehen. Für die Bewertung steht eine Likert-Skala mit sieben Bewertungsstufen zur Verfügung [20]. Da die Skala eine ungerade Anzahl an Bewertungsmöglichkeiten bietet ist auch eine neutrale Eingabe möglich, welche sich in keiner Verbesserung und keiner Verschlechterung äußert.

5.2 Technische Aspekte

Der Quellcode von *Shades Of Noise* wird mit dem Versionsverwaltung *Git*[4] in einem eigenständig *Bitbucket*-Repository verwaltet. Um die Commits einheitlich zu halten werden *Conventional Commits* genutzt [21]. Dies ermöglicht eine sowohl für Menschen als auch für Maschinen interpretierbare Semantik für die Commits. Dadurch lassen sich beispielsweise Changelogs automatisch generieren. *Conventional Commits* haben eine vorgeschrieben Form, welche in Abbildung 5.1 beschrieben ist.

<type> beschreibt dabei den Typ des Commits. Dieser kann beispielsweise *feat* für das Implementieren von Funktionen oder *refactor* für Codeüberarbeitungen sein. Der *[optional scope]* beschreibt welcher Bereich der Anwendung geändert wird. Die Berei-

[3] `https://developer.apple.com/documentation/uikit/uialertview`, zuletzt besucht: 23. April 2020

[4] `https://git-scm.com/`, zuletzt besucht: 15. Mai 2020

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

Abbildung 5.1: Conventional Commits

che, welche für die Entwicklung von *Shades Of Noise* genutzt werden sind: *GUI* für Änderungen an den Views und teilweise an den ViewControllern, *Logic* für Änderungen, welche die generelle Logik der Anwendung betreffen und der Scope *Persistence*. Dieser wird für Änderungen an den Models oder der lokalen Datenbank genutzt. Scopeübergreifende Änderungen werden mit dem Scope *General* markiert. Die *<description>* gibt eine einzeilige Erklärung über die Änderungen des Commits und der *[optional body]* ist eine, falls nötig, ausführliche Beschreibung der Änderungen. Ein ausführliches Cheat Sheet zu *Conventional Commits* ist im Appendix unter Abbildung B.1 angehängt.

Das Vorgehen der Entwicklung an *Shades Of Noise* lässt sich als Mischung aus Wasserfallmodell und SCRUM beschreiben [22, Ch. 2-3]. Zu Beginn der Entwicklung wurde eine ausführliche Anforderungsanalyse durchgeführt. Diese lässt sich in das Wasserfallmodell einordnen. Nach der Anforderungsanalyse wurde allerdings auf eine inkrementelle Entwicklung umgelenkt, welche sich mehr dem SCRUM-Prozess zuordnen lässt. So wurde frühzeitig ein Prototyp als Entwurf entwickelt, um diesen dem Betreuer und den Verantwortlichen vorzustellen und dadurch Anpassungen an den Anforderungen zu entdecken. Die Besprechungen fanden im wöchentlichen bis zweiwöchentlichen Rhythmus statt.

5.3 Integrierte und Standalone Version

Zur Realisierung der zwei Versionen von *Shades Of Noise* werden sogennante *Build Schemes* und *Build Configurations* genutzt [23].
Ein *Build Scheme* beschreibt eine Blaupause für den Build-Prozess. Das *Build Scheme*

definiert, welche *Build Configurations* genutzt werden sollen um die Entwicklungs-, Test- und Produktionsbuilds für ein gegebenes *Target* zu erstellen.
Die *Build Configurations* spezifizieren eine Menge an Einstellungen, welche für beliebige *Targets* verwendet werden können.
Ein *Target* definiert eine Liste an Einstellungen für das Projekt. Darüberhinaus können einzelne Klassen und Ressourcen in einem Target inkludiert oder ausgeschlossen werden.

In *Shades Of Noise* sind zwei *Build Schemes* definiert: eines für die *Standalone* Version und eines für die *Integrated* Version. Um letztendlich zwei verschiedene Versionen für die verschiedenen *Build Schemes* zu erstellen, ist ein benutzerdefiniertes Flag nötig. Dies ist den Build Settings des Build Target gesetzt. Hierfür ist in der IDE XCode in den Projekteinstellungen das Flag *-DINTEGRATED* für die integrierte Version gesetzt. -D ist hierbei ein Parameter des Kommandozeilenbefehls, welcher ausgeführt wird um die Anwendung zu bauen. -D erlaubt das Übergeben von benutzerdefinierten Flags. Abbildung 5.2 zeigt diese Einstellung.

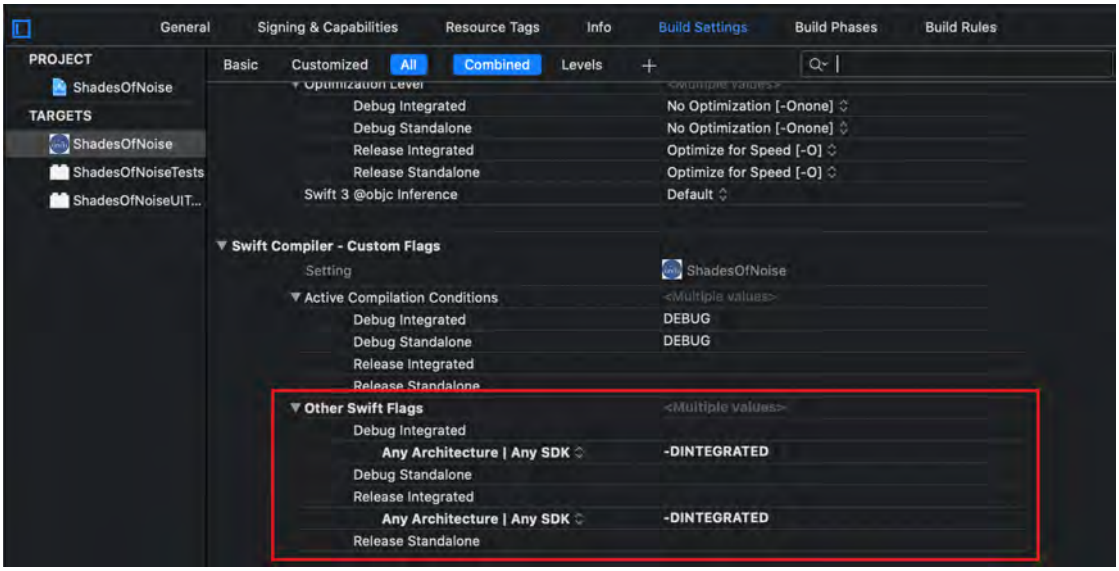

Abbildung 5.2: Userdefined Flag

Die benutzerdefinierten Flags können im Code genutzt werden, um festzustellen welche Version genutzt wird. In *Shades Of Noise* wird dafür das Custom Flag zu Kompilierzeit in einen konstanten Boolschen Wert übersetzt, wie Quellcode 5.1 zeigt. Durch den

gegebenen Boolschen Wert wird an wenigen[5] Stellen im Quelltext definiert, ob es sich um die integrierte Version oder die Standalone Version handelt und entsprechend eine andere Oberfläche dargestellt oder eine andere Logik angewandt.

```
1 // [..]
2 #if INTEGRATED
3 let IS_INTEGRATED_VERSION = true
4 #else
5 let IS_INTEGRATED_VERSION = false
6 #endif
7 // [..]
```

Listing 5.1: Übersetzen der Flag in einen Boolean

[5] Stand 25. April 2020: an drei Stellen

6

Vorstellung von Shades Of Noise

Ziel der App ist es, mithilfe von auditorischer Stimulation den Tinnitus kurzzeitig zu unterdrücken. Dazu stehen eine Vielzahl an Geräuschen zur Verfügung. Diese sind in verschiedene Kategorieren unterteilt. Eine Kategorie bildet zum Beispiel natürliche Geräusche ab, wohingegen eine andere Kategorie computererzeugte Töne sind.

