



universität
ulm

Fakultät für
Ingenieurwissenschaften,
Informatik und
Psychologie
Institut für Datenbanken
und Informationssyste-
me

Entwicklung einer webbasierten Anwen- dung zur Präsentation von stressreduzie- renden Inhalten bzw. Inhalten zur Ernäh- rungsunterstützung

Abschlussarbeit an der Universität Ulm

Vorgelegt von:

Anna Marburger
anna.marburger@uni-ulm.de
1085208

Gutachter:

Prof. Dr. Manfred Reichert

Betreuer:

Dr. Michael Winter

2025

Fassung 26. Februar 2025

© 2025 Anna Marburger

Satz: PDF-L^AT_EX 2_ε

Kurzfassung

Besonders in Zeiten von beruflichem oder privatem Stress und emotionalen Belastungen fällt es vielen Menschen zunehmend schwer, an einem gesunden und ausgewogenen Ernährungsverhalten festzuhalten. Geht es um Stressbewältigung, wird bereits vermehrt auf digitale Lösungen gesetzt, Beispiel dafür sind verschiedene Meditations- oder Moodtracker-Apps.

Im Rahmen dieser Arbeit wird eine Cross-Plattform-Lösung entwickelt und implementiert, die durch eine Wissensdatenbank Nutzern helfen soll, sich über die Verbesserung ihres Wohlbefindens zu informieren. Darüber hinaus leiten verschiedene Fragebögen in der Applikation den Nutzer zur Reflexion an und helfen dabei, Trends in den eigenen Verhaltensweisen zu erkennen und einzuschätzen. Um einen breiten Nutzerkreis zu erfassen, ist die Anwendung auf möglichst vielen Plattformen verfügbar, dazu zählen Android- und iOS-Endgeräte sowie alle Endgeräte mit Browser. Weiterhin ist die Applikation aus diesem Grund dreisprachig in Englisch, Deutsch und Spanisch implementiert.

Gendererklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Arbeit die Sprachform des generischen Maskulinums verwendet. Diese sprachliche Vereinfachung impliziert jedoch ausdrücklich alle Geschlechter und ist keinesfalls als Ausschluss oder Benachteiligung zu verstehen. Alle Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.3 Struktur	2
2 Technologische Grundlagen	4
2.1 Entwicklungsumgebung VS Code	4
2.2 Ionic	5
2.3 TypeScript	8
2.4 Webtechnologien	12
3 Verwandte Arbeiten	15
3.1 Stressmonitor - Moodpress	15
3.2 BeeHealthy	17
4 Anforderungsanalyse	19
4.1 Anwendungsfälle	19
4.2 Funktionale Anforderungen	21
4.3 Nicht-funktionale Anforderungen	28
5 Entwurf und Architektur der Anwendung	31
5.1 Entwurf und Konzept	31
5.2 Architektur	33
5.3 Serverkommunikation	35
6 Implementierung	39
6.1 Anmeldung und Registrierung	39
6.2 Tab-Seiten	41

Inhaltsverzeichnis

6.3 Fragebogen-Feature	44
7 Anforderungsabgleich	49
7.1 Funktionale Anforderungen	49
7.2 Nicht-funktionale Anforderungen	56
8 Zusammenfassung und Ausblick	59
8.1 Zusammenfassung	59
8.2 Mögliche Limitationen und Ausblick	60
Literatur	62

1 Einleitung

Im ersten Kapitel wird zum Thema dieser Ausarbeitung hingeführt. Dazu wird zunächst auf die Motivation der Arbeit eingegangen, anschließend die Zielsetzung diskutiert und abschließend die Struktur der Ausarbeitung beschrieben.

1.1 Motivation

In der heutigen Gesellschaft ist ständiger Stress allgegenwärtig, sei es durch berufliche Herausforderungen oder private Verpflichtungen und Belastungen. Dies hat nicht nur Auswirkungen auf das allgemeine Wohlbefinden, sondern beeinflusst auch das Ernährungsverhalten vieler Menschen. Stress führt häufig zu ungesunden Essgewohnheiten, was langfristig negative gesundheitliche Folgen haben kann [32].

Gleichzeitig gewinnen digitale Lösungen an Bedeutung, wenn es darum geht, Menschen in ihrem Alltag zu unterstützen [22, 20]. Im Bereich der Ernährungsunterstützung können solche Anwendungen dabei helfen, gesunde Gewohnheiten zu fördern, fundierte Informationen bereitzustellen und Strategien zur Stressbewältigung zu vermitteln. Dabei setzen Entwickler vermehrt auf Cross-Plattform-Technologien [9]. Frameworks wie Ionic ermöglichen die Entwicklung von Anwendungen, die mit einer einzigen Codebasis auf unterschiedlichen Betriebssystemen lauffähig sind. Dies reduziert den Entwicklungsaufwand, vereinfacht die Wartung und gewährleistet eine konsistente Nutzererfahrung über verschiedene Plattformen hinweg. Besonders für Anwendungen mit einem breiten Nutzerkreis stellt dies einen entscheidenden Vorteil dar.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll eine Anwendung entwickelt werden, welche Smartphones als Plattform nutzt, um den Zugang zu relevanten Informationen bezüglich Stresserkennung sowie -bewältigung und gesunder Ernährung zu erleichtern. Mit Fragebögen soll es dem Nutzer möglich sein, eigene Verhaltensmuster zu reflektieren und zu überschauen. Dadurch können ungesunde Gewohnheiten entdeckt und mit den passenden, bereitgestellten Informationen erste Schritte zur Bewältigung dieser gegangen werden. Die Anwendung nutzt dabei zur Speicherung der Nutzerdaten und Informationen den Health Study Project Server (HSP)¹. Mit der Verwendung des Ionic-Frameworks soll die Anwendung plattformübergreifend entwickelt werden und auf möglichst vielen Endgeräten zur Verfügung stehen.

1.3 Struktur

In den folgenden Kapiteln wird ein umfassender Einblick in die Konzeptentwicklung und Implementierung der webbasierten Ionic-Anwendung gegeben, die dieser Arbeit zugrunde liegt. Das 2. Kapitel bietet zunächst einen kurzen Einblick zu den verschiedenen, verwendeten Technologien und liefert alle Hintergrundinformationen, die zum Verständnis der Implementierung in den folgenden Kapiteln der Arbeit erforderlich sind. Kapitel 3 stellt zwei ausgewählte verwandte Arbeiten beziehungsweise Realisierungen vor, um den Kontext dieser Arbeit einzuordnen und zu zeigen, welche bestehenden Ansätze als Grundlage oder Inspiration dienen können.

Um den Teil der Arbeit einzuleiten, in dem es um die Implementierungslösung geht, werden in Kapitel 4 alle Anwendungsfälle der Applikation definiert und anschließend sowohl die funktionalen als auch nicht-funktionalen Anforderungen basierend auf den Anwendungsfällen aufgestellt. Kapitel 5 widmet sich dem Entwurf und der konzeptionellen Lösung der Anwendung, dabei wird auf Design, Architektur sowie Serverkommunikation und Datenaustausch eingegangen. Die entworfenen Lösungen werden anschließend in der in Kapitel 6 vorgestellten Implementierung praktisch umgesetzt. Dabei wird zunächst ein weitgefasster Überblick über den Anmeldungs-

¹Zu finden unter: <https://eu1.hsc.health/>

1 Einleitung

und Registrierungsprozess sowie die Hauptfunktionen und -seiten der App geboten, bevor genauer auf die Lösung des Fragebogen-Features eingegangen wird.

Um die Arbeit zu vervollständigen, wird in Kapitel 7 auf die vorher definierten Anforderungen erneut eingegangen und die Applikation auf Vollständigkeit in Erfüllung ebendieser geprüft. Abschließend rundet das letzte Kapitel mit einer Zusammenfassung des Projekts und einem Ausblick über die mögliche Weiterentwicklung der Applikation diese Arbeit ab.

2 Technologische Grundlagen

Das folgende Kapitel geht auf die grundlegenden Technologien ein, die zur Implementierung der vorgestellten Applikation verwendet wurden. Damit werden die benötigten Hintergrundinformationen zum Verständnis der darauffolgenden Kapitel geliefert.

2.1 Entwicklungsumgebung VS Code

Visual Studio Code (VS Code) ist ein leichtgewichtiges, leistungsstarkes Entwicklungswerkzeug, welches sich besonders für die Entwicklung von Cloud- und Webanwendungen eignet. Es werden mehr als 30 unterschiedliche Programmier-, Auszeichnungs- und Datenbanksprachen unterstützt, darunter HTML, CSS und TypeScript, was den Code-Editor für das Entwickeln und Debuggen dieses Projekts besonders geeignet macht. Die Open-Source-Software ist plattformübergreifend für Linux, OS X sowie Windows kostenlos verfügbar [19].

VS Code bietet eine umfassende Sammlung an Erweiterungen an, die unter anderem weitere Programmiersprachen, Runtimes und Environments (wie beispielsweise Docker) einschließen [17]. Zudem vereinfachen Funktionen wie die Code-Vervollständigung mit IntelliSense [26], die Git-Integration zur Versionskontrolle [28] sowie Refactoring-Optionen wie "Extract Method" und "Extract Variable" [27] den Entwicklungsprozess erheblich. Darüber hinaus bietet VS Code ein vollständig integriertes Terminal, welches verschiedene Shells nutzen kann, die auf dem Betriebssystem installiert sind, wie beispielsweise Bash, Zsh oder PowerShell. Kommandozeilenbefehle können dadurch direkt in der Entwicklungsumgebung ausgeführt werden, was den Arbeitsaufwand verringert. Auch Funktionen wie Links zu Arbeitsbereichsdateien und Fehlererkennung werden im VS Code Terminal durch Integration mit dem Editor unterstützt [25].

Um die Arbeit mit dem Code-Editor speziell für dieses Projekt zu optimieren, wurden folgende Erweiterungen verwendet:

- Ionic
- React Native Tools
- Typescript React Code Snippets

In Abbildung 2.1 ist die Benutzeroberfläche des Editors zu sehen, mit geöffnetem Ionic-Plugin in der linken Randspalte.

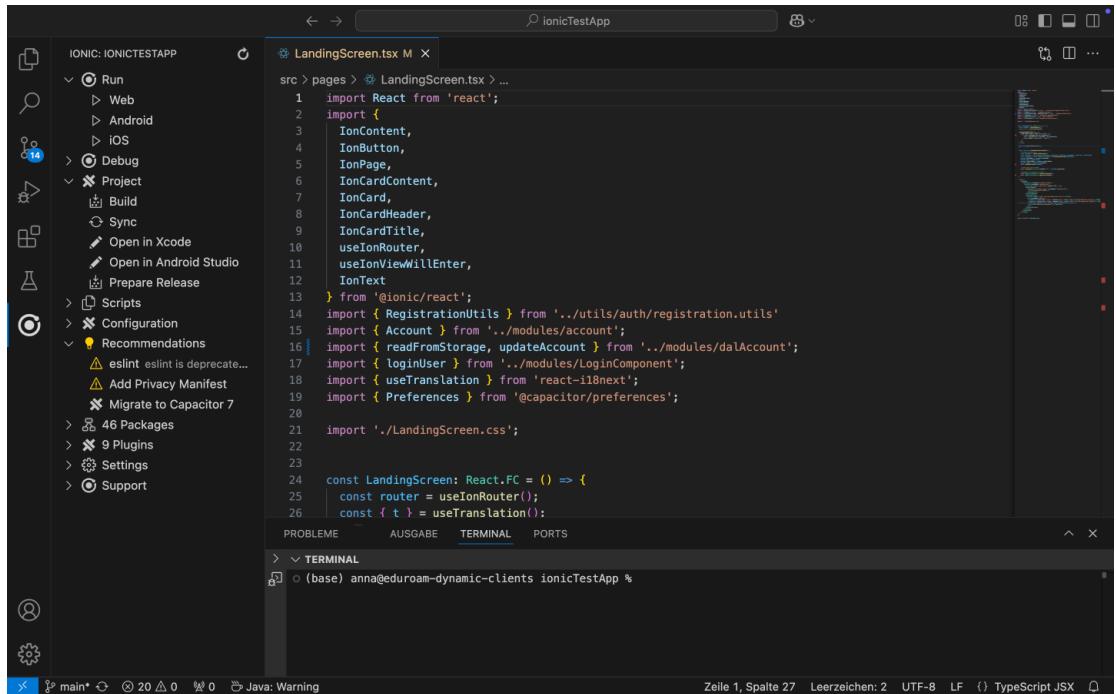


Abbildung 2.1: Benutzeroberfläche Visual Studio Code

2.2 Ionic

Ionic ist ein Open-Source-Framework und UI-Toolkit, welches seit seiner Gründung im Jahr 2012 von vielen Teams und Firmen eingesetzt wird, um die Entwicklung von hybriden, mobilen Applikationen sowie Web-Anwendungen zu vereinfachen und zu verbessern [35]. Mit einer einzigen Codebasis können Webentwickler durch Ionic

Anwendungen für Android, iOS sowie Desktopanwendungen und Progressive Web Apps schaffen, ohne separate Teams für jede Plattform zu benötigen [3]. Dabei wird Adaptive Styling eingesetzt, das heißt plattformspezifische Stile basierend auf dem Gerät, auf dem die Anwendung ausgeführt wird. Die Gestaltung der Komponenten geschieht nach den Geräterichtlinien und ermöglicht so, trotz einer einzigen Codebasis die Anwendung auf jeder Plattform nativ wirken zu lassen [14]. Als UI-Toolkit bietet Ionic eine große Sammlung an Highlevel-UI-Komponenten, wie Buttons, Listen, Tabs oder Alerts [11]. Dadurch kann besonders schnell und einheitlich das Interface der Anwendung zusammengestellt werden.

