ulm university universität ulm

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

Masterarbeit
im Studiengang Informatik

Konzeption einer modernen Web-Application zur Verwaltung von Dynamischen Mobile Crowdsensing Plattformen im Healthcare Bereich

vorgelegt von

Pascal Damasch

Oktober 2019

1. Gutachter	Prof. Dr. Manfred Reichert
2. Gutachter	Prof. Dr. Thomas Probst
Betreuer:	Dr. Rüdiger Pryss
Matrikelnummer	833251
Arbeit vorgelegt am:	29.10.2019

Kurzfassung

Steigender Stress und Leistungsdruck in der Schule oder später im Berufsleben können unter anderem zu psychischen Erkrankungen, Tinnitus oder Bluthochdruck führen. Viele Betroffene finden keine Zeit mehr für sich selber um den angesammelten Stress und Leistungsdruck abbauen zu können. Durchgeführte Studien belegen Zusammenhänge zwischen Stress und Tinnitus und zeigen die Auswirkungen auf die Betroffenen.
Dieses Projekt wurde gegründet, um noch mehr spezifische Daten zu erhalten, diese zu analysieren und um den Betroffenen anschließend direkt persönliche Informationen hierzu bereitstellen zu können. Die Basis hierfür bildet eine Vielzahl an bereits vorhandenen Studien, sowie die darin enthaltenen Fragebögen, welche intervallabhängig wiederholt werden können. Die Benutzer der Webapplikation können bequem von zu Hause aus die Fragebögen beantworten und erhalten anschließend individuelle, auf den Benutzer angepasste Informationen. Dabei werden die Informationen graphisch dargestellt um eine schnelle und übersichtliche Anschauung zu ermöglichen und um Veränderungen direkt erkennen zu können.

Danksagung

Zunächst möchte ich mich bei *Prof. Dr. Manfred Reichert* und *Prof. Dr. Thomas Probst* für die Begutachtung dieser Arbeit bedanken.

Ein besonderer Dank gilt meinem Betreuer *Dr. Rüdiger Pryss* für seine freundliche Betreuung und tatkräftige Unterstützung während der gesamten Arbeit.

Ebenfalls möchte ich mich bei Johannes Schobel für seine hilfsbereite und freundliche Unterstützung, sowie der bereitstellung der API bedanken.

Außerdem möchte ich mich bei meiner Schwester *Fabienne* für das Korrekturlesen dieser Arbeit bedanken.

Abschließend will ich auch noch meinen *Eltern* dafür danken, dass sie mich auch während meinem Master noch immer tatkräftig unterstützt haben.

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 29.10.2019 Pascal Damasch

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Zielsetzung . . . 1
- 1.2 Aufbau der Arbeit . . . 1

2 Hintergrundinformationen **5**
- 2.1 Tinnitus . . . 5
- 2.2 Stress . . . 6
- 2.3 Verwandte Arbeiten . . . 7
  - 2.3.1 Track Your Tinnitus . . . 7
  - 2.3.2 Track Your Stress . . . 7
- 2.4 Projekt . . . 9

3 Anforderungsanalyse **11**
- 3.1 Funktionale Anforderungen . . . 11
- 3.2 Nicht-funktionale Anforderungen . . . 13

4 Architektur **15**
- 4.1 Frontend . . . 15
  - 4.1.1 Angular . . . 15
  - 4.1.2 Node.js . . . 16
  - 4.1.3 TypeScript . . . 16
  - 4.1.4 Visual Studio Code . . . 17
- 4.2 Backend . . . 17
  - 4.2.1 API . . . 17
  - 4.2.2 MariaDB . . . 18
- 4.3 Versionsverwaltung . . . 19

5 Implementierungsverlauf **21**

6 Verwendete API Funktionen **25**
- 6.1 Authentifizierung . . . 25
- 6.2 Benutzer . . . 26
- 6.3 Editor . . . 29
- 6.4 Fragebögen / Antwortbögen . . . 29
- 6.5 Nachrichten . . . 31
- 6.6 Notizen . . . 32
- 6.7 Studien . . . 34
- 6.8 Verschiedenes . . . 35

7 Ausgewählte Implementierungsaspekte **37**
- 7.1 HTML . . . 37
  - 7.1.1 Ergebnisse Benutzer . . . 37
  - 7.1.2 Fragebogen anzeigen . . . 39

7.2 TypeScript . . . 44
7.2.1 Aufruf Dialog . . . 44
7.2.2 Aufruf API-Funktion . . . 45
7.2.3 Fragebogen Klasse . . . 46
7.2.4 Graph erstellen . . . 47
7.2.5 Trigger: Wechsel zwischen TrackYourTinnitus2 und TrackYourStress . . . 48
7.3 API . . . 48
7.3.1 GET . . . 48
7.3.2 POST . . . 49
7.3.3 PATCH . . . 49
7.3.4 DELETE . . . 50
7.4 Sprachen . . . 50
7.5 Deployment . . . 51

8 Walkthrough **53**
8.1 Nicht registrierter Benutzer . . . 53
8.2 Registrierter Benutzer . . . 54
8.3 Editor . . . 61

9 Abgleich der Anforderungen **69**
9.1 Abgleich funktionale Anforderungen . . . 69
9.2 Abgleich nicht-funktionale Anforderungen . . . 71

10 Zusammenfassung & Ausblick **73**
10.1 Zusammenfassung . . . 73
10.2 Ausblick . . . 74

Abbildungsverzeichnis

1.1 Aufbau der Arbeit 3

2.1 Track your Tinnitus APP [7] 8

4.1 Architektur mit Frontend und Backend 15
4.2 Rückgabe API auf Anfrage aller Antwortbögen eines Fragebogens im JSON-Format 18
4.3 Sourcetree Oberfläche 19

7.1 Pop-up-Fenster 45
7.2 Dateien Webserver 52

8.1 Navigation nicht registrierter Benutzer 54
8.2 Liniendiagramm Ergebnisseite registrierter Benutzer 55
8.3 Balkendiagramm Ergebnisseite registrierter Benutzer 55
8.4 Ausschnitt Oberfläche Fragebogen 56
8.5 Oberfläche meine Nachrichten 57
8.6 Oberfläche Nachrichtenverlauf unter meine Nachrichten 57
8.7 Oberfläche meine Notizen 58
8.8 Oberfläche meine Statistiken 59
8.9 Navigation registrierter Benutzer 60
8.10 Oberfläche Statistiken 62
8.11 Oberfläche Studie erstellen 63
8.12 Oberfläche Fragebogen erstellen 66
8.13 Navigation Editor 67

1 Einleitung

"Termin folgt auf Termin. Immer die Deadlines im Nacken. An die privaten Pflichten erinnert zwischendurch das Piepen des Handys: Kurznachrichten, E-Mails, soziale Netzwerke. Dort protzen Bekannte mit Reisen oder absolvierten Marathonläufen um die Wette. Bei den Werbemodels kneift kein Hosenbund. So jagt ein Reiz den nächsten. Dass das heutige Leben kaum noch Pausen kennt, kann unter Umständen an der seelischen Gesundheit nagen, fürchten Psychiatrie-Experten."[1] Ein Ausschnitt aus den Stuttgarter Nachrichten vom 24.11.2016. Dieser Absatz beschreibt den aktuellen Trend im Berufsleben sehr nahe. Viele Menschen stehen unter Dauerstress und finden keine Zeit um sich um sich selber zu kümmern. Vielen Betroffenen fehlt der Ausgleich zu Ihrem hektischen Arbeitsalltag, was unter anderem zu psychischen Erkrankungen, Tinnitus oder Bluthochdruck führen kann. Zusammenhänge zwischen Erkrankungen und potentiellen Auswirkungen werden in Studien durch Fragebögen ermittelt. Durch die Beantwortung von intervallabhängigen Fragebögen, ganz gemütlich von zu Hause aus, könnten über betroffene Menschen sehr viele Daten ermittelt werden. Diese Daten könnten dann wiederum als ein großes Gesamtbild analysiert werden oder aber für jeden einzelnen Betroffenen. Zudem könnte hierdurch für jeden Betroffenen ein persönlicher Werdegang ermittelt werden.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist das Erstellen einer Webapplikation zur Bereitstellung von Fragebögen zu verschiedenen Studien und deren Analyse. Registrierten Benutzern soll es ermöglicht werden, sich für verschiedene Studien anmelden zu können und anschließend die darin enthaltenen Fragebögen auszufüllen. Diese sollen in unterschiedlichen Intervallen auch erneut beantworten werden können. Durch die Wiederholung von Fragebögen soll dem Benutzer eine auf ihn zugeschnittene Ergebnisanalyse bereitgestellt werden. Durch die Ergebnisanalyse kann der Benutzer dann mögliche Veränderungen bei der Beantwortung der Fragebögen erkennen.

1.2 Aufbau der Arbeit

Die Struktur der Arbeit ist in Abbildung 1.1 dargestellt und wird im folgenden Schritt für Schritt genauer erläutert.
Die Einleitung ist nach diesem Kapitel abgeschlossen und es folgen in Kapitel 2 die Hintergrundinformationen. Dabei werden zu aller erst die Begriffe Tinnitus 2.1 und Stress 2.2 genauer erläutert, welche in dem Abschnitt Verwandte Arbeiten 2.3 danach benötigt werden. Zum Schluss werden in diesem Kapitel noch kurz ein paar grundlegende Informationen zu dem Pro-

jekt und dessen Ablauf aufgelistet (2.4). Anschließend werden in Kapitel 3 die Anforderungen an das System genauer erläutert mit den Funktionalen und Nicht-funktionalen Anforderungen. Wie das System genau aufgebaut ist, wird dann im Kapitel Architektur 4 präziser erläutert. Dabei werden alle vorhandenen Teile des Frontends (4.1) und des Backends (4.2) beschrieben und anschließend noch kurz die Versionsverwaltung (4.3) angesprochen. Anschließend wird in Kapitel 5 der Implementierungsverlauf dargestellt und im folgenden Kapitel 6 die verwendeten API-Funktionen der verschiedenen Bereiche wie Authentifizierung, der Fragebögen oder der Studien beschrieben. Danach werden in Kapitel 7 ein paar der Implementierungsinhalte genauer beschrieben, gefolgt vom Walkthrough (8) über die erstellte Website aus der Sicht eines nicht registrierten Benutzers (8.1), eines registrierten Benutzers (8.2) und eines Editors (8.3). Als vorletztes Kapitel, in Kapitel 9, werden die Anforderungen aus Kapitel 3 auf ihre Umsetzung bewertet. Zum Abschluss wird im letzten Kapitel 10 noch eine kurze Zusammenfassung über die Arbeit und ein möglicher Ausblick auf zukünftige Veränder- und Verbesserungen des Systems gegeben.

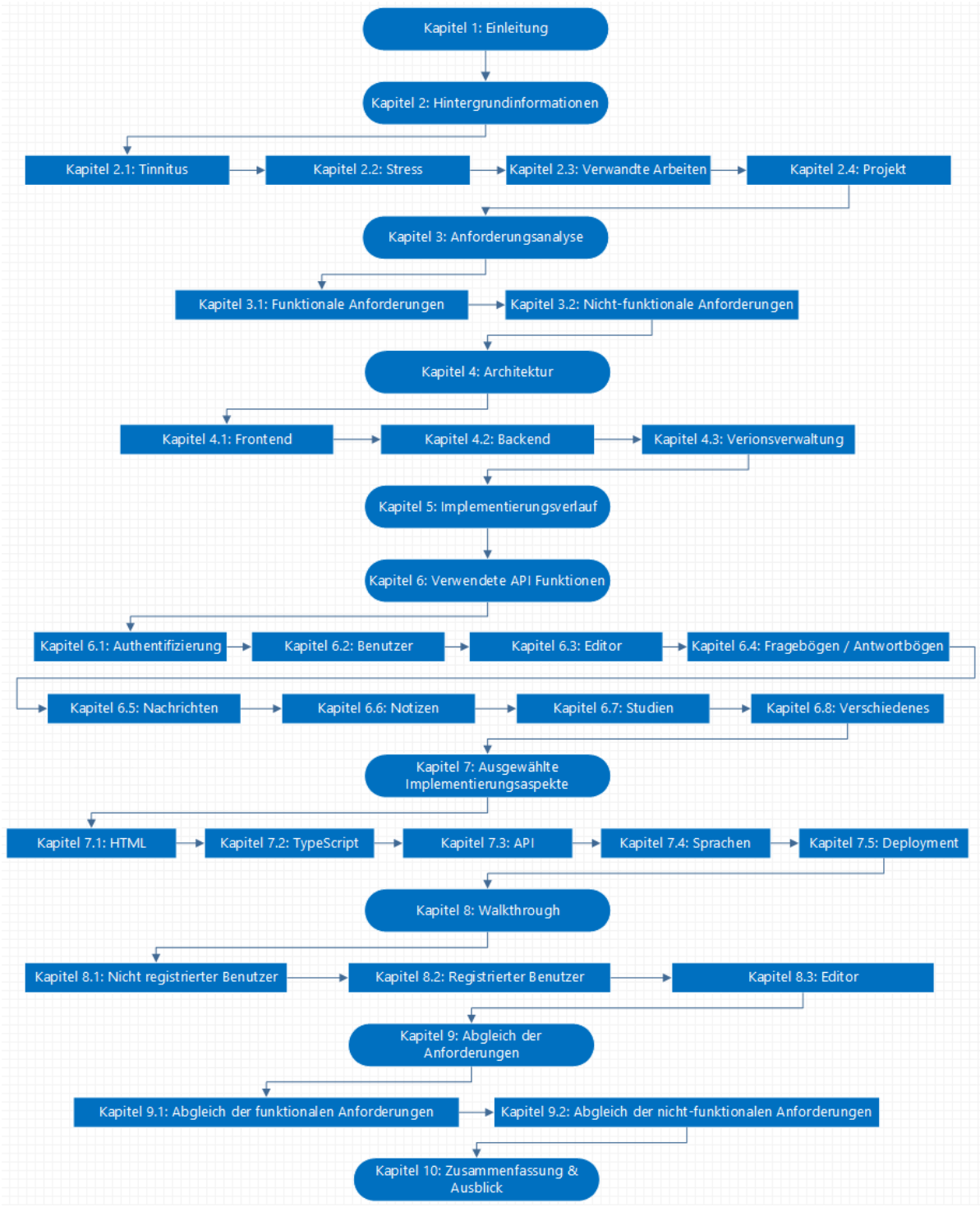


Abbildung 1.1: Aufbau der Arbeit

2 Hintergrundinformationen

In diesem Kapitel werden die Begriffe Tinnitus und Stress, sowie genauere Hintergrundinformationen zu diesem Projekt und anderen verwandten Arbeiten beschrieben.

2.1 Tinnitus

Tinnitus wird aus dem lateinischen Wort **tinnīre** abgeleitet, was im lateinischen so viel bedeutet wie klingeln [2]. So gut wie jeder kennt auch das Brummen oder Summen im Ohr, welches ein paar Sekunden bis Stunden anhalten kann und meistens nach einem lauten Konzert- oder Discobesuch auftritt. Diese Symptome sind ganz normal, da die hohe Schallbelastung die Haarzellen des Innenohres belasten.
Halten diese Geräusche jedoch über einen längeren Zeitraum an, so spricht man von Tinnitus. Dieser wird nicht durch die Umwelt hervorgerufen, sondern vom Patienten selber. Hierbei gibt es zwei verschiedene Arten von Tinnitus [3]:

- **Akuter Tinnitus**: Der Tinnitus besteht seit maximal drei Monaten und kann von selber wieder verschwinden.
- **Chronischer Tinnitus**: Der Tinnitus hält länger als drei Monate an.

Dazu wird der Tinnitus auch noch in unterschiedliche Grade eingestuft [3]:

- **1. Grad**: Der Tinnitus ist fast nicht wahrnehmbar und stört den Betroffenen nicht.
- **2. Grad**: Der Tinnitus ist bei Stille oder in Stresssituationen wahrnehmbar und störend.
- **3. Grad**: Der Tinnitus belastet den Betroffenen sehr im Berufs- und Privatleben. Betroffene leiden unter Schlaf- und Konzentrationsstörungen, Muskelverspannungen oder Kopfschmerzen.
- **4. Grad**: Dauerbelastung des Betroffenen durch den Tinnitus und massive Einschränkung der Lebensqualität. Der Betroffene kann nicht mehr arbeiten, zieht sich aus dem sozialen Leben zurück und leidet unter massiven psychischen Störungen wie Angst oder Depression.

Viele Menschen können mit dem Tinnitus leben und haben sich an die Geräusche gewöhnt. Die Ursachen für Tinnitus sind oft eine starke Lärmbelästigung oder Stress 2.2. Tinnitus kann in jedem Alter auftreten, jedoch tritt er meistens zwischen 40 und 50 Jahren auf. In Deutschland leiden über 3 Millionen Menschen an einem chronischen Tinnitus [4].

Betroffene Menschen zu behandeln, ist sehr schwierig, da es wie bereits beschrieben verschiedene Ursachen, Arten und Grade von Tinnitus gibt. Zudem können die Ursachen für den Tinnitus, auch durch sehr viele Tests, meistens nicht bestimmt werden. Je früher der Tinnitus jedoch behandelt wird, desto besser stehen die Chancen diesen wieder in den Griff zu bekommen. Mögliche Behandlungsarten sind zum Beispiel:

- Entspannungstechniken wie Yoga
- Infusionstherapie mit Medikamenten zur Durchblutungsförderung
- Psychologische Betreuung
- Unterdrückung des Tinnitus durch Hörgeräte (Tinnitusmasker)

2.2 Stress

Unter Stress versteht man psychische und körperliche Anspannungen, die durch äußerer Reize entstehen und den Körper in eine erhöhte Alarmbereitschaft versetzen. Früher waren diese Reize zum Beispiel Hunger, Kälte, die Gefahr vor Angriffen oder Schwerstarbeit. Heutzutage entsteht Stress eher durch Reizüberflutung, Zeit- und Leistungsdruck oder Konflikte.
Jeder Mensch reagiert anders auf Stress und geht damit unterschiedlich um. Stress muss nicht immer negativ sein, in den meisten Fällen schadet Stress aber dem Körper. Wie zum Beispiel in Kapitel 2.1 bereits erwähnt, kann Stress zu Tinnitus, Schlafstörungen, Verdauungsproblemen bis hin zu Herz-Kreislauferkrankungen führen.
Stress wird dabei in 4 verschiedene Phasen unterteilt [5]:

- **Vorphase**: Der Körper fährt sofort alle Stoffwechselvorgänge herunter um den Körper auf eine bevorstehende Aktivierung vorzubereiten. Diese ist auch als Schrecksekunde bekannt und verursacht eine kurze Handlungsunfähigkeit.
- **Alarmphase**: Der Körper sammelt alle ihm zur Verfügung stehende Energie, durch die Zufuhr von Stresshormonen (Adrenalin). Dadurch wird der Adrenalinspiegel, Herzschlag, Blutdruck und Blutzuckerspiegel erhöht. Das Gehirn und die Muskeln werden verstärkt mit Blut und Sauerstoff versorgt, was zu einem anstieg der Aktivität führt.
- **Handlungsphase**: In dieser Phase reagiert der Körper auf den äußerlichen Reiz. Sei es die Antwort auf eine spontane Frage während einer Präsentation, eine sportliche Leistung oder eine Erhöhung der Aufmerksamkeit während dem Autofahren.
- **Erholungsphase**: Der Körper erholt sich von der Stressreaktion durch das Normalisieren des Hormonspiegels und der Zufuhr von Energie an die Muskeln.

Wie bereits erwähnt, gibt es aber auch durchaus positiven Stress. Manche Menschen können unter Stress produktiver arbeiten und erfahren nach dem erfolgreichen Abschluss einer Aufgabe ein befriedigendes Gefühl [6].

Aus diesem Grund ist es um so wichtiger, Stressfaktoren gezielt zu ermitteln um negativen Stress zu erkennen, abzubauen und um den Körper vor einer zu starken Belastung zu schützen. Gegen Stress gibt es eine Vielzahl von Behandlungsarten, zum Beispiel:

- Ein Spaziergang an der frischen Luft um dem Körper Ruhe zu zuführen.

- Sport um den angesammelten Stress abzubauen oder überschüssige Energie los zu werden
- Entspannungsübungen wie Yoga
- Organisiertes Arbeiten
- Pflanzliche Arznei wie Kamille oder Baldrian

2.3 Verwandte Arbeiten

Nennenswert sind hier vor allem zwei weitere Projekte des **Instituts für Datenbanken und Informationssysteme (DBIS) der Universität Ulm**:
Track Your Tinnitus und **Track Your Stress**
Auf diese beiden Projekte, möchte ich im Folgenden Nähere eingehen.

2.3.1 Track Your Tinnitus

Vor dem Projekt Track Your Tinnitus entstand 2007 in Zusammenarbeit mit dem **Universitätsklinikum Regensburg**, der **Tinnitus Research Initiative** und der **Universität Magdeburg** das Projekt **Tinnitus Database**. Bei Tinnitus Database werden von Tinnituskliniken, auf freiwilliger Basis der Patienten, Informationen zum Tinnitus gespeichert. Darunter fallen zum Beispiel Informationen über die Form des Tinnitus, Symptome, Behandlungsarten und Ergebnisse. Alle Informationen können dann vom Patienten über eine Website mit persönlichem Log-in angeschaut werden. Auf dieser Website werden dem Benutzer graphische Oberflächen zur besseren Veranschaulichung bereitgestellt. Nachdem eine graphische Oberflächen für die Benutzer verfügbar war entstand dann im Jahr 2014 Track Your Tinnitus als mobile Anwendung für die Benutzer im GooglePlay- und iOS-Store. Hier können die Benutzer jetzt auch selber Fragebögen in regelmäßigen Abständen ausfüllen und sich ihre Ergebnisse anzeigen lassen. Zu sehen ist die APP in Abbildung 2.1.