Shades Of Noise wird in diesem Kapitel aus der Sicht des Benutzers dargestellt. Dabei werden die einzelnen Funktionen der Anwendung aufgezeigt. Die App unterstützt iOS ab Version 10.3, dies entspricht dem iPhone 5 und neueren Geräten.

Da die Standalone und die integrierte Version nicht alle Ansichten gemeinsam nutzen, folgen zunächst die Ansichten, welche beide Anwendungen verwenden, gefolgt von den versionsspezifischen Ansichten.

6.1 Übersicht Audiodateien

Die Hauptanwendung von *Shades Of Noise* bildet der Audio-Player. Hierfür ist zunächst eine Übersicht nötig, welche die Audiodateien enthält. Wie in Abbildung 6.1 zu sehen ist, sind die verschiedenen Dateien in Kategorien eingeteilt. Die Kategorien und Audiodateien sind von *UNITI* vorgegeben. Eine Kategorie entspricht beispielweise synthetisch erstellten Geräuschen, wohingegen eine andere Kategorie natürlichen Geräuschen wie Regen oder einem Wasserfall entspricht. Über den Pfeil, welcher sich bei den Kategorieüberschriften rechts befindet, lässt sich die Kategorie aus- und einklappen. Die einzelnen Geräusche werden mit einem passenden Bild, dem Titel und der Länge des

Geräusches in Tabellenform dargestellt. Klickt ein Benutzer auf ein Geräusch, so gelangt er zum Audio-Player.

6.2 Audio-Player

Der Audio-Player besteht aus dem Bild der Audiodatei, sowie einem Fortschrittsbalken mit einer Anzeige über den Sekundenfortschritt, sowie die Gesamtlänge der Datei. Der Titel der Audiodatei ist unter dem Fortschrittsbalken zu sehen. Darunter ist die Funktionsleiste.

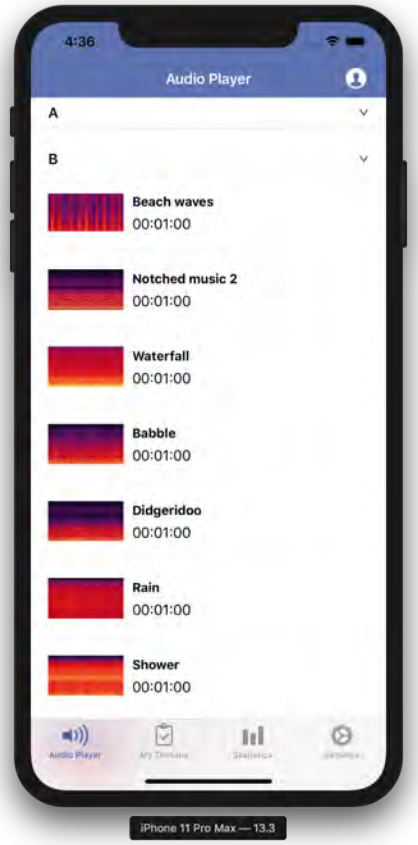

Abbildung 6.1: Übersicht

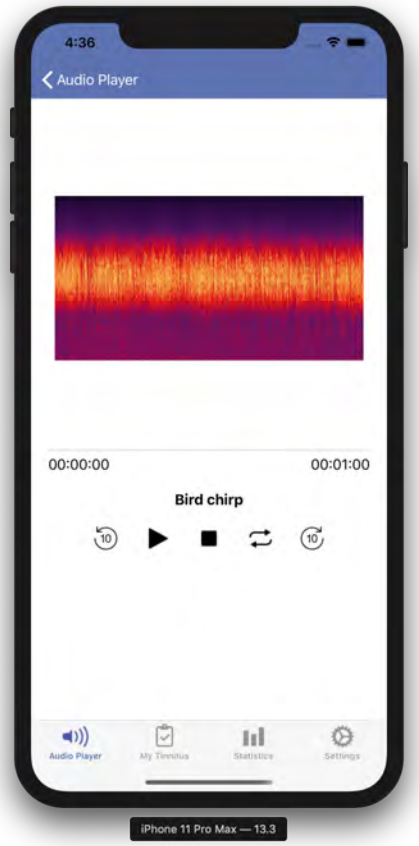

Abbildung 6.2: Audio-Player

Diese besteht aus einem *Zurück- und Vorspulen-Button* und einem *Play-Button*, welcher, wenn das Geräusch abgespielt, wird zu einem *Pause-Button* wird. Daneben ist

der *Stop-Button*: Dieser pausiert die Wiedergabe und setzt den Sekundenfortschritt zurück auf 00:00. Drückt der Benutzer den *Loop-Button*, so wird ihm eine Auswahl an Zeitintervallen angezeigt, zwischen denen er wählen kann. Die Wiedergabe der Datei wird dann, entsprechend der Auswahl, fortgesetzt. Wird das Ende der Audiodatei erreicht, so startet sie von vorne. Ist die Loop-Funktion aktiv, so ist der *Loop-Button* farbig hervorgehoben. Die Loop-Funktion lässt sich dann durch einen Klick auf den *Loop-Button* wieder deaktivieren. Verlässt der Benutzer den Audio-Player durch Klicken auf den Zurück-Button in der Navigationsleiste, so wird er aufgefordert eine Bewertung der kurzfristigen Verbesserung durch die Geräuschstimulation abzugeben. Hierfür steht eine Likert-Skala zur Verfügung, welche sieben gestaffelte Bewertungsmöglichkeiten anbietet. Diese zeigt Abbildung 6.4.

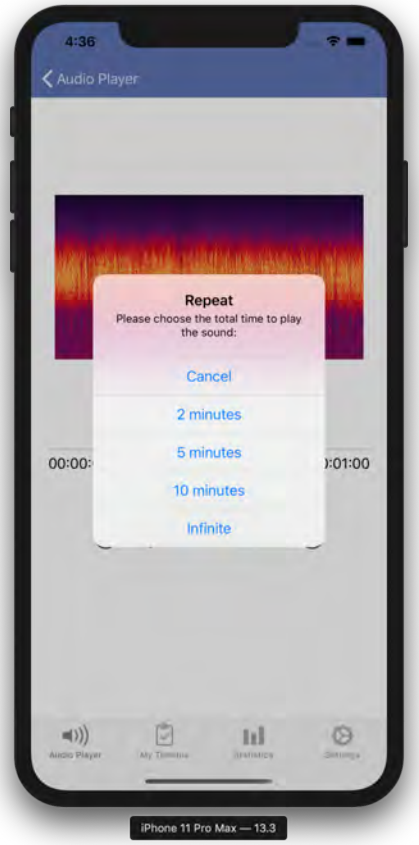

Abbildung 6.3: Loop Funktion

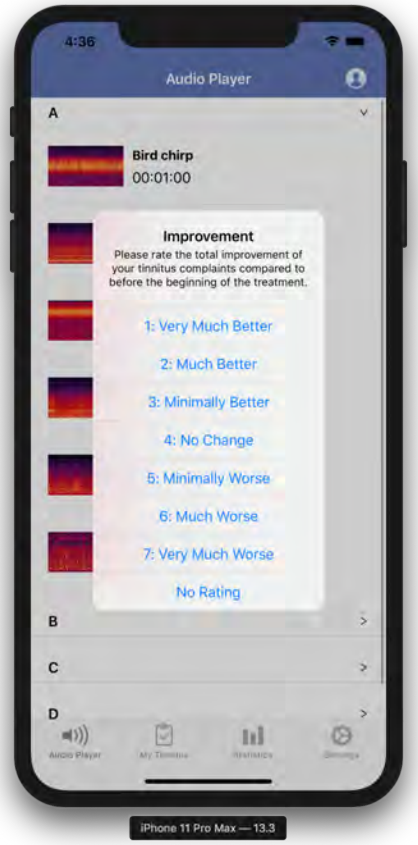

Abbildung 6.4: Bewertung

Zusätzlich kann der Benutzer die Bewertung überspringen. Hierfür steht die Option *No Rating*. Da *Shades Of Noise* auf die kurzfristige Tinnitusunterdrückung ausgelegt ist, verschwindet das Bewertungsfenster automatisch nach einem festgelegten Zeitraum[1].