Funktionsweise und Aufbau

Um aus einer Codebasis sowohl mobile Applikationen als auch Webanwendungen zu erstellen, nutzt Ionic WebViews. Dies sind voll funktionsfähige Web-Browser, die innerhalb einer App verwendet werden [15]. Ionic-Apps werden mit Web-Technologien erstellt und als native Anwendung mit WebViews gerendert, die sowohl im iOS- als auch im Android Development Kit (SDKs) enthalten sind [12]. Um Zugriff auf native SDK-Funktionen wie Kamera, GPS, Bluetooth oder Speicher zu erhalten, kann Capacitor verwendet werden, einer nativen Open-Source-Runtime [33]. Dafür werden Plugins eingesetzt, welche JavaScript eine direkte, einheitliche Schnittstelle zu den nativen SDKs bieten. Nicht nur offizielle Plugins werden von Capacitor angeboten, auch Community-Plugins werden unterstützt und erweitern die Auswahl [34]. Folgende Abbildung 2.2 zeigt die Architektur einer Ionic-Applikation, die mithilfe von WebView zu einer nativen App wird.

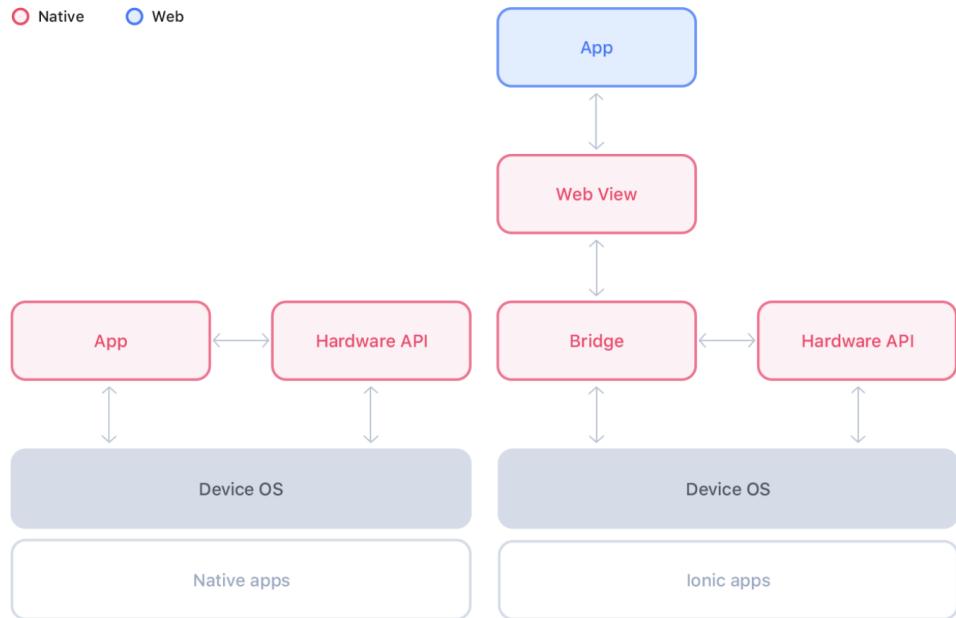


Abbildung 2.2: WebView-Architektur [15]

Ionic in Verwendung mit React

Seit Version 4.x ist Ionic nicht mehr eng an das JavaScript-Framework Angular gekoppelt, sondern wurde als eigenständige Web-Komponentenbibliothek neu gestaltet. Ebenso wurden die aktuellsten JavaScript-Frameworks wie React oder Vue ebenfalls integriert, sodass Entwickler flexibel in der Wahl ihres Frontend-Frameworks sind [5].

Ionic React ist äquivalent zu React-Native und baut auf das Setup, welches normalerweise in einer Create React App (CRA)-App zu finden ist. So wird beispielsweise für Routing und Navigation in der Anwendung React Router verwendet [13]. React kommt dabei mit einigen Vorteilen, wie beispielsweise verbesserter Leistungs- und Update-Geschwindigkeit durch die Verwendung von Virtual DOM (Document Object Model), was im Gegensatz zum herkömmlichen DOM nicht alle Komponenten, sondern nur tatsächlich geänderte Elemente aktualisiert [18].

Der Vorteil, den das Verwenden von Ionic React bietet, ist zum Beispiel die größere Bibliothek an vorgefertigten UI-Komponenten, die Ionic im Vergleich zu React Native liefert [3]. Nachteilig hingegen gegenüber Frameworks, die zu nativem Code kompilieren, ist die verschlechterte Performance, die mit dem Verwenden von

WebViews einhergehen kann [4].

Der verringerte Aufwand durch die wiederverwendbaren Komponenten und einer einzigen Codebasis für eine Crossplattform-Applikation spielte eine wichtige Rolle bei der Wahl des Technologie-Stacks. Im Zusammenspiel mit Capacitor, um Funktionen wie Push-Benachrichtigungen in der Applikation umzusetzen, bietet Ionic als Toolkit alles, was es für das in dieser Arbeit vorgestellte Projekt braucht.

2.3 TypeScript

Die Programmiersprache TypeScript wurde von Microsoft entwickelt und 2012 als Open-Source-Version veröffentlicht. Sie wird als Übermenge von JavaScript gesehen, da sie JavaScript um statische Typisierung erweitert (siehe Abbildung 2.3).

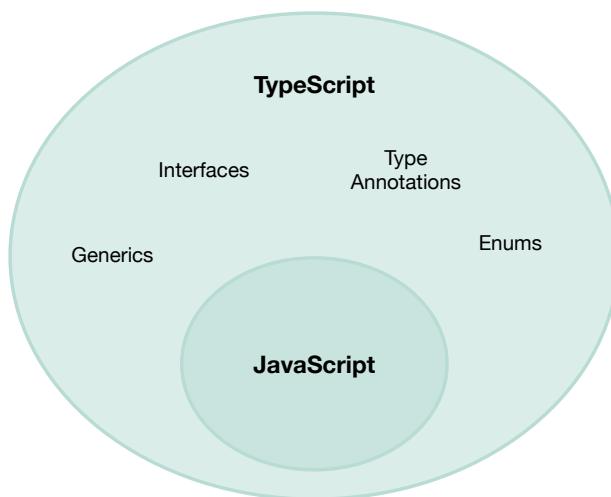


Abbildung 2.3: TypeScript als Obermenge von JavaScript

TypeScript vs. JavaScript

Zusätzlich zur vollständigen JavaScript-Syntax enthält TypeScript also auch Syntax zum Definieren und Nutzen von Typen [16, S. 6]. Folglich fügt TypeScript Regeln hinzu, wie verschiedene Arten von Werten im Code verwendet werden können. Dabei wird jedoch nicht das Laufzeitverhalten von JavaScript-Code verändert. Es wird garantiert, dass funktionierender JavaScript-Code auf die gleiche Weise in Type-

Script ausgeführt wird, auch wenn TypeScript Typfehler im Code entdeckt [38]. Um dies anschaulich zu erklären, soll folgendes Beispiel dienen:

```
let a = "hallo";
let b = 5;
console.log( a / b );
```

Dieser JavaScript-Beispielcode läuft ohne Fehler und gibt `NaN` (Not-a-Number) aus. `NaN` beschreibt hierbei Nummern, die keinen legalen numerischen Wert vertreten [40]. Ein weiteres Beispiel für die Ausgabe `NaN` ist die Division durch Null:

```
console.log(0/0);
```

TypeScript reagiert zur Laufzeit ebenso, da TypeScript-Code zu JavaScript-Code kompiliert, jedoch wird TypeScript noch vor dem Kompilieren des Codes die Typen der Operation überprüfen und folgenden Fehler feststellen, welcher in Abbildung 2.4 zu sehen ist. War es nicht gerade die Absicht des Programmierers, einen String durch eine Zahl zu teilen, wurde durch die Typisierung von TypeScript vorzeitig ein Programmierfehler entdeckt, der in einer reinen JavaScript-Umgebung möglicherweise erst deutlich später aufgefallen wäre.

The screenshot shows a code editor window with three lines of TypeScript code:

```
1 var a = "hallo";
2 var b = 5;
3 var c = a / b;
```

A red squiggly underline is under the division operator `/` in the third line. A tooltip above the line reads: "Die linke Seite einer arithmetischen Operation muss den Typ "any", "number" oder "bigint" aufweisen oder ein Enumerationstyp sein. ts(2362)". The status bar at the bottom of the editor shows the file name "example.ts" and the message "1 von 1 Problemen".

Abbildung 2.4: TypeScript Typfehler

TypeScript Features

Um Fehler, wie im vorherigen Abschnitt beschrieben, zu vermeiden und JavaScript-Programme typsicher zu machen, hat TypeScript unter anderem folgende Features eingebunden:

- *Type Annotations*

TypeScript erkennt bei der Initialisierung einer Variable den entsprechenden Typ selbst. Um den Typ einer Variable anzugeben, beispielsweise vorteilhaft, wenn die Variable zunächst ohne Wert nur deklariert wird, können Typ-Anmerkungen verwendet werden [16, S. 22-23].

```
let a: number = 10;  
a = "hello"; // type error
```

Dies ist auch bei Funktionsparametern möglich. Dadurch können beispielsweise falsche Funktionsverwendungen abgefangen werden, was besonders die Zusammenarbeit in größeren Projekten mit vielen Beteiligten erleichtert.

```
function doubleIt(a: number): number{  
    return a + a; // takes a number, returns a number  
}  
  
doubleIt(5); // output: 10  
doubleIt("hello"); // type error
```

- *Interfaces*

Neben den primitiven Datentypen wie `string`, `number` oder `boolean` können auch komplexere Datentypen für Typprüfungen definiert werden. Dazu gibt es unter anderem das Interface-Feature von TypeScript. Definierte Interfaces können implementiert werden, ohne `implements` zu verwenden, indem das Objekt lediglich der Form des Interfaces entspricht [37].

```
interface Person {  
    name: string;  
    age: number;  
}  
  
function printPerson(person: Person) {  
    console.log(person.name + ": " + person.age + " years");  
}  
  
let user = { name: "Anna", age: 23 };  
printPerson(user); // output: "Anna: 23 years"
```

- *Enums*

TypeScript erweitert JavaScript ebenfalls um das Enum-Feature. Mit Enums können Sets an benannten Werten definiert werden, die möglicherweise Code-

Absichten deutlicher, konsistenter und Fallunterscheidungen einfacher machen [36].

```
enum AlertMessage {  
    Fail = "The action failed",  
    Success = "The action was successfully done"  
};  
  
function notifyUser(message: AlertMessage){  
    console.log(message);  
}  
  
notifyUser(AlertMessage.Fail); // output: "The action failed"
```

TypeScript im Kontext dieser Arbeit

Wie bereits im Abschnitt 2.2 erläutert wurde, werden Ionic-Projekte als Webanwendung entwickelt und basieren so auf der Programmiersprache JavaScript. Ebenso wurde bereits darauf eingegangen, welche Vorteile das Verwenden des Frontend-Frameworks React mit sich bringt. In Verbindung mit TypeScript zeigen sich weitere Vorteile, die sich aus den beschriebenen Features im letzten Absatz herauskristallisieren.

Zunächst führt die Typisierung durch die Sprache zu einer verbesserten Code-Qualität und -Klarheit. Die Verwendung von typisierten Variablen, Funktionen und Objekten lässt leichter Rückschlüsse darauf ziehen, was der entsprechende Code bezweckt und fängt Fehler frühzeitig ab. Dies macht das Zusammenarbeiten bei größeren Softwareprojekten und das Warten von Code erheblich leichter. Außerdem werden weniger Ressourcen für Debugging aufgewendet [31]. Durch TypeScripts Support für objekt-orientierte Features wie Interfaces und Classes ist es leichter, Code zu modularisieren, das heißt, Code-Elemente wiederzuverwenden [1].

Dass TypeScript eine Obermenge der weit verbreiteten Sprache JavaScript ist, sorgt dafür, dass Entwickler, die in JavaScript bereits programmieren, eine sehr geringe, um fast zu sagen keine Lernkurve in Kauf nehmen müssen. Weiter können dadurch ebenfalls alle JavaScript-Bibliotheken verwendet und auf eine große Community für Support zurückgegriffen werden [39].

TypeScript fügt wichtige Elemente für die Entwicklung von Cross-Plattform-Anwendungen hinzu, die den Prozess des Programmierens effizienter sowie das Ergebnis qualitativ hochwertiger gestalten. Dadurch ist die Sprache als Ergänzung zu Ionic ein wertvoller Teil in Mobil- sowie Webanwendungen und kaum mehr wegzudenken.

2.4 Webtechnologien

Neben Javascript beziehungsweise TypeScript sind die Webtechnologien HTML und CSS für eine Ionic-Applikation ebenfalls von zentraler Bedeutung und sollen in diesem Unterkapitel beschrieben werden.

HTML

HTML steht für HyperText Markup Language und ist der grundlegende Baustein im Webdevelopment. Mit HTML wird die Struktur und der Aufbau von Inhalten auf Webseiten und Webanwendungen beschrieben. HyperText bezieht sich dabei auf die Verlinkung von Webseiten untereinander, eine fundamentale Funktion im gesamten World Wide Web [7]. Markup- oder auch Auszeichnungssprachen sind definierte Strukturen, um Dokumente zu gliedern und zu kommentieren. Dabei werden Tags oder Codes verwendet, um beispielsweise Überschriften, Absätze, Listen, Bilder oder Links zu definieren [23].

HTML verwendet Tags, um Markup von tatsächlichem Inhalt zu trennen. Diese Elemente unterteilen sich in HTML5 in folgende Gruppen [24]:

- Semantische Elemente: Dazu gehören unter anderem Überschriften (<h1>), Footer(<footer>) oder Navigations Elemente (<nav>).
- Graphische Elemente: Binden beispielsweise Vektorgraphiken (<svg>) ein oder erstellen ein neues Canvas (<canvas>), auf welchem mit JavaScript graphische Elemente gemalt werden können.
- Multimedia Elemente: Hierzu zählen als Beispiel Video (<video>) oder Audio (<audio>) Dateien, die in die Website eingebunden werden.