2.3.2 Track Your Stress

Das Projekt Track Your Stress findet zum ersten Mal in zwei Bachelorthesen von 2017 Anwendung. Zum einen von Michael Schrempp in „Konzeption und Realisierung einer mobilen Anwendung zur Erfassung des Stresslevels am Beispiel von iOS“ [8]. Herr Schrempp entwickelte im Rahmen seiner Bachelorarbeit eine iOS Applikation zur Ermittlung von Stress. Analog hierzu, entwickelte Julian Haug unter dem Titel „Track Your Stress: Konzeption und Realisierung einer mobilen (Android) Anwendung zur Messung des Stresslevels“ [9], eine Android Applikation. Beide Applikationen sind ähnlich der zuvor genannten von Track Your Tinnitus. Michael Schrempp entwickelte dann in seinem Master Projekt eine vorläufige Website für Track Your Stress.

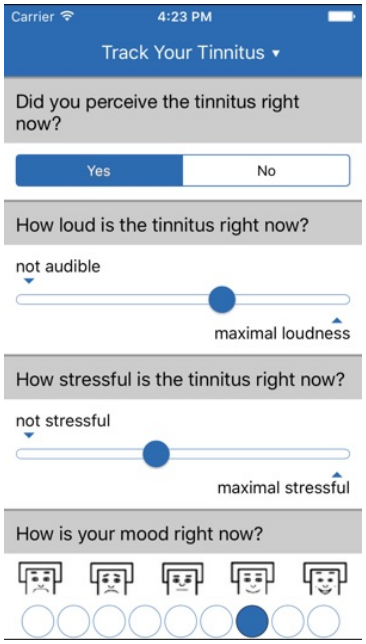


Abbildung 2.1: Track your Tinnitus APP [7]

Enthaltene Funktionalitäten waren:

Startseite

- Willkommenstext der Webseite.

Navbar

- Eine Navigationsbar im Header der Seite. Von hier aus ist es möglich auf alle anderen Seiten zu zugreifen.

Footer

- Copyright, Uni und Programmierer.

Registrierung

- Hier können sich neue Benutzer auf der Seite mit Emailadresse, Nickname und Passwort registrieren.
- Nach der Registrierung wird an die Emailadresse des Benutzers eine Bestätigungsemail versendet. Diese enthält einen Bestätigungslink zur Aktivierung des Kontos.
- Ein Klick auf den Bestätigungslink schließt die Registrierung ab und aktiviert den Account. Der Benutzer kann sich jetzt einloggen.

Login / Logout

- Eine Loginseite für registrierte Benutzer. Zur Verifizierung ist ein Tokensystem implementiert worden um zu überprüfen ob der Benutzer wirklich registriert und eingeloggt ist.

Benutzerbereich im Header enthalten

- **Mein Profil**
  Hier wird das Profil des Benutzers angezeigt mit Nickname, Emailadresse, Vor-/ Nachname und dem Geschlecht. Ebenfalls ist es hier für den Benutzer möglich das Passwort oder Informationen seines Profils zu ändern.
- **Meine Studien**
  Auf dieser Seite werden alle aktiven Studien des Benutzers angezeigt. Der Benutzer kann neuen Studien beitreten, sich von registrierten Studien abmelden und seinen Highscore für gesammelte Punkte einer Studie anschauen.
- **Meine Fragebögen**
  Dem Benutzer werden auf dieser Seite alle aktiven Fragebögen angezeigt. Durch die Auswahl eines Fragebogens kann dieser auf einer neuen Seite ausgefüllt und abgeschickt werden.

Überseite

- Informationen über das Projekt, die Datenschutzbestimmungen und das Team.

Sprachenkompatibilität im Footer enthalten

- Deutsche und Englische Version vorhanden.
- Kann über Buttons gewechselt werden.

2.4 Projekt

Das Projekt wurde von dem **Institut für Datenbanken und Informationssysteme der Universität Ulm** ins Leben gerufen und von **Dr. Rüdiger Pryss** als Masterarbeit ausgeschrieben. Das Ziel des Projektes ist eine Webapplikation, die es den Benutzern ermöglichen soll Fragebögen zu verschiedenen Studien auszufüllen. Eine Studie ist zum Beispiel Track your Tinnitus mit den aus der APP 2.3.2 bereits vorhandenen Fragebögen. Diese sollen nun auch am Computer ausfüllbar sein und dem Benutzer somit weitere zusätzliche zusätzliche Funktionalitäten bereitstellen.
Aufbauen soll das Projekt auf die bisherigen Projekte **TrackYourTinnitus** 2.3.1 und **TrackYourStress** 2.3.2. Als Grundlage der Implementierung dient der vorhandene Code von TrackYourStress. Dieser wurde jedoch an einigen Stellen angepasst und erweitert. Ebenfalls aus diesen Projekten soll die vorhandene API verwendet werden, welche bereits einen großen Funktionsumfang besitzt und die Schnittstelle zur Datenbank darstellt. Dazu mehr im Kapitel 4.2.
Nach den Anpassungen des Codes von TrackYourStress wurde die Website immer weiter mit neuen Funktionalitäten, Texten und Bildern erweitert. Nähere Informationen hierzu werden im Kapitel **Implementierungsverlauf** 5 erläutert.

3 Anforderungsanalyse

In diesem Kapitel werden die funktionalen und nicht-funktionalen Anforderungen an das zu entwickelnde System definiert und näher erläutert. Zudem wird die Wichtigkeit der Anforderung bewertet. Hierbei entspricht eine hohe Bewertung einer hohen Relevanz. In Kapitel 9 wird dann aufgeführt wie die angegeben Anforderungen realisiert wurden.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die Aufgaben, welche das System nach der Implementierung erfüllen soll.

Abkürzung	Anforderung	Wichtigkeit
FA01	Antwortbögen auswerten	+
FA02	Datenbankanbindung	+
FA03	Fragebögen ausfüllen	+
FA04	Fragebögen verwalten	+
FA05	Hilfe für Benutzer	+
FA06	Oberflächenverwaltung	o
FA07	Sprachunterstützung	+
FA08	Statistiken	+
FA09	Studien erstellen	o

Tabelle 3.1: Tabelle der funktionalen Anforderungen mit Wichtigkeit (-/unwichtig, o/neutral, +/wichtig)

FA01 Antwortbögen auswerten

Dem Benutzer soll es möglich sein, die Ergebnisse der Antwortbögen einzusehen. Die Antworten einer Frage sollen dann miteinander verglichen werden.

FA02 Datenbankanbindung

Die Antworten zu den Fragebögen sollen in einer Datenbank abgespeichert werden.

FA03 Fragebögen ausfüllen

Es sollen unterschiedliche Studien zur Verfügung stehen, wobei jede einzelne Studie unterschiedliche Fragebögen enthält, die der Benutzer dann ausfüllen kann.

FA04 Fragebögen verwalten

Es soll möglich sein neue Fragebögen für Studien zu erstellen, vorhandene Fragebögen zu bearbeiten und nicht mehr benötigte Fragebögen zu entfernen.

FA05 Hilfe für Benutzer

Sollte der Benutzer Probleme bei der Nutzung der Website oder Anwendung haben, soll ihm eine Hilfe zur Verfügung stehen.

FA06 Oberflächenverwaltung

Die Oberfläche soll für den PC optimiert sein.

FA07 Sprachunterstützung

Eine deutsche, englische und niederländische Oberfläche sollen vorhanden sein.

FA08 Statistiken

Es sollen Statistiken zu relevanten Informationen angezeigt werden.

FA09 Studien erstellen

Es soll möglich sein neue Studien zu erstellen.

3.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben die Qualitätseigenschaften des Systems.

Abkürzung	Anforderung	Wichtigkeit
NFA01	Benutzerfreundlichkeit	+
NFA02	Erweiterbarkeit	+
NFA03	Korrektheit und Zuverlässigkeit	+
NFA04	Performanz	+
NFA05	Verfügbarkeit	+
NFA06	Wartbarkeit	+

Tabelle 3.2: Tabelle der nicht-funktionalen Anforderungen mit Wichtigkeit (-/unwichtig, o/neutral, +/wichtig)

NFA01 Benutzerfreundlichkeit

Die Anwendung soll einfach und intuitiv zu bedienen sein.

NFA02 Erweiterbarkeit

Das System soll einfach zu erweitern sein im Bezug auf die Oberfläche und die Funktionalität.

NFA03 Korrektheit und Zuverlässigkeit

Das System soll stets so funktionieren, wie es vom Benutzer erwartet wird und mit auftretenden Fehlern korrekt umgehen.

NFA04 Performanz

Berechnungen und Seitenwechsel sollen in angebrachter Zeit durchgeführt werden.

NFA05 Verfügbarkeit

Die Webseite soll nur minimale Ausfallzeiten aufweisen.

NFA06 Wartbarkeit

Durch eine strukturierte und kommentierte Implementierung soll die Wartung des Systems vereinfacht werden.

4 Architektur

Dieses Kapitel beinhaltet die grundlegende Architektur des Projekts aufgeteilt in Frontend, Backend und Versionsverwaltung. In Abbildung 4.1 ist der Aufbau bildlich dargestellt.

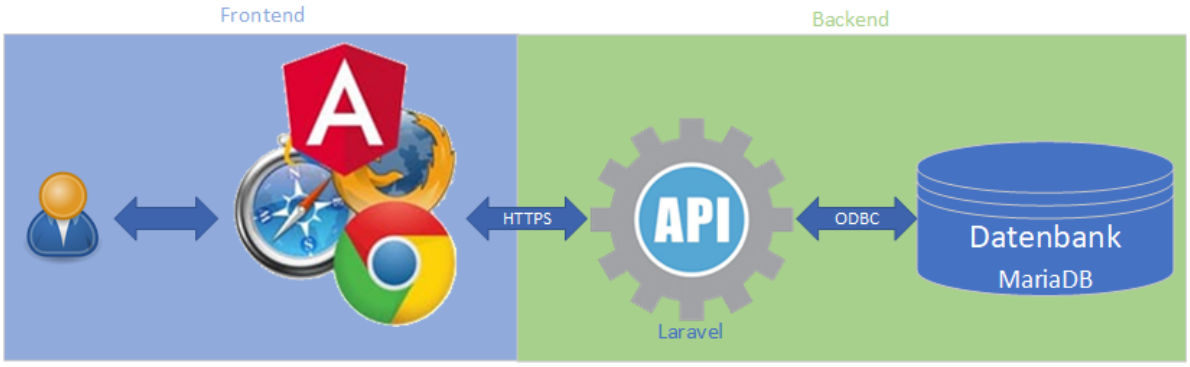


Abbildung 4.1: Architektur mit Frontend und Backend

4.1 Frontend

Als Frontend bezeichnet man die Präsentationsebene oder Benutzeroberfläche der Applikation, also den Teil welchen der Benutzer schlussendlich sehen kann. Auf dieser Ebene werden dem Benutzer Texte, Bilder oder andere Informationen zur Verfügung gestellt.

4.1.1 Angular

Zur Entwicklung des Frontends wurde das Open-Source Front-End-Webapplikationsframework **Angular**, in der Version 8.0.1, von Google verwendet. Es ermöglicht Oberflächen für das Web und mobile Geräte zu entwickeln. Durch seine große Fanbase gibt es auch genügend Tutorials die einem den Einstieg erleichtern. Zudem existiert eine enorme Menge an externen Paketen / Plugins die sehr simpel in das Projekte eingebunden werden können.
Eine Auflistung der in diesem Projekt verwendeten Pakete:

Angular5-csv Export (Version 0.2.11)

- Einfaches erstellen und herunterladen von CSV-Dateien.

Angular Google Maps (Version 1.0.0)

- Angular Google Maps ermöglicht das Anzeigen der Google Maps Karte in einem Angular Projekt. Dabei ist es auch möglich verschiedene Standorte zu markieren oder ganze Routen anzuzeigen. Zu sehen ist dies in der Oberfläche in Abbildung 8.10 in Kapitel 8.3.

Angular Material (Version 8.1.2)

- Angular Material enthält eine große Anzahl an UI-Komponenten, welche das Erstellen und Designen von ansprechenderen Oberflächen ermöglicht.

Angular Social Button (Version 1.0.0)

- Durch Angular Social Button kann die Website simple per Button weiterempfohlen werden. Diese sind im Footer enthalten.

Bootstrap (Version 3.4.1)

- Bootstrap ist ein Open Source CSS-Framework. Es bietet für HTML, CSS und JS bereits vorgefertigte Vorlagen für verschiedene Elemente wie Buttons, Formulare und Tabellen. Es ermöglicht dadurch ein einfaches, allgemeines designen der Website.

Chart.js (Version 2.8.0)

- Mit Chart.js können verschiedene Diagramme erstellt werden.

Device Detector (Version 1.3.6)

- Der Device Detector ermöglicht es Informationen über den Browser, das Betriebssystem und andere verwendete Geräte zu ermitteln.

Font Awesome (Version 4.7.0)

- Font Awesome stellt verschiedene Icons und Logos zur Verfügung.

Die erste Version hieß AngularJS. Diese wurde dann aber in der zweiten Version auf Angular 2 umbenannt [10]. Die aktuelle Version ist **Angular 8**, welche in diesem Projekt auch verwendet wurde.

4.1.2 Node.js

Um Webapplikation auf dem eigenen Computer anzeigen zu können, benötigt man einen virtuellen Server auf dem Computer. Hierbei wurde **Node.js** verwendet um einen eigenständigen Server zu realisieren. Node.js wurde entwickelt um skalierbare Netzwerkanwendungen zu erstellen. Enthalten ist auch der Node Package Manager (NPM), welcher es ermöglicht neue Pakete / Plugins simple im Projekt zu installieren und diese, wie auch andere verwendete Tools wie Angular, regelmäßig zu updaten.

4.1.3 TypeScript

Als Programmiersprache wird **TypeScript** empfohlen. TypeScript ist eine von Microsoft entwickelte Programmiersprache, welche kompatibel zu JavaScript ist. Dadurch können die Bibliotheken von JavaScript auch in Typescript verwendet werden. In diesem Projekt wurde die Version 3.5.2 verwendet.

4.1.4 Visual Studio Code

Als Editor wurde **Visual Studio Code**, in der Version 1.37.1, von Microsoft verwendet. Dieser unterstützt eine große Anzahl an Programmiersprachen wie C++, CSS, HTML, Java (-Script), PHP, Python, SQL, TypeScript und XML. Ebenfalls ist es möglich Plugins zu installieren, die das Programmieren vereinfachen.
Eine Übersicht der in diesem Projekt verwendeten Plugins:

Icon Fonts (Version 2.1.5)

- Schnelleres Hinzufügen von Symbolen in HTML durch eine Ergänzung der Autovervollständigung

TODO Highlight (Version 1.0.4)

- Farbliche Markierung von TODOs

4.2 Backend

Als Backend bezeichnet man den Teil der Applikation, der für den Benutzer nicht sichtbar ist. Dieser Teil beinhaltet zum Beispiel den Zugriff auf gespeicherte Daten in der Datenbank.

4.2.1 API

Im Backend fungierte eine bereits aus den Projekten **Track Your Tinnitus** (2.3.1) und **Track Your Stress** (2.3.2), des Instituts für Datenbanken und Informationssysteme (DBIS) der Universität Ulm, verwendete **API**. Die Abkürzung API steht für Application Programing Interface und bezeichnet eine Programmierschnittstelle. Sie stellt die Schnittstelle zwischen dem Frontend und der Datenbank dar. Die API wurde mittels des **PHP-Webframeworks Laravel** erstellt. PHP wird meistens als serverseitige Programmiersprache verwendet. Services der API werden per URL angesprochen. Hinzu kommt dann noch eine HTTP-Methode wie **GET**, **POST**, **PATCH** oder **DELETE** damit die API weiß welche Operation ausgeführt werden soll.

GET

- Daten werden vom Server angefordert.

POST

- Daten werden an den Server übermittelt.

PATCH

- Vorhandene Daten auf dem Server werden verändert.

DELETE

- Daten werden auf dem Server gelöscht.

Anschließend werden entsprechende SQL Funktionen auf der Datenbank aufgerufen. Als Antwort liefert die API dann verschiedene Daten zurück zum Frontend im JSON-Format (JavaScript Object Notation). Darunter fallen zum Beispiel geladene Daten aus der Datenbank,

Fehlermeldungen oder Bestätigungen über erfolgreich abgeschlossene Aufgaben.
Zu sehen ist eine Rückgabe der API in Abbildung 4.2. Die Rückgabe beinhaltet hier alle Antwortbögen eines Fragebogens. Enthalten sind die Daten der einzelnen Antwortbögen, Informationen zu dem verwendeten Gerät des Benutzers, das Ausfülldatum, der Standort des Benutzers zu dem Zeitpunkt, die ID des Benutzers und weitere Informationen im JSON-Format.

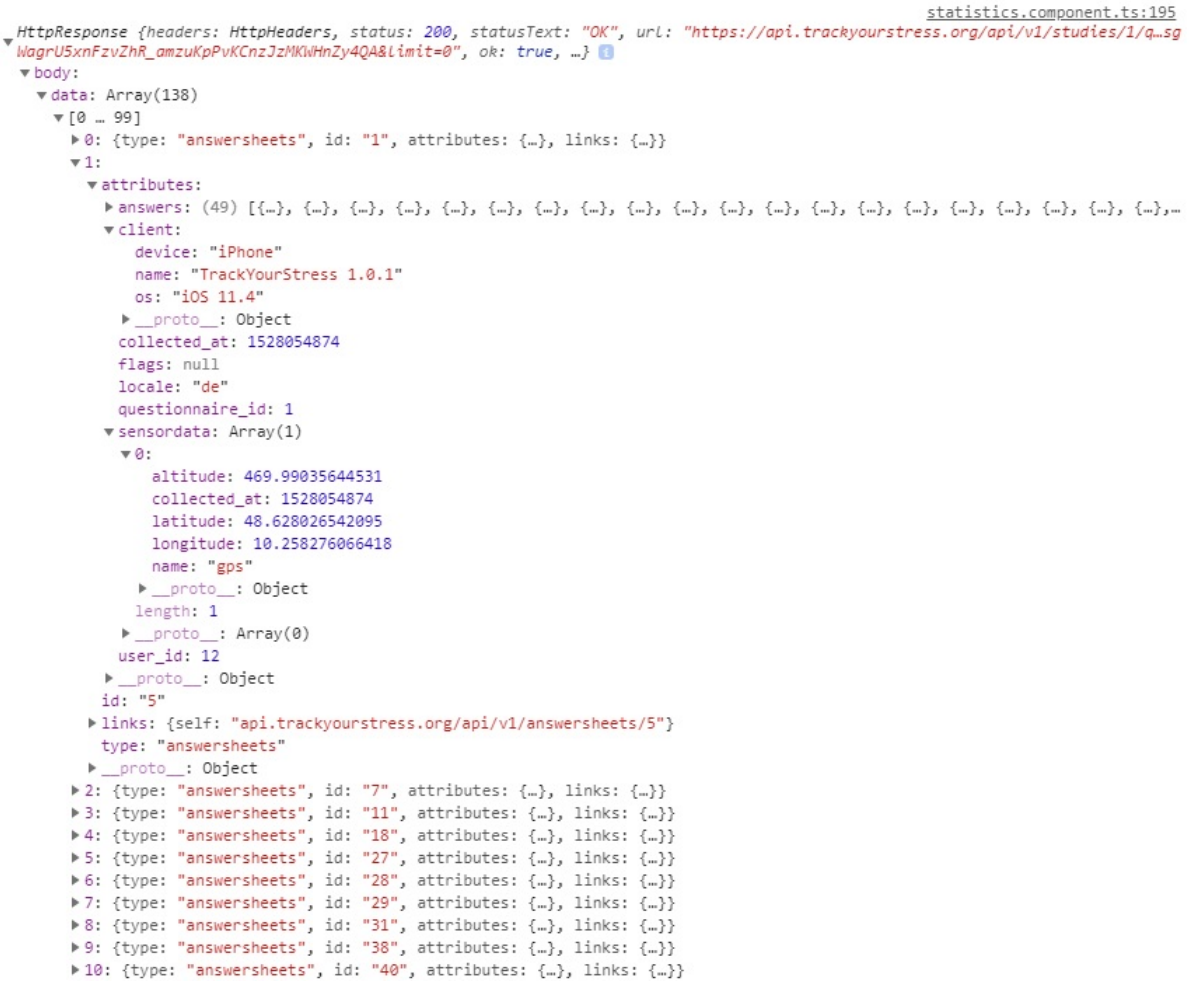


Abbildung 4.2: Rückgabe API auf Anfrage aller Antwortbögen eines Fragebogens im JSON-Format

Genauere Details zu den verwendeten Funktionen der API werden in Kapitel 6 erklärt.

4.2.2 MariaDB

Das Herzstück des gesamten Projekts ist die Datenbank **MariaDB** im Hintergrund. Hier werden alle Informationen gespeichert. Per API können die einzelnen, gespeicherten Daten dann modifiziert, gelöscht oder geladen werden. Verbunden ist die API per **ODBC-Treiber** mit der Datenbank. **ODBC** steht hierbei für **Open Database Connectivity** und bezeichnet eine standardisierte, offene Schnittstelle für den Zugriff auf verschiedene Datenbanksysteme.