6.3 Statistik

Durch Anhören und Bewerten der Audiodateien werden Statistiken gebildet. Diese sind aufgeteilt in zwei verschiedene Statistiken. Beide Statistiken werden im Tab *Statistics* angezeigt. Die erste Statistik bilden die zuletzt gehörten Audiodateien. In der gemeinsamen Übersicht, welche in Abbildung 6.5 gezeigt wird, sind die letzten fünf angehörten Geräusche abgebildet. Die zweite Statistik bilden die Bewertungen aller Audiodateien.

6.3.1 Zuletzt gehört

Ein Eintrag in der Statistik *Zuletzt gehört* wird aus dem Titel der Datei, der Anhördauer und dem Zeitstempel gebildet. Über *Show All* gelangt der Benutzer zu einer erweiterten Ansicht, welche alle und nicht nur die letzten fünf, angehörten Geräusche darstellt. Dies zeigt Abbildung 6.6. Über die Navigationsleiste kann der Benutzer von dort aus wieder zurück zur gemeinsamen Statistikübersicht gelangen.

6.3.2 Bewertungen

Die Statistik der Bewertungen listet alle Geräusche, standardmäßig nach Namen sortiert, auf. Wie in Abbildung 6.7 zu sehen ist, werden dabei die Namen der Audiodateien, die Anzahl an Bewertungen (*Countings*) und die berechnete durchschnittliche Bewertung (*Average*) angezeigt. Durch Drücken des in Abbildung 6.8 gezeigten Button kann der Benutzer die Bewertungen nach den dargestellten Werten sortieren, also dem Namen, der Anzahl der Bewertungen oder der durchschnittlichen Bewertung. Hierfür wählt der Benutzer über die Auswahlbox den gewünschten Wert aus.

[1] Stand 15. April 2020: 30 Sekunden

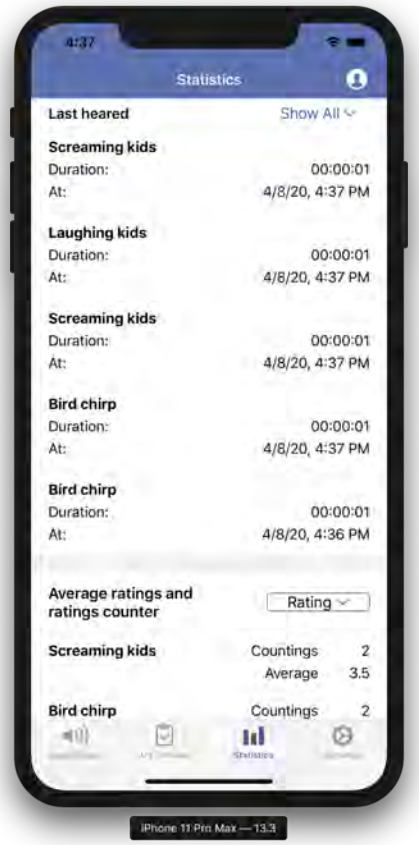

Abbildung 6.5: Statistik

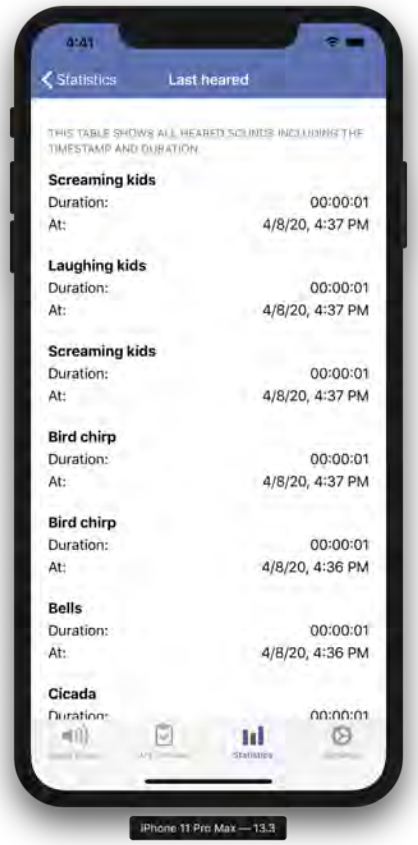

Abbildung 6.6: Zuletzt gehört

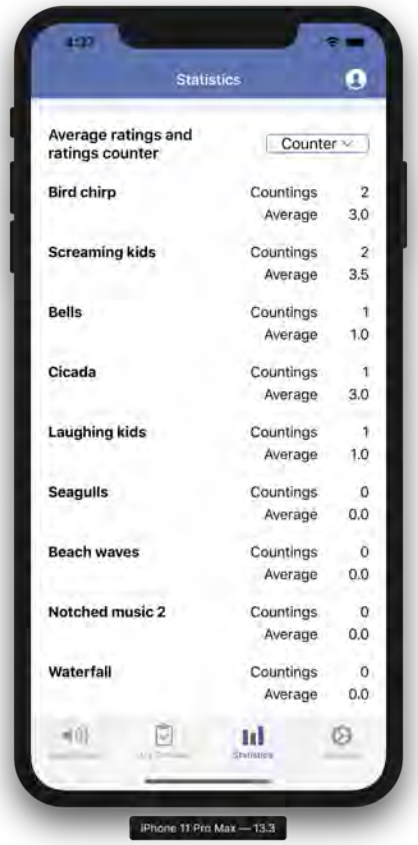

Abbildung 6.7: Bewertungen

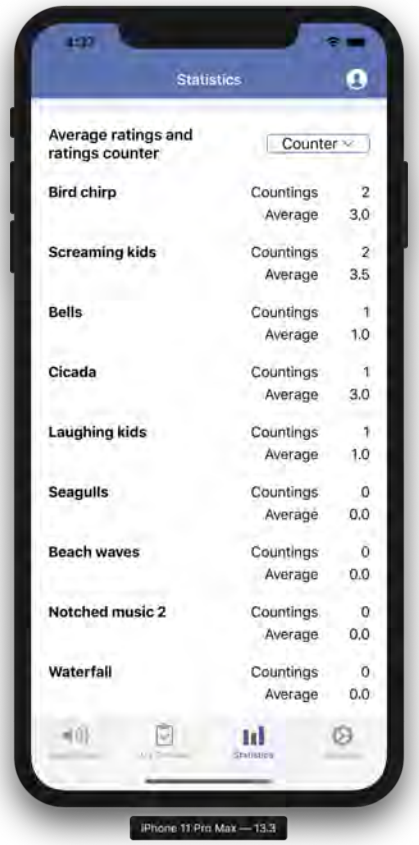

Abbildung 6.8: Sortierung

6.4 Standalone Version

Im Folgenden sind die für die Standalone Version spezifischen Interaktionsmöglichkeiten beschrieben.