Dabei ist HTML plattformübergreifend kompatibel, was dafür sorgt, dass HTML-basierte Webseiten auf allen Endgeräten gleichermaßen funktionieren. Auf die-

se Kompatibilität baut auch Ionic als fortgeschrittenes HTML5-Hybrid-Mobile-App-Framework.

CSS

Um Webseiten und -anwendungen nicht nur strukturieren sondern auch gestalten zu können, gibt es CSS (Cascading Style Sheets). In solchen Stylesheets wird beschrieben, wie Elemente auf dem Bildschirm dargestellt werden sollen. Dabei beziehen sich die sogenannten Rules auf Farben und Schriften, bis hin zu Spacing und Animationen. Genau wie HTML ist CSS unter den gängigen Webbrowsersn standardisiert [6].

CSS ist eine regelbasierte Sprache, dabei werden Gruppen von Stilen angegeben, die auf bestimmte Elemente oder Gruppen von Elementen der Webseite angewendet werden sollen [8]. Folgender Beispiel-Code zeigt, wie ein HTML-Element selektiert und gestylt werden kann.

```
h1 {  
    color: white;  
    font-size: 28px;  
}
```

Sollen nicht alle h1-Überschriften weiß werden, können beispielsweise Klassen verwendet werden:

```
.white {  
    color: white;  
    font-size: 28px;  
}
```

Die Klasse wird im Stylesheet mit einem Punkt selektiert und im HTML-Dokument folgendermaßen verwendet:

```
<h1 class="white"> This is a Header </h1>
```

2 Technologische Grundlagen

CSS bietet deutlich mehr Möglichkeiten, Inhalte zu selektieren und anzupassen, als es in einem kurzen Überblick möglich wäre vorzustellen. Gemeinsam mit JavaScript und HTML gehört CSS jedoch zu den grundlegenden Webtechnologien und spielt so auch im Stylen der Benutzeroberfläche einer Ionic-Anwendung eine wichtige Rolle.

3 Verwandte Arbeiten

Im Bereich der mentalen Gesundheit werden zunehmend mobile Applikationen eingesetzt. Mehrere Studien haben die Nützlichkeit von Apps für die psychische Gesundheit als Ergänzung traditioneller Interventionen aufgezeigt, beispielsweise durch Aufklärung über Behandlungstechniken, durch die Erleichterung von Symptomüberwachung und durch fortgesetzten Zugang zu Interventionen [20]. Im folgenden Kapitel sollen zwei Applikationen vorgestellt werden, die ähnlich zu der im Rahmen dieser Arbeit entwickelten Lösung implementiert wurden und sich auf Stressreduzierung beziehungsweise Ernährungsunterstützung fokussieren.

3.1 Stressmonitor - Moodpress

Die Anwendung Moodpress ist sowohl im PlayStore als auch im AppStore für iOS sowie macOS verfügbar und ist dadurch eine plattformübergreifende Applikation [21].

Neben einem Moodtracker, der als eine Art Tagebuch fungiert, importiert die in Abbildung 3.1 zu sehende iOS-Version der App Gesundheitsdaten, die zur Einschätzung des Stresslevels dienen. Dazu zählen unter anderem die Herzfrequenzvarianz, die durch eine Fitness-Uhr aufgezeichnet wird, sofern vorhanden.

Im Erkunden-Tab der Applikation werden verschiedene Inhalte zur Stressreduzierung angeboten. Beispielsweise gehört dazu eine Bibliothek an beruhigenden Klängen und Geräuschen. Unter „Selbstfürsorge“ findet der Nutzer Tipps und Informationen zu verschiedenen Themen, bei denen es um Self-Care geht.

Weiter erklären Infoseiten dem Nutzer den Einfluss der entsprechenden, getrackten Gesundheitsfaktoren. Sie sind über die Fragezeichen-Symbole im Home-Tab zu erreichen. Neben der Herzfrequenzvarianz werden ebenfalls Schlaf, Schrittanzahl

3 Verwandte Arbeiten

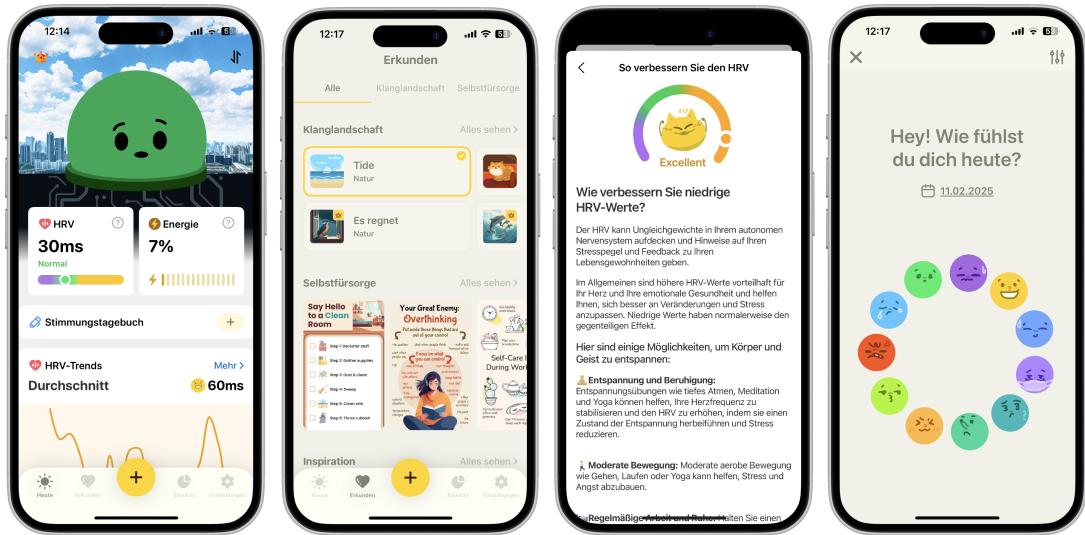


Abbildung 3.1: Verschiedene Seiten der Moodpress-App

und die verbrachte Zeit im Tageslicht getrackt. Jedoch können auch manuell Aktivitäten hinzugefügt werden, was besonders hilfreich ist, wenn keine Fitness-Uhr vorhanden ist, von welcher die Applikation automatisch die entsprechenden Daten importieren kann. Je nach Auswertung der Daten passt sich das Emoji auf dem Homescreen an und symbolisiert dem Nutzer mit einem Blick eine Einschätzung seines Stresslevels.

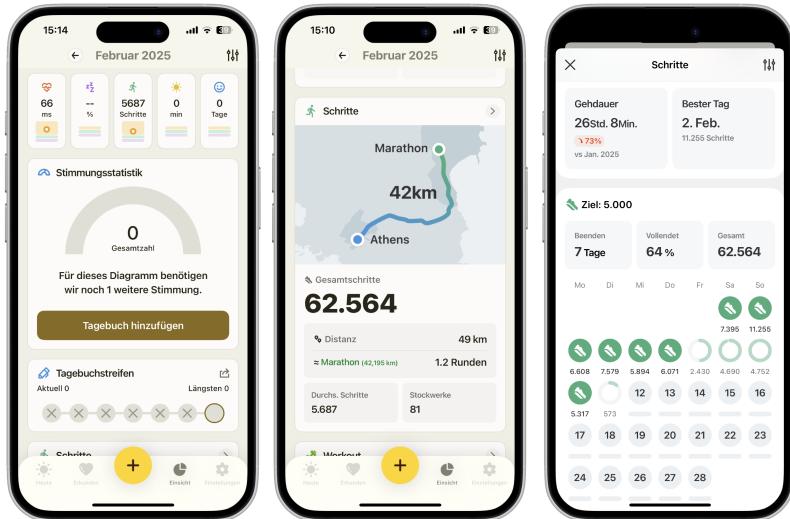


Abbildung 3.2: Verschiedene Screens des Einsicht-Tabs

Im Einsicht-Tab ist es möglich, genauer verschiedene Statistiken einzusehen, die sich auf die bereits beschriebenen, gemessenen Daten beziehen. In mehreren Karten werden graphisch die Werte für unter anderem Schlafdauer, Schrittzahl und Workout-Zeit dargestellt. Klickt man auf die Schritte-Karte, wird noch mehr Einblick geboten, so unter anderem eine Kalender-Ansicht, in der die Tage markiert sind, an welchen der Nutzer sein Schritte-Ziel erreicht hat (Abbildung 3.2). Das Ziel lässt sich dabei in den Einstellungen selbst festlegen.

Zusammenfassend lässt sich sagen, dass der Nutzer mit Hilfe von Moodstep einen Überblick über seine Gewohnheiten, Stimmungen und Aktivitäten erlangt und sich zu diesen Themen informieren kann, um bei Schwierigkeiten entsprechende Tools zur Hand zu haben.

3.2 BeeHealthy

Mit der App BeeHealthy sollen junge Menschen spielerisch gesunde Ernährung, psychische Gesundheit und ausgewogene Bewegung erlernen. Sie wurde ursprünglich für Kinder und Jugendliche mit Migrationshintergrund konzipiert, da häufige Wohnortwechsel, mangelnde Spielmöglichkeiten und der eingeschränkte Zugang zu Schulen sowie Sprachbarrieren im fremden Land sich häufig negativ auf die psychische und physische Gesundheit auswirken [10].

Die Applikation ist bisher nur als Pilotversion für Androidgeräte im PlayStore verfügbar, jedoch werden weitere Funktionen und Verbesserungen bereits in Aussicht gestellt und nach Unterstützung dafür gesucht. [10].

Die verschiedenen Screens der Anwendung sind in Abbildung 3.3 zu sehen. Die App startet auf dem Homescreen, von welchem aus der Nutzer in verschiedene Bereiche navigieren kann. Über „Profil bearbeiten“ kann der Avatar sowie der Name des Users verändert werden. Unter der Sprachen-Flagge kann die Sprache der Benutzeroberfläche geändert werden. Dabei stehen Englisch, Ukrainisch und Deutsch zur Verfügung.

Über die drei Balken „Eat“, „Breathe“ und „Move“ gelangt man in den Lernbereich der Applikation. Es steht jeweils eine Liste zu jedem Bereich bereit, in der verschiedene Themen aufgegriffen werden (siehe Abbildung 3.3). Zu jedem der Themen gibt es weiterhin mehrere Lektionen, in denen durch Quizzes spielerisch Informatio-

3 Verwandte Arbeiten

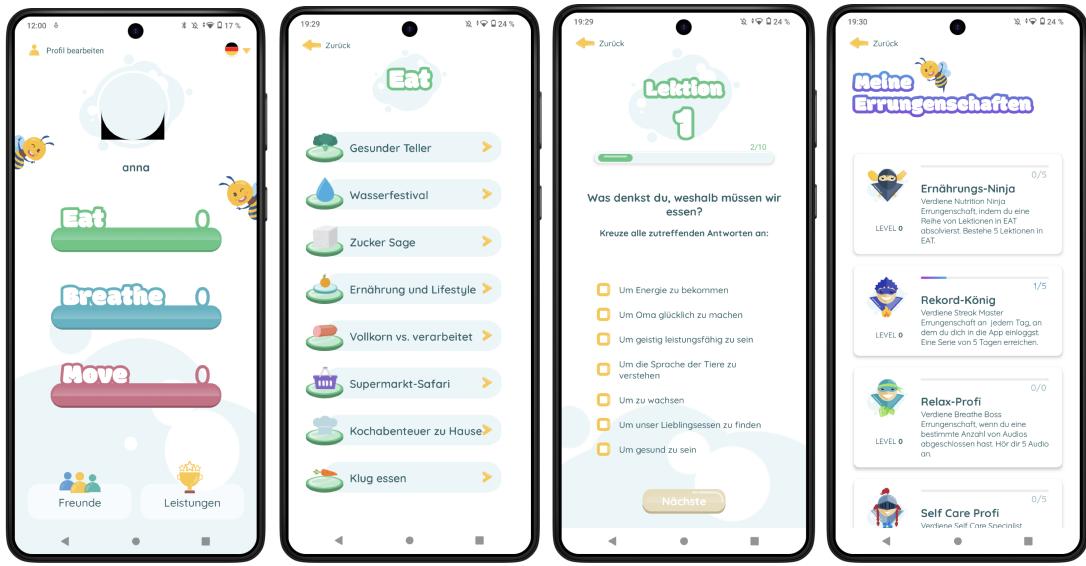


Abbildung 3.3: Verschiedene Seiten der BeeHealthy-App

nen vermittelt werden. Über „Leistungen“ auf dem Homescreen kommt der Nutzer zu dem Bereich seiner Errungenschaften. Für das Erreichen verschiedener Ziele erhält der Nutzer Auszeichnungen, die das fortlaufende Nutzen der Applikation spielerisch fördern sollen.

Durch das verspielte User Interface und der Gamification¹ in der Vermittlung der Lerninhalte ist die Applikation besonders für Kinder ausgelegt, um ihnen schon früh einen gesunden Umgang mit Ernährung, Sport und Stress zu vermitteln. Da sich die Anwendung jedoch in der Pilotversion befindet, sind einige Funktionen noch eingeschränkt, wie beispielsweise die Auswahl des Avatars (siehe Abbildung 3.3, in welcher der Avatar nicht korrekt angezeigt wird), oder die „Freunde“-Funktion.

¹Unter Gamification versteht man die Integration von Spielementen in spielfremde Umgebungen [2].

4 Anforderungsanalyse

Im Kapitel der Anforderungsanalyse werden zunächst die verschiedenen Anwendungsfälle der App analysiert, um mögliche Benutzerinteraktionen und den Kontext darzustellen. Weiter erläutern die funktionalen Anforderungen im nächsten Abschnitt genau, welche Kernfunktionalitäten von der Anwendung erfüllt werden müssen. Abschließend werden die Qualitätsanforderungen an die Applikation definiert.

4.1 Anwendungsfälle

Die folgenden Anwendungsfälle beschreiben die Hauptfunktionen der implementierten App und sollen mögliche Interaktionen mit dem System darstellen. Dabei werden auch unterschiedliche Akteure und ihre Rollen aufgezeigt. Das Anwendungsfalldiagramm 4.1 visualisiert die möglichen Nutzeraktionen, die durch die App realisiert werden.