4.3 Versionsverwaltung

Zur Versionsverwaltung des Projekts wurde **GIT** in Kombination mit **Bitbucket** als Filehosting-Dienst verwendet. Dadurch kann auf verschiedenen Systemen immer mit der aktuellsten Version gearbeitet werden und bei Problemen kann der Code auf eine frühere Version zurück gesetzt werden. Hinzu kommt noch **Sourcetree** als Graphische Oberfläche für den Benutzer. Zu sehen ist die Oberfläche in Abbildung 4.3. Zentral in der Mitte werden die einzelnen Commits angezeigt mit Beschreibung und Datum. Darunter werden die vom ausgewählten Commit veränderten Dateien und präzisere Informationen zu den jeweiligen Änderungen der Datei dargestellt.

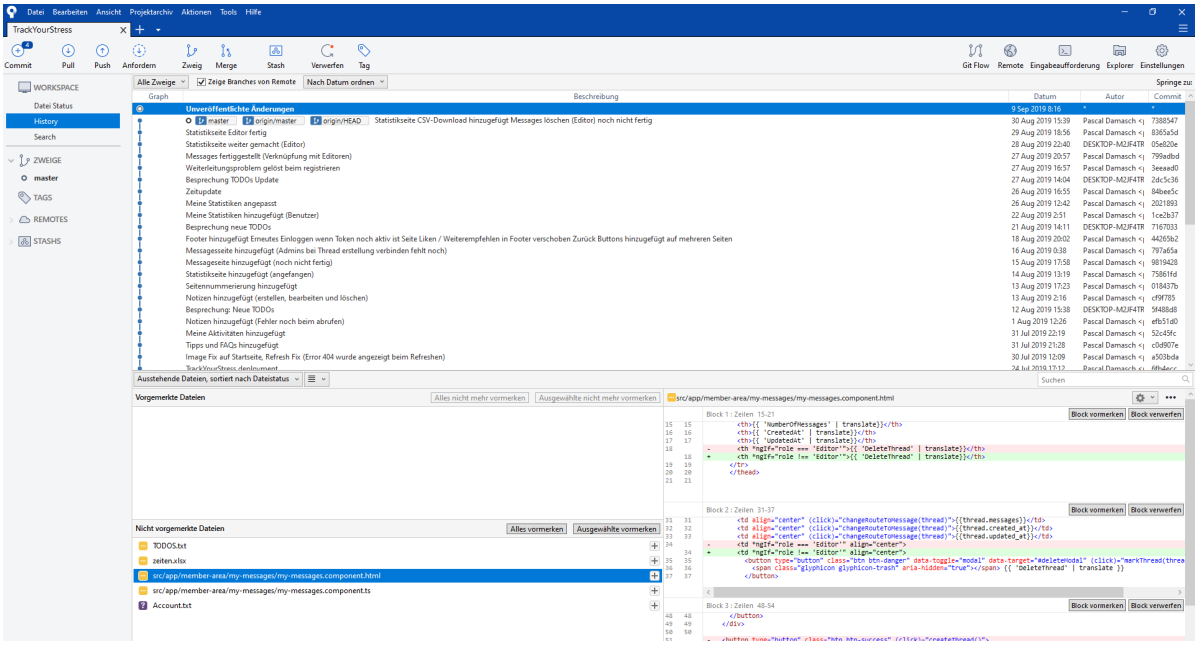


Abbildung 4.3: Sourcetree Oberfläche

5 Implementierungsverlauf

In diesem Kapitel wird die Reihenfolge der Implementierung beschrieben. Angefangen mit der Übernahme des vorhandenen Codes aus TrackYourStress, den ersten Implementierungen bis hin zu den jüngsten Ergänzungen.

1. **Anpassungen an die Vorlage von TrackYourStress (2.3.2)**
   - Anlegen eines neues Projektes mit aktueller Angular und TypeScript Version
   - Kopieren und anpassen des vorhandenen Codes von TrackYourStress. Neben funktionalen Änderungen wurde die Oberfläche angepasst:
     - **Startseite**: Text wurde angepasst und Verlinkungen zu den beteiligten Universitäten hinzugefügt
     - **Navbar**: Enthält jetzt Umstellung der Sprachen
     - **Footer**: Text wurde angepasst und Auswahl der Sprachen wurde entfernt
     - **Überseite**: Text wurde angepasst
2. **Meine Ergebnisse hinzugefügt**
   Zusätzlicher Abschnitt im Benutzerbereich. Alle aktiven Studien des Benutzers werden hier aufgelistet. Durch die Auswahl einer Studie gelangt der Benutzer auf die Ergebnisseite. Hier kann der Benutzer einen Fragebogen der zuvor ausgewählten Studie auswählen und sich alle Ergebnisse dieses Fragebogens graphisch und textuell anzeigen lassen.
3. **Impressum und Datenschutz hinzugefügt**(Nur auf deutsch)
4. **Plattformkompatibilität hinzugefügt**
   Es wurde eine Switch-Methode für das System hinzugefügt um die Oberfläche schnell zwischen TrackYourStress und TrackYourTinnitus2 wechseln zu können. Diese Funktion wurde hinzugefügt um für andere Projekte schneller eine Website generieren zu können, welche mit derselben API im Hintergrund kompatibel ist, aber oberflächliche Änderungen beinhaltet.
5. **Zu den Sprachen Deutsch und Englisch wurde Niederländisch als dritte Sprache hinzugefügt**
6. **Impressum und Datenschutz wurden aktualisiert und für die englische und niederländische Oberfläche hinzugefügt**
7. **Die Ergebnisseite wurde um eine Ansicht für die Registrierungsfragebögen ergänzt**

8. **CSV-Export hinzugefügt**
   Auf der Ergebnisseite wurde eine Download-Funktion hinzugefügt. Dadurch ist es für den Benutzer möglich alle seine Ergebnisse des ausgewählten Fragebogens als CSV/Excel-Datei herunterzuladen.
9. **Seite Weiterempfehlen hinzugefügt**
   Im Footer wurden mehrere Elemente hinzugefügt um die Seite auf verschiedenen Plattformen weiterempfehlen zu können, darunter zum Beispiel Facebook und Twitter.
10. **FAQs und Tipps wurden hinzugefügt**
11. **Meine Aktivitäten hinzugefügt**
    Zusätzlicher Abschnitt im Benutzerbereich. Eine Übersicht aller Aktivitäten des Benutzers seit der Registrierung mit Zeitstempel und einer kurzen Beschreibung.
12. **Notizen hinzugefügt**
    Zusätzlicher Abschnitt im Benutzerbereich. Der Benutzer kann für sich Notizen erstellen, diese als erledigt markieren, bearbeiten und löschen.
13. **Seitennummerierung hinzugefügt**
    Bei großen Datenmengen wurde eine Unterteilung auf mehrere Seiten hinzugefügt um eine übersichtlichere Nutzung zu ermöglichen. Dabei werden maximal 15 Elemente pro Seite angezeigt. Enthalten ist das zum Beispiel unter meine Aktivitäten, meine Notizen oder Tipps.
14. **Messages hinzugefügt**
    Ein zusätzlicher Abschnitt im Benutzerbereich. Der Benutzer kann bei Problemen oder Fragen ein Thema mit einer Nachricht erstellen. Diese wird dann an die Administration der ausgewählten Studie weitergeleitet. Dadurch wird eine interne Chatfunktion bereitgestellt, die es dem Benutzer ermöglicht mit der Administration zu kommunizieren. Wurde dem Benutzer geholfen kann dieser das Thema auch wieder löschen.
15. **Meine Statistiken hinzugefügt**
    Zusätzlicher Abschnitt im Benutzerbereich. Der Benutzer kann sich verschiedene Statistiken zu seinem Profil anschauen:
    - **Allgemeine Informationen**
      - Anzahl aller registrierten Benutzer
      - Anzahl vorhandener Studien
      - Anzahl vorhandener Fragebögen
    - **Benutzerspezifische Informationen**
      - Anzahl bisheriger Aktivitäten
      - Anzahl ausgefüllter Fragebögen
      - **Graphische Darstellung im Donut-Diagramm**
        * Anzahl beigetretener Studien zu allen vorhandenen Studien
        * Anzahl ausgefüllter Fragebögen zu allen vorhandenen Fragebögen (Jeder Fragebogen zählt hier nur einmal)
        * Anzahl aller erledigten Notizen zu allen vorhandenen Notizen

16. **Statistikseite für Editor hinzugefügt**
    Der Editor kann sich verschiedene Statistiken zu den Benutzern, Studien und Fragebögen anschauen:
    - **Allgemeine Informationen**
      - Anzahl vorhandener Studien
      - Anzahl vorhandener Fragebögen
      - Anzahl ausgefüllter Fragebögen durch die Benutzer
      - Durchschnittliche Anzahl ausgefüllter Fragebögen der Benutzer
    - **Benutzerbezogene Informationen**
      - Anzahl aller registrierten Benutzer
      - Durchschnittliches Alter der Benutzer
      - Standartabweichung beim Alter der Benutzer
    - **Graphische Darstellung im Balken-Diagramm**
      - Auflistung aller beteiligten Länder mit Anzahl
    - **Graphische Darstellung im Donut-Diagramm**
      - Aufteilung der Geschlechter (Männlich, Weiblich, Keine Angabe)
      - Aufteilung der Plattformen (iOS, Android, Computer)
17. **Statistikseite des Editors einen CSV-Export hinzugefügt**
    Auf der Statistikseite des Editors wurde eine Download-Funktion hinzugefügt. Dadurch ist es für den Editors möglich die Statistiken als CSV/Excel-Datei herunterzuladen.
18. **Ergänzung der Statistikseite des Editors mit Google Maps**
    Auf der Statistikseite des Editors wurde zusätzlich eine Google Maps Karte hinzugefügt, welche die ungefähren Standorte der Benutzer darstellt. Dadurch ist es für die Editoren möglich herauszufinden in welchen Ländern und unter welchen Gegebenheiten sich die meisten Benutzer befinden.
19. **Fragebögen als Editor erstellen**
    Dem Editor ist es nun möglich einen neuen Fragebogen für eine vorhandene Studie zu erstellen. Dabei kann der Editor verschiedene Frageelemente hinzufügen und diese nach belieben ordnen. Mehr dazu im Kapitel 8.3. Die Funktionalität wurde aber noch nicht mit der API verknüpft, da in der API noch nicht alle Informationen/ Daten übernommen werden können.
20. **Fragebögen als Editor bearbeiten**
    Der Editor kann jetzt vorhandene Fragebögen laden und diese bearbeiten. Die Oberfläche und Funktionalität ist dabei wie beim Erstellen eines neuen Fragebogens. Wie bei der vorherigen Funktionalität ist diese aber auch noch nicht mit der API verknüpft.
21. **Fragebögen als Editor aktivieren und löschen**
    Fragebögen können jetzt durch den Editor für die Benutzer freigegeben werden und gegebenenfalls auch wieder gelöscht werden. Aus Sicherheitsgründen ist diese Funktionalität aber auch noch nicht mit der API verknüpft.

22. **TrackYourStress alle Funktionalitäten aktiviert**
    Bisher waren für **TrackYourStress** bestimmte Abschnitte nicht freigeschaltet. Darunter fallen zum Beispiel der Editorbereich, die Statistikseite des Benutzers, Nachrichten und Notizen. Jetzt sind alle Funktionalitäten verfügbar.

23. **Studie erstellen**
    Als Editor ist es nun möglich neue Studien zu erstellen. Dabei werden der **Titel** und die **Beschreibung** auf deutsch, englisch und niederländisch benötigt, sowie optional die Eingabe eines Passworts um der Studie beizutreten. Diese Funktionalität ist aber noch nicht mit der API verbunden, da diese keine passende Funktionalität zur Verfügung stellt.

24. **Fragebogen erstellen/ bearbeiten Sprachen hinzugefügt**
    Bisher konnte ein Fragebogen nur in einer Sprache erstellt werden. Da die Website jedoch neben der deutschen auch eine englische und niederländische Oberfläche bereitstellt, musste die Oberfläche und Funktionalität beim Erstellen und Bearbeiten eines Fragebogen überarbeitet werden. Jetzt ist es notwendig das Konstrukt des Fragebogens in allen drei Sprachen auszufüllen. Die Funktionalität ist aber immer noch nicht mit der API verbunden.

25. **Meine Kontextdaten hinzugefügt**
    Der Benutzer kann jetzt seine Deezer, Netflix, Spotify und Twitter Accounts zu seinem Profil hinzufügen. Dadurch können neue Informationen über die Suchanfragen und favorisierten Kategorien des Benutzers ermittelt werden. Die Funktionalität ist aber immer noch nicht mit der API verbunden.

6 Verwendete API Funktionen

In diesem Teil der Arbeit werden die verwendeten API-Funktionen aufgelistet und kurz beschrieben. Es sind noch weitere Funktionen vorhanden, welche aber in diesem Projekt, bis jetzt, nicht benötigt wurden. Um die Auflistung übersichtlicher zu gestalten werden die Funktionen in verschiedene Bereiche aufgeteilt. Genaue Angaben zum jeweiligen API-Link werden aus Sicherheitsgründen nicht angegeben.

6.1 Authentifizierung

- **Benutzer Registrieren**
  - Registriert einen neuen Benutzer im System.
  - Keine Rolle notwendig
  - Methode: POST
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:
    - type: string (Hier 'users', Benötigt)
    - attributes: array (Benötigt)
      - email: string (Benötigt)
      - password: string (Benötigt)
      - password_confirmation: string (Benötigt)
      - name: string (Benötigt)
      - settings: array (Zusätzliche Angaben (Sprache), nicht benötigt)
- **Benutzer Login**
  - Authentifiziert einen Benutzer durch seine Emailadresse oder Benutzernamen und generiert ein Token für den Benutzers. Dieses Token wird zur Verifizierung der API-Funktionen verwendet.
  - Keine Rolle notwendig
  - Methode: POST
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:

        - type: string (Hier 'users', Benötigt)
        - attributes: array (Benötigt)
            - email: string (Email oder Benutzername wird benötigt)
            - name: string (Email oder Benutzername wird benötigt)
            - password: string (Benötigt)

- **Benutzer Logout**
    - Loggt den Benutzer aus dem System. Dabei wird das Token deaktiviert.
    - Rolle: User
    - Methode: DELETE
    - Kein Data-Element notwendig
- **Neues Passwort**
    - Setzt ein neues Passwort für den Benutzer.
    - Rolle: User
    - Methode: POST
    - Keine spezifischen IDs für API-Link
    - Parameter Data-Element:
        - type: string (Hier 'users', Benötigt)
        - attributes: array (Benötigt)
            - reset_token: string (Token des Benutzers, Benötigt)
            - password: string (Benötigt)
- **Passwort zurücksetzen**
    - Schickt dem Benutzer per Mail einen Link um sein Passwort zurückzusetzen.
    - Keine Rolle notwendig
    - Methode: POST
    - Keine spezifischen IDs für API-Link
    - Parameter Data-Element:
        - type: string (Hier 'users', Benötigt)
        - attributes: array (Benötigt)
            - email: string (Benötigt)

6.2 Benutzer

- **Alle registrierten Benutzer**
    - Lädt alle registrierten Benutzer.

  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Meine Aktivitäten**
  - Lädt alle Aktivitäten des Benutzers seit dem Registrieren.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Meine Antwortbögen**
  - Lädt alle vorhandenen Antwortbögen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Meine Fragebögen**
  - Lädt alle vorhandenen Fragebögen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Meine Notizen**
  - Lädt alle vorhandenen Notizen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Mein Profil**
  - Lädt das Profil des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig

- **Meine Rollen**
  - Lädt die angegebenen Rollen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Meine Studien**
  - Lädt alle vorhandenen Studien des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Passwort bearbeiten**
  - Aktualisiert das Passwort des Benutzers.
  - Rolle: User
  - Methode: PATCH
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:
    - type: string (Hier 'users', Benötigt)
    - attributes: array (Benötigt)
      - password: string (Benötigt)
- **Profil bearbeiten**
  - Aktualisiert das Profil des Benutzers.
  - Rolle: User
  - Methode: PATCH
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:
    - type: string (Hier 'users', Benötigt)
    - attributes: array (Benötigt)
      - name: string (Nicht benötigt)
      - firstname: string (Nicht benötigt)
      - lastname: string (Nicht benötigt)
      - sex: integer (Zahl steht für Geschlecht, Nicht benötigt)
      - picture: array (Link zum Bild und Größe, Nicht benötigt)

· settings: array (Zusätzliche Angaben, Nicht benötigt)

6.3 Editor

- **Alle Antwortbögen eines Fragebogens**
  - Lädt alle vorhandenen Antwortbögen aller Benutzer zu einem spezifischen Fragebogen.
  - Rolle: Editor
  - Methode: GET
  - Notwendige IDs:
    - studyID (ID der Studie)
    - questionnaireID (ID des Fragebogens)
  - Kein Data-Element notwendig

6.4 Fragebögen / Antwortbögen

- **Antwortbogen anzeigen**
  - Lädt einen spezifischen Antwortbogen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - answersheetID (ID des Antwortbogens)
  - Kein Data-Element notwendig
- **Antwortbogen einreichen**
  - Überträgt einen ausgefüllten Fragebogen des Benutzers.
  - Rolle: User
  - Methode: POST
  - Notwendige IDs:
    - questionnaireID (ID des Fragebogens)
  - Parameter Data-Element:
    - type: string (Hier 'answersheets', Benötigt)
    - attributes: array (Benötigt)
      - locale: string (Ausgewählte Sprache, Benötigt)
      - answers: array (Antworten zum Fragebogen, Nicht benötigt)
      - sensordata: array (Antworten zum Fragebogen, Nicht benötigt)

        - client: array (Informationen zum Client, Benötigt)
        - collected_at: date (Benötigt)

- **Antwortbögen des Fragebogens**
  - Lädt alle Antwortbögen eines Benutzers zu einem spezifischen Fragebogen.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - questionnaireID (ID des Fragebogens)
  - Kein Data-Element notwendig
- **Antworten einer spezifischen Frage**
  - Lädt alle Antworten eines Benutzers zu einer spezifischen Frage eines Fragebogens.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - questionnaireID (ID des Fragebogens)
    - label (Label des Elements)
  - Kein Data-Element notwendig
- **Alle Fragebögen**
  - Lädt alle vorhandenen Fragebögen.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Elemente des Fragebogens**
  - Lädt alle Elemente eines Fragebogens.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - questionnaireID (ID des Fragebogens)
  - Kein Data-Element notwendig
- **Fragebogen anzeigen**
  - Lädt einen spezifischen Fragebogen.
  - Rolle: User
  - Methode: GET

  - Notwendige IDs:
    - questionnaireID (ID des Fragebogens)
  - Kein Data-Element notwendig

6.5 Nachrichten

- **Nachricht erstellen**
  - Erstellt eine neue Nachricht im angegebenen Thema.
  - Rolle: User
  - Methode: POST
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:
    - type: string (Hier 'messages/messages', Benötigt)
    - attributes: array (Benötigt)
      - body: string (Hier 'message', Benötigt)
      - deliver_at: date (Nicht benötigt)
- **Postfach anzeigen**
  - Lädt alle Themen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - threadID (ID des Themas)
  - Kein Data-Element notwendig
- **Teilnehmer zu Thema hinzufügen**
  - Fügt andere Benutzer zum Thema hinzu damit mit diesen kommuniziert werden kann.
  - Rolle: User
  - Methode: POST
  - Notwendige IDs:
    - threadID (ID des Themas)
  - Parameter Data-Element:
    - type: string (Hier 'users', Benötigt)
    - attributes: array (Benötigt)
      - participants: array (IDs der Benutzer, Benötigt)

- **Thema anzeigen**
  - Lädt alle Nachrichten des Themas.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - threadID (ID des Themas)
  - Kein Data-Element notwendig
- **Thema erstellen**
  - Erstellt ein neues Thema mit dem Benutzer als Besitzer.
  - Rolle: User
  - Methode: POST
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:
    - type: string (Hier 'messages/threads', Benötigt)
    - attributes: array (Benötigt)
      - subject: string (Benötigt)
- **Thema löschen**
  - Löscht ein Thema mit allen dazugehörigen Nachrichten. Nur für den Besitzer des Themas möglich.
  - Rolle: User
  - Methode: DELETE
  - Notwendige IDs:
    - threadID (ID des Themas)
  - Kein Data-Element notwendig

6.6 Notizen

- **Alle Notizen**
  - Lädt alle Notizen des Benutzers.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Neue Notiz**
  - Erstellt eine neue Notiz für den Benutzer.