6.4.1 Anmeldung mit Anmeldedaten

Nach dem Start der App ist die erste Ansicht der Login-Bildschirm, wie in Abbildung 6.9 zu sehen ist. Hier kann sich ein Benutzer mit vorhandenen Anmeldedaten, welche entweder aus Nutzername und Passwort oder aus E-Mail und Passwort bestehen, anmelden. Alternativ ist auch eine Nutzung ohne Anmeldedaten vorgesehen.

Ist der Nutzername oder das Passwort ungültig, so bekommt der Benutzer eine entsprechende Warnmeldung angezeigt. Nach erfolgreicher Anmeldung befindet sich der Benutzer in der Übersicht der Audiodateien, wie in Abschnitt 6.1 bereits beschrieben ist.

6.4.2 Passwort zurücksetzen

Hat ein Benutzer sein Passwort vergessen, so kann er, unter Angabe der E-Mail-Adresse, sein Passwort zurücksetzen lassen. Dies zeigt Abbildung 6.10. Hat der Benutzer bei der Anmeldung bereits die E-Mail als Eingabe gewählt und das Feld gefüllt, so ist das Feld bei der *Passwort Vergessen*-Ansicht vorgefüllt. Weitere Informationen zum Zurücksetzen erhält der Benutzer per E-Mail.

6.4.3 Registrierung

Hat ein Benutzer noch keine Anmeldedaten, so kann er sich direkt in der Standalone Version von *Shades Of Noise* registrieren. Hierfür muss er einen Nutzernamen, eine E-Mail-Adresse und ein Passwort eingeben. Die Felder werden auf Validität geprüft. Zum Beispiel muss die E-Mail-Adresse einem regulären Ausdruck genügen, welcher überprüft, ob es sich generell um eine gültige E-Mail Adresse handeln kann. Wie Abbildung 6.11 zeigt, muss das Passwort zudem bestätigt werden.

Abbildung 6.9: Login

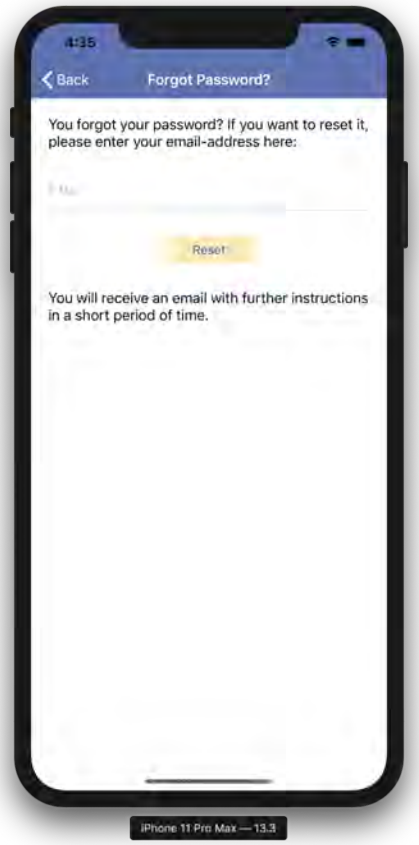

Abbildung 6.10: Passwort Vergessen

6.4.4 Fragebogen

Um einige demographische Daten zu erfassen und den Benutzer über die Charakteristik seines Tinnitus abzufragen, bietet die Standalone Version einen Fragebogen. Zur Eingabe stehen verschiedene Eingabefelder zur Verfügung. Abbildung 6.12 zeigt diese an einem Beispiel.

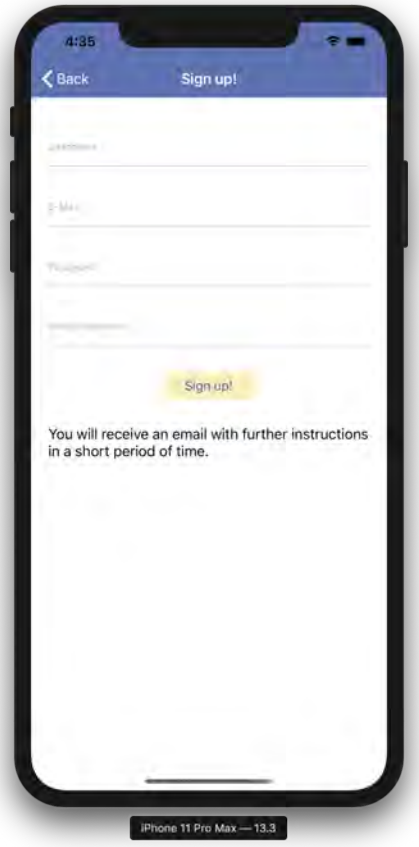

Abbildung 6.11: Registrieren

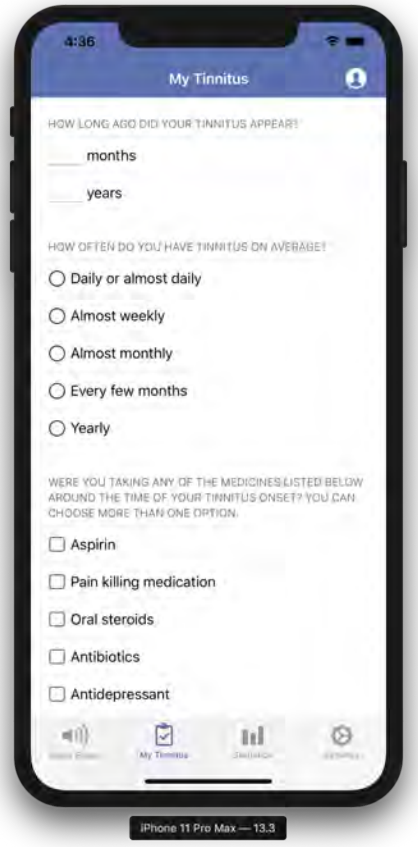

Abbildung 6.12: Fragebogen

Die Eingabemethoden sind Texteingabe und Single-Choice- sowie Multiple-Choice-Felder. Durch erneutes Klicken eines Single-Choice- oder Multiple-Choice-Feldes kann die Eingabe wieder zurückgenommen werden. Änderungen durch den Benutzer werden sofort in der lokalen Datenbank abgespeichert.

6.4.5 Profil

Über das Profil-Icon, welches sich rechts in der Navigationbar befindet, gelangt der Benutzer zur Profilübersicht. Wie Abbildung 6.13 zeigt, bekommt der Benutzer hier seinen Nutzername und seine E-Mail-Adresse angezeigt. Über den *Logout-Button* kann sich der Benutzer von *Shades Of Noise* ausloggen und befindet sich danach wieder in der Login-Ansicht, welche bereits in Abschnitt 6.4.1 vorgestellt wurde. Über das X links in der Navigationbar gelangt der Benutzer zurück zur zuletzt geöffneten Ansicht. Über den *Reload-Button* rechts in der Navigationbar kann der Benutzer sein Profil erneut laden. Möchte ein Benutzer sein Passwort ändern, so gelangt er über den *Change Password!-Button* zur Verwaltungsseite für das Passwort.

6.4.6 Passwort ändern

Um sein Passwort ändern zu können, muss der Benutzer sein aktuelles Passwort und ein neues Passwort eingeben. Das neue Passwort muss bestätigt werden, um zu verhindern, dass sich der Benutzer aus Versehen vertippt hat. Die Eingabe der Felder werden dementsprechend auf Gleichheit überprüft. Über den *Change Password-Button* ändert der Benutzer schließlich sein Passwort. Abbildung 6.14 zeigt diese Ansicht.

6.4.7 Einstellungen

In den Einstellungen findet der Benutzer unter anderem die Datenschutzerklärung, weitere Informationen über das Team und das Projekt, sowie das Impressum und eine Kontaktmöglichkeit. Abbildung 6.15 zeigt die Übersicht dazu.