Die Applikation startet im Landing-Screen, welcher dem Nutzer drei Handlungsmöglichkeiten bietet. Bei bereits bestehendem Konto kann der User sich dort mit der Angabe seines Nutzernamens und seines korrekten Passworts anmelden. Ist dies nicht der Fall, besteht die Möglichkeit, ein neues Konto anzulegen. Dabei wird der Nutzer ebenfalls nach Username und Passwort gefragt. Zuletzt kann sich auch anonym registriert werden. Dies erfordert keine Angabe von Daten und leitet den Nutzer direkt weiter. Bei Erstinstallation und -anmeldung wird der Nutzer darüber hinaus zum Akzeptieren des Disclaimers und von Benachrichtigungen aufgefordert. Ist der Nutzer nun erfolgreich angemeldet, stehen ihm die weiteren Funktionen der App offen. Auf der Profilseite können Disclaimer, Impressum und die Datenschutzerklärung eingesehen werden. Des Weiteren wird dem Benutzer dort ermöglicht, die Sprache der App zu ändern und Benachrichtigungen zu (de-)aktivieren, sowie

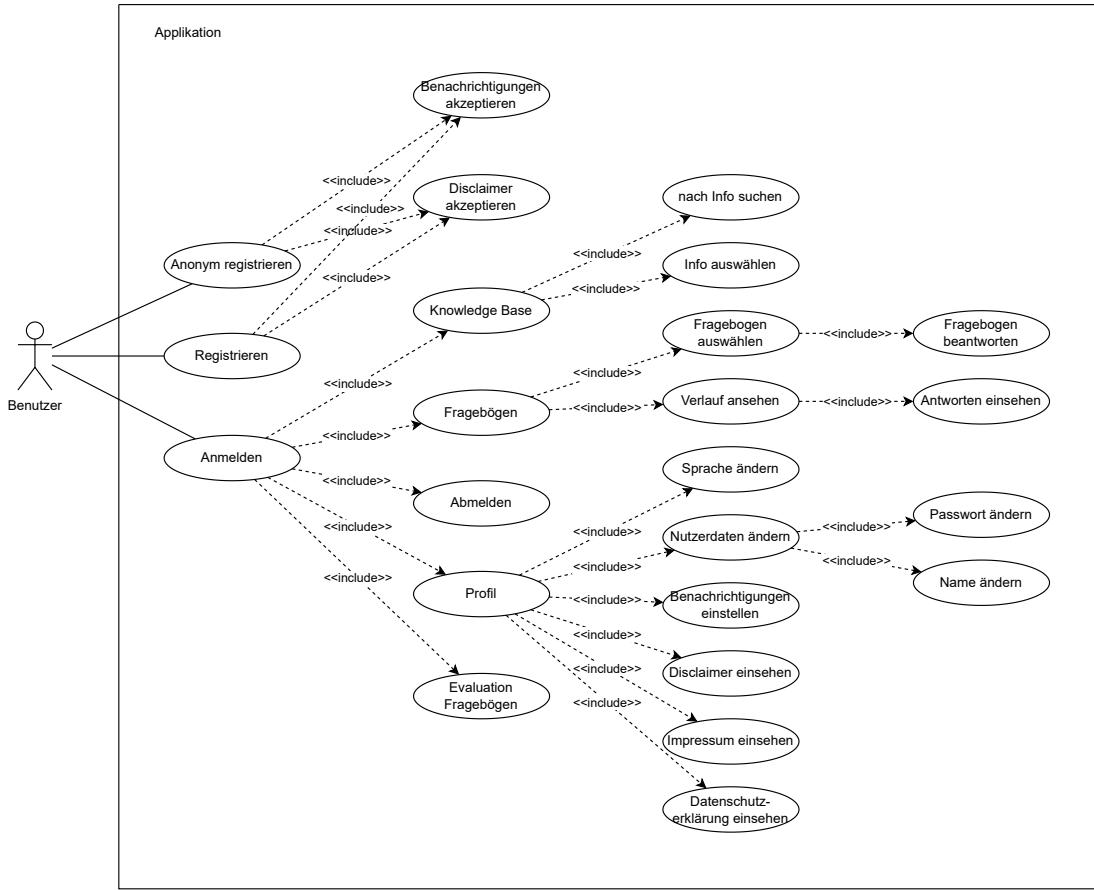


Abbildung 4.1: Anwendungsfalldiagramm

die Uhrzeit dieser anzupassen. Auch Passwort und Name können über das Profil geändert werden.

Ein zentraler Teil der Anwendung sind die Fragebögen, die durch den Nutzer ausgefüllt und eingesehen werden können. Eine Übersicht davon ist im Fragebögen-Tab zu finden, ebenso wie der Verlauf bereits beantworteter Fragebögen. Die Fragen sollen den Nutzer zur Reflexion anleiten und lassen eine Nachverfolgung von Angewohnheiten und Trends zu. Zu Letzterem trägt die Auswertung ausgefüllter Fragebögen bei, welche auf dem Home-Tab durch einen Graphen visualisiert wird. Auf der Homepage sind weiter Shortcuts zu den wichtigsten Funktionen der Anwendung zu finden. Dazu zählt neben dem User-Screen und dem Fragebögen-Tab die Knowledge Base. In einem eigenen Tab werden hier in einer Liste verschiedene Informationen angeboten, die der Nutzer durchsuchen und auswählen kann, um

mehr über ein Thema zu erfahren. Zuletzt ist es dem Nutzer zu jeder Zeit ermöglicht, sich von der Anwendung abzumelden. Dafür befindet sich im Userscreen ein entsprechender Button.

4.2 Funktionale Anforderungen

Im folgenden Kapitel sollen die funktionalen Anforderungen (FAs) an die Anwendung definiert werden. Sie beschreiben die wesentlichen Funktionen, welche die Anwendung implementieren muss, um die definierten Ziele der Nutzer und Anwendungsfälle zu erfüllen. Sie bilden die Grundlage für die Evaluation der Anwendung, indem sie mit den Ergebnissen des Implementierungsprozesses abgeglichen werden. Die Tabelle 4.1 bietet eine Übersicht zu den definierten Anforderungen.

ID	Titel	Seite
FA01	Registrierung	22
FA02	Anmeldung	22
FA03	Anonyme Registrierung	22
FA04	Abmeldung	23
FA05	Navigation mit Tabs	23
FA06	Nutzerseite anzeigen	23
FA07	Nutzerdaten verwalten	24
FA08	Passwort ändern	24
FA09	Sprache ändern	24
FA10	Benachrichtigungen verwalten	25
FA11	Disclaimer	25
FA12	Impressum, Datenschutzerklärung	25
FA13	Fragebögen-Seite	26
FA14	Fragebogen ausfüllen	26
FA15	Knowledge-Base-Seite	27
FA16	Einzelne Info-Seite ansehen	27
FA17	Schnellzugriffe Home-Seite	27
FA18	Evaluation auf der Home-Seite	28

Tabelle 4.1: Verzeichnis der funktionalen Anforderungen

ID	FA01
Titel	Registrierung
Beschreibung	Bisher nicht registrierte Nutzer können sich mit einem Passwort und Namen registrieren. Das Passwort muss mind. 8 Zeichen enthalten.
Begründung	Neue Nutzer können so ein sicheres Konto anlegen, mit welchem ihre Nutzerdaten gesichert werden.

ID	FA02
Titel	Anmeldung
Beschreibung	Bereits registrierte Nutzer können sich in ihr Konto mit Nutzernamen und richtigem Passwort einloggen.
Begründung	Nutzer können so die App mit ihren bisherigen Daten verwenden.

ID	FA03
Titel	Anonyme Registrierung
Beschreibung	Bisher nicht registrierte Nutzer können ein anonymes Konto erstellen, wozu weder Passwort noch Name benötigt werden. Mit Abmelden des Nutzers geht der Verlust seiner Daten einher.
Begründung	Nutzer können die App ohne Aufwand einer Anmeldung nutzen und die Verknüpfung von Daten mit ihrem Namen vermeiden.

ID	FA04
Titel	Abmeldung
Beschreibung	Wenn der Nutzer in der App eingeloggt ist, soll er sich jederzeit ausloggen können. Bei anonymen Nutzern gehen dadurch ihre Daten verloren, bei regulär registrierten Nutzern bleiben alle Daten erhalten.
Begründung	Auf einem Gerät kann so zwischen verschiedenen Nutzern gewechselt werden. Anonyme Nutzer können dadurch ihr Konto entfernen.

ID	FA05
Titel	Navigation mit Tabs
Beschreibung	Die App soll eine Navigation durch Tabs enthalten. Die verschiedenen Tabs sollen einen Homescreen, eine Fragebogen-Seite, einen Info-Tab und einen Userscreen beinhalten.
Begründung	Der Nutzer kann mit den Tabs schnell zwischen den Hauptseiten der Anwendung wechseln.

ID	FA06
Titel	Nutzerseite anzeigen
Beschreibung	Der Userscreen soll dem Nutzer Informationen zu seinem Konto, Einstellungsmöglichkeiten und Informationen zur Anwendung bereitstellen.
Begründung	Der Nutzer kann durch einen Userscreen sein Konto und Einstellungen der Applikation verwalten.

ID	FA07
Titel	Nutzerdaten verwalten
Beschreibung	Der Nutzer soll seinen Namen (Vor- und Nachname) in der App ändern können. Die Änderungen sollen serverseitig gespeichert werden.
Begründung	Unter anderem kann der Nutzer so Fehler in seinem Namen korrigieren.
ID	FA08
Titel	Passwort ändern
Beschreibung	Das Passwort soll geändert werden können. Dazu muss das bisherige Passwort korrekt eingegeben werden und das neue Passwort mind. 8 Zeichen lang sein. Ansonsten wird eine Fehlermeldung ausgegeben.
Begründung	Das regelmäßige Ändern des Passworts trägt zur Sicherheit eines Kontos bei.
ID	FA09
Titel	Sprache ändern
Beschreibung	Die Applikation soll in den folgenden drei Sprachen zur Verfügung stehen: Englisch, Deutsch, Spanisch. Der Nutzer soll dies eigenhändig in den Einstellungen auswählen können.
Begründung	Durch Multilanguage-Support kann eine größere Nutzerzahl die App sinnvoll verwenden.

ID	FA10
Titel	Benachrichtigungen verwalten
Beschreibung	Bei Erstanmeldung bzw. -installation soll der Nutzer zum Akzeptieren von Benachrichtigungen aufgefordert werden. In den Einstellungen soll der Nutzer Benachrichtigungen an- und ausschalten, sowie bei angeschalteten Benachrichtigungen die Uhrzeit der Benachrichtigung einstellen können.
Begründung	Benachrichtigungen erinnern den Nutzer an das regelmäßige Ausfüllen der Fragebögen, das Ausschalten dient dem Nutzer zur Vermeidung ungewollter Benachrichtigungen.

ID	FA11
Titel	Disclaimer
Beschreibung	Bei Erstanmeldung bzw. -installation muss der Nutzer vor dem Verwenden der App einen Disclaimer akzeptieren, welcher den Haftungsausschluss bzgl. der App beschreibt. Den Disclaimer soll der Nutzer ebenfalls über den Userscreen einsehen können.
Begründung	Der Nutzer wird über den Haftungsausschluss der Anwendung aufgeklärt und akzeptiert diesen.

ID	FA12
Titel	Impressum, Datenschutzerklärung einsehen
Beschreibung	Der Nutzer kann über den Userscreen das Impressum und die Datenschutzerklärung einsehen.
Begründung	Dem Nutzer wird transparent Impressum und Datenverarbeitung der Anwendung vorgelegt.

ID	FA13
Titel	Fragebögen-Seite
Beschreibung	Es soll ein Tab implementiert werden, der eine Übersicht von einerseits ausfüllbaren Fragebogen-Vorlagen und andererseits bereits ausgefüllten Fragebögen gibt.
Begründung	Der Nutzer kann durch die Übersicht zu einzelnen Fragebögen gelangen.
ID	FA14
Titel	Fragebogen ausfüllen
Beschreibung	Durch Auswählen eines Fragebogen-Items in der Übersichtseite kann der Nutzer einen Fragebogen ausfüllen. Die Antworten des Nutzers werden durch das Klicken auf einen Button an den Server zurückgesendet und dort gespeichert. Es soll mit Buttons durch die Fragebogenseiten navigiert werden können. Ein Ladebalken soll den Fortschritt visualisieren. Es wird erst eine neue Fragebogen-Instanz erstellt, wenn die vorherige vollständig ausgefüllt wurde. Ansonsten wird die vorherige Instanz geladen, und der Nutzer kann diese weiter bearbeiten.
Begründung	Der Nutzer kann seine aktuellen Antworten festhalten.

ID	FA15
Titel	Ausgefüllten Fragebogen einsehen
Beschreibung	Durch Auswählen eines Fragebogen-Items in der Verlaufsliste kann der Nutzer seine Antworten zu einem bereits ausgefüllten Fragebogen einsehen. In dieser Ansicht soll der Nutzer den Fragebogen nicht weiter bearbeiten können.
Begründung	Der Nutzer kann vorherige Antworten einsehen und vergleichen.

ID	FA16
Titel	Knowledge-Base-Seite
Beschreibung	Es soll ein Tab als Übersicht für Informationen implementiert werden. Die Übersichtsseite soll durchsucht werden können und Titel nach der Nutzereingabe filtern.
Begründung	Der Nutzer erhält eine Informationsübersicht und kann gezielt nach Informationen suchen.

ID	FA17
Titel	Einzelne Info-Seite ansehen
Beschreibung	Die Listenelemente der Knowledge-Base-Seite sollen zu den jeweiligen Informationsseiten navigieren. Die Seiten sollen Text und Bild darstellen können, welche sie vom Server laden.
Begründung	Der Nutzer kann sich genauer zu einem Thema informieren.