  - Rolle: User
  - Methode: POST
  - Keine spezifischen IDs für API-Link
  - Parameter Data-Element:
    * type: string (Hier 'notes', Benötigt)
    * attributes: array (Benötigt)
      · is_done: boolean (Nicht benötigt)
      · title: string (Nicht benötigt)

- **Notiz anzeigen**
  - Lädt eine Notiz des Benutzers.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    * noteID (ID der Notiz)
  - Kein Data-Element notwendig
- **Notiz bearbeiten**
  - Aktualisiert eine Notiz des Benutzers.
  - Rolle: User
  - Methode: PATCH
  - Notwendige IDs:
    * noteID (ID der Notiz)
  - Parameter Data-Element:
    * type: string (Hier 'notes', Benötigt)
    * attributes: array (Benötigt)
      · is_done: boolean (Nicht benötigt)
      · title: string (Nicht benötigt)
- **Notiz löschen**
  - Löscht eine Notiz des Benutzers.
  - Rolle: User
  - Methode: DELETE
  - Notwendige IDs:
    * noteID (ID der Notiz)
  - Kein Data-Element notwendig

6.7 Studien

- **Alle Studien**
  - Lädt alle vorhandenen Studien.
  - Rolle: User
  - Methode: GET
  - Keine spezifischen IDs für API-Link
  - Kein Data-Element notwendig
- **Fragebögen einer Studie**
  - Lädt alle vorhandenen Fragebögen einer spezifischen Studie.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - studyID (ID der Studie)
  - Kein Data-Element notwendig
- **Studie abmelden**
  - Benutzer meldet sich von einer Studie ab.
  - Rolle: User
  - Methode: DELETE
  - Notwendige IDs:
    - studyID (ID der Studie)
  - Kein Data-Element notwendig
- **Studie anzeigen**
  - Lädt eine spezifische Studie.
  - Rolle: User
  - Methode: GET
  - Notwendige IDs:
    - studyID (ID der Studie)
  - Kein Data-Element notwendig
- **Studie einschreiben**
  - Benutzer meldet sich zu einer Studie an.
  - Rolle: User
  - Methode: POST
  - Notwendige IDs:

        * studyID (ID der Studie)
    - Kein Data-Element notwendig

6.8 Verschiedenes

- **Alle FAQs**
    - Lädt alle vorhandenen FAQs
    - Keine Rolle notwendig
    - Methode: GET
    - Keine spezifischen IDs für API-Link
    - Kein Data-Element notwendig
- **Alle Tipps**
    - Lädt alle vorhandenen Tipps.
    - Keine Rolle notwendig
    - Methode: GET
    - Keine spezifischen IDs für API-Link
    - Kein Data-Element notwendig
- **Bestenliste Studie**
    - Lädt die Bestenliste einer Studie.
    - Rolle: User
    - Methode: GET
    - Notwendige IDs:
        * studyID (ID der Studie)
    - Kein Data-Element notwendig
- **FAQ anzeigen**
    - Lädt eine spezifische FAQ.
    - Keine Rolle notwendig
    - Methode: GET
    - Notwendige IDs:
        * faqID (ID der FAQ)
    - Kein Data-Element notwendig
- **Tipp anzeigen**
    - Lädt einen spezifischen Tipp
    - Keine Rolle notwendig

- Methode: GET
- Notwendige IDs:
  - tippID (ID des Tipps)
- Kein Data-Element notwendig

7 Ausgewählte Implementierungsaspekte

In diesem Kapitel werden die wichtigsten Funktionalitäten des HTML-, TypeScript und API-Codes genauer beschrieben und betrachtet.

7.1 HTML

Per HTML, und CSS-Styling, wurde die Oberfläche der Website erstellt. Die interessantesten Bereiche sind hierbei das Anzeigen der Ergebnisse des Benutzers und der Fragebögen des Benutzers.

7.1.1 Ergebnisse Benutzer

Auf der Ergebnisseite werden die Ergebnisse der Fragebögen für den Benutzer angezeigt. Dabei wird unterschieden ob es ein einmaliger Fragebogen ist oder ob dieser mehrfach ausgefüllt werden kann. In Listing 7.1 ist der Aufbau der Ergebnisseite von mehrfach ausfüllbaren Fragebögen dargestellt. Auf einmalige Fragebögen wird hier nicht genauer eingegangen, da deren Ergebnisseite der der normalen Fragebögen gleicht. Der einzige Unterschied besteht darin, dass alle Felder deaktiviert wurden und dadurch keine Eingabe möglich ist.
Auf der Ergebnisseite wird, nachdem entschieden wurde ob der Fragebogen mehrfach oder einmalig ausfüllbar ist, jedes einzelne Element des Fragebogens durchlaufen (Zeile 1). Danach wird das Konstrukt des Fragebogens aufgebaut indem der Typ des Elements ausgelesen und dann ein passender Abschnitt dafür erstellt wird. Bis auf Headline, Text, TextString/TextArea und TextDate wird alles in Graphen (canvas) angezeigt. Dabei wird eine **id** zu dem jeweiligen Graphen erstellt, welche aus dem Namen des Elements besteht (**questStruct.label**). Die Art des Graphen wird dann erst beim Erstellen an sich festgelegt. Momentan wird nur ein Feld freigehalten wo ein Graph mit einer **id** erstellt werden kann. Dazu genauer in Abschnitt 7.2.4. Bei den anderen Typen wird ein einfaches mehrzeiliges Textfeld erzeugt in welchem die Ergebnisse mit Datum aufgelistet werden.

```
1  <div *ngFor="let questStruct of questionnaireStructures">
2
3    <!-- HEADLINE -->
4    <ng-template [ngIf]="questStruct.headline">
5      <li class="list-group-item center" style="background-color:powderblue;">
6        <div innerHTML={{questStruct.headline}}></div>
7      </li>
8    </ng-template>
9
10   <!-- TEXT -->
```

```
11  <ng-template [ngIf]="questStruct.text">
12    <li class="list-group-item">
13      <div innerHTML={{questStruct.text}}></div>
14    </li>
15  </ng-template>
16
17  <!-- QUESTION -->
18  <ng-template [ngIf]="questStruct.question">
19    <li class="list-group-item">
20      <div innerHTML={{questStruct.question}}></div>
21
22      <!-- SingleChoice -->
23    <ul *ngIf="questStruct.questionType === 'SingleChoice'" class="list-group">
24      <canvas id="{{questStruct.label}}"></canvas>
25    </ul>
26
27    <!-- MultipleChoice -->
28    <ul *ngIf="questStruct.questionType === 'MultipleChoice'" class="list-group
        ">
29      <canvas id="{{questStruct.label}}"></canvas>
30    </ul>
31
32    <!-- Slider -->
33    <div *ngIf="questStruct.questionType === 'Slider'">
34      <canvas id="{{questStruct.label}}"></canvas>
35    </div>
36
37    <!-- TextString/TextArea -->
38    <div *ngIf="questStruct.questionType === 'TextString' || questStruct.
        questionType === 'TextArea'">
39      <textarea class="form-control" name="textString{{questStruct.id}}" form="
          usrform" id="{{questStruct.label}}" disabled=true></textarea>
40    </div>
41
42    <!-- TextDate -->
43    <div *ngIf="questStruct.questionType === 'TextDate'">
44      <textarea class="form-control" name="textDate{{questStruct.id}}" form="
          usrform" id="{{questStruct.label}}" disabled=true></textarea>
45    </div>
46
47    <!-- YesNoSwitch -->
48    <ul *ngIf="questStruct.questionType === 'YesNoSwitch'" class="list-group">
49      <canvas id="{{questStruct.label}}"></canvas>
50    </ul>
51
52    <!-- SAMScaleFace -->
53    <div *ngIf="questStruct.questionType === 'SAMScaleFace'" class="SAMBox">
54      <canvas id="{{questStruct.label}}"></canvas>
55    </div>
56
57    <!-- SAMScaleBody -->
58    <div *ngIf="questStruct.questionType === 'SAMScaleBody'" class="SAMBox">
59      <canvas id="{{questStruct.label}}"></canvas>
60    </div>
61    </li>
62  </ng-template>
```

```
63 </div>
```

Listing 7.1: Ergebnisse Benutzer

7.1.2 Fragebogen anzeigen

Ein Fragebogen enthält ein **form** Element, was zur Kontrolle benötigt wird und überprüft ob alle Eingaben der enthaltenen Elemente korrekt und vollständig sind (Listing 7.2). Alle Elemente innerhalb werden mit der **form** verknüpft. Erst wenn alle Eingaben der enthaltenen Elemente korrekt und vollständig sind, wird die Funktion **generateJsonData()** aufgerufen, welche die Daten aus den verschiedenen Elementen lädt und weiterverarbeitet. Aufgerufen wird die Überprüfung durch den **submit** Button am Ende der Seite.

```
1 <form (ngSubmit)="answersForm.valid && generateJsonData()" #answersForm="ngForm
      " novalidate>
2   ...
3   <br><button type="submit" class="btn btn-success">{{ 'Save' | translate }}</
      button>
4 </form>
```

Listing 7.2: Fragebogen form

Nach dem **form** Element folgt die Erstellung der einzelnen Elemente des Fragebogens. Als Schleife, um alle Elemente des Fragebogens zu erhalten, wird die spezielle Angular Funktion ***ngFor** verwendet (Listing 7.3). Dadurch wird jedes einzelne Element von **questionnaire-Structures** aufgerufen.

```
1 <div *ngFor="let questStruct of questionnaireStructures">
2   ...
3 <\div>
```

Listing 7.3: Fragebogen *ngFor Schleife

Das Element wird jetzt auf den Typ überprüft und dann entsprechend dargestellt:

- **Headline**
  Eine Überschrift im Fragebogen in Textform (Listing 7.4). Wird hervorgerufen durch einen blauen Hintergrund und enthält den Text aus **questStruct.headline**.

```
1 <ng-template [ngIf]="questStruct.headline">
2   <li class="list-group-item center" style="background-color:powderblue;">
3     <div innerHTML={{questStruct.headline}}></div>
4   </li>
5 </ng-template>
```

Listing 7.4: Fragebogen Headline

- **Text**
  Ähnlich dem Headline Element, aber ohne den farbigen Hintergrund und der Text wird aus **questStruct.text** geladen (Listing 7.5).

```
1 <ng-template [ngIf]="questStruct.text">
2   <li class="list-group-item">
3     <div innerHTML={{questStruct.text}}></div>
4   </li>
```

```
5 </ng-template>
```

Listing 7.5: Fragebogen Text

- **SingleChoice**
  Beim Typ **SingleChoice** werden alle Antwortelemente geladen und mit einem **clickevent** verknüpft, so das beim Auswählen eines Elements dieses als gesetzt markiert wird und das vorherig ausgewählte Element in den Daten überschrieben wird. Oberflächlich werden die Antwortmöglichkeiten als **Radiobuttons** dargestellt, wobei nur eines davon aktiv sein kann. Der Text für den **Radiobutton** ist in **answer** enthalten und wird daraus geladen (Listing 7.6).

```
1  <ul *ngIf="questStruct.questionType === 'SingleChoice'" class="list-group"
       >
2    <li *ngFor="let answer of questStruct.answers; let i = index" class="
         list-group-item" (click)="checkSingleChoice(questStruct, i)">
3      <div class="radio">
4        <label>
5          <input type="radio" name="singleChoice{{questStruct.id}}"
6            (change)="checkSingleChoice(questStruct, i)"
7            [value]="questStruct.id + i" ngModel #singleChoiceModel="ngModel
                 "
8            [required]="questStruct.required === 1">
9          {{answer}}
10
11         <div *ngIf="answersForm.submitted && singleChoiceModel.invalid"
               class="alert alert-danger">
12           {{ 'OneAnswerIsRequired' | translate }}
13         </div>
14       </label>
15     </div>
16   </li>
17 </ul>
```

Listing 7.6: Fragebogen SingleChoice

- **MultipleChoice**
  MultipleChoice ist wieder ähnlich zu **SingleChoice**, nur das dieses mal anstatt der **Radiobuttons** eine **Checkbox** verwendet wird, welche mehrere Auswahlmöglichkeiten gleichzeitig zulässt (Listing 7.7).

```
1  <ul *ngIf="questStruct.questionType === 'MultipleChoice'" class="list-
       group">
2    <li *ngFor="let answer of questStruct.answers; let i = index" class="
         list-group-item" (click)="checkMultipleChoice(questStruct, i)">
3      <div class="radio">
4        <label>
5          <input type="checkbox" name="multChoice{{questStruct.id}}"
6            (change)="checkMultipleChoice(questStruct, i)"
7            [value]="questStruct.id + i" ngModel #multipleChoiceModel="
                 ngModel"
8            [required]="questStruct.required === 1">
9          {{answer}}
10
11         <div *ngIf="answersForm.submitted && multipleChoiceModel.invalid"
               class="alert alert-danger">
```

```
12            {{ 'OneAnswerIsRequired' | translate }}
13          </div>
14        </label>
15      </div>
16    </li>
17  </ul>
```

Listing 7.7: Fragebogen MultipleChoice

- **Slider**
  Der **Slider** enthält einen Namen, Minimum Wert, Maximum Wert und eine Schritteinheit für das Bewegen des Sliders. Immer wenn der Slider bewegt wurde, wird die Funktion **changeSliderappearance(questStruct)** aufgerufen, welche den aktuellen Wert des Sliders speichert. Zum besseren Verständnis der Einheit wird noch eine textuelle Beschreibung des Minimums und Maximums angezeigt (Listing 7.8).

```
1  <div *ngIf="questStruct.questionType === 'Slider'">
2    <input type="range" (click)="changeSliderappearance(questStruct)"
3      name="slider{{questStruct.id}}"
4      [className]="questStruct.sliderclass"
5      min="{{questStruct.sliderValues[0]}}"
6      max="{{questStruct.sliderValues[1]}}"
7      step="{{questStruct.sliderValues[2]}}"
8      [(ngModel)]="questStruct.sliderValue" #sliderModel="ngModel"
9      [required]="questStruct.required === 1">
10
11   <div>
12     <div class="sliderMinValue Sameline">
13       {{questStruct.answers[0]}}
14     </div>
15     <div class="sliderMaxValue pull-right Sameline">
16       {{questStruct.answers[1]}}
17     </div>
18   </div>
19
20   <div *ngIf="answersForm.submitted && sliderModel.invalid" class="alert
       alert-danger">
21     {{ 'RequiredField' | translate }}
22   </div>
23 </div>
```

Listing 7.8: Fragebogen Slider

- **TextString/TextArea**
  Neben der einzeiligen Texteingabe **TextString** gibt es auch eine mehrzeilige Texteingabe **TextArea**. Beide sind vom Konstrukt her gleich aufgebaut. Bei einer Texteingabe wird bei jedem Tastendruck die Funktion **updateCollectedat(questStruct)** aufgerufen, welche den momentanen Inhalt des Textfeldes abspeichert (Listing 7.9).

```
1  <div *ngIf="questStruct.questionType === 'TextString' || questStruct.
     questionType === 'TextArea'">
2    <input type="text" class="form-control"
3      name="textString{{questStruct.id}}"
4      [(ngModel)]="questStruct.textStringAnswer" #textStringModel="ngModel"
5      (keypress)="updateCollectedat(questStruct)"
6      [required]="questStruct.required === 1">
```

```
7
8     <div *ngIf="answersForm.submitted && textStringModel.invalid" class="
          alert alert-danger">
9       {{ 'RequiredField' | translate }}
10    </div>
11  </div>
```

Listing 7.9: Fragebogen TextString/TextArea

- **TextDate**
  TextDate zeigt dem Benutzer einen Kalender an, aus welchem dieser dann ein Datum auswählen kann. Nach einer Auswahl wird das ausgewählte Datum über die Funktion **updateCollectedat(questStruct)** gespeichert (Listing 7.10).

```
1  <div *ngIf="questStruct.questionType === 'TextDate'">
2    <input type="date" class="form-control" name="textDate{{questStruct.id}}
         "
3      [(ngModel)]="questStruct.textDateAnswer" #textDateModel="ngModel"
4      (keypress)="updateCollectedat(questStruct)"
5      (click)="updateCollectedat(questStruct)"
6      [required]="questStruct.required === 1">
7
8    <div *ngIf="answersForm.submitted && textDateModel.invalid" class="alert
         alert-danger">
9      {{ 'RequiredField' | translate }}
10   </div>
11 </div>
```

Listing 7.10: Fragebogen TextDate

- **YesNoSwitch**
  Beim **YesNoSwitch** gibt es nur, wie der Name bereits sagt, die Auswahlmöglichkeiten Ja und Nein. Der Aufbau ist gleich wie bei den **SingleChoice** Radiobuttons, daher wird auch die selbe Funktion **checkSingleChoice(questStruct, i)** bei Veränderungen verwendet (Listing 7.11).

```
1  <ul *ngIf="questStruct.questionType === 'YesNoSwitch'" class="list-group">
2    <li *ngFor="let answer of questStruct.answers; let i = index" class="
         list-group-item"
3      (click)="checkSingleChoice(questStruct, i)">
4      <div class="radio">
5        <label>
6          <input type="radio" name="singleChoice{{questStruct.id}}"
7            (change)="checkSingleChoice(questStruct, i)"
8            [value]="questStruct.id + i"
9            ngModel #singleChoiceModel="ngModel"
10           [required]="questStruct.required === 1">
11         {{answer}}
12
13         <div *ngIf="answersForm.submitted && singleChoiceModel.invalid"
               class="alert alert-danger">
14           {{ 'RequiredField' | translate }}
15         </div>
16       </label>
17     </div>
18   </li>
```

```
19 </ul>
```

Listing 7.11: Fragebogen YesNoSwitch

- **SAMScaleFace**
  Das Element **SAMScaleFace** beinhaltet neun verschiedene Stufen von Gesichtern. Die Gesichter sind als png Bild vorhanden und werden als **Image** auf der Seite angezeigt. Eine Schleife zählt die Zahlen Eins bis Neun durch, was dem Index Null bis Acht entspricht. Bei einem geraden Index, oder Null, ist die variable **even** auf **true** gesetzt. Wenn **even true** ist wird ein Bild neben dem Radiobutton angezeigt. Im anderen Falle wird zwar ein Bild geladen, dieses wird aber nicht angezeigt und nur zur Einhaltung der Oberflächenform verwendet. Wie beim **SingleChoice** Element kann auch hier nur ein **Radiobutton** gleichzeitig aktiv sein. Wenn ein neuer Radiobutton als markiert gesetzt wird, wird dieser durch die Funktion **checkSam(questStruct, index-1, true)** auf markiert gesetzt und das vorherig ausgewählte Element in den Daten überschrieben (Listing 7.12).

```
1  <div *ngIf="questStruct.questionType === 'SAMScaleFace'" class="SAMBox">
2    <div *ngFor="let index of [1,2,3,4,5,6,7,8,9]; let even = even; let
         pictureIndex = index" class="floatingSAM">
3      <img *ngIf="even" src="/assets/images/SAMScaleFace/SAM-VP-9_{{((
         pictureIndex + 1) / 2) + 0.5}}.png" width="10%">
4
5      <img *ngIf="!even" src="/assets/images/SAMScaleFace/SAM-VP-9_1.png"
         style="visibility: hidden;" width="10%">
6
7      <input type="radio" class="radioStyleClass"
8        name="samFace{{questStruct.id}}"
9        (change)="checkSam(questStruct, index-1, true)"
10       [value]="questStruct.id + index-1"
11       ngModel #samFaceModel="ngModel"
12       [required]="questStruct.required === 1">
13
14     <div *ngIf="answersForm.submitted && samFaceModel.invalid" class="
         alert alert-danger">
15       {{ 'OneAnswerIsRequired' | translate }}
16     </div>
17   </div>
18 </div>
```

Listing 7.12: Fragebogen SAMScaleFace

- **SAMScaleBody**
  Das **SAMScaleBody** Element ist gleich aufgebaut wie das **SAMScaleFace** Element. Der einzige Unterschied liegt darin, dass es sich hier nicht um Gesichter sondern um Körper handelt und dadurch andere Bilder geladen werden (Listing 7.13).

```
1  <div *ngIf="questStruct.questionType === 'SAMScaleBody'" class="SAMBox">
2    <div *ngFor="let index of [1,2,3,4,5,6,7,8,9]; let even = even; let
         pictureIndex = index" class="floatingSAM">
3      <img *ngIf="even" src="/assets/images/SAMScaleBody/SAM-A-9_{{((
         pictureIndex + 1) / 2) + 0.5}}.png" width="10%">
4
5      <img *ngIf="!even" src="/assets/images/SAMScaleBody/SAM-A-9_1.png"
         style="visibility: hidden;" width="10%">
6
```

```
7       <input type="radio" class="radioStyleClass"
8         name="samBody{{questStruct.id}}"
9         (change)="checkSam(questStruct, index-1, true)"
10        [value]="questStruct.id + index-1"
11        ngModel #samBodyModel="ngModel"
12        [required]="questStruct.required === 1">
13
14        <div *ngIf="answersForm.submitted && samBodyModel.invalid" class="
              alert alert-danger">
15        {{ 'OneAnswerIsRequired' | translate }}
16      </div>
17    </div>
18  </div>
```

Listing 7.13: Fragebogen SAMScaleBody

7.2 TypeScript

Angular basiert auf der Programmiersprache TypeScript. Alle Funktionalitäten, welche nicht per HTML geschrieben werden konnten, wurden daher in TypeScript geschrieben. Darunter fallen zum Beispiel der Aufbau und die Verwendung von Klassen, der Aufruf der API oder allgemeine Funktionalitäten. Ein paar der wichtigsten Implementierungen werden nachstehend etwas genauer beschrieben.