Zudem kann der Benutzer in den Einstellungen die Sprache der Anwendung ändern. Hierfür stehen verschiedene Sprachen zur Auswahl[2]. Ändert der Benutzer die Sprache, so muss er die Anwendung neu starten. Die Anwendung erzwingt dies dadurch, dass die Navigations- und die Tableiste ausgeblendet werden, sobald der Benutzer die Sprache

[2]Stand 14.05.2020: Deutsch und Englisch

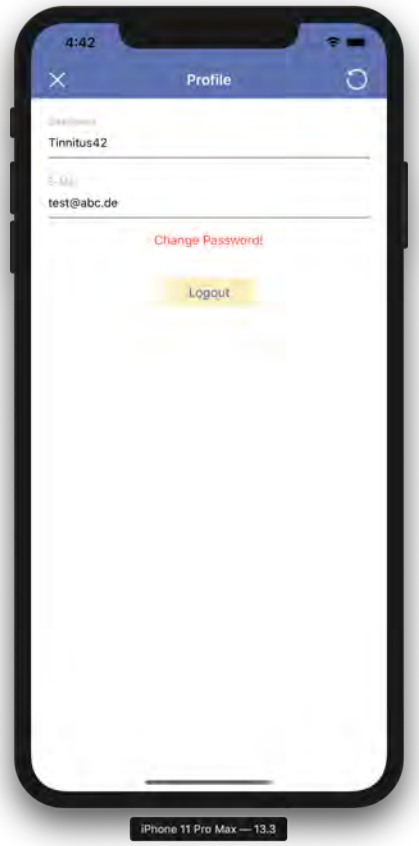

Abbildung 6.13: Profil

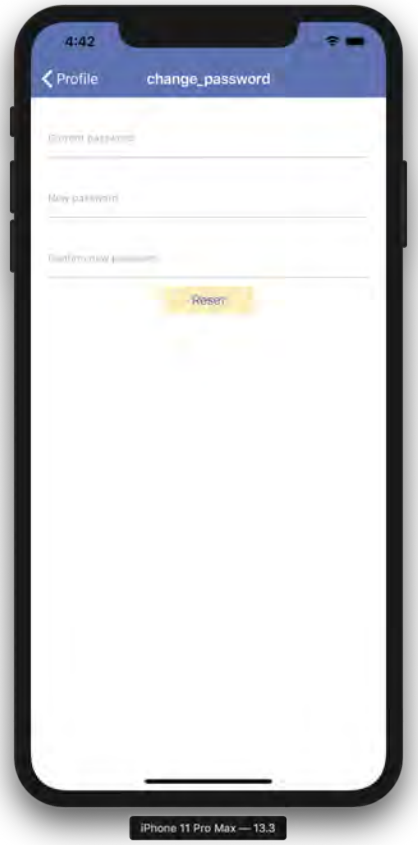

Abbildung 6.14: Passwort Ändern

ändert. Abbildung 6.16 zeigt die geänderte Sprache mit den ausgeblendeten Leisten und einem zusätzlichen Hinweis, dass die Anwendung neu gestartet werden muss.

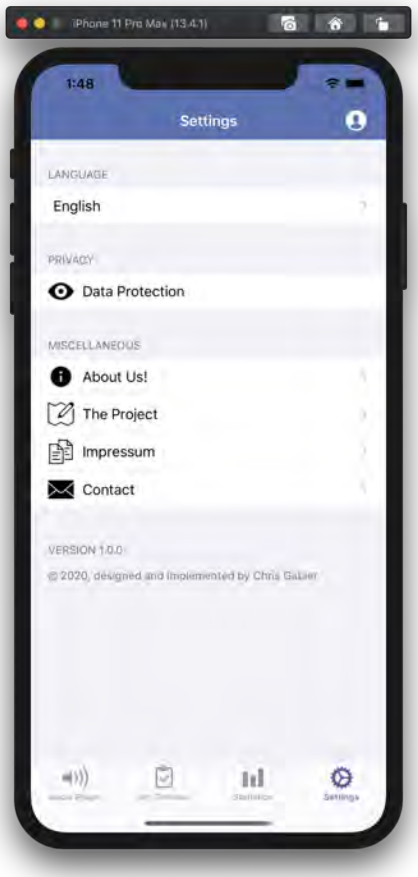

Abbildung 6.15: Einstellungen

Abbildung 6.16: Sprache Ändern

6.5 Integrierte Version

Da *Shades Of Noise* in die gemeinsame *UNITI*-App integriert werden soll, ist es nötig eine angepasste Version bereitzustellen, sodass sich Funktionen wie das Profil und ein Fragebogen nicht überschneiden.

6.5.1 Angepasste Hauptansicht

Hierfür wurde eine angepasste Hauptansicht implementiert, welche in Abbildung 6.17 zu sehen ist. Die von *Shades Of Noise* benötigten Funktionen sind der Audio-Player und die Statistikübersicht. Durch Tippen auf den entsprechenden Eintrag gelangt der Benutzer zu den bereits erläuterten Ansichten des Audio-Player bzw. der Statistikübersicht.

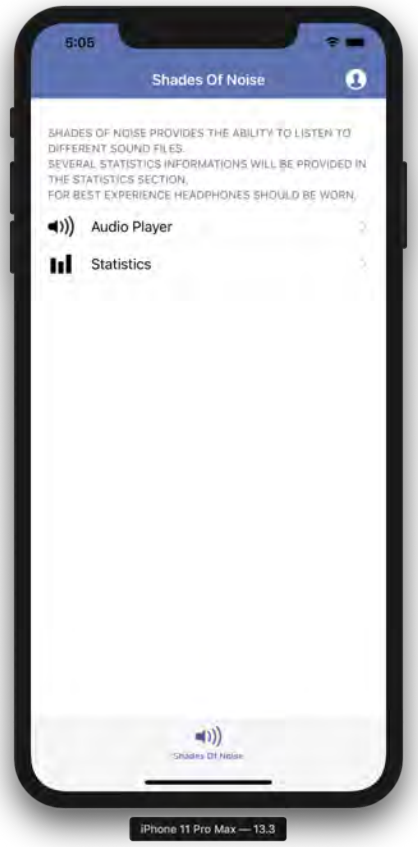

Abbildung 6.17: Integrierte Ansicht

7

Anforderungsabgleich

In diesem Kapitel werden die an die Anwendung gestellten Anforderungen aus Kapitel 3 mit den implementierten Funktionen der Anwendung abgeglichen. Die Anforderungen sind wie in Kapitel 3 in funktionale und nicht-funktionale Anforderungen gegliedert und die Reihenfolge der Anforderungen stimmt überein.

7.1 Funktionale Anforderungen

Der folgende Abschnitt gleicht die erhobenen, funktionalen Anforderungen mit dem momentanten Stand der Implementierung ab. Bei teilweiser Erfüllung oder Nichterfüllung ist jeweils eine Begründung mit aufgeführt.