ID	FA18
Titel	Schnellzugriffe Home-Seite
Beschreibung	Auf der Homepage sollen Schnellzugriffe auf verschiedene Inhalte und Funktionen der App eingebunden werden.
Begründung	Eine schnelle und einfache Navigation zu den wichtigsten Funktionen der App.

ID	FA19
Titel	Evaluation Home-Seite
Beschreibung	Ein Graph soll die Scores der zuletzt ausgefüllten Fragebögen auf dem Homescreen visualisieren.
Begründung	Evaluation der Fragebögen für den Nutzer.

4.3 Nicht-funktionale Anforderungen

Neben den funktionalen Anforderungen setzen die in diesem Kapitel definierten nicht-funktionalen Anforderungen (QAs) Maßstäbe für die Applikation, beziehen sich dabei aber auf die Qualität der implementierten Funktionen.

ID	Titel	Seite
QA01	Robustheit	29
QA02	Usability und Intuitivität	29
QA03	Kompatibilität	29
QA04	Wartbarkeit	29
QA05	Client-Server-Architektur	30
QA06	Implementierungssprache	30

Tabelle 4.2: Verzeichnis der nicht-funktionalen Anforderungen

ID	QA01
Titel	Robustheit
Beschreibung	Die Anwendung soll nicht abstürzen. Das Verwenden der App und das Nutzen aller Funktionen darf keine Fehler hervorrufen. Eingaben des Nutzers müssen dafür überprüft werden und davon ungültige angemessen abgefangen werden.

ID	QA02
Titel	Usability und Intuitivität
Beschreibung	Das User Interface der App soll ansprechend intuitiv sein, das heißt, der Nutzer muss ohne vorherige Einweisung die App eigenständig nutzen können. Dafür werden gängige Navigationselemente wie Tabs und Buttons verwendet und übersichtlich eingesetzt.

ID	QA03
Titel	Kompatibilität
Beschreibung	Die Anwendung soll im Browser und auf allen aktuellen Android- und iOS-Versionen funktionieren. Dafür wird das plattformübergreifende Framework Ionic verwendet.

ID	QA04
Titel	Wartbarkeit
Beschreibung	Die Anwendung soll mit geringem Aufwand zu Warten sein. Dafür wird der Quellcode klar und deutlich dokumentiert und strukturiert, um anderen Entwicklern das Lesen und Warten zu erleichtern.

ID	QA05
Titel	Client-Server-Architektur
Beschreibung	Die Anwendung stellt einen Client dar, der Informationen des HSC Servers erhält, verarbeitet und gegebenenfalls zurücksendet, um sie dort zu speichern.

ID	QA06
Titel	Implementierungssprache
Beschreibung	Die Anwendung soll als Web-App entwickelt werden und die herkömmlichen Webtechnologien HTML, TypeScript und CSS verwenden. Zudem soll das UI-Toolkit Ionic zusammen mit der Javascript-Bibliothek React verwendet werden. Die Implementierungssprache ist Englisch.

5 Entwurf und Architektur der Anwendung

Neben der Anforderungsanalyse ist die Konzeption der Anwendung vor der Implementierung ein wichtiger Teil des Projekts. In diesem Kapitel wird daher zunächst auf den Entwurf und das Konzept der App eingegangen. In den darauffolgenden Abschnitten ist die Architektur des Projekts und abschließend die Kommunikation mit dem Server, welcher die Datenverwaltung der Applikation übernimmt, Thema.

5.1 Entwurf und Konzept

Um einen Entwurf für das User Interface der Applikation zu designen, wurde mit Hilfe des cloudbasierten Design-Tools Figma ein Prototyp erstellt. Dieser umfasst Designs der Kernseiten der Anwendung, wie den Landing-Screen, die verschiedenen Tabs im eingeloggten Zustand und die Seite zum Ausfüllen von Fragebögen. Eine Übersicht der Seitenentwürfe ist in Abbildung 5.1 zu sehen.

Mit verschiedenen Elementen können in Figma beispielsweise Buttons, Textfelder, Überschriften und Listen erstellt werden und sind, sofern sie als Komponenten gespeichert werden, ohne den Aufwand des erneuten Designens wiederverwendbar. Dieses Feature trägt dazu bei, das Design einheitlich zu halten und Änderungen sofort auf alle Instanzen einer Komponente anzuwenden, was den Workflow erheblich erleichtert. Um Icons darzustellen, wie sie zum Beispiel in der Tab-Bar zu sehen sind, wurde das kostenlose Open Iconic Icon Set - Plugin¹ verwendet, welches 223 gängige Icons zur Verfügung stellt.

¹<https://www.figma.com/community/plugin/1146185659935567786/open-iconic-icon-set-by-iconduck>

5 Entwurf und Architektur der Anwendung



Abbildung 5.1: UI-Entwurf der Hauptseiten mit Figma

Der UI-Entwurf konnte mit den von Ionic zur Verfügung gestellten UI-Komponenten ohne größere Abweichungen erfolgreich umgesetzt werden. So ähneln beispielsweise die Shortcuts auf dem Homescreen, dargestellt als Kacheln, der Card-Komponente von Ionic, der Slider auf der Fragebogen-Seite dem Range-Slider und die Datumsanzeigen im Fragebögen-Tab dem Chip-Element von Ionic.

Neben dem Erstellen von verschiedenen Screens erlaubt Figma als Prototyping-

5 Entwurf und Architektur der Anwendung

Tool auch das Entwerfen von Navigationspfaden und bietet so noch vor der Implementierung die Möglichkeit, verschiedene Nutzerflows zu testen. Um dies zu demonstrieren, sind in Abbildung 5.2 alle möglichen Pfade dargestellt, die entweder durch Nutzerinteraktion vom Homescreen weg oder von anderen Seiten zurück zum Homescreen führen.

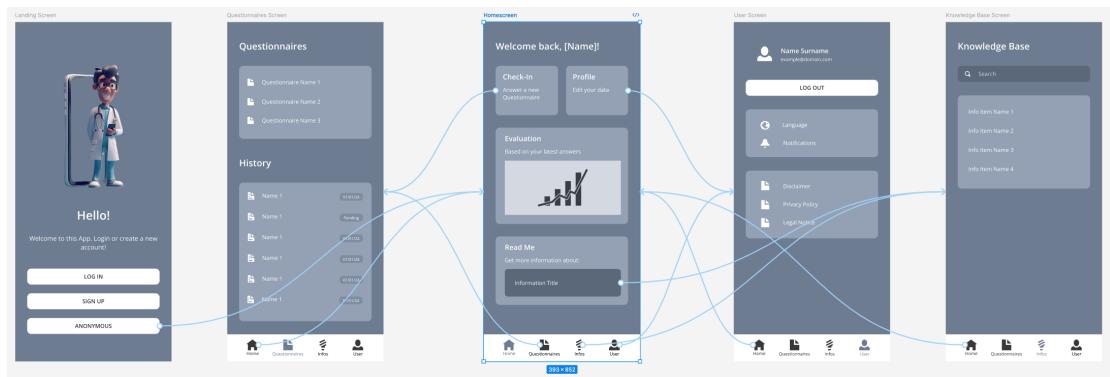


Abbildung 5.2: Navigationselemente, die von und zum Homescreen führen

Mit Hilfe dieses Mock-Ups konnte also erfolgreich die erste Idee der Anwendung visualisiert sowie getestet und später in der Implementierung umgesetzt werden.

5.2 Architektur

Die Architektur der Anwendung soll im folgenden Abschnitt beschrieben werden. Dazu wird zunächst auf das UML-Paketdiagramm eingegangen, welches die Strukturierung von Klassen und Komponenten aufzeigt. Eine Visualisierung davon ist in Abbildung 5.3 zu sehen.

Den Hauptteil der Anwendung beinhaltet das `pages`-Paket. Dort sind alle Seiten, oder auch Screens, der Anwendung zu finden, die mit Ionic durch sogenannte page-Komponenten realisiert werden. Dazu gehören beispielsweise die verschiedenen Tabs, die Landing-Page, der Login- sowie der SignUp-Screen. Zu jeder Seite beinhaltet das Paket auch die passende `.css`-Datei, in welcher seitenspezifische Styles festgelegt werden.

Das Package enthält einen `sub`-Ordner, zu dem alle weiteren Screens gehören, die

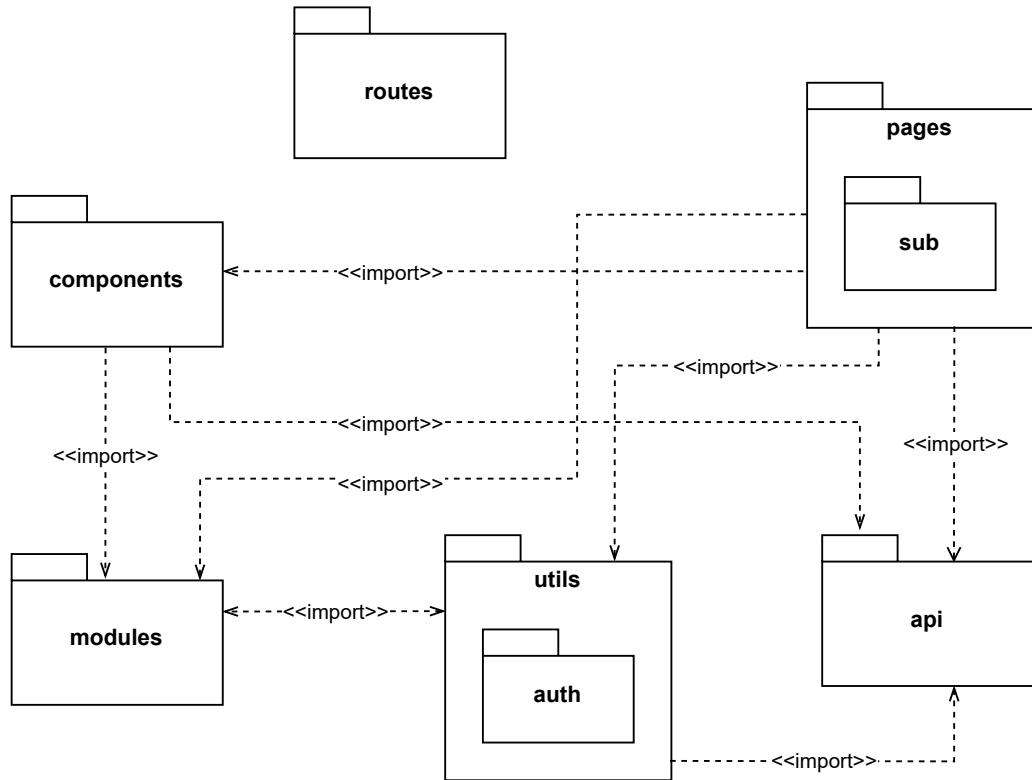


Abbildung 5.3: vollständiges Package-Diagramm der Anwendung

nicht zur Hauptnavigation der Anwendung gehören. Unter anderem befinden sich so dort die Questionnaire-Seite, auf welcher die Fragebögen ausgefüllt werden, die Infopage, zu welcher man über den Knowledge-Tab gelangt, Policy-, Disclaimer- und Legal-Notice-Screen, sowie eine Not-Found-Seite. Letztere dient für alle ungültigen Navigationen, die der Nutzer möglicherweise durchführt, was vor allem bei der Nutzung der App als Webanwendung eine Rolle spielt.

Das `pages`-Package importiert von verschiedenen anderen Paketen, dazu zählt unter anderem das `components`-Package. Dieses Paket enthält Komponenten, die verschiedene Seiten verwenden. Ein Beispiel dafür sind die verschiedenen Fragebogen-Elemente, die auf dem Questionnaire-Screen zu finden sind, wie ein Textfeld, eine Likert- oder Slider-Frage. Durch die Modularisierung der Elemente sind sie dynamisch einsetz- und wiederverwendbar. Darüber hinaus befinden sich Funktionen für die Evaluation der Fragebögen, die sowohl auf dem Homescreen als auch auf der Fragebogen-Seite benötigt werden, in diesem Paket.

Auch von `modules` wird importiert. Dieses Paket beinhaltet Module, die das lokale Speichern von Nutzerdaten übernehmen. Dazu zählen beispielsweise Nutzernname, Passwort, ID des Nutzers und das Session-Token, welches die Anwendung nach erfolgreichem Login des Nutzers vom Server enthält und Zugriff auf Nutzerdaten und weiteren App-spezifischen Inhalt auf dem Server ermöglicht. Für die Kommunikation mit dem Server sind im `api`-Package Gateway- und Tenant-API-Clients zu finden, welche verschiedene Klassen für Nutzer-, Fragebögen- oder Informationsseiten-spezifische Anfragen beinhalten. Von diesem Package machen ebenfalls die Funktionen aus dem `utils/auth`-Paket Gebrauch, die unter anderem für die Registrierung neuer Nutzer beim Server mit Hilfe der API zuständig sind. `utils/auth` importiert weiter von `modules`, um nach der Registrierung die lokale Account-Instanz zu updaten. Neben dem `auth`-Ordner wurden keine weiteren Packages im Rahmen der Anwendung dort angelegt.

Zuletzt befindet sich im `route`-Paket die Private-Route-Komponente, welche dafür sorgt, dass ohne erfolgreiche Anmeldung des Nutzers nicht auf den Kerninhalt der Anwendung zugegriffen werden kann und der Nutzer stattdessen zurück zur Anmeldeseite geleitet wird.

5.3 Serverkommunikation

Die Anwendung nutzt zur externen Speicherung von Daten den Health Study Project Server (HSP)². Dabei zählen zu den Daten einerseits Nutzerdaten, die der Server in einer User-Database verwaltet, zum anderen Fragebögen und Infoseiten, die im Tenant, welcher für die Applikation beim Server angelegt wurde, erstellt werden können. Um die Infrastruktur des Servers nachvollziehen zu können, ist ein Auszug der HSP-Dokumentation diesbezüglich in Abbildung 5.4 zu sehen.