7.2.1 Aufruf Dialog

Ein Dialog, oder auch Pop-up-Fenster genannt, wird verwendet um einfache Tätigkeiten zusätzlich auf einer Seite anzuzeigen. Dabei bleibt im Hintergrund die aktuell geöffnete Seite geöffnet und nur im Vordergrund wird ein neues Fenster angezeigt. Zu sehen ist ein Beispiel in Abbildung 7.1 vom Erstellen einer neuen Nachricht eines Benutzers. Aufgerufen wird die Komponente durch eine andere Komponente. Beim Beispiel des angegebenen Bildes ruft eine Funktion der Meine Nachrichten Komponente den Konstruktor der Dialog Komponente auf. Dies ist in Listing 7.14 zu sehen. Dabei werden zuerst allgemeine Einstellungen für den Dialog festgelegt. Zum Beispiel dass das Fenster in der Mitte des Bildschirmes erscheinen soll, oder dass wenn man außerhalb des Bildes klickt, das Fenster dadurch geschlossen wird. Anschließend wird der Dialog erstellt und geöffnet. Hier wäre es auch noch möglich gewesen zusätzliche Attribute zu übergeben. Zum Schluss kann noch angegeben werden, ob beim Schließen des Dialogs zusätzlich noch etwas ausgeführt werden soll. Dabei ist es ebenfalls möglich wieder Daten zu übergeben.

```
1 createThread() {
2     const dialogConfig = new MatDialogConfig();
3     dialogConfig.disableClose = false;
4     dialogConfig.autoFocus = true;
5     const dialogRef = this.dialog.open(MessageDialogComponent, dialogConfig);
6
7     dialogRef.afterClosed().subscribe(
8       data => this.ngOnInit()
9     );
```

```
10    }
```

Listing 7.14: Aufruf Dialog

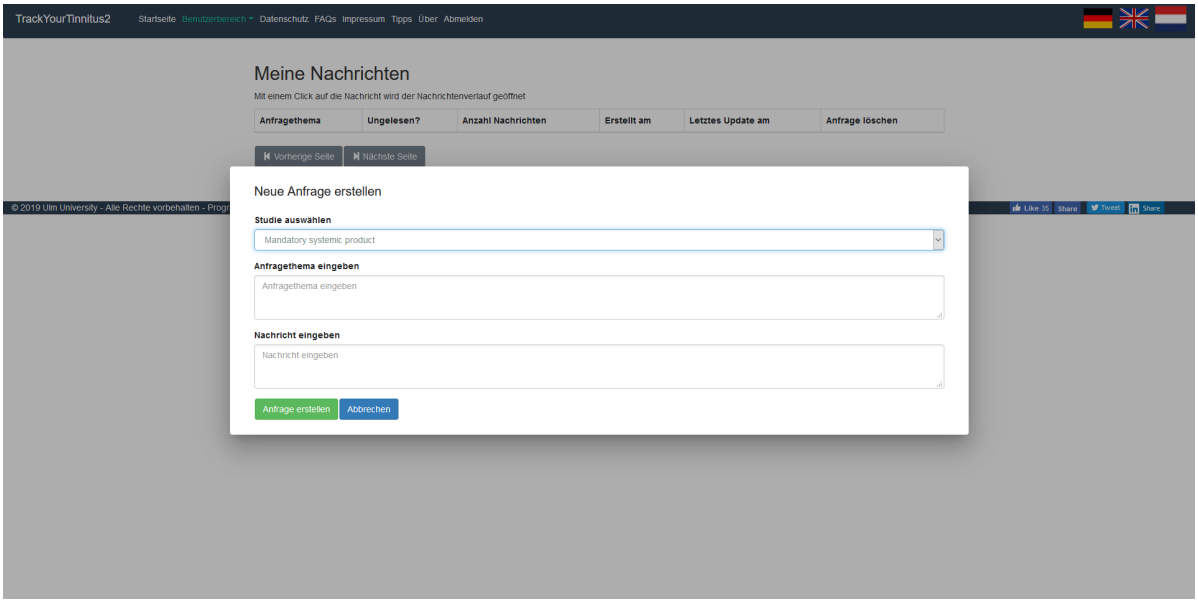


Abbildung 7.1: Pop-up-Fenster

7.2.2 Aufruf API-Funktion

Um Servicefunktionen der API aufzurufen wird ein besonderes Konstrukt benötigt. Zu sehen ist dies in Listing 7.15. Dabei wird in Zeile 1 die Funktion **this.questionnaireApiService.getQuestionnaire(questionnaire.id)** aufgerufen und die Verbindung in der Variable **reqQuestionnaire** gespeichert. In Zeile 2 wird dann auf eine Antwort der Funktion mittels **subscribe** gewartet. Da die Daten auf einem anderen Server liegen, kann dies einige Zeit dauern. Wenn bei dem Aufruf keine Probleme auftreten, dann sind Daten in **respQuestionnaire** enthalten. Diese können dann, wie in Listing 7.15 aufgezeigt, zum Beispiel dafür verwendet werden um alle Elemente eines Fragebogens zu laden. Tritt bei der Übertragung ein Fehler auf, wird ein Error (Zeile 7) aufgerufen. Hier wird dann für den Benutzer die Fehlerursache angezeigt. Die Funktion **this.questionnaireApiService.getQuestionnaire(questionnaire.id)** wird in Abschnitt 7.3.1 noch gezeigt.

```
1  const reqQuestionnaire = this.questionnaireApiService.getQuestionnaire(
       questionnaire.id);
2  reqQuestionnaire.subscribe(respQuestionnaire => {
3    let questionnaire = respQuestionnaire['body']['data'];
4    for (const element of questionnaire) {
5      ...
6    }
7  }, err => {
8    const status = err['status'];
9    this.alertService.enterMessage(status, 1);
10 });
```

Listing 7.15: Aufruf API-Funktion

7.2.3 Fragebogen Klasse

Die Klasse des Fragebogens enthält alle Informationen über ein Element, zum Beispiel eine Frage, des Fragebogens. Dieses wird individuell je nach Typ erstellt, da nicht jeder Typ jede Variable der Klasse benötigt. Enthalten sind aber immer die Variablen für den **general** (Zeile 3), **attributes** (Zeile 6) und **content** (Zeile 10) Bereich. Die restlichen Variablen werden je nach Element spezifisch ausgewählt. Aus dieser Klasse werden die Elemente aus Abschnitt 7.1.1 und 7.1.2 geladen, welche dort in Array Form vorhanden sind. Da ein Fragebogen mehrere dieser Elemente besitzt.

```
1  export class QuestionnaireStructure {
2    constructor(
3      // general
4      public id: number,
5
6      // attributes
7      public name: string,
8      public elementtype: string,
9
10     // content
11     public headline: string,
12     public text: string,
13     public question: string,
14     public required: number,
15     public questionType: string,
16     public label: string,
17     public answers: string[],
18
19     // values
20     public singleAndMultValues: string[],
21     public sliderValues: number[],
22     public samValues: number[],
23     public yesNoValues: string[],
24
25     // needed for slider answers
26     public sliderclass: string,
27     public sliderValue: number,
28
29     // needed for singleChoice answers
30     public isSingleChecked: boolean[],
31
32     // needed for multipleChoice answers
33     public isMultipleChecked: boolean[],
34
35     // needed for TextString
36     public textStringAnswer: string,
37
38     // needed for TextDate
39     public textDateAnswer: string,
40
41     // needed for TextInteger
42     public textInteger: number,
43
44     // needed for samFace
45     public isSamFaceChecked: boolean[],
46
```

```
47      // needed for samBody
48      public isSamBodyChecked: boolean[],
49
50      // save the current time of answering the question
51      public collected_at: number
52    ) { }
53  }
```

Listing 7.16: Fragebogen Klasse

7.2.4 Graph erstellen

Nachdem der Graph im HTML-Code platziert und mit einer id versehen wurde (Abschnitt 7.1.1) kann dieser Graph nun in TypeScript erstellt werden. Zu sehen ist ein Beispiel in 7.17. Nachdem der Graph in Zeile 1 durch die **canvasID** gefunden wurde und in Zeile 3 die zugehörige Variable zum bearbeiten des Graphen geladen wurde, kann in Zeile 5 die eigentliche Einstellung des Graphen erfolgen. Dafür wird ein **Typ**, eine **Benennung** der jeweiligen Werte, die **Werte** an sich und für jeden Wert eine **Farbe** benötigt. Danach kann noch festgelegt werden ob der Graph einen **Titel** besitzt und ob dieser, sowie die Legende, angezeigt werden soll. Anschließend wird der Graph mit der Bearbeitungsvariable und den Einstellungen des Graphen erstellt.

```
1  let canvas = <HTMLCanvasElement>document.getElementById(canvasID);
2  canvas.height = 100;
3  let ctx = canvas.getContext("2d");
4
5  let myChartConfig = {
6    // pie, bar, line, doughnut, ...
7    type: type,
8    data: {
9      labels: [label1, label2, label3, ...],
10     datasets: [
11       {
12         data: [data1, data2, data3, ...],
13         backgroundColor: [color1, color2, color3, ...]
14       }
15     ]
16   },
17   options: {
18     title: {
19       display: true,
20       text: title,
21       fontSize: 18
22     },
23     legend: {
24       display: true,
25       labels: {
26         fontSize: 18
27       }
28     }
29   }
30 }
31
32 let myChart = new Chart(ctx, myChartConfig);
```

Listing 7.17: Graph erstellen

7.2.5 Trigger: Wechsel zwischen TrackYourTinnitus2 und TrackYourStress

Es wurde ein **Trigger** hinzugefügt, welcher das einfache Wechseln der Oberfläche von TrackYourTinnitus2 und TrackYourStress ermöglicht. Dabei muss lediglich in der **globals.ts** die Variable **platform** auf **tinnitus** oder **stress** gesetzt werden und die **apiEndPoint** Variable der jeweiligen Seite aktiviert werden. Dadurch ist es zudem möglich noch schneller, weitere Oberflächen mit den selben Funktionalitäten für andere Bereiche zu erstellen. Werden bestimmte Funktionalitäten aber nicht benötigt, können diese auch durch eine Abfrage der **platform** Variable deaktiviert werden. Dadurch kann die Oberfläche mit dem selben Programm, je nach Plattform, individuell erstellt und gestaltet werden.
Zum Schluss muss noch in der **index.html** der Titel und das Logo der Website angepasst werden. Der Titel wird im **title** Element festgelegt und das Logo im **link** Element. Für Stress wird das Logo **stresskopf.png** und für Tinnitus **tinnitus.jpg** verwendet.

7.3 API

Die Funktionalität der API wurde bereits in Kapitel 4.2.1 und die verwendeten Funktionen in Kapitel 6 beschrieben. Jede der vier HTML-Methoden wird nachfolgend genauer erklärt.

7.3.1 GET

Mittels GET-Methode können Daten vom Server angefordert werden. Zu sehen in Listing 7.18. Bei dieser Methode werden genauere Informationen zu einem Fragebogen mit der übergebenen **questionnaireID** geladen. Der **token** ist ein Schlüssel des Benutzers, welcher beim Einloggen geladen und für API-Anfragen benötigt wird. Dieser Schlüssel dient der Überprüfung, ob der Benutzer eingeloggt ist und wird zur Übermittlung an das Ende der **URL** gesetzt. Der **header** übermittelt die Sprache des Benutzers, zum Beispiel bei den Fragebögen in welcher Sprache die Elemente geladen und zurückgegeben werden sollen. Danach wird die URL aufgebaut, auf welche die Anfrage laufen soll. Zum Schluss wird eine **http.get** Methode auf die angegebene URL und dem erstellten header aufgerufen. Als Antwort erhält man dann eine Datei im JSON-Format, welche dann von der Funktion zurück gegeben wird.

```
1 getQuestionnaire(questionnaireID: number) {
2     const token = localStorage.getItem('token');
3     const headers = new HttpHeaders().set('Accept-Language', this.translate.
          currentLang);
4     const questionnaireStructureURL = this.globals.apiEndPoint + '/api/v1/
          questionnaires/' + questionnaireID + '?token=' + token;
5     return this.http.get(questionnaireStructureURL, {observe: 'response',
          headers: headers});
6   }
```

Listing 7.18: API GET

7.3.2 POST

Die POST-Methode besitzt ähnliche Elemente wie die GET-Methode. Zu sehen ist dies in Listing 7.19. Es wird ebenfalls der **token** des Benutzers geladen, sowie die momentan ausgewählte Sprache der Website. Nachdem die URL erzeugt wurde, wird hier nun zusätzlich eine JSON-Datei erstellt. Die JSON-Datei enthält je nach API-Methode unterschiedliche Elemente. Beim Senden der Antworten eines Fragebogens, werden neben dem Typ, auch das Datum, die ausgewählte Sprache, die Antworten des Benutzers und Informationen des Clients benötigt. Anschließend wird wieder der **header** erstellt, jedoch diesmal mit der Information, dass eine JSON-Datei mit übertragen werden soll. Zum Schluss wird eine **http.post** Methode mit der URL, dem header und der JSON-Datei in JSON-String Notation aufgerufen. Zurückgegeben wird auch hier eine Datei im JSON-Format, welche angibt ob alles erfolgreich ausgeführt wurde oder ob es zu Problemen kam.

```
1  postAnswersheet(questionnaireID: number, answers, clientInfo) {
2      const token = localStorage.getItem('token');
3      const currentLan = this.translate.currentLang;
4      const submitAnswerURL = this.globals.apiEndPoint + '/api/v1/questionnaires/
           ' + questionnaireID + '/answersheets?token=' + token;
5
6      const reqJsonData = {
7        'data' : {
8          'type' : 'answersheets',
9          'attributes' : {
10           'collected_at' : Math.floor(Date.now() / 1000),
11           'locale' : currentLan,
12           'answers' : answers,
13           'client' : clientInfo
14         }
15       }
16     };
17     const headers = new HttpHeaders().set('Content-Type', 'application/json');
18     return this.http.post(submitAnswerURL, JSON.stringify(reqJsonData), {
           observe: 'response', headers: headers});
19   }
```

Listing 7.19: API POST

7.3.3 PATCH

Die PATCH-Methode ist identisch zur POST-Methode. Zu sehen in Listing 7.20. Die einzigen Unterschiede sind, dass im angegeben Beispiel nicht die ausgewählte Sprache der Website benötigt wird und das anstatt einer **http.post** Methode eine **http.patch** Methode ausgeführt wird.

```
1  updatePassword(password: string, passwordConf: string) {
2      const token = localStorage.getItem('token');
3      const updatePasswordURL = this.globals.apiEndPoint + '/api/v1/my/profile/
           password?token=' + token;
4      const reqJsonData = {
5        'data' : {
6          'type' : 'users',
7          'attributes' : {
```

```
8             'password' : password,
9             'password_confirmation' : passwordConf
10          }
11        }
12      };
13      const headers = new HttpHeaders().set('Content-Type', 'application/json');
14      return this.http.patch(updatePasswordURL, reqJsonData, {observe: 'response'
            , headers: headers});
15    }
```

Listing 7.20: API PATCH

7.3.4 DELETE

Die DELETE-Methode ist am einfachsten aufgebaut und in Listing 7.21 zu sehen. Im Beispiel soll eine Notiz gelöscht werden. Dafür wird lediglich die ID der Notiz und der **token** des Benutzers benötigt. Nachdem die URL erstellt wurde, wird eine **http.delete** Methode darauf aufgerufen und ähnlich wie bei POST und PATCH eine Datei im JSON-Format zurückgegeben. Die wiederum angibt ob alles erfolgreich ausgeführt wurde oder dabei Probleme aufgetreten sind.

```
1  deleteNote(noteID: number){
2      const token = localStorage.getItem('token');
3      const myNotesUrl = this.globals.apiEndPoint + '/api/v1/notes/' + noteID + '
           ?token=' + token;
4      return this.http.delete(myNotesUrl, {observe: 'response'});
5    }
```

Listing 7.21: API DELETE

7.4 Sprachen

Die Oberfläche unterstützt momentan drei verschiedene Sprachen:

- Deutsch
- Englisch
- Niederländisch

Gespeichert sind die jeweiligen Texte und Wörter in drei verschiedenen JSON-Dateien im **assets** Ordner des Programms. Die Dateien besitzen bisher jeweils mehr als 800 verschiedene Elemente. Aufgebaut sind die Elemente in den JSON-Dateien wie folgt:

```
1  "Bezeichner" : "Text"
```

Listing 7.22: Sprachen JSON

Der **Bezeichner** wird hierbei für das Aufrufen des Elements verwendet. Dabei wird aber unterschieden ob das Element in HTML oder TypeScript aufgerufen wird:

```
1  // HTML
2  {{ 'Bezeichner' | translate }}
```

Listing 7.23: Sprachen Aufruf HTML

```
1 //TypeScript
2 this.translate.get("Bezeichner").subscribe((element: string) => {
3   ...
4 });
```

Listing 7.24: Sprachen Aufruf TypeScript

Um die **translate** Komponente verwenden zu können muss der **TranslateService** von **@ngx-translate/core** importiert werden. Die wichtigsten Funktionen der translate Komponente sind hierbei:

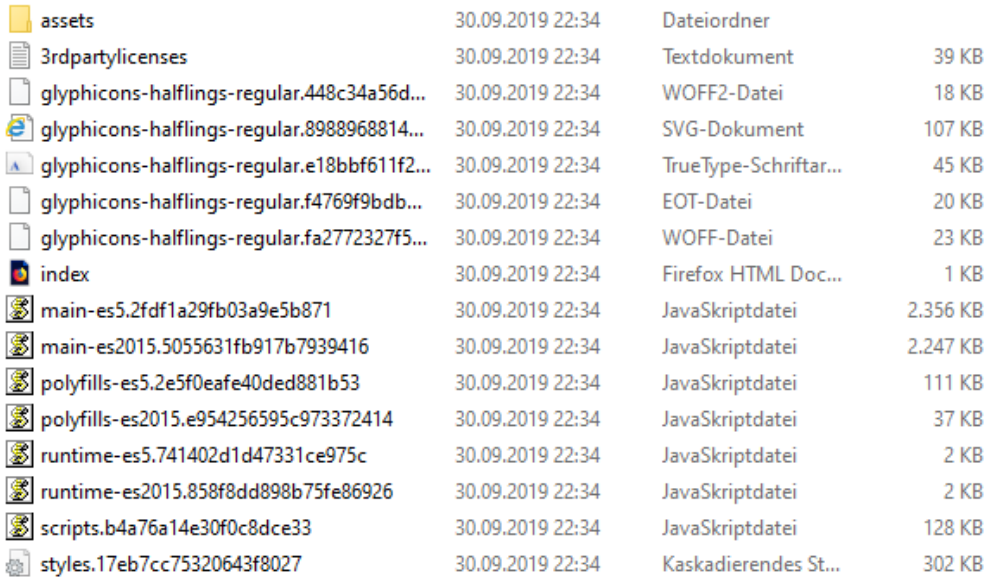
```
1 translate.addLangs(['en', 'de', 'nl']);
2
3 translate.setDefaultLang('en');
4
5 translate.use(translate.getBrowserLang());
```

Listing 7.25: Translate Komponente

Die Funktion in Zeile 1 definiert die verschiedenen Sprachen und Abkürzungen, um die entsprechenden Sprachen anzusprechen. Die einzelnen JSON-Dateien heißen dann wie die angegebenen Abkürzungen **en.json**, **de.json** und **nl.json**.
Mit der Funktion in Zeile 3 und 5 kann die vordefinierte oder momentan ausgewählte Sprache anhand einer der fest angegebenen Abkürzungen festgelegt werden. Zusätzlich kann die Sprache auch durch die Sprache des verwendeten Browsers zugeordnet werden.

7.5 Deployment

Um das Programm auf einen Webserver zu laden benötigt es noch einen Schritt zuvor. Der Befehl **ng build –prod** muss in der Konsole des Programmordners ausgeführt werden. Wurde der Befehl ohne Fehler ausgeführt, so entstand ein neuer Ordner mit dem Namen **dist** im Hauptverzeichnis des Programms. In diesem befindet sich ein Unterordner mit dem Titel des erstellten Angular Projekts und darin befinden sich die compilierten Dateien für den Webserver. Zu sehen in Abbildung 7.2. Diese Dateien müssen jetzt nur noch auf den Webserver geladen werden.



assets	30.09.2019 22:34	Dateiordner	
3rdpartylicenses	30.09.2019 22:34	Textdokument	39 KB
glyphicons-halflings-regular.448c34a56d...	30.09.2019 22:34	WOFF2-Datei	18 KB
glyphicons-halflings-regular.8988968814...	30.09.2019 22:34	SVG-Dokument	107 KB
glyphicons-halflings-regular.e18bbf611f2...	30.09.2019 22:34	TrueType-Schriftar...	45 KB
glyphicons-halflings-regular.f4769f9bdb...	30.09.2019 22:34	EOT-Datei	20 KB
glyphicons-halflings-regular.fa2772327f5...	30.09.2019 22:34	WOFF-Datei	23 KB
index	30.09.2019 22:34	Firefox HTML Doc...	1 KB
main-es5.2fdf1a29fb03a9e5b871	30.09.2019 22:34	JavaSkriptdatei	2.356 KB
main-es2015.5055631fb917b7939416	30.09.2019 22:34	JavaSkriptdatei	2.247 KB
polyfills-es5.2e5f0eafe40ded881b53	30.09.2019 22:34	JavaSkriptdatei	111 KB
polyfills-es2015.e954256595c973372414	30.09.2019 22:34	JavaSkriptdatei	37 KB
runtime-es5.741402d1d47331ce975c	30.09.2019 22:34	JavaSkriptdatei	2 KB
runtime-es2015.858f8dd898b75fe86926	30.09.2019 22:34	JavaSkriptdatei	2 KB
scripts.b4a76a14e30f0c8dce33	30.09.2019 22:34	JavaSkriptdatei	128 KB
styles.17eb7cc75320643f8027	30.09.2019 22:34	Kaskadierendes St...	302 KB

Abbildung 7.2: Dateien Webserver

8 Walkthrough

Auf der Website gibt es drei verschiedene Arten von Benutzern, welche nachstehend im Bezug auf ihre Navigationsmöglichkeiten, sprich Funktionsmöglichkeiten, genauer beschrieben werden.

8.1 Nicht registrierter Benutzer

Der nicht registrierte Benutzer ist in seinen Navigationsmöglichkeiten sehr eingeschränkt. Zu sehen sind seine Navigationsmöglichkeiten in Abbildung 8.1. Ausgehend von der Startseite, welche einen Begrüßungstext und Verlinkungen zu den beteiligten Universitäten besitzt, kann der nicht registrierte Benutzer:

- Den Datenschutz der Website anschauen.
- Die FAQs (Frequently Asked Questions) anschauen, also die meist gestellten Fragen der Website.
- Das Impressum der Website anschauen.
- Die Überseite anschauen, welche Informationen über das Thema der Website, das Projekt und das Team beinhaltet.
- Zwischen deutscher, englischer und niederländischer Oberfläche wechseln.
- Die Website auf Facebook liken und teilen, einen Tweet für Twitter erstellen oder auf Linkedin teilen.
- Sich registrieren um ein registriertes Benutzerprofil zu erstellen. In diesem Fall muss der Benutzer eine E-Mailadresse, einen Benutzernamen und ein Passwort hinterlegen. Danach erhält er eine E-Mail an seine angegebene E-Mailadresse, welche einen Bestätigungslink enthält den der Benutzer betätigen muss um sein Konto zu aktivieren. Anschließend kann er sich anmelden um zum registrierten Benutzerprofil zu wechseln 8.2.
- Sich auf der Seite anmelden um zu einem registrierten Benutzerprofil zu wechseln. Hier kann er auch im Falle eines Passwortverlusts sein Passwort zurücksetzen lassen.