Tabelle 7.1: Anforderungsabgleich funktionale Anforderungen

ID	**FA01**
Titel	**Ansicht Player**
Beschreibung	Anforderung erfüllt in Abschnitt 6.1

ID	**FA02**
Titel	**Kategorien anzeigen und durchstöbern**
Beschreibung	Anforderung erfüllt in Abschnitt 6.1

ID	**FA03**
Titel	**Geräusche anzeigen und durchstöbern**
Beschreibung	Anforderung erfüllt in Abschnitt 6.1

ID	**FA04**
Titel	**Audio-Player starten**
Beschreibung	Anforderung erfüllt in Abschnitt 6.1 und 6.2

ID	**FA05**
Titel	**Geräusch abspielen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.2

ID	**FA06**
Titel	**Geräusch pausieren**
Beschreibung	Anforderung erfüllt in Abschnitt 6.2

ID	**FA07**
Titel	**Geräusch stoppen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.2

ID	**FA08**
Titel	**Geräusch spulen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.2

ID	**FA09**
Titel	**Geräusch in Dauerschleife abspielen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.2

ID	FA10
Titel	**Bewertung Clinical Global Impression (CGI)**
Beschreibung	Anforderung erfüllt in Abschnitt 6.2

ID	FA11
Titel	**Ansicht Statistiken**
Beschreibung	Anforderung erfüllt in Abschnitt 6.3

ID	FA12
Titel	**Zuletzt gehört**
Beschreibung	Anforderung erfüllt in Abschnitt 6.3.1

ID	FA13
Titel	**Durchschnittliche Bewertung**
Beschreibung	Anforderung erfüllt in Abschnitt 6.3.2

ID	FA14
Titel	**Ansicht Fragebogen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.4

ID	FA15
Titel	**Fragebogen darstellen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.4

ID	FA16
Titel	**Antwort zurücksetzen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.4

ID	FA17
Titel	**Anzeige beantwortete Frage**
Beschreibung	Teilweise erfüllt. Es wird kein seperates Zeichen, wie ein Check-Haken oder ein grüner Hintergrund, eingeblendet. Da die Fragen aber in Tabellenform darliegen und nicht eingeklappt werden, ist durch einen kurzen Blick ebenfalls erkenntlich, ob die Frage bereits beantwortet ist oder nicht. Siehe Abschnitt 6.4.4

ID	FA18
Titel	**Antworten speichern**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.4. Die Anforderung wurde durch eine automatische Speicherung der Antworten implementiert.

ID	FA19
Titel	**Anzeige Vollständigkeit**
Beschreibung	Anforderung nicht erfüllt. Diese Anforderung ist eine optionale Anforderung, welche in fortlaufenden Projekten implementiert werden kann.

ID	FA20
Titel	**Einloggen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.1

ID	FA21
Titel	**Profil anzeigen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.5

ID	**FA22**
Titel	**Ausloggen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.5

ID	**FA23**
Titel	**Ansicht Einstellungen**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.7

ID	**FA24**
Titel	**Sprache ändern**
Beschreibung	Anforderung erfüllt in Abschnitt 6.4.7

ID	**FA25**
Titel	**Navigation**
Beschreibung	Anforderung erfüllt. Die Navigationsleiste wird beispielsweise in Abschnitt 6.2 dargestellt.

7.2 Nicht-funktionale Anforderungen

Der folgende Abschnitt bewertet die Erfüllung der erhobenen, nicht-funktionalen Anforderungen. Diese ist sowohl bei Erfüllung als auch bei Nichterfüllung begründet.

Tabelle 7.2: Anforderungsabgleich nicht-funktionale Anforderungen

ID	**NFA01**
Titel	**Modularität**
Beschreibung	Anforderung erfüllt. Siehe hierfür Abschnitt 6.4 und 6.5.

ID	**NFA02**
Titel	**Schnittstelle**
Beschreibung	Anforderung teilweise erfüllt. Diverse Codestellen sind implementiert und Anwendungsflüsse definiert. Auf vollständige Erfüllung kann erst geprüft weden, wenn das Framework bereitgestellt wurde und alle nötigen Schnittstellen implementiert sind.

ID	**NFA03**
Titel	**Multilingual**
Beschreibung	Anforderung erfüllt. Über Swift-eigene .strings-Dateien lassen sich zusätzliche Sprachen einbinden. Aktuell[1] sind Deutsch und Englisch implementiert.

[1] Stand 28. April 2020

ID	**NFA04**
Titel	**Erweiterbarkeit**
Beschreibung	Anforderung erfüllt. Sowohl Geräusche als auch Fragen werden über einfache JSON-Dateien eingelesen. Sprachen können über .strings-Dateien hinzugefügt werden. Erweiterbarkeit über die REST-Schnittstelle kann geprüft werden, sobald das entsprechende Framework zur Verfügung steht.

ID	**NFA05**
Titel	**Alternative Oberfläche**
Beschreibung	Anforderung erfüllt. Siehe hierfür Abschnitt 6.4 und 6.5 und 5.3.

8

Fazit

Die erste Phase der Entwicklung von *Shades Of Noise* ist mit dieser Arbeit abgeschlossen. Die definierten Anforderungen wurden überwiegend in der Anwendung umgesetzt. In Abschnitt 8.1 wird die Erkenntnis dieser Arbeit geschildert. Abschnitt 8.2 behandelt einen Ausblick über die weiteren Entwicklungsphasen und anstehende Implementierungen.

8.1 Zusammenfassung

Ziel dieser Arbeit war die Konzeption und Realisierung einer mobilen Anwendung zur Unterstützung von Tinnituspatienten mithilfe gezielter auditorischer Stimulation.

Zunächst wurde dazu in die Thematik und Relevanz der Erkrankung eingeführt und darauf folgend die Anforderungen an die Anwendung erhoben. Diese wurden in Abstimmung mit dem Betreuer zunächst initial ausführlich definiert. Während der Projektlaufzeit wurden diese in regelmäßigen Besprechungen, sowohl mit dem Betreuer als auch mit den Verantwortlichen, angepasst und erweitert. Anhand der gegebenen Anforderungen wurde ein Architekturkonzept erarbeitet, welches auch den nicht-funktionalen Anforderungen entspricht. Auf einige Prozess- und Implementierungsdetails wurde weiter eingegangen. Als eine Art Benutzerdokumentation wurde die Funktionalität der Anwendung aus Sicht des Benutzers beschrieben. Als Abschluss wurde die umgesetzte Funktionalität mit den erhobenen Anforderungen verglichen.

8.2 Ausblick

Dieser Abschnitt beschreibt geplante Erweiterungen der *Shades Of Noise* App. Diese sind teilweise schon zu Beginn der Arbeit und teilweise während des Projektverlaufes aufgekommen. Ziel ist es, aus der prototypischen Anwendung eine voll funktionsfähige Anwendung zu entwickeln. Hierfür sind unter anderem die Anbindung an das Backend und die Integration in die UNITI App nötig. Weitere Verbesserungen können beispielsweise eine Favoritenliste oder die Beachtung der persönlichen Tinnitusfrequenz sein.

8.2.1 Anbindung REST Api

Im Rahmen dieser Bachelorarbeit wurde die Anbindung an das Backend ausgenommen. Um die Datensätze für Forschungszwecke nutzen zu können, erfolgt in einem weiteren Projekt die Anbindung an das Backend. Hierfür wird die bereits bestehende TrackYourHealth Struktur[1] verwendet. Die Anbindung soll mit Swift-nativen Methoden und Klassen umgesetzt werden, um die möglichst geringe Abhängigkeit an Drittanbietersoftware beizubehalten.

8.2.2 Integration in UNITI App

Für den UNITI Projektrahmen und die darin erwähnte Studie müssen bereits vorhandene Anwendungen in eine gemeinsame Anwendung überführt werden. Hierzu zählt unter anderem die *Shades Of Noise* Anwendung. Vorbereitend für die Harmonisierung und Integration in die Gesamtanwendung wurde bei der Implementierung bereits auf die Modularität geachtet und die integrierte und Standalone Version bereitgestellt. Die Integration ist voraussichtlich für Herbst 2020 geplant. Hierfür wird die integrierte Version von *Shades Of Noise* genutzt.