Für die Kommunikation mit dem Server wurden verschiedene API-Clients zur Verfügung gestellt. Wie durch das oben genannte Schaubild deutlich wird, ist die Gateway-API für die Authentifizierung und das User Management verantwortlich. Über sie können neue Nutzer registriert, sich in bereits angelegte Konten eingeloggt und Nutzerdaten wie Name, Passwort und E-Mail-Adresse verwaltet werden. Die Nutzerdatenbank ist dabei unabhängig von den verschiedenen Tenants, welche jeweils

²Die Dokumentation findet sich hier: <https://healthservice.atlassian.net/wiki/spaces/HSPDOCS/>

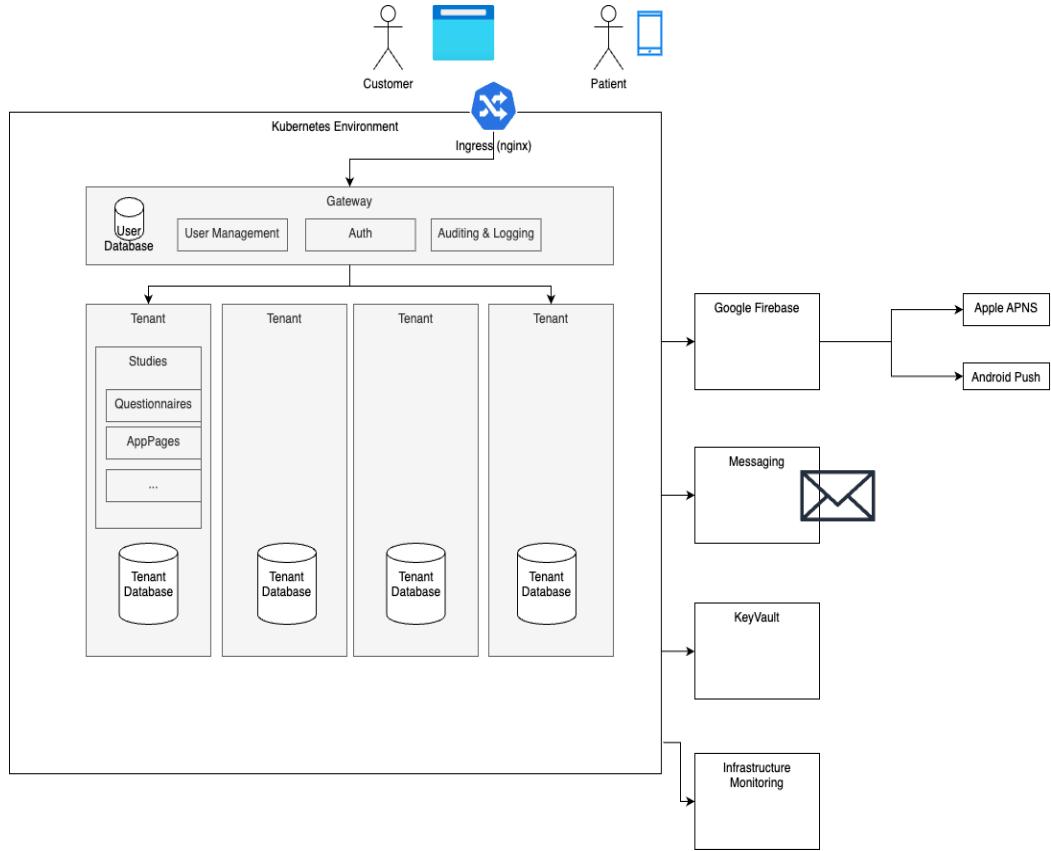


Abbildung 5.4: Übersicht der Server Infrastruktur aus den HSP-Docs

Bereiche für die Anwendungen, die den Dienst nutzen, widerspiegeln.

Für die Authentifizierung stehen zwei Möglichkeiten zur Wahl, ein Resource Owner Password Flow oder ein Client Credentials Flow. Bei ersterem über gibt der Nutzer seine persönlichen Zugangsdaten der Anwendung, die anschließend eine Anfrage mit den entsprechenden Daten an den Server schickt, um diese zu validieren und ein Session Token zu erhalten, mit welchem weitere Anfragen an den Server gestellt werden können [30]. Der Client Credentials Flow hingegen nutzt einen Authentifizierungsservice. Das heißt, der Nutzer wird an den entsprechenden Authentifizierungsservice weitergeleitet, gibt dort seine Daten ein, ohne dass die Anwendung dabei involviert ist, und kehrt nach erfolgreicher Verifizierung zur Anwendung zurück, welche vom Service nun das Zugriffstoken erhält, ohne die Zugangsdaten je selbst abgefragt zu haben [29].

Zunächst sollte letztere Option implementiert werden, da hierbei keine sensiblen

5 Entwurf und Architektur der Anwendung

Daten zwischen Anwendung und Server verschickt werden müssen, jedoch hätte dies eine Registrierung der Anwendung beim Server gefordert, was zum Zeitpunkt der Implementierung nicht möglich war, sodass schließlich der Resource Owner Password Flow für die Authentifizierung von Nutzern verwendet wurde. Der Ablauf des Registrierens ist im Sequenzdiagramm in Abbildung 5.5 dargestellt.

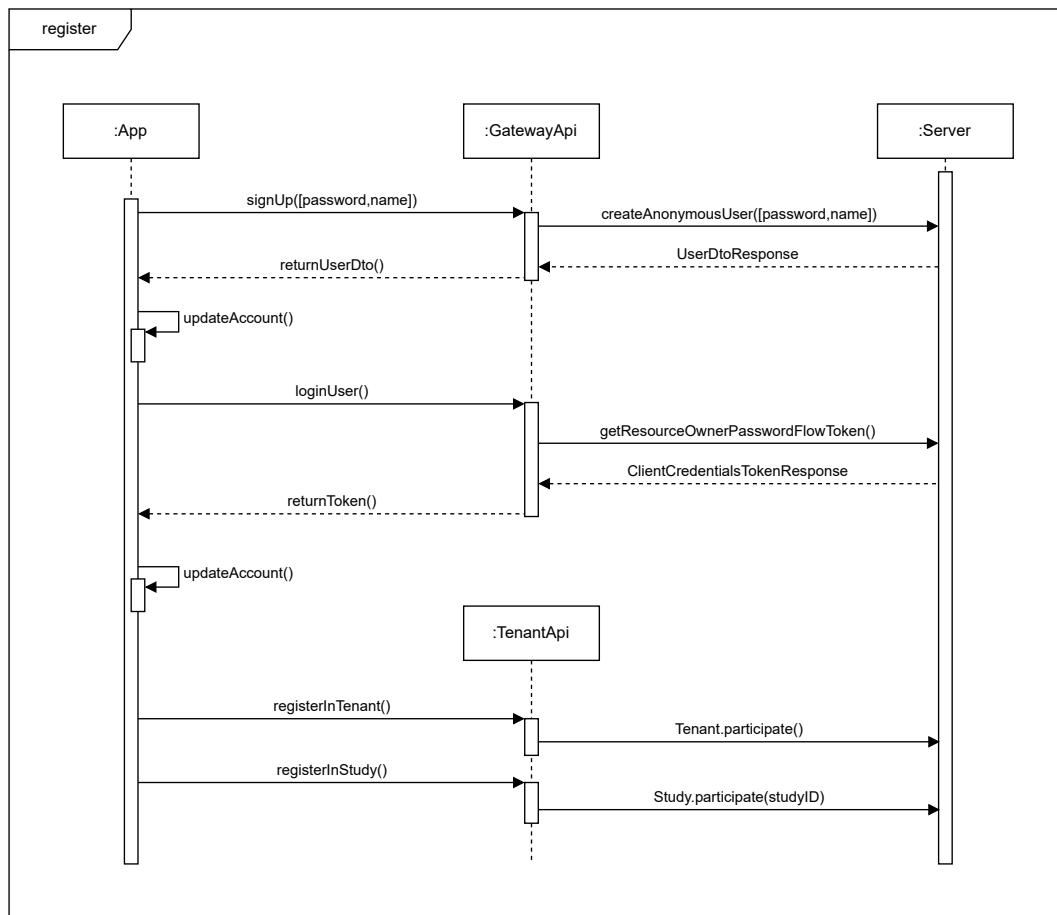


Abbildung 5.5: Sequenzdiagramm bzgl. Registrierung und Login beim Server

Die App erzeugt dafür in der `signUp`-Methode eine Instanz des Gateway-API-Clients, auf welcher die `createAnonymousUser()`-Funktion aufgerufen wird. Bei einer anonymen Registrierung mit der Angabe eines randomisiert generierten Passworts durch die App und bei herkömmlicher Registrierung mit Passwort- und Namenseingabe des Nutzers. Der Server antwortet mit einer `UserDto`, einem Datenobjekt, welches unter anderem den vom Server generierten Nutzernamen und Nutzer-ID ent-

hält. Die Anwendung vervollständigt daraufhin die lokale Accountinstanz mit eben-diesen Daten, um anschließend damit eine Login-Anfrage erneut über die Gateway-API zu versenden. Geschieht dies erfolgreich und kann der Server Nutzernname und Passwort validieren, gibt er ein Zugriffstoken zurück, welches ebenfalls von der Anwendung in der Accountinstanz gespeichert wird. Um die Registrierung abzuschließen, muss der neue User zuletzt in Tenant und Studie eingeschrieben werden, sodass er Zugriff auf die Fragebögen und Informationsseiten erhält. Dies geschieht über die Tenant-API, da es sich dabei um Tenant-spezifische Anfragen handelt und nicht mehr um die Authentifizierung des Users.

Nun kann von der Anwendung mit dem erhaltenen Zugriffstoken Anfragen wie `useTenantApi().appPagesApi.getAppPages(projectId)` an den Server gestellt werden, um beispielsweise die Informationsseiten zu erhalten. Da das Zugriffstoken nur eine begrenzte Gültigkeit hat, führt die API bei einer Ablehnung durch den Server automatisch eine erneute Anmeldung durch, um ein neues Token zu erhalten.

6 Implementierung

In diesem Kapitel wird die Implementierung der in den vorherigen Kapiteln beschriebenen Anforderungen und Konzepte vorgestellt und erläutert. Dazu wird zunächst auf den Anmeldungs- und Registrierungs-Teil eingegangen, anschließend wird ein Überblick über die verschiedenen Tabseiten der Anwendung gegeben und zuletzt soll die Umsetzung des Fragebogen-Features genauer aufgezeigt werden. Zum Programmieren wurde Visual Studio Code in der Version 1.96.4 verwendet und Ionic mit der Version 7.2.0. Zur Visualisierung werden Screenshots der Applikation auf einem Smartphone mit Android 13 als Betriebssystem verwendet.

6.1 Anmeldung und Registrierung

Die Applikation startet mit dem Landing-Screen (Abbildung 6.1a), auf welchem dem Nutzer drei Optionen zum Fortfahren offenstehen: Über den Login-Button kann sich in ein bereits bestehendes Konto angemeldet werden (Abbildung 6.1b), während „Sign Up“ zum Registrierungsscreen (Abbildung 6.1c) führt, auf welchem ein neues Konto mit der Angabe eines Namens und eines Passworts erstellt werden kann. Die Option „Anonymous“ lässt den Nutzer eine Registrierung umgehen und die Applikation mit einem anonymen Konto nutzen.

Intern unterscheidet sich die anonyme und die reguläre Registrierung kaum, beide Optionen verwenden die `createAnonymousUser()`-Funktion der Server-API. Bei einer anonymen Registrierung wird dabei das Erstellen des Passworts durch die Anwendung übernommen und es wird kein Name gesetzt.

Wird eine Anmeldung bei einem nicht existierenden Konto oder falscher Passworteingabe versucht, erhält der Nutzer Feedback mit einem „Login failed“-Alert. Bei der Registrierung wird das Passwort auf eine Mindestanzahl von acht Zeichen geprüft und außerdem, ob beide Passworteingaben übereinstimmen. Andernfalls wird

6 Implementierung

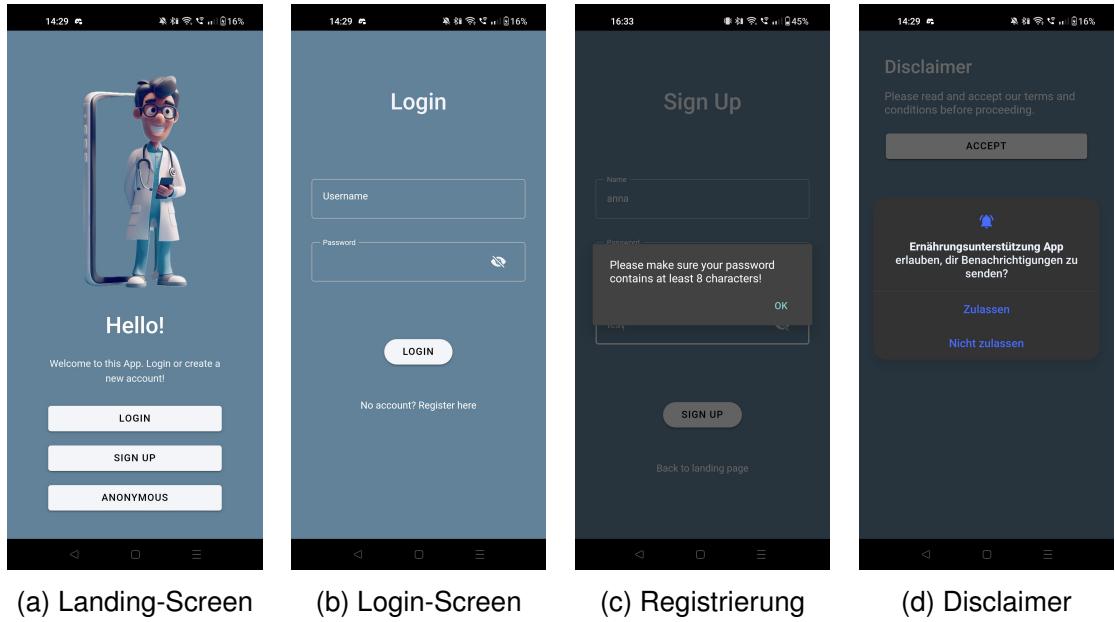


Abbildung 6.1: Screens, die im Login- und Registrierungs-Prozess involviert sind.

eine entsprechende Fehlermeldung angezeigt (Abbildung 6.1c). Dies passiert auch, wenn der eingegebene Name aus weniger als zwei Zeichen besteht.