Zusammenfassend kann also werden, dass die Funktionsmöglichkeiten des nicht registrierten Benutzers sehr eingeschränkt sind.

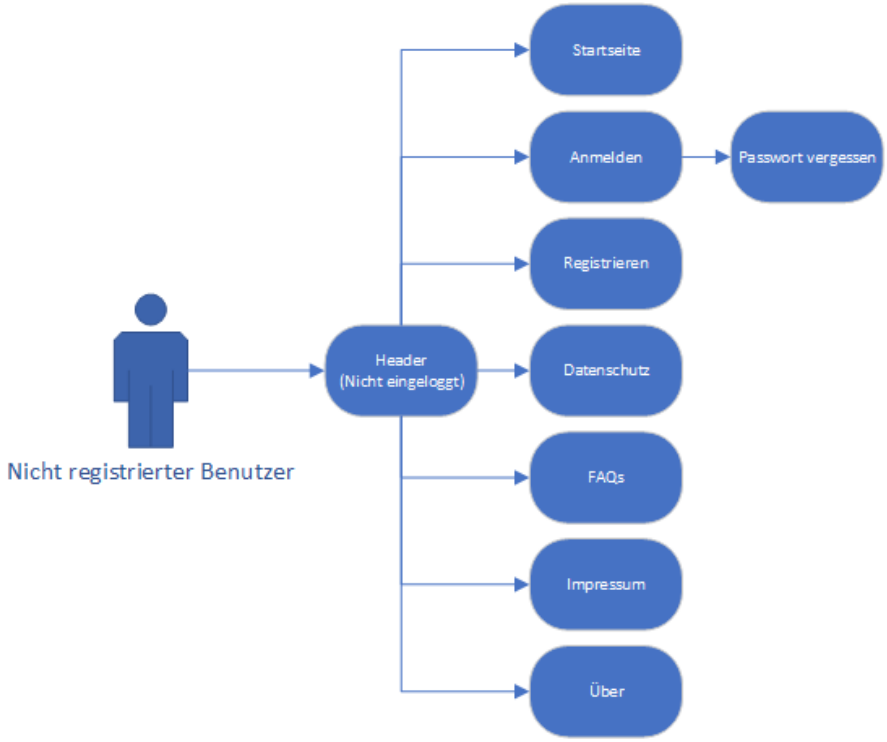


Abbildung 8.1: Navigation nicht registrierter Benutzer

8.2 Registrierter Benutzer

Der registrierte Benutzer hat nach erfolgreicher Registrierung, im Vergleich zu einem nicht registrierten Benutzer, viele zusätzliche Navigationsmöglichkeiten. Zu sehen in Abbildung 8.9. Neben einer neuen Seite mit allgemeinen Tipps, ist nun auch der Benutzerbereich mit nachfolgenden Elementen freigeschaltet:

- **Meine Aktivitäten**
  Ein Ereignisprotokoll des Benutzers mit allen Aktivitäten seit der Registrierung mit Zeitstempel und Beschreibung.
- **Meine Ergebnisse**
  Nach der Auswahl einer Studie und eines spezifischen Fragebogens daraus, werden dem Benutzer die Ergebnisse seiner Antwortbögen präsentiert. Einmalige Fragebögen, wie der Registrierungsfragebogen der Studie, enthalten dabei nur die jeweiligen Antworten des Benutzers zu den Fragen. Fragebögen die mehrfach ausgefüllt werden können, erhalten zusätzlich graphische Darstellungen, wie Linien- oder Balkendiagramm, um die Resultate anschaulicher darzustellen. Zwei Beispiele befinden sich nachstehend in den Abbildungen 8.2 und 8.3. Zusätzlich ist es dem Besitzer möglich die Ergebnisse als CSV/Excel-Datei herunterzuladen.

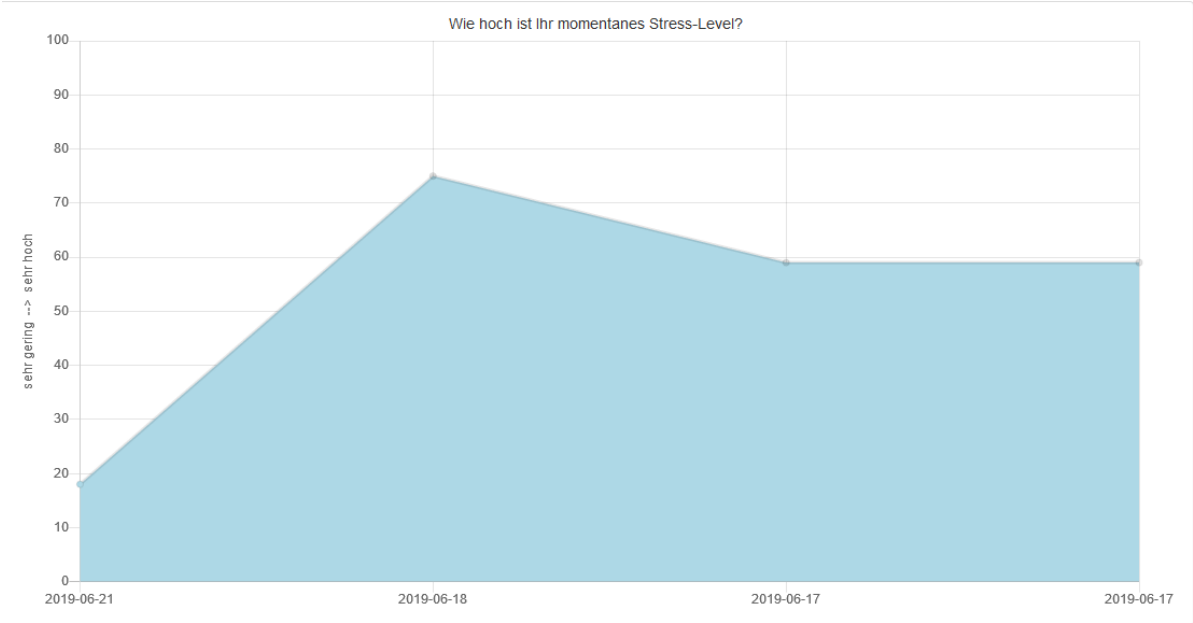


Abbildung 8.2: Liniendiagramm Ergebnisseite registrierter Benutzer

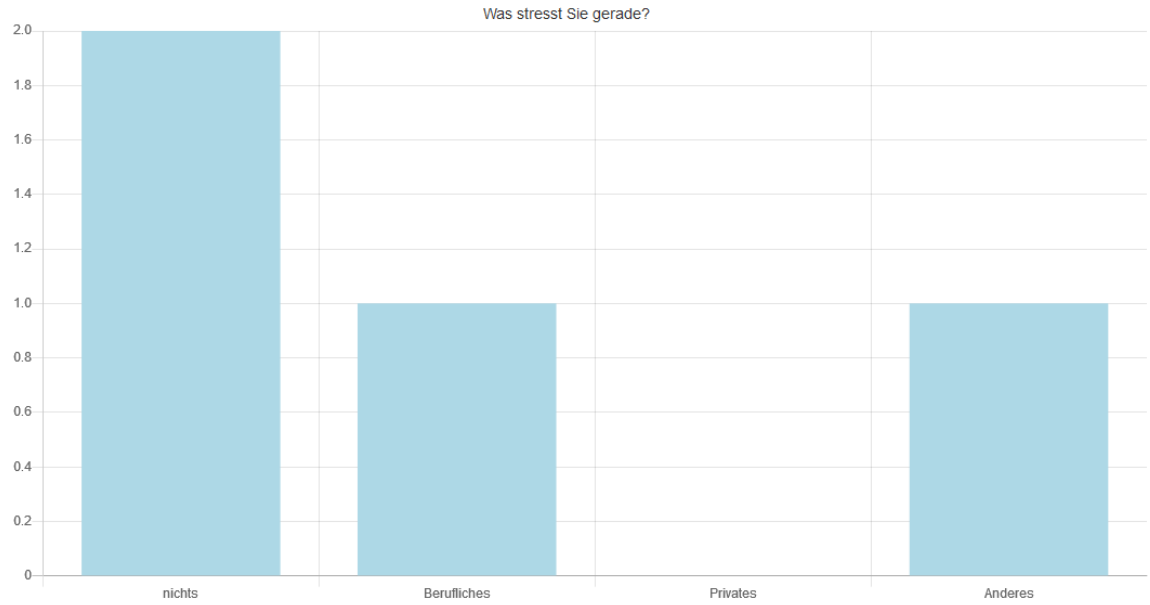


Abbildung 8.3: Balkendiagramm Ergebnisseite registrierter Benutzer

- **Meine Fragebögen**
  Das wichtigste Element der Website, eine Auflistung aller offenen Fragebögen. Hier kann der registrierte Benutzer Fragebögen ausfüllen. Die Fragen werden in Form von Texteingaben, einstellbaren Slidern (Schiebereglern), Single- / Multiplechoice oder der Auswahl eines Datums angezeigt. Ein Ausschnitt der Oberfläche ist in Abbildung 8.4 zu sehen.

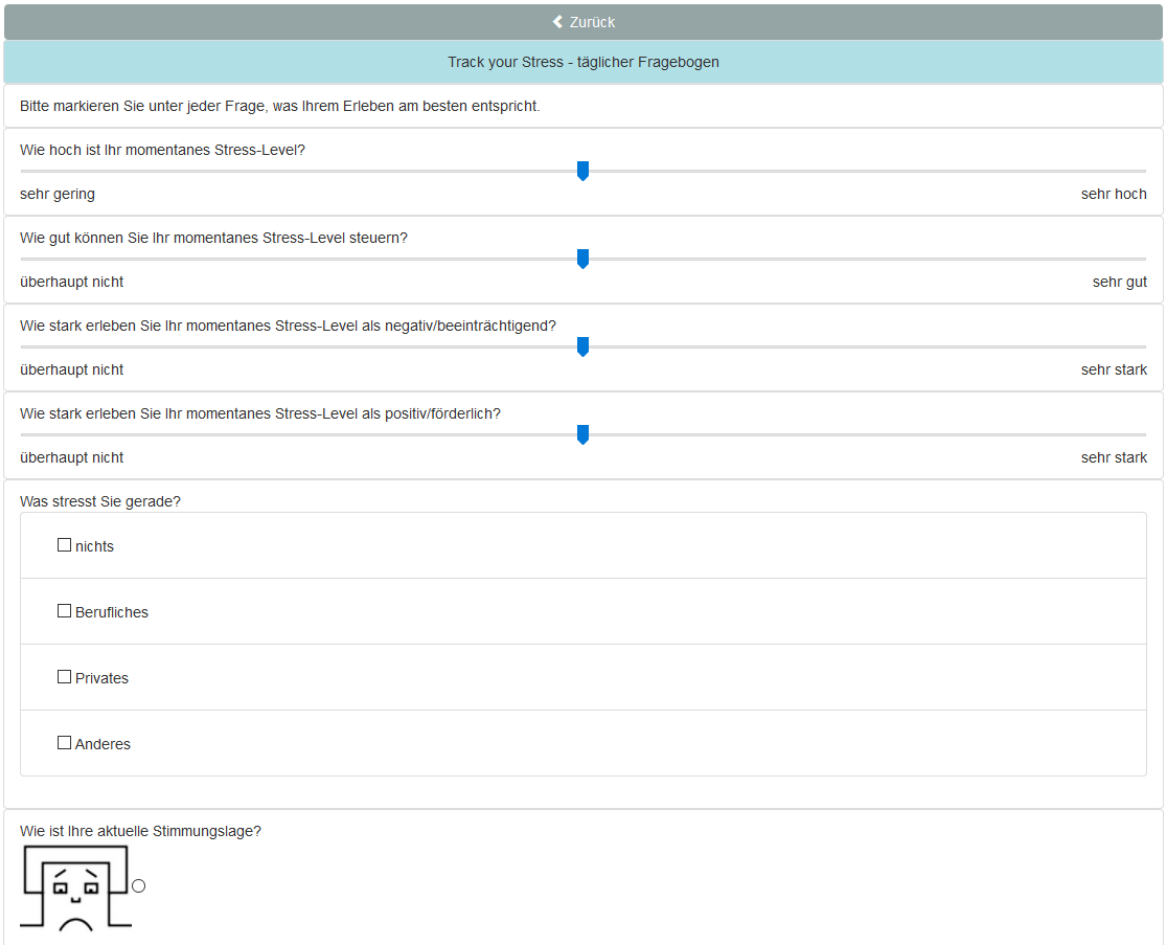


Abbildung 8.4: Ausschnitt Oberfläche Fragebogen

- **Meine Kontextdaten**
  Hier hat der Benutzer die Möglichkeit verschiedene Accounts mit seinem Profil zu verknüpfen. Aktuell ist die Verlinkung mit Deezer, Netflix, Spotify und Twitter möglich. Erlaubt der Benutzer das Abrufen verschiedener Profilinformationen, wie Suchkategorien oder Favoriten, so können neue Informationen über mögliche Hilfen oder Warnungen ermittelt und an den Benutzer weitergegeben werden.

  Wichtiger Hinweis: Diese Funktionalität ist oberflächlich vorhanden, aber noch nicht mit der API verknüpft, da diese noch keine passende Funktionalität zur Verfügung stellt. Daher können momentan noch keine Accounts mit dem Benutzerprofil verknüpft werden.

- **Meine Nachrichten**
  In diesem Bereich werden alle Anfragen des registrierten Benutzers an die Studienadministration angezeigt. Zu sehen in Abbildung 8.5. Der registrierte Benutzer kann eine Anfrage zu einer zutreffenden Studie mit einem Anfragethema und einer dazugehörigen Nachricht erstellen. Die Anfrage wird dann an die zugehörige Administration der Studie weitergeleitet und kann im Chatverlauf eingesehen werden (Abbildung 8.6). Der registrierte Benutzer kann nach erfolgreicher Hilfe die Anfrage, falls diese nicht mehr benötigt wird, auch wieder löschen.

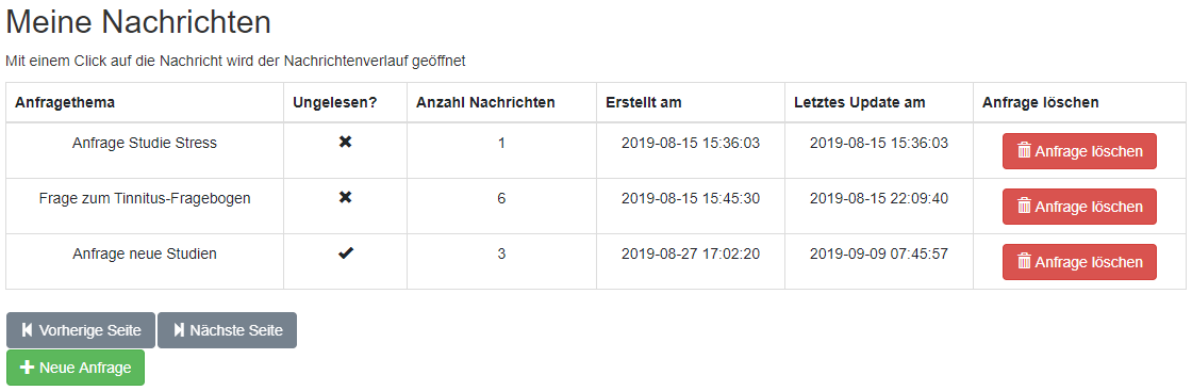


Abbildung 8.5: Oberfläche meine Nachrichten

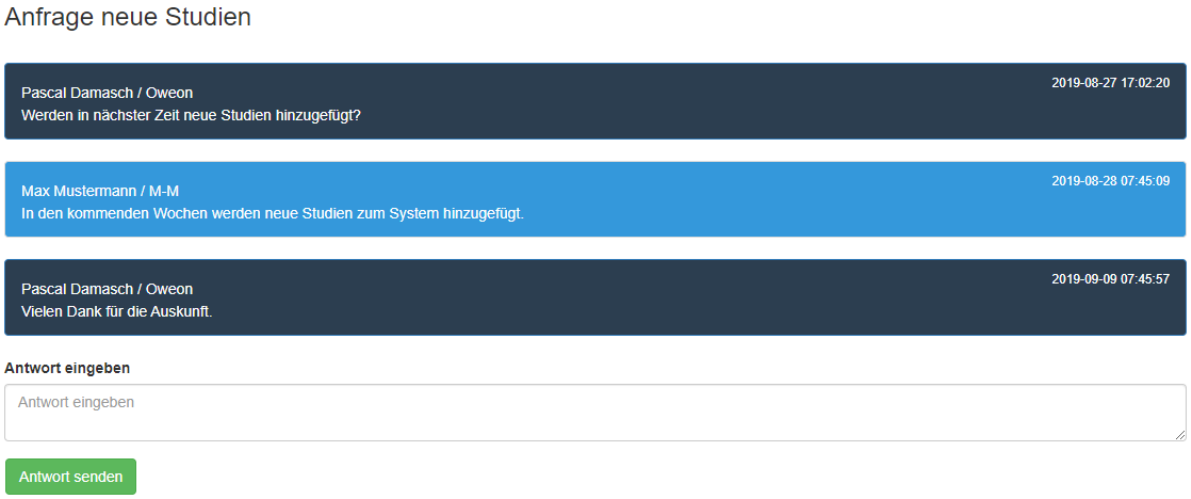


Abbildung 8.6: Oberfläche Nachrichtenverlauf unter meine Nachrichten

- **Meine Notizen**
  Der registrierte Benutzer hat zudem die Möglichkeit eigene Notizen zu erstellen, zu bearbeiten, als erledigt zu markieren oder zu löschen. Die Oberfläche ist in Abbildung 8.7 zu sehen.

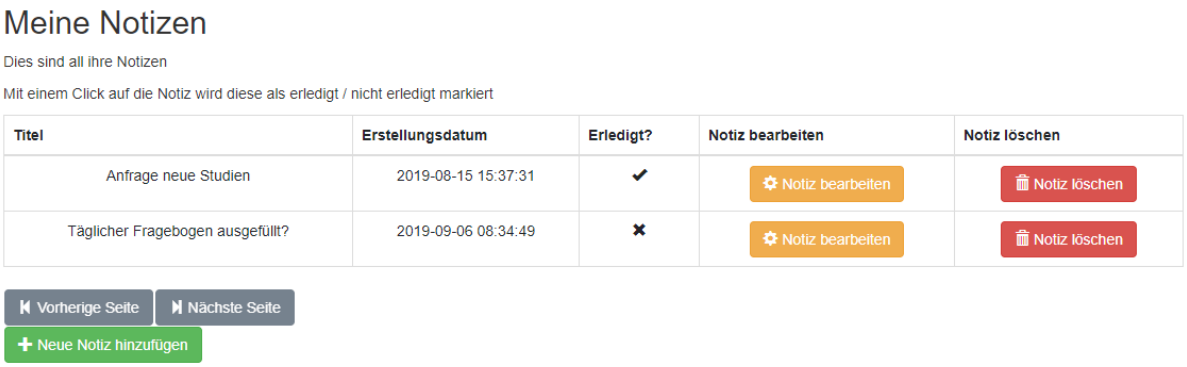


Abbildung 8.7: Oberfläche meine Notizen

- **Mein Profil**
  Auf dieser Seite werden dem registrierten Benutzer verschiedene Informationen zu seinem Profil angezeigt:
  - Nickname
  - Email Adresse
  - Vorname
  - Nachname
  - Geschlecht

  Außerdem kann hier das Passwort geändert und das eigene Profil angepasst werden.
- **Meine Statistiken**
  In Abbildung 8.8 zu sehen. Hier werden dem registrierten Benutzer verschiedene Informationen zur allgemeinen Website und sich selber angezeigt:
  - **Allgemeine Informationen**
    - Anzahl der registrierten Benutzer
    - Anzahl vorhandener Studien
    - Anzahl vorhandener Fragebögen
  - **Benutzerbezogene Informationen**
    - Anzahl Aktivitäten
    - Anzahl ausgefüllter Fragebögen
    - Anzahl beigetretener Studien im Vergleich zu allen vorhandenen Studien
    - Anzahl ausgefüllter Fragebögen im Vergleich zu allen vorhandenen Fragebögen (Jeder Fragebogen zählt hier nur einmal)
    - Anzahl aller erledigten Notizen im Verhältnis zu allen vorhandenen Notizen