[1] `https://api.dummy.trackyourhealth.net/dingodocs/v1.html`, zuletzt besucht: 16. Mai 2020

Literaturverzeichnis

[1] Hoang, J.K., Loevner, L.A.: Evaluation of Tinnitus and Hearing Loss in the Adult. Diseases of the Brain, Head and Neck, Spine 2020–2023: Diagnostic Imaging. Springer International Publishing (2020) 193–201

[2] Neff, P., Michels, J., Meyer, M., Schecklmann, M., Langguth, B., Schlee, W.: 10 Hz Amplitude Modulated Sounds Induce Short-Term Tinnitus Suppression. Frontiers in Aging Neuroscience **9** (2017)

[3] Neff, P., Zielonka, L., Meyer, M., Langguth, B., Schecklmann, M., Schlee, W.: Comparison of Amplitude Modulated Sounds and Pure Tones at the Tinnitus Frequency: Residual Tinnitus Suppression and Stimulus Evaluation. Trends in Hearing **23** (2019)

[4] Feldmann, H.: Homolateral and Contralateral Masking of Tinnitus by Noise-Bands and by Pure Tones. Audiology **10** (1971) 138–144

[5] Schoisswohl, S., Arnds, J., Schecklmann, M., Langguth, B., Schlee, W., Neff, P.: Amplitude Modulated Noise for Tinnitus Suppression in Tonal and Noise-Like Tinnitus. Audiol Neurotol **24** (2019) 309–321

[6] Herrmann, J.: Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten. Diploma thesis, Ulm University (2014)

[7] Stampf, A.: Konzeption und Realisierung einer mobilen Anwendung zur Bestimmung der Tinnitusfrequenz am Beispiel von Android. Bachelor thesis, Ulm University (2018)

[8] Vogel, C.: Conception and realization of a mobile data acquisition and assistance application for intersession processes of patients in psychotherapeutic treatments at the example of the ios platform. Master thesis, Ulm University (2019)

[9] Gablonski, T., Pryss, R., Probst, T., Vogel, C., Andreas, S.: Intersession-Online: A Smartphone Application for Systematic Recording and Controlling of Intersession Experiences in Psychotherapy. J **2** (2019) 480–495

[10] Stach, M., Vogel, C., Gablonski, T., Andreas, S., Probst, T., Reichert, M., Schickler, M., Pryss, R.: Technical Challenges of a Mobile Application Supporting Intersession Processes in Psychotherapy. (2020) accepted for publication

[11] Mehdi, M., Riha, C., Neff, P., Dode, A., Pryss, R., Schlee, W., Reichert, M., Hauck, F.J.: Smartphone Apps in the Context of Tinnitus: Systematic Review. Sensors **1725** (2020) 20

[12] Jun, H.J., Park, M.K.: Cognitive Behavioral Therapy for Tinnitus: Evidence and Efficacy. Korean Journal of Audiology **17** (2013) 101–104

[13] O'Rourke, T., Vogel, C., John, D., Pryss, R., Schobel, J., Haug, F., Haug, J., Pieh, C., Nater, U.M., Feneberg, A.C., Reichert, M., Probst, T.: The Impact of Coping Strategies on Situational Coping in Daily Life: An Ecological Momentary Assessment Study with the mHealth application TrackYourStress. JMIR Preprints **1** (2020)

[14] Adjamian, P.: The application of electro- and magneto-encephalography in tinnitus research – methods and interpretations. Frontiers in Neurology **5** (2014) 228

[15] Kraft, R., Schlee, W., Stach, M., Reichert, M., Langguth, B., Baumeister, H., Probst, T., Hannemann, R., Pryss, R.: Combining mobile crowdsensing and ecological momentary assessments in the healthcare domain. Frontiers in Neuroscience **14** (2020) 164

[16] Apple Inc.: Storyboard [online]. `https://developer.apple.com/library/archive/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html` (2018) Zuletzt abgerufen: 2020-04-16.

[17] Apple Inc.: Core data [online]. `https://developer.apple.com/documentation/coredata` (2020) Zuletzt abgerufen: 2020-04-22.

[18] Apple Inc.: Persistent store types and behaviors [online]. https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CoreData/PersistentStoreFeatures.html (2018) Zuletzt abgerufen: 2020-04-22.

[19] Apple Inc.: Interface builder built-in [online]. https://developer.apple.com/xcode/interface-builder/ (2020) Zuletzt abgerufen: 2020-04-23.

[20] Bertram, D.: Likert scales. Retrieved November **2** (2007) 10

[21] Netlify: Conventional commits [online]. https://www.conventionalcommits.org/en/v1.0.0-beta.4/ (2020) Zuletzt abgerufen: 2020-04-23.

[22] Sommerville, I.: Software Engineering. 9 edn. Pearson Studium, München (2012)

[23] Ted Bendixson: Some practical uses for xcode build schemes and build configurations (swift) [online]. https://medium.com/practical-ios-development/some-practical-uses-for-xcode-build-schemes-and-build\-configurations-swift-e50d15a1304f (2016) Zuletzt abgerufen: 2020-04-23.

[24] albelop: Conventional commits cheat sheet [online]. https://cheatography.com/albelop/cheat-sheets/conventional-commits/ (2018) Zuletzt abgerufen: 2020-04-23.