Die Komponenten des Login- und Registrierungs-Screens konnten mit dem IonInput-Element für die Textfelder und IonButton für jegliche Buttons umgesetzt werden. Dabei bietet das Input-Element die Möglichkeit, es um eine IonInputPasswordToggle-Komponente zu erweitern, welche das Passwort automatisch als Sternchen anzeigt und einen Toggle-Button hinzufügt, der die Sichtbarkeit der Eingabe umstellen lässt (Abbildung 6.1b).

Nach dem Login bzw. der Registrierung wird der Nutzer zum Disclaimer weitergeleitet (Abbildung 6.1d), welcher akzeptiert werden muss, um weiter zur App zu gelangen. Außerdem wird auf diesem Screen der Nutzer ebenfalls zum Akzeptieren von Push-Benachrichtigungen aufgefordert. Dies ist jedoch nicht zum Nutzen der Anwendung notwendig, lediglich das Benachrichtigungs-Feature bleibt bei Ablehnen der Aufforderung verwehrt. Für Letzteres wird das Capacitor-Plugin Local Notifications verwendet, welches lokale Benachrichtigungen für mobile Apps ermöglicht, die direkt auf dem Gerät ausgelöst werden, ohne einen externen Server oder eine Internetverbindung zu benötigen.

6.2 Tab-Seiten

War die Anmeldung oder Registrierung erfolgreich und wurde der Disclaimer akzeptiert, gelangt der Nutzer weiter zum eigentlichen Teil der Applikation.

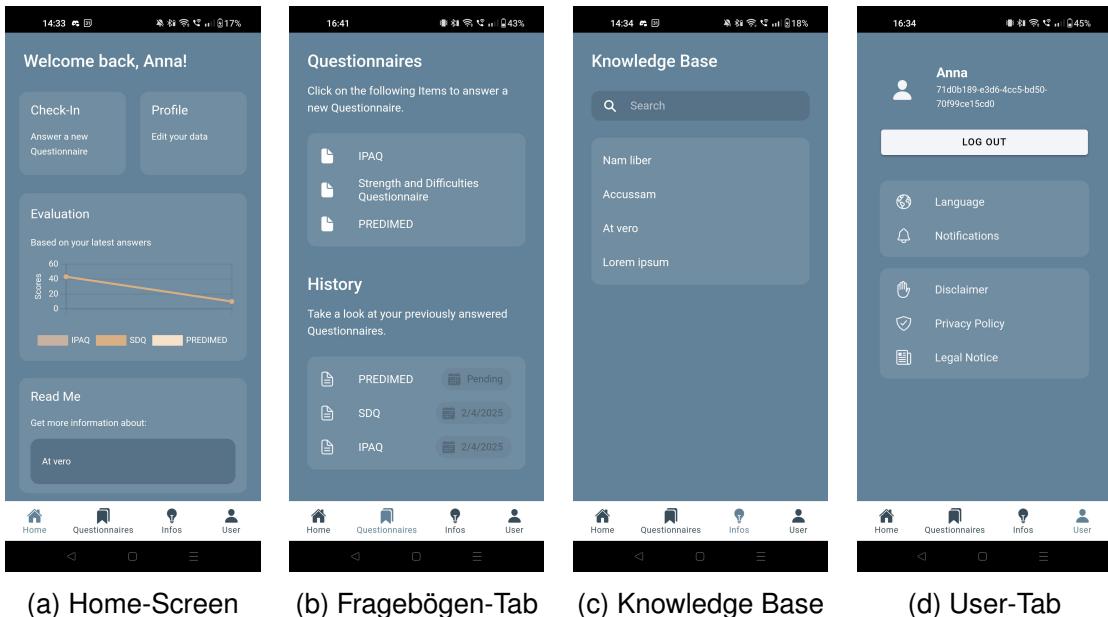


Abbildung 6.2: Die verschiedenen Tabs der Anwendung

Dieser gliedert sich in vier Tabs und startet auf der Homepage, dem ersten Tab (Abbildung 6.2a). Um zu den anderen Screens zu navigieren, wurde eine Tableiste im Footer der App implementiert, Ionic bietet dafür die IonTabBar an. Auf der Homepage besteht außerdem die Möglichkeit, über die drei verschiedenen Kacheln (Check In, Profile, Read Me) zu den jeweiligen Screens zu gelangen. Ein weiteres Element des Homescreens ist die Evaluation-Kachel, auf welcher die Scores der letzten, vollständig ausgefüllten Fragebögen angezeigt werden. Dazu wird die Bibliothek `react-chartjs-2` verwendet, welche React-Komponenten für `Chart.js`, einer JavaScript-Graphenbibliothek, zur Verfügung stellt. Auf die Methode zur Berechnung der Scores soll in Abschnitt 6.3 eingegangen werden, wenn die Implementierung des Fragebogen-Features behandelt wird.

6 Implementierung

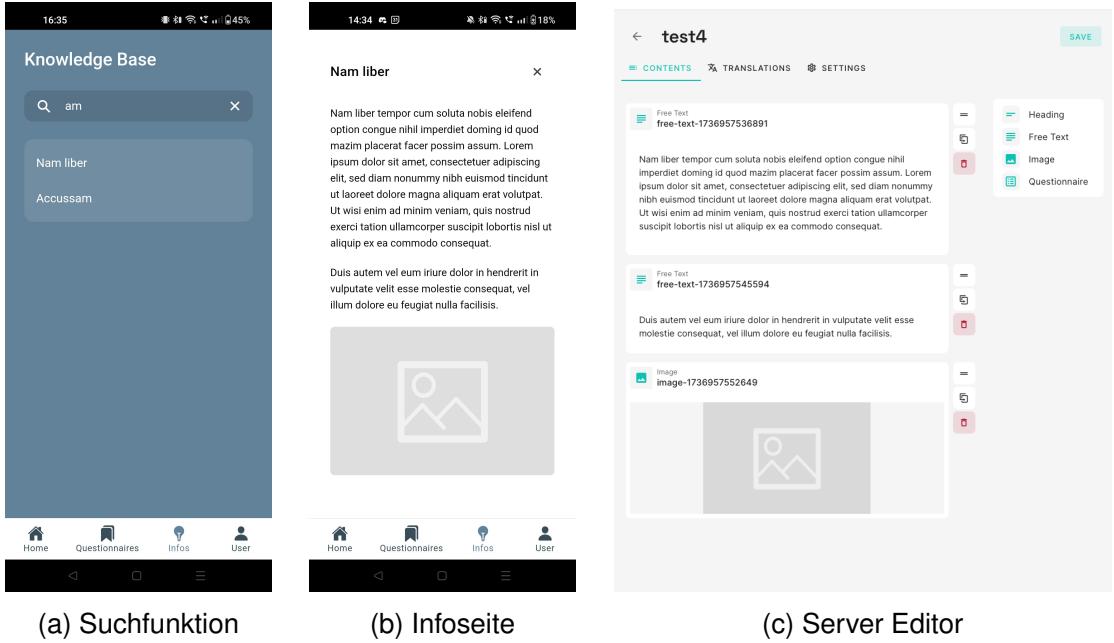


Abbildung 6.3: Funktionen des Knowledge-Base-Tabs

Neben den Fragebögen in Tab Zwei (Abbildung 6.2b), auf welche im nächsten Abschnitt eingegangen wird, befindet sich im dritten Tab die Knowledge Base (Abbildung 6.2c). Dort findet der Nutzer Informationsseiten zu wichtigen Themen bezüglich Stressvermeidung und Ernährungsunterstützung. Diese können auf der Übersichtsseite durchsucht werden, indem der Nutzer Schlüsselwörter in die Suchleiste eingibt und die Titel nach diesen Wörtern durchsucht und gefiltert werden (siehe Abbildung 6.3a). Da das Erstellen der entsprechenden Informationen nicht Teil des Projekts war, wurden Blindtexte eingesetzt, um mögliche Inhalte zu imitieren.

Abbildung 6.3c zeigt das Serverinterface, auf welchem die entsprechenden Informationsseiten angelegt und gespeichert werden können. Das serverseitige Speichern der Texte und Bilder bringt den Vorteil, die Applikation deutlich leichtgewichtiger und flexibler zu halten. Inhalte können so ohne Umstände angepasst werden, ohne dass dazu Änderungen in der Applikation notwendig sind. Die Anwendung lädt die Informationsseiten über die Tenant-API des Servers mithilfe des AppPages-Clients.

Im letzten Tab, dem User-Screen (Abbildung 6.2d), kann der Nutzer verschiedene Einstellungen treffen sowie Appinformationen einsehen. Zunächst kann über das „Language“-Item in der oberen Liste die Sprache der Anwendung geändert wer-

6 Implementierung

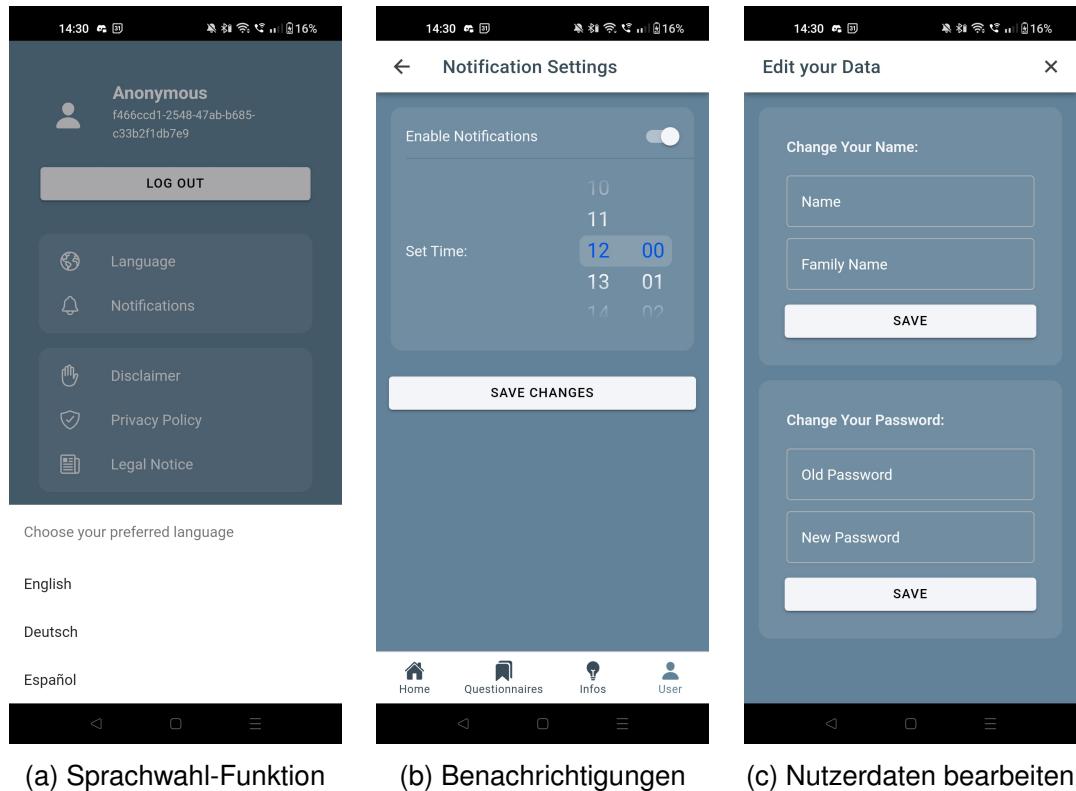


Abbildung 6.4: Funktionen des User-Tabs

den. Die Auswahl besteht dabei aus Englisch, Deutsch und Spanisch, welche in einem IonActionSheet dargestellt wird (Abbildung 6.4a). Über das nächste Element in der Liste gelangt der Nutzer zu den Benachrichtigungseinstellungen, die in Abbildung 6.4b zu sehen sind. Sind diese aktiviert, kann ebenfalls eine Uhrzeit ausgewählt werden, zu welcher täglich eine Benachrichtigung geschickt werden soll. Wie in Abschnitt 6.1 bereits erwähnt wurde, wird dies in der Anwendung mit Hilfe des Plugins Local Notifications umgesetzt.