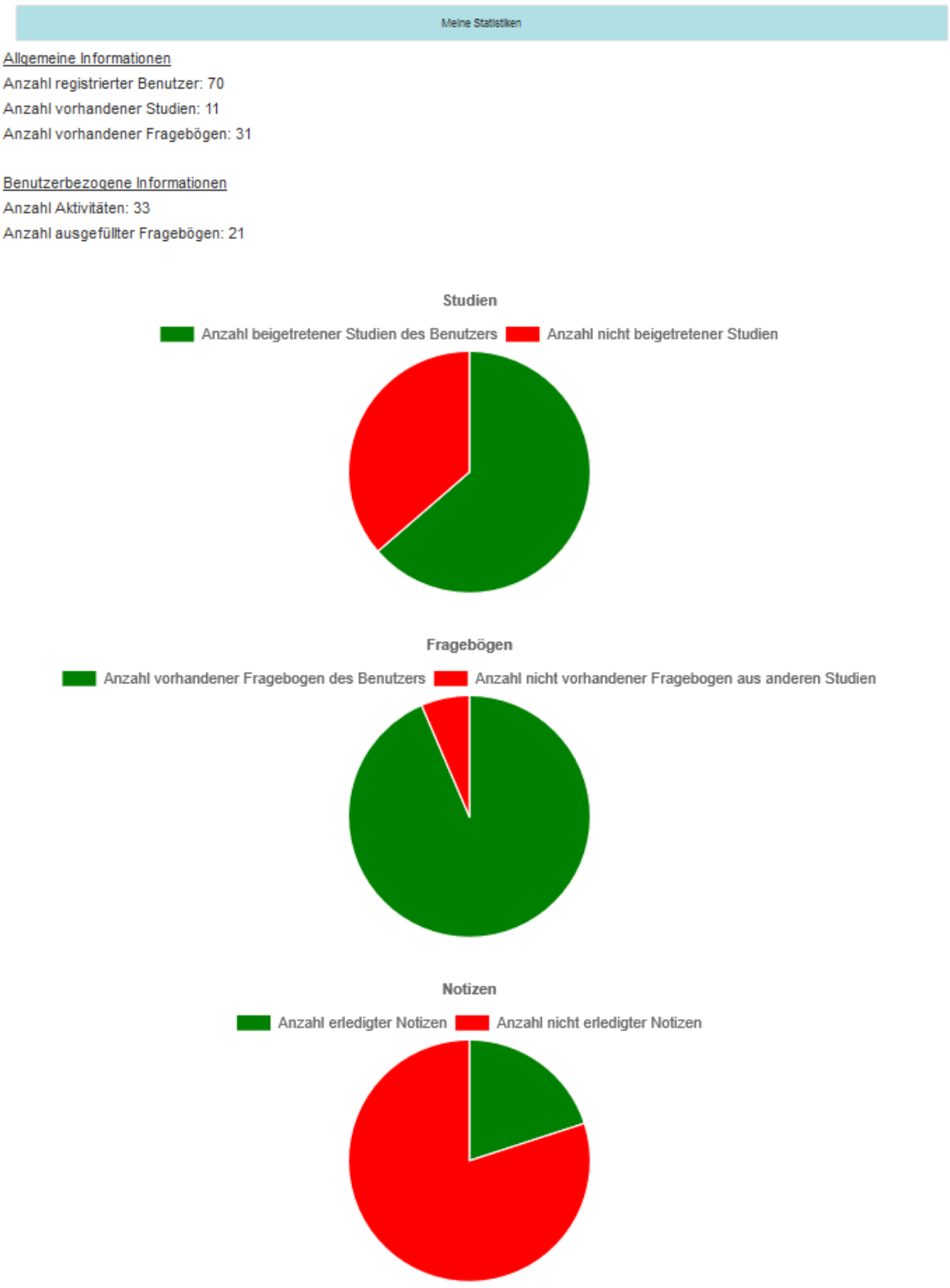


Abbildung 8.8: Oberfläche meine Statistiken

- **Meine Studien**
  Auf dieser Seite werden dem registrierten Benutzer alle seine beigetretenen Studien angezeigt. Zudem kann er hier neuen Studien beitreten, sich von beigetretenen Studien abmelden oder Errungenschaften aktivieren. Wenn die Errungenschaften aktiviert worden sind, erhält der Benutzer für jeden ausgefüllten Fragebogen einer Studie Punkte. Diese Punkte werden dann in der Bestenliste angezeigt und mit anderen registrierten Benutzern der Studie verglichen.

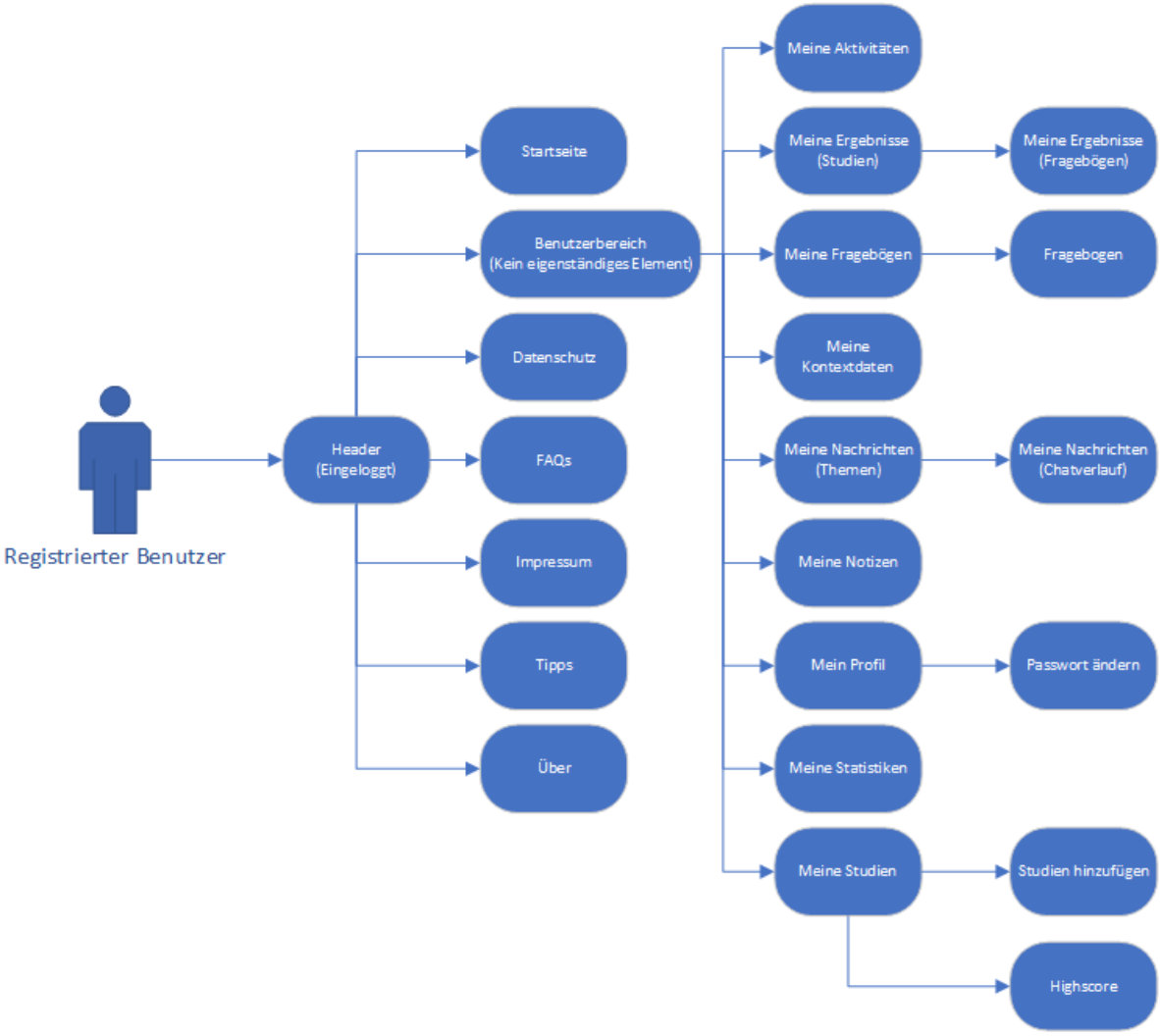


Abbildung 8.9: Navigation registrierter Benutzer

8.3 Editor

Der Editor besitzt alle Funktionalitäten des registrierten Benutzers, sowie weitere zusätzliche Funktionen und Navigationsmöglichkeiten. Diese werden in der Abbildung 8.13 in grün dargestellt. Der Editor hat zudem Zugriff auf eine globale Statistikseite über alle Studien und Fragebögen und besitzt zudem eine eigene Verwaltungsseite für die vorhandenen Studien.

Statistikseite

Die Statistikseite, welche in Abbildung 8.10 zu sehen ist, listet folgende Informationen auf:

- **Allgemeine Informationen**
  - Anzahl vorhandener Studien
  - Anzahl vorhandener Fragebögen
  - Anzahl ausgefüllter Fragebögen durch die Benutzer
  - Durchschnittliche Anzahl ausgefüllter Fragebögen der Benutzer
- **Benutzerbezogene Informationen**
  - Anzahl aller registrierten Benutzer
  - Durchschnittliches Alter der Benutzer
  - Standartabweichung beim Alter der Benutzer
- **Google Maps Karte mit Standorten**
  - Graphische Darstellung der ungefähren Standorte der Benutzer zum Zeitpunkt als der Registrierungsfragebogen ausgefüllt wurde
- **Graphische Darstellung im Balken-Diagramm**
  - Auflistung aller beteiligten Länder mit Anzahl
- **Graphische Darstellung im Donut-Diagramm**
  - Aufteilung der Geschlechter (Männlich, Weiblich, Keine Angabe)
  - Aufteilung der Plattformen (iOS, Android, Computer)

Diese Informationen kann der Editor auch als CSV/Excel-Datei herunterladen.

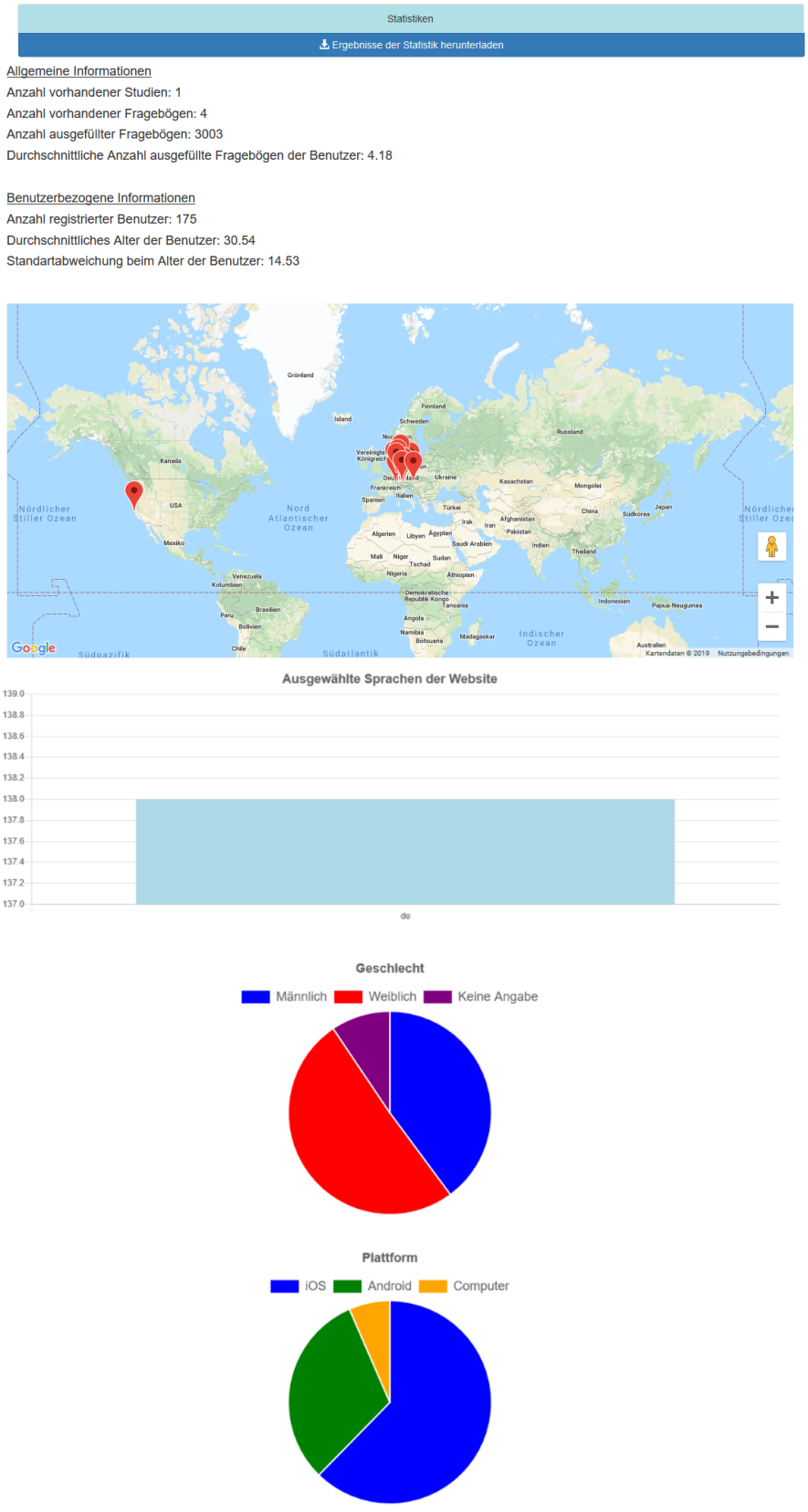


Abbildung 8.10: Oberfläche Statistiken

Verwaltung Studien

Die zweite zusätzliche Funktion ist das Verwalten der Studien. Hier besitzt der Editor unterschiedliche Funktionsmöglichkeiten. Direkt auf der Seite **Studien verwalten** werden dem Editor alle vorhandenen Studien angezeigt. Hier kann der Editor neue Studien erstellen und bestehende Studien bearbeiten. Zu sehen ist in Abbildung 8.11 die Oberfläche zum Erstellen einer neuen Studie. Hier muss der Editor den **Titel** und die **Beschreibung** der Studie auf deutsch, englisch und niederländisch angeben, da die Oberfläche diese drei Sprachen bereitstellt. Zusätzlich kann noch ein **Passwort** festgelegt werden, welches der Benutzer beim beitreten zur Studie eingeben muss. Die Oberfläche zur Bearbeitung einer bestehenden Studie ist identisch zur Oberfläche beim Erstellen einer neuen Studie.

Wichtiger Hinweis: Diese Funktionalitäten sind oberflächlich vorhanden, aber noch nicht mit der API verknüpft, da diese keine passende Funktionalität zur Verfügung stellt um neue Studien zu erstellen oder bestehende Studien zu bearbeiten. Daher können momentan noch keine Studien über die Website erstellt oder bearbeitet werden.

Abbildung 8.11: Oberfläche Studie erstellen

Verwaltung Fragebögen

Die letzte zusätzliche Funktion ist das Verwalten von Fragebögen zu den einzelnen Studien. Der Editor wählt eine der vorhandenen Studien aus und erhält dann eine Auflistung aller Fragebögen der Studie. Der Editor hat jetzt verschiedene Möglichkeiten:

- **Neuen Fragebogen zur Studie hinzufügen**
  Der Editor kann neue Fragebögen zur Studie hinzufügen. Zu sehen ist ein Beispiel in Abbildung 8.12. Beim Erstellen stehen folgende Elemente zur Verfügung:
  - Der Titel des Fragebogens

- Soll der Fragebogen einmalig oder mehrfach ausfüllbar sein. Bei mehrfach kann zusätzlich noch angegeben werden, in welchen Intervallen der Fragebogen ausgefüllt werden soll
  - Täglich
  - Wöchentlich
  - Monatlich
- Anzahl der Punkte für den Benutzer wenn dieser den Fragebogen ausfüllt. Diese Punkte werden dann nach jedem Ausfüllen summiert und auf der Bestenliste angezeigt
- Ab hier können jetzt Elemente frei wählbar hinzugefügt werden. Enthalten sind die folgenden Frageelemente:
  - SingleChoice
    Eine Frage mit mehreren vorgegebenen Antwortmöglichkeiten, wobei nur eine Antwort ausgewählt werden kann.
  - MultipleChoice
    Eine Frage mit mehreren vorgegebenen Antwortmöglichkeiten. Bei MultipleChoice können gegenüber SingleChoice mehrere Antworten ausgewählt werden.
  - Slider
    Eine Frage mit einem Schieberegler als Auswahlmöglichkeit. Der Editor legt beim Erstellen fest was das Minimum und Maximum ist und die jeweiligen Beschriftungen für diese.
  - TextString
    Eine Frage mit einer Antwort in Textform.
  - TextDate
    Eine Frage welche mit einem Datum beantwortet wird. Das Datum kann hierbei in einem Kalender ausgewählt werden.
  - YesNoSwitch
    Eine Frage mit den Antwortmöglichkeiten ja oder nein.
  - SAMScaleFace
    Eine Frage mit neun verschiedenen Antwortmöglichkeiten, wobei jede Antwort für ein bestimmtes Wohlbefinden des Benutzers steht. Oberflächlich werden dem Benutzer dabei verschiedene Gesichter angezeigt, welche jeweils das aktuelle Wohlbefinden darstellen.
  - SAMScaleBody
    Eine Frage mit neun verschiedenen Antwortmöglichkeiten, wobei jede Antwort für ein bestimmtes Ruhelevel des Benutzers steht. Oberflächlich werden dem Benutzer dabei verschiedene Körper angezeigt, welche jeweils das aktuelle Ruhelevel darstellen.
  - Text
    Dieses Element ist an sich kein Frageelement. Es kann verwendet werden um

Bereiche zu deklarieren indem eine Überschrift und ein Text hinzugefügt werden können.

Wichtig: Da eine deutsche, englische und niederländische Oberfläche unterstützt wird, muss beim Erstellen des Fragebogens auch jedes Element in jeder Sprache angegeben werden. Durch den Tab mit den Bereichen **Deutsch**, **Englisch** und **Niederländisch** wechselt man zur jeweiligen Sprache. Entfernt oder bearbeitet man ein Element wird die Änderung in allen Tabs gleichermaßen behandelt. Dadurch wird die Struktur in jeder Sprache aktuell und für den Editor das Erstellen so einfach wie möglich gehalten.

- **Vorhandenen Fragebogen bearbeiten**
  Nach dem Auswählen eines Fragebogens kann dieser bearbeitet werden. Die Oberfläche sieht hierbei genau so aus wie beim Erstellen eines neuen Fragebogens und besitzt den selben Funktionalitätsumfang.
- **Fragebögen für Benutzer freischalten**
  Durch das Aktivieren eines Fragebogens wird dieser für die Benutzer freigeschaltet und kann nun von Benutzers ausgefüllt werden.
- **Fragebögen löschen**
  Der ausgewählte Fragebogen wird gelöscht.

Wichtiger Hinweis: Diese Funktionalitäten sind oberflächlich vorhanden, aber noch nicht mit der API verknüpft, da diese noch eingeschränkt ist und nicht alle Informationen übernehmen kann. Daher können momentan noch keine Fragebögen über die Website erstellt und bearbeitet werden.