A

Quelltexte

In diesem Anhang sind einige wichtige Quelltexte aufgeführt.

```
1  //
2  //  AudioPlayerDetailViewController.swift
3  //  ShadesOfNoise
4  //
5  //  Created by Chris on 06.01.20.
6  //  Copyright 2020 Uni Ulm DBIS. All rights reserved.
7  //
8
9  import UIKit
10 import AVFoundation
11
12 var audioPlayer: AVAudioPlayer = AVAudioPlayer()
13
14 /*
15 This View Controller belongs to the Audio Player Page of the
16 standalone and the integrated version of Shades Of Noise.
17 */
18 class AudioPlayerDetailViewController: UIViewController,
19                 AVAudioPlayerDelegate {
20     // MARK: - Class variables
21         // [..]
```

```
22      var timestampStart: Int64 = 0
23          // Used to set timestamps,
24          // when player is starting to play
25      var timestampStop: Int64 = 0
26          // Used to set timestamps, when player is stopping
27      var timestampDifference: Int64 = 0
28          // Used to calculate the difference
29          // between two timestamps
30
31      // [..]
32
33      // MARK: - IBActions
34      // Switching the image and function of the play
35      // / pause button,
36      // depending if the audioPlayer is currently playing
37      @IBAction func playTapped(_ sender: Any) {
38          if(audioPlayer.isPlaying) {
39              pause()
40          } else {
41              audioPlayer.play()
42              timestampStart = Date.currentTimeStamp
43              play.setImage(UIImage(named: "icons8-pause-30"),
44                  for: .normal)
45          }
46      }
47
48      // If stop is tapped, the audioPlayer should switch
49      // to the play button and reset current time
50      @IBAction func stopTapped(_ sender: Any) {
51          if audioPlayer.isPlaying {
52              pause()
```

```
53          }
54          audioPlayer.pause()
55          audioPlayer.currentTime = 0
56      }
57
58      // Looping the amount of time the user
59      // has put in using the alert
60      @IBAction func repeatTapped(_ sender: Any) {
61          alert(title: "SON_repeat".localized,
62          message: "SON_repeat_detail".localized,
63          buttons: repeatData, handler: self.handlerRepeat,
64          vc: self)
65      }
66
67      // Sets currentTime to currentTime+10,
68      // does not change the current state
69      // regarding playing
70      @IBAction func forwardTapped(_ sender: Any) {
71          audioPlayer.currentTime += 10
72      }
73      // Sets currentTime to max(currentTime+10, 0),
74      // does not change the current state
75      // regarding playing
76      @IBAction func replayTapped(_ sender: Any) {
77          audioPlayer.currentTime -= 10
78      }
79
80      /*
81       Updating the view frequently to display
82       // a smooth progressbar and currentTime
83       */
```

```
84      func updateView(){
85          timer = Timer.scheduledTimer(withTimeInterval: 0.01,
86           repeats: true) {_ in
87              // Updates the view according to the current player
88              self.current.text =
89              timeFormatter.string(from: audioPlayer.currentTime)
90              self.progress.progress =
91              Float(audioPlayer.currentTime/audioPlayer.duration)
92              if !audioPlayer.isPlaying {
93                  self.play.setImage(
94                          UIImage(named: "icons8-play-30"),
95                          for: .normal)
96              }
97
98              // When the end of the loop is reached,
99              // the player should take care of the remainder,
100             // when remainder is reached, the sound is paused
101             // and some variables are resetted
102             if self.numberOfLoops == 0 {
103                 if self.remainderOfLoops >= 0 &&
104                 audioPlayer.currentTime > self.remainderOfLoops{
105                     self.repeatB.setImage(
106                         UIImage(named: "icons8-repeat-30"),
107                         for: .normal)
108                     self.pause()
109                     self.remainderOfLoops = -1
110                     self.numberOfLoops = 0
111                 }
112             }
113         }
114     }
```

```
115
116     /*
117      Pauses the audioPlayer, gets a timestamp
118      and adds the timestampDifference.
119      Sets the image of the play-button to "play"
120      */
121     func pause() {
122         audioPlayer.pause()
123         timestampStop = Date.currentTimeStamp
124         timestampDifference += timestampStop - timestampStart
125         // Add differnce of last play period to differnce
126         play.setImage(UIImage(named: "icons8-play-30"),
127                 for: .normal)
128     }
129
130     // MARK: - Initialization
131     override func viewDidLoad() {
132         super.viewDidLoad()
133
134         // load audioPlayer and prepare to play
135         guard
136             let path = Bundle.main.path(
137                         forResource: audio.audioName,
138                         ofType: "wav")
139             else {
140                 fatalError("Couldn't load audio
141                 \(audio.audioName ?? "") from main bundle.")
142         }
143
144         do {
145             audioPlayer = try AVAudioPlayer(
```

```
146                 contentsOf: URL(fileURLWithPath: path))
147         } catch{
148             fatalError("Couldn't load audio
149             \(audio.audioName ?? "") from main bundle.")
150         }
151         audioPlayer.prepareToPlay()
152         audioPlayer.delegate = self
153
154
155         // set labels and images according to the loaded file
156         nameLabel.text = audio.name
157         imageV.image = UIImage(data: audio.imageFile!)
158         current.text =
159         timeFormatter.string(from: audioPlayer.currentTime)
160         duration.text =
161         timeFormatter.string(from: audioPlayer.duration)
162         progress.progress =
163         Float(audioPlayer.currentTime/audioPlayer.duration)
164     }
165
166     // [..]
167
168     /*
169      When the detailView will dismiss the cgi-rating alert
170      will appear and the AudioPlayer will stop playing,
171      furthermore a entry to lastHeared will be added.
172      */
173     override func willMove(toParent parent: UIViewController?) {
174         super.willMove(toParent: parent)
175         if parent == nil && (audioPlayer.isPlaying ||
176         timestampDifference > 0) {
```

```
177             if audioPlayer.isPlaying {
178                 timestampStop = Date.currentTimeStamp
179                 timestampDifference +=
180                         timestampStop - timestampStart
181                         // Add differnce of last play
182                         // period to differnce
183             }
184             audioPlayer.stop()
185 
186             alert(title: "SON_improvement".localized, message:
187              "SON_improvement_detail".localized, buttons:
188                 ["SON_1_very_much_better".localized,
189                  "SON_2_much_better".localized,
190                  "SON_3_minimally_better".localized,
191                  "SON_4_no_change".localized,
192                  "SON_5_minimally_worse".localized,
193                  "SON_6_much_worse".localized,
194                  "SON_7_very_much_worse".localized,
195                  "SON_8_no_rating".localized],
196                   handler: self.handlerRating,
197                   vc: self,
198                   cgiRating: true)
199         }
200 
201     }
202 
203     // [..]
204 
205     func calculateLoopsAndRemainder(minutes: Double) {
206         var addOne = false
207         var time = minutes * 60
```

```
208
209         if time
210         <= audioPlayer.duration - audioPlayer.currentTime {
211             numberOfLoops = 0
212             remainderOfLoops = audioPlayer.currentTime + time
213         } else {
214             time -=
215             (audioPlayer.duration - audioPlayer.currentTime)
216             addOne = true
217             numberOfLoops =
218             (time / audioPlayer.duration).rounded(.down)
219             remainderOfLoops =
220             time - (numberOfLoops * audioPlayer.duration)
221             if addOne {
222                 numberOfLoops += 1
223             }
224         }
225     }
226
227     // [..]
228 }
```

Listing A.1: Audio-Player Detail Ansicht View Controller

B

Appendix

In diesem Anhang sind einige weiterführende Informationen aufgeführt.

Cheatography

Conventional Commits Cheat Sheet

by albelop via cheatography.com/39450/cs/15616/

Types		
`feat`	Features	A new feature
`fix`	Bug Fixes	A bug fix
`docs`	Documentation	Documentation only changes
`style`	Styles	Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)
`refactor`	Code Refactoring	A code change that neither fixes a bug nor adds a feature
`perf`	Performance Improvements	A code change that improves performance
`test`	Tests	Adding missing tests or correcting existing tests
`build`	Builds	Changes that affect the build system or external dependencies (example scopes: gulp, broccoli, npm)
`ci`	Continuous Integrations	Changes to our CI configuration files and scripts (example scopes: Travis, Circle, BrowserStack, SauceLabs)
`chore`	Chores	Other changes that don't modify src or test files
`revert`	Reverts	Reverts a previous commit

Commit message structure

<type>[optional scope]**: <description>**
[optional body]
[optional footer]

A commit that has the text `BREAKING CHANGE:` at the beginning of its optional body or footer section introduces a breaking API change

Specification

1. Commits MUST be prefixed with a type, which consists of a verb, `feat`, `fix`, etc., followed by a colon and a space.

2. The type `feat` MUST be used when a commit adds a new feature to your application or library.

3. The type `fix` MUST be used when a commit represents a bug fix for your application.

4. An optional scope MAY be provided after a type. A scope is a phrase describing a section of the codebase enclosed in parenthesis, e.g., `fix(parser):`

5. A description MUST immediately follow the type/scope prefix. The description is a short description of the pull request, e.g., `fix: array parsing issue when multiple spaces were contained in string.`

6. A longer commit body MAY be provided after the short description. The body MUST begin one blank line after the description.

7. A footer MAY be provided one blank line after the body. The footer SHOULD contain additional meta-information about the pull-request (such as the issues it fixes, e.g., `fixes #13, #5`).

8. Breaking changes MUST be indicated at the very beginning of the footer or body section of a commit. A breaking change MUST consist of the uppercase text `BREAKING CHANGE`, followed by a colon and a space.

9. A description MUST be provided after the `BREAKING CHANGE: `, describing what has changed about the API, e.g., `BREAKING CHANGE: environment variables now take precedence over config files.`

10. Types other than `feat` and `fix` MAY be used in your commit messages.

By **albelop**
cheatography.com/albelop/

Published 3rd May, 2018.
Last updated 3rd May, 2018.
Page 1 of 1.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Abbildung B.1: Conventional Commits [24]

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Chris Gabler Matrikelnummer: 20 00 43 3

Erklärung

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

Ulm, den 16 Juni 2020, Chris Gabler

Chris Gabler