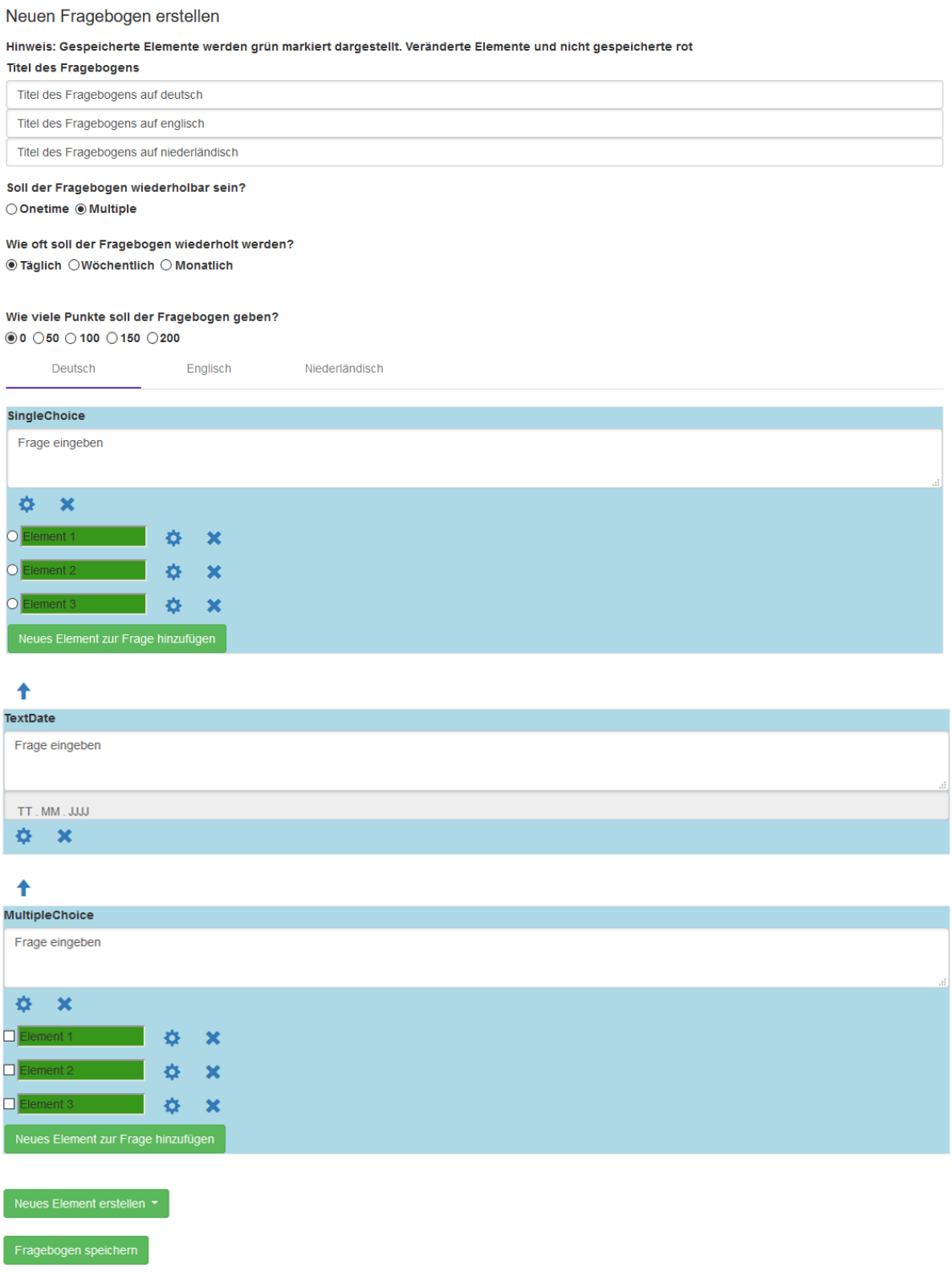


Abbildung 8.12: Oberfläche Fragebogen erstellen

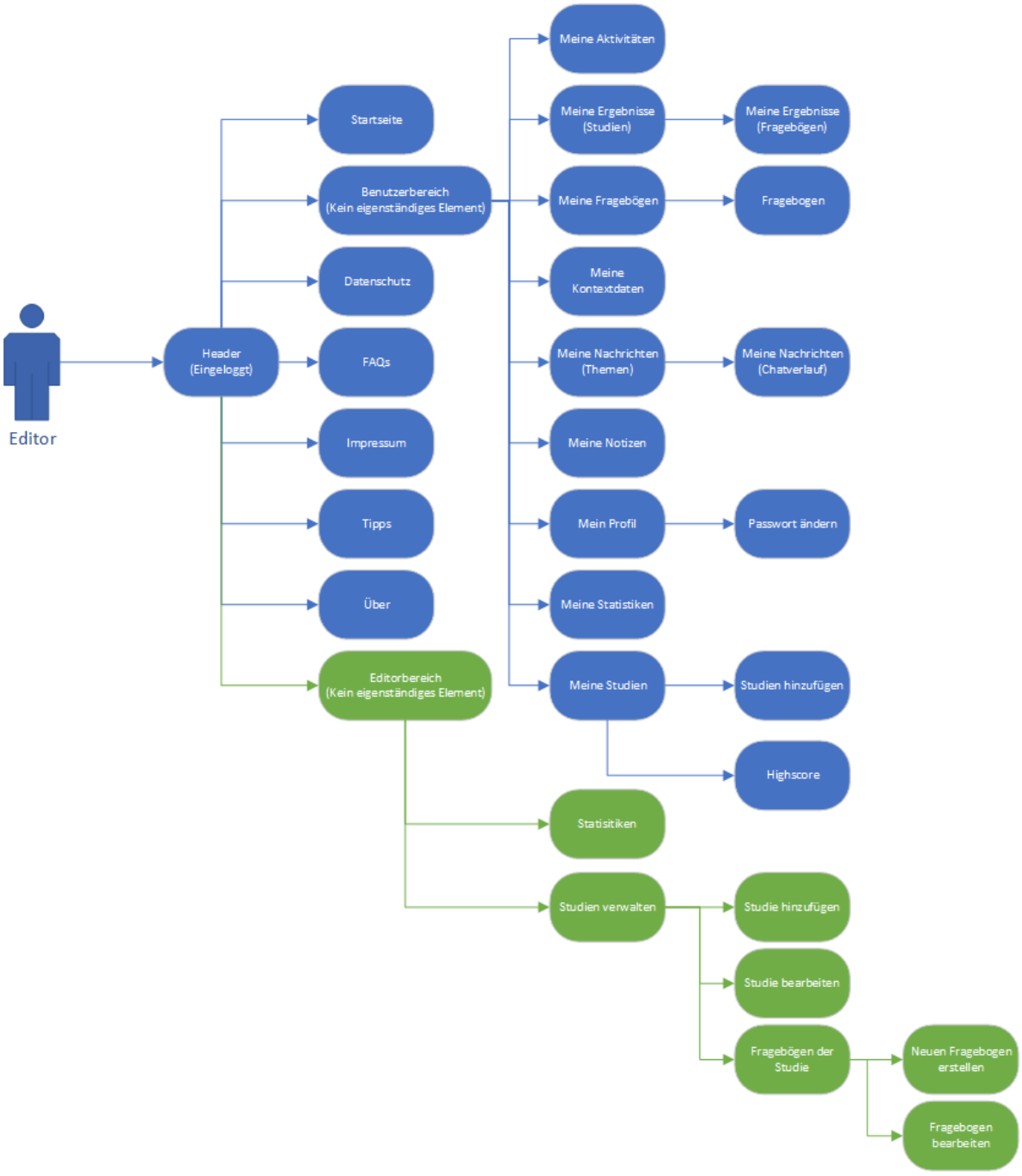


Abbildung 8.13: Navigation Editor

9 Abgleich der Anforderungen

In diesem Kapitel werden die Anforderungen aus Kapitel 3 mit dem entwickelten System verglichen.

9.1 Abgleich funktionale Anforderungen

Abkürzung	Anforderung	Wichtigkeit	Umsetzung
FA01	Antwortbögen auswerten	+	+
FA02	Datenbankanbindung	+	+
FA03	Fragebögen ausfüllen	+	+
FA04	Fragebögen verwalten	+	o
FA05	Hilfe für Benutzer	+	+
FA06	Oberflächenverwaltung	o	o
FA07	Sprachunterstützung	+	+
FA08	Statistiken	+	+
FA09	Studien erstellen	o	o

Tabelle 9.1: Tabelle der funktionalen Anforderungen mit Wichtigkeit (-/unwichtig, o/neutral, +/wichtig) und Umsetzung (-/schlecht, o/neutral, +/gut)

Wie in Tabelle 9.1 zu sehen ist wurden alle funktionalen Anforderungen, bis auf die Oberflächenverwaltung und das Erstellen von Studien, neutral bis gut umgesetzt.

FA01 Antwortbögen auswerten

Dem Benutzer wird eine Funktion zur Verfügung gestellt die es ihm erlaubt seine Antwortbögen einzusehen. Diese Antwortbögen werden je nach Frage miteinander verglichen und textuell oder grafisch dem Benutzer dargestellt. Zu sehen in Abbildung 8.2 und 8.3.

FA02 Datenbankanbindung

Als Schnittpunkt zur Datenbank dient eine bereits vorhandene API. Detaillierte Informationen dazu gab es bereits in Abschnitt 4.2.

FA03 Fragebögen ausfüllen

Dem Benutzer werden unterschiedliche Studien, mit verschiedenen Fragebögen zu spezifischen Themen angeboten. Zu sehen ist ein Fragebogen der Studie Track your Tinnitus in Abbildung 8.4.

FA04 Fragebögen verwalten

Dem Editor ist es möglich neue Fragebögen für bestehenden Studien zu erstellen, diese zu bearbeiten und wenn notwendig zu entfernen. Durch eine momentane Einschränkung seitens der API ist diese Funktionalität zwar oberflächlich vorhanden, aber noch nicht mit der API verknüpft. Dadurch können zum jetzigen Zeitpunkt keine neuen Fragebögen über die Website erstellt oder bestehende Fragebögen bearbeitet werden. Hier wird noch eine Anpassung an der API benötigt bevor diese miteinander verknüpft werden können. Daher kann dieser Funktionalität aktuell keine gute Umsetzung zugeordnet werden.

FA05 Hilfe für Benutzer

Dem Benutzer werden auf jeder Seite alle notwendigen Funktionalitäten beschrieben. Falls trotzdem Probleme auftreten sollten, kann sich der Benutzer mit der Administration in Verbindung setzen und über das Erstellen eines Themas kommunizieren.

FA06 Oberflächenverwaltung

Eine Oberfläche für den PC war gefordert und wurde umgesetzt. Die Website funktioniert auch auf mobilen Geräten, wurde hierfür aber nicht ausgelegt.

FA07 Sprachunterstützung

Eine deutsche, englische und niederländische Oberfläche ist wie gewünscht vorhanden.

FA08 Statistiken

Für den Benutzer und den Editor wurden eigene Statistikseiten hinzugefügt. Unter meine Statistiken erfährt der Benutzer relevante Informationen über die Website und sich selber. Zu sehen in Abbildung 8.8. Für den Editor wurde zudem nochmal eine weitere spezielle Statistikseite hinzugefügt, auf welcher dieser weitere Informationen über Studien und Benutzer erhält. Zu sehen ist diese in Abbildung 8.10.

FA09 Studien erstellen

Die Website ist normalerweise für ein besonderes Thema oder Studie ausgelegt. Momentan zum Beispiel TrackYourTinnitus2 oder TrackYourStress. Um Nebenstudien in diesen Bereichen durchzuführen wäre eine Funktionalität zur Erstellung von neuen Studien sinnvoll gewesen.

Durch die Einschränkung der API können zum jetzigen Zeitpunkt keine Studien erstellt werden. Es konnte zwar eine Oberfläche zur Erstellung von neuen Studien erstellt werden, jedoch ist keine Verknüpfung zur API vorhanden. Aus diesem Grund kann aktuell über die Website keine neue Studie erstellt werden und es gibt nur eine neutrale Bewertung der Umsetzung.

9.2 Abgleich nicht-funktionale Anforderungen

Abkürzung	Anforderung	Wichtigkeit	Umsetzung
NFA01	Benutzerfreundlichkeit	+	+
NFA02	Erweiterbarkeit	+	+
NFA03	Korrektheit und Zuverlässigkeit	+	+
NFA04	Performanz	+	+
NFA05	Verfügbarkeit	+	?
NFA06	Wartbarkeit	+	+

Tabelle 9.2: Tabelle der nicht-funktionalen Anforderungen mit Wichtigkeit (-/unwichtig, o/neutral, +/wichtig) und Umsetzung (-/schlecht, o/neutral, +/gut)

Wie bei den funktionalen Anforderungen wurden auch die nicht-funktionalen Anforderungen gut umgesetzt. Zu sehen in Tabelle 9.2.

NFA01 Benutzerfreundlichkeit

Die Bedienung der Website wurde versucht so einfach und intuitiv wie möglich zu gestalten. Es wurde darauf geachtet dem Benutzer nicht zu viele Informationen auf einmal zu präsentieren.

NFA02 Erweiterbarkeit

Es wurde ein Trigger zum Wechseln der Oberfläche hinzugefügt (Kapitel 7.2.5). Dieser kann momentan zwischen den Oberflächen und Funktionalitäten von Track your Tinnitus 2 und Track your Stress wechseln.

NFA03 Korrektheit und Zuverlässigkeit

Alles arbeitet soweit überprüft korrekt- und zuverlässig. Sicherlich werden aber neue Fehler nach Onlinestellung und intensiver Nutzung der Website auftreten.

NFA04 Performanz

Die Performanz ist in einem verträglichen Zustand für den Benutzer. Abhängig wird dieser Punkt aber schlussendlich vom ausgewählten Server beeinträchtigt.

NFA05 Verfügbarkeit

Die Verfügbarkeit konnte noch nicht getestet werden, da das System noch nicht auf einen Server gespielt wurde.

NFA06 Wartbarkeit

Die Implementierung wurde ausführlich kommentiert und so übersichtlich wie möglich durchgeführt. Vorhandene TODOs wurden als Kommentar mit einer Beschreibung hinzugefügt.

10 Zusammenfassung & Ausblick

Der letzte Abschnitt widmet sich in einer kurzen Zusammenfassung dem Ergebnis des Projekts und gibt zudem einen Ausblick über mögliche nächste Schritte und zukünftige Verbesserungen.

10.1 Zusammenfassung

Ziel dieser Arbeit war das Erstellen einer Webapplikation zur Beantwortung von Fragebögen zu verschiedenen Studien. Die Anforderungen an das System wurden in Kapitel 3 genauer beschrieben und in Kapitel 9 dann nach ihrer Umsetzung bewertet. Alle Anforderungen wurden wie gewünscht umgesetzt und mögliche Verbesserungen werden im letzten Abschnitt dieses Kapitels 10.2 noch genauer aufgeführt.
Nach den Anforderungen folgte ein Überblick über die Architektur des Systems in Kapitel 4. Darin wurden zuerst die Bestandteile des Frontends beschrieben, mit Angular, Node.js und TypeScript, und anschließend das Backend der Webapplikation mit der verknüpften API und der darin enthaltenen MariaDB.
Danach folgten detailliertere Informationen über den Implementierungsverlauf, eine Auflistung aller verwendeten API Funktionen mit Beschreibung und eine Beschreibung der wichtigsten Implementierungsaspekte. Die Implementierungsaspekte wurden dabei aufgeteilt in HTML-Code, TypeScript-Code, den Aufruf der API Funktionen, die Verwaltung der verwendeten Oberflächensprachen und wie das Programm schlussendlich auf einem Webserver deployed werden kann.
Nachdem der Aufbau und der funktionale Hintergrund des Systems beschrieben wurde, folgte der Abschnitt über das schlussendlich aussehende und funktionierende System. Dabei wurden der Funktionalitätsumfang der verschiedenen Rollen einzeln beschrieben. Der nicht registrierte Benutzer mit seinem sehr eingeschränkten Funktionsumfang, der registrierte Benutzer mit dem Großteil der Funktionsmöglichkeiten und der Editor mit zusätzlichen administrativen Funktionalitäten.
Zusammenfassend kann gesagt werden, dass das System mit allen Anforderungen wie gewünscht umgesetzt wurde.

10.2 Ausblick

Wie bereits in Abschnitt 9 zu sehen war wurden alle Anforderungen wie gewünscht umgesetzt. Zusätzlich hinzugefügte Funktionalitäten die noch nicht mit der API verknüpft sind müssen noch auf Anpassungen der API warten. Darunter fallen:

- **Meine Kontextdaten**
  Hier muss eine passende API Funktionalität hinzugefügt und verknüpft werden.
- **Neue Studie erstellen und bestehende Studien bearbeiten**
  Hier muss eine passende API Funktionalität hinzugefügt und verknüpft werden.
- **Neuen Fragebogen erstellen und bestehende Fragebögen bearbeiten**
  Hier muss die vorhandene API Funktionalität noch überarbeitet und angepasst werden. Anschließend muss diese dann noch verknüpft werden.

Auch nach den gerade erwähnten Anpassungen ist ein System dieser Art nie fertig gestellt. Es gibt immer Bereiche die verbessert werden und neue Ideen die implementiert werden müssen. Der nächste Schritt für das System wäre eine erste Inbetriebnahme des Systems. Ende Oktober 2019 sollen die zugehörigen Apps von Track Your Tinnitus 2 und Track Your Stress verfügbar sein. Danach soll Anfang November 2019 die hier erstellten Webseiten online gehen. Das bedeutet eine erste wirkliche Beanspruchung und Verwendung des Systems. Hier werden höchstwahrscheinlich neue Fehler auftreten, welche behoben werden müssen, und neue Einsichten und Verbesserungsvorschläge aufkommen, welche dann implementiert werden müssen. Bis dahin können noch neue Studien und Fragebögen zur Datenbank hinzugefügt werden um den Benutzern eine größere Auswahl an Themen bereitzustellen. Für weitere Implementierungen wird aber auf die Inbetriebnahme des Systems gewartet. Denn erst dann ist ersichtlich welche zusätzlichen Funktionalitäten von den Benutzern benötigt werden und welche Anpassungen an das System noch getroffen werden müssen.

Literaturverzeichnis

[1] stuttgarter-nachrichten.de. (2016). Die optimierte Stress-Gesellschaft: Warum Muße so wichtig ist. https://www.stuttgarter-nachrichten.de/inhalt.die-optimierte-stress-gesellschaft-warum-experten-musse-fuer-sehr-wichtig-halten.08041c45-6f13-449f-994a-172ab21b4281.html. 25.09.2019

[2] T-t-z.de. Tinnitus – TTZ – Tinnitus Therapie Zentrum. https://t-t-z.de/tinnitus/. 07.09.2019

[3] Fux, C. und Felchner, C. (2018). Tinnitus: Behandlung, Ursachen, Symptome. NetDoktor. https://www.netdoktor.de/symptome/tinnitus/. 07.09.2019

[4] Hno-aerzte-im-netz.de. Was ist ein Tinnitus? » Tinnitus » Krankheiten » HNO-Ärzte-im-Netz ». https://www.hno-aerzte-im-netz.de/krankheiten/tinnitus/was-ist-ein-tinnitus.html. 07.09.2019

[5] Kahle, C. (2019). Stress - Definition, Ursachen und Symptome | Meine Gesundheit. Meine-gesundheit.de. https://www.meine-gesundheit.de/krankheit/krankheiten/stress. 07.09.2019

[6] Internisten-im-netz.de. Stress » Psyche » Körper » Fachgebiete » Internisten im Netz ». https://www.internisten-im-netz.de/fachgebiete/psyche-koerper/stress.html. 07.09.2019

[7] Google Play. Track Your Tinnitus. https://play.google.com/store/apps/details?id=com. jochenherrmann.trackyourtinnitus. 08.09.2019

[8] Schrempp, Michael (2017) Konzeption und Realisierung einer mobilen Anwendung zur Erfassung des Stresslevels am Beispiel von iOS. Bachelor thesis, Ulm University. http://dbis.eprints.uni-ulm.de/1571/. 26.08.2019

[9] Haug, Julian (2017) Track Your Stress: Konzeption und Realisierung einer mobilen (Android) Anwendung zur Messung des Stresslevels. Bachelor thesis, Ulm University. http://dbis.eprints.uni-ulm.de/1572/. 26.08.2019

[10] Drilling, T. und Augsten, S. (2018). Angular und AngularJS. Dev-insider.de. https://www.dev-insider.de/angular-und-angularjs-a-694857/. 23.05.2019

[11] Kraft, Robin and Birk, Ferdinand and Reichert, Manfred and Deshpande, Aniruddha and Schlee, Winfried and Langguth, Berthold and Baumeister, Harald and Probst, Thomas and Spiliopoulou, Myra and Pryss, Rüdiger. (2019). Design and Implementation of a Scalable Crowdsensing Platform for Geospatial Data of Tinnitus Patients. In: 32nd IEEE CBMS International Symposium on Computer-Based Medical Systems (CBMS 2019), 5-7 June 2019, Cordoba. (Accepted for Publication). 14.10.2019

[12] Pryss, Rüdiger and Probst, Thomas and Schlee, Winfried and Schobel, Johannes and Langguth, Berthold and Neff, Patrick and Spiliopoulou, Myra and Reichert, Manfred. (2018). Prospective crowdsensing versus retrospective ratings of tinnitus variability and tinnitus–stress associations based on the TrackYourTinnitus mobile platform. International Journal of Data Science and Analytics . ISSN 2364-4168. 14.10.2019

[13] Pryss, Rüdiger and Schlee, Winfried and Langguth, Berthold and Reichert, Manfred. (2017). Mobile Crowdsensing Services for Tinnitus Assessment and Patient Feedback. In:

6th IEEE International Conference on AI & Mobile Services (IEEE AIMS 2017), June 25-30, 2017, Honolulu, Hawaii, USA. (Accepted for Publication). 14.10.2019

[14] Probst, Thomas and Pryss, Rüdiger and Langguth, Berthold and Spiliopoulou, Myra and Landgrebe, Michael and Vesala, Markku and Harrison, Stephen and Schobel, Johannes and Reichert, Manfred and Stach, Michael and Schlee, Winfried. (2017). Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples? Frontiers in Aging Neuroscience, 9 . p. 113. ISSN 1663-4365. 14.10.2019

[15] Schickler, Marc and Reichert, Manfred and Pryss, Rüdiger and Schobel, Johannes and Schlee, Winfried and Langguth, Berthold. (2015). Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health. eXamen.press . Springer Vieweg. ISBN 978-3642330568. 14.10.2019

[16] Pryss, Rüdiger and Schobel, Johannes and Reichert, Manfred (2018) Requirements for a Flexible and Generic API Enabling Mobile Crowdsensing mHealth Applications. In: 4th Int'l Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems (RESACS), RE'18 Workshops, August, 20-24, Banff, Canada. (Accepted for Publication). 14.10.2019

[17] Pryss, Rüdiger and John, Dennis and Reichert, Manfred and Hoppenstedt, Burkhard and Schmid, Lukas and Schlee, Winfried and Spiliopolou, Myra and Schobel, Johannes and Kraft, Robin and Schickler, Marc and Langguth, Berthold and Probst, Thomas. (2019). Machine Learning Findings on Geospatial Data of Users from the TrackYourStress mHealth Crowdsensing Platform. In: IEEE 20th International Conference on Information Reuse and Integration for Data Science, July 30 - August 1, 2019, Los Angeles, California, USA. (Accepted for Publication). 14.10.2019

[18] Pryss R, John D, Schlee W, Schlotz W, Schobel J, Kraft R, Spiliopoulou M, Langguth B, Reichert M, O'Rourke T, Peters H, Pieh C, Lahmann C, Probst T. (2019). Exploring the Time Trend of Stress Levels While Using the Crowdsensing Mobile Health Platform, TrackYourStress, and the Influence of Perceived Stress Reactivity: An Ecological Momentary Assessment Pilot Study. https://preprints.jmir.org/preprint/13978. 14.10.2019

[19] Beierle, Felix and Tran, Vinh Thuy and Allemand, Mathias and Neff, Patrick and Schlee, Winfried and Probst, Thomas and Zimmermann, Johannes and Pryss, Rüdiger. (2019). What data are smartphone users willing to share with researchers? Journal of Ambient Intelligence and Humanized Computing . pp. 1-13. ISSN 1868-5137. 14.10.2019

[20] Felix Beierle, Vinh Thuy Tran, Mathias Allemand, Patrick Neff, Winfried Schlee, Thomas Probst, Rüdiger Pryss and Johannes Zimmermann. (2018). Context Data Categories and Privacy Model for Mobile Data Collection Apps. 14.10.2019