In der zweiten Itemgruppe befinden sich Disclaimer, die Datenschutzrichtlinie und der Haftungsausschluss. Auf den entsprechenden Unterseiten werden die zugehörigen Texte angezeigt, die den Nutzer bezüglich der Applikation informieren sollen. Zuletzt besteht für den Nutzer die Möglichkeit, sowohl seinen Namen (Vor- und Nachnamen) als auch sein Passwort über diesen Tab zu ändern. Dafür wird auf den Namen des Nutzers neben dem Profil-Icon getippt, woraufhin sich ein Modal öffnet (siehe Abbildung 6.4c). Die beiden Funktionen, die mit den jeweiligen Save-Buttons verknüpft und für das Speichern des neuen Passworts und Namen lokal

sowie serverseitig zuständig sind, werden in den folgenden Abbildungen 6.5a und 6.5b dargestellt. Dabei muss bei der Eingabe eines neuen Passworts auf die Mindestlänge von acht Zeichen und auf die korrekte Eingabe des bisherigen Passworts geprüft werden. Ist dies nicht der Fall, wird der Nutzer mit entsprechenden Alerts auf die jeweiligen Fehler aufmerksam gemacht.

```
async function savePasswordChange() {
    if (account?.password !== formValues.password) {
        alert(t("UserScreen.EditUserDtoModal.AlertPasswordIncorrect"));
    } else if (formValues.newpassword.length < 8) {
        alert(t("UserScreen.EditUserDtoModal.AlertPasswordTooShort"));
    } else {
        const uc = new UpdateCurrentUserPasswordCommand();
        uc.oldPassword = formValues.password;
        uc.newPassword = formValues.newpassword;
        try {
            await useGatewayApi().currentUserApi.updateCurrentUserPassword(uc);
            const na = account;
            na.password = formValues.newpassword;
            setAccount(na);
            updateAccount(account);
            alert(t("UserScreen.EditUserDtoModal.AlertSuccessPassword"));
        } catch (error) {
            alert(t("UserScreen.EditUserDtoModal.AlertFailPassword"));
        }
    }

    // reset form values
    handleInputChange("newpassword", "");
    handleInputChange("password", "");
}
```

(a) savePasswordChange() -Funktion

```
async function saveNameChange() {
    if (account) {
        const uc = new UpdateCurrentUserCommand();
        uc.familyName = formValues.familyName;
        uc.givenName = formValues.name;
        try {
            const response = await useGatewayApi().currentUserApi.updateCurrentUser(uc);
            const na = account;
            na.familyName = response.familyName;
            na.name = response.givenName;
            setAccount(na);
            updateAccount(account);
            alert(t("UserScreen.EditUserDtoModal.AlertSuccessName"));
        } catch (error) {
            alert(t("UserScreen.EditUserDtoModal.AlertFailName"));
        }
    }
}
```

(b) saveNameChange() -Funktion

Abbildung 6.5: Code der Save-Methoden auf der Bearbeitungsseite

6.3 Fragebogen-Feature

Geht es um Stressbewältigung, spielt die Ernährung eine wichtige Rolle. Empfohlen wird eine weitestgehend mediterrane Ernährung (Kreta-Diät), die erwiesenermaßen auch bei der Vorbeugung von Herz-Kreislauf-Krankheiten, Typ-2-Diabetes-mellitus und Darm-, Brust- und Prostata-Krebs hilft [32]. Um den Nutzer hinsichtlich einer solchen Ernährung zu unterstützen und anzuleiten, wurde unter anderem der 14-teilige Fragebogen zur Einhaltung der mediterranen Diät¹ implementiert.

Im zweiten Tab findet der Nutzer eine Übersichtsseite zum Fragebogen-Feature der Anwendung. Zunächst kann über die obere Itemliste, siehe Abbildung 6.2b, eine Fragebogenvorlage zum Ausfüllen ausgewählt werden. Gibt es bereits eine

¹Die Fragen können beispielsweise unter <https://dietamediterranea.com/en/test-de-la-dieta-mediterranea/> eingesehen werden

6 Implementierung

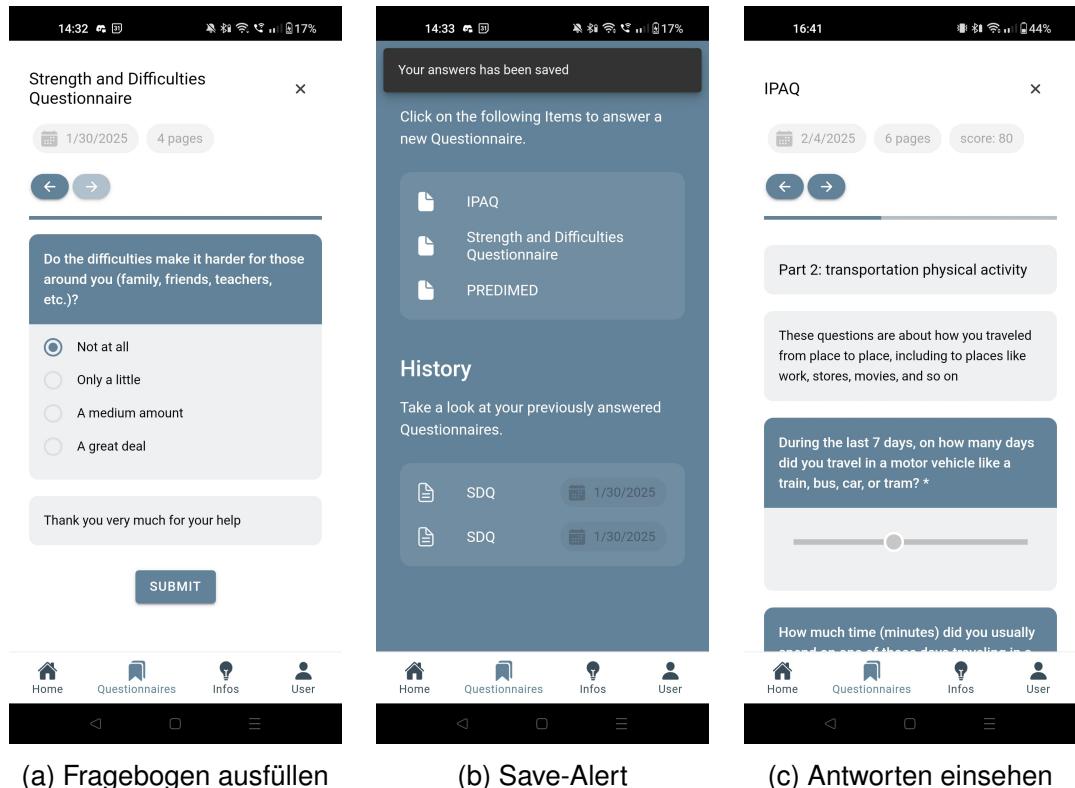


Abbildung 6.6: Screens des Fragebogen-Features

Instanz, die vom Nutzer noch nicht vollständig beantwortet wurde, öffnet die App ebendiese mit den zuletzt angegebenen Antworten und bietet so dem Nutzer die Möglichkeit, das Ausfüllen auch zu einem späteren Zeitpunkt noch fortzusetzen. Gibt es keine unvollständige Instanz, wird eine leere Vorlage geöffnet.

Durch die Seiten des Fragebogens kann mit den Pfeilen im Header des Screens navigiert werden. Damit dies auch bei gescrolltem Content möglich ist, ist der Header, welcher bis zur Progress-Bar reicht, am oberen Bildschirmrand fixiert (Abbildung 6.6a). Neben den Buttons und der Progress-Bar befinden sich zwei Infochips im Header, einer zeigt das Startdatum der Instanz, der andere die Seitenanzahl an.

Am Ende des Fragebogens, sprich auf der letzten Seite unter der letzten Frage, befindet sich bei jedem Fragebogen der Submit-Button. Drückt der Nutzer diesen, werden seine Antworten an den Server gesendet und dort gespeichert. Geschieht dies erfolgreich, wird der Nutzer zurück zum Fragebögen-Tab geroutet und mit einem Toast über das erfolgreiche Speichern seiner Antworten benachrichtigt (Abbildung 6.6b).

Auf der Übersichtsseite ist unter der Überschrift „History“ eine weitere Liste an Fragebögen zu sehen (Abbildung 6.2b). Hier werden alle bisher erstellten Fragebogeninstanzen mit dem jeweiligen Abschlussdatum aufgeführt. Ist eine Instanz noch nicht vollständig beantwortet, ist statt dem Enddatum ein „Pending“ zu sehen. Mit Klicken auf eines der Listenitems kann zu der entsprechenden Instanz im Viewmodus navigiert werden (Abbildung 6.6c). Hierbei kann der Nutzer seine abgegebenen Antworten einsehen, jedoch nicht bearbeiten. Zusätzlich wird in diesem Screen ein weiterer Infochip angezeigt, welcher den Antworten-Score des Fragebogens anzeigt und durch die Funktion in Abbildung 6.7 berechnet wird. Durch dieses Feature ist es dem Nutzer möglich, seine Antworten nachzuverfolgen und Trends im eigenen Verhalten zu erkennen.

```
export function evaluateQuestionnaire(questionnaire: QuestionnaireInstanceDetailsDto): number {
    let score: number = 0;

    questionnaire.pages.forEach(page => {
        page.contents.forEach(item => {
            if (item instanceof SliderQuestionDto || item instanceof NumberQuestionDto) {
                const sliderValue = item.answer?.value ?? 0;
                score += sliderValue;
            } else if (item instanceof ChoiceQuestionDto) {
                const id = item.answer?.values ? item.answer.values[0] : null;
                const choiceValue = id ? Number(item.choices.find(c => c.id === id)?.value ?? 0) : 0;
                score += choiceValue;
            } else if (item instanceof LikertQuestionDto) {
                const id = item.answer?.value ?? null;
                const scaleValue = id ? Number(item.scale.find(c => c.id === id)?.value ?? 0) : 0;
                score += scaleValue;
            }
        });
    });
    return score;
}
```

Abbildung 6.7: evaluateQuestionnaire-Funktion

Der Fragebogen besteht aus verschiedenen Elementen, Beispiele dafür sind Likert-Fragen, Textfragen oder Range-Fragen, welche in der Anwendung mit einem Slider dargestellt werden.

Abbildung 6.8 zeigt, welche Antwort die Anwendung vom Server erhält, wenn eine spezifische Fragebogeninstanz geladen wird. Unter `pages.contents[]` sind die Inhalte der verschiedenen Fragebogenseiten zu sehen, wie `TextDisplay` (für Überschriften), `RichTextDisplay` (für Erklärungsblöcke), `ChoiceQuestion` oder auch `SliderQuestion`. Um die verschiedenen Elemente in dem einzelnen `contents-`

6 Implementierung

```

{
  "id": "30312188-c4d8-4cb6-844b-d90f43c0dc70",
  "name": "IPAQ",
  "language": "de",
  "version": "2.0",
  "status": "published",
  "lastModified": "2025-02-04T17:40:26Z",
  "languages": [
    {
      "id": "30312188-c4d8-4cb6-844b-d90f43c0dc70"
    }
  ],
  "pages": [
    {
      "id": "ab51c96a-ceb5-4828-9140-3ab6a252e4ce",
      "name": "page1",
      "elementType": "ContentPage"
    }
  ],
  "sections": [
    {
      "id": "6a88cf0a-21b4-47ef-9935-5b0fe9143d6a",
      "name": "header1",
      "elementType": "TextDisplay"
    },
    {
      "id": "e3bdd88c-e63d-429e-b99e-488e7676a045",
      "name": "text3",
      "elementType": "RichTextDisplay"
    }
  ],
  "questions": [
    {
      "id": "6b52d954-5687-4592-8d17-c59f580cd2dd",
      "name": "question1",
      "elementType": "ChoiceQuestion"
    },
    {
      "id": "fb0f7164-6d52-4e53-a348-3064193432f5",
      "name": "text4",
      "elementType": "RichTextDisplay"
    },
    {
      "id": "86159cc-f71a-493b-a244-3d19a43ed2a3",
      "name": "question2",
      "elementType": "SliderQuestion"
    }
  ],
  "answers": [
    {
      "id": "a20ef1fd-6b90-497f-96df-b1d1b8915e5c",
      "name": "question4",
      "elementType": "NumberQuestion"
    },
    {
      "id": "45bb3f54-8dc8-4068-9854-cf81f34bc45",
      "name": "question5",
      "elementType": "NumberQuestion"
    },
    {
      "id": "a67f4980-69cb-4055-baef-47315e6f7b4",
      "name": "question6",
      "elementType": "NumberQuestion"
    },
    {
      "id": "1e0c08b5-4597-4a6c-8b00-b3c96e240c39",
      "name": "question7",
      "elementType": "NumberQuestion"
    }
  ],
  "responses": [
    {
      "id": "8d0ff31d3-ad04-4546-ab7f-cded8006c6e",
      "name": "page3",
      "elementType": "ContentPage"
    },
    {
      "id": "756bd03f-362d-4618-9500-9a599f9ae352",
      "name": "page4",
      "elementType": "ContentPage"
    },
    {
      "id": "5ab260a5-63bf-4194-8a16-3a08a55a1965",
      "name": "page5",
      "elementType": "ContentPage"
    },
    {
      "id": "9e01baei-f3dd-4c12-9db5-164de5b6dd77",
      "name": "page6",
      "elementType": "ContentPage"
    }
  ],
  "sharingOptions": []
}

```

Abbildung 6.8: Server Antwort auf Fragebogenanfrage

Array zu speichern, implementieren sie alle den abstrakten Typ ContentDto.

Nun muss die Anwendung den Typ erkennen und dynamisch auf die entsprechende UI-Komponente mappen. Die Schwierigkeit, hierbei eine Lösung zu finden, zeichnete sich dadurch aus, dass nicht alle ContentDto-Elemente die gleichen Variablen beinhalten. Ein Beispiel dafür ist die answer-Variable, welche es nur in Frage-Komponenten gibt. TypeScript weiß dabei nicht, um welche Komponenten es sich jeweils handelt, sondern sieht nur den abstrakten Typ. Dies führt zu Compiler-Fehlern, wenn auf entsprechende Variablen zugegriffen wird - selbst dann, wenn das Element zuvor explizit überprüft wurde.

Um dieses Problem zu umgehen, muss also zunächst vom Typ ContentDto in die

6 Implementierung

jeweiligen konkreten Content-Typen geparsst werden. Abbildung 6.9 zeigt, wie anhand der ElementType-Variable auf die entsprechende Komponente gemappt wird, um die Antwort des Servers in die entsprechenden UI-Elemente zu übersetzen, während in Abbildung 6.10 zu sehen ist, wie diese Map im Fragebogenseiten-Code eingesetzt wird. PageSegment entspricht dabei einer Unterseite des Fragebogens.

```
const componentMap: {
  [key in ElementType]?: React.FC<{ questionItem: any,
    onAnswerChange: (questionId: string, answer: any) => void,
    answer: AnswerDto | null,
    viewOnly: boolean }>
} = {
  [ElementType.LikertQuestion]: LikertQuestion as React.FC<{ questionItem: ContentDto }>,
  [ElementType.ChoiceQuestion]: ChoiceQuestion as React.FC<{ questionItem: ContentDto }>,
  [ElementType.RichTextDisplay]: TextItem as React.FC<{ questionItem: ContentDto }>,
  [ElementType.TextDisplay]: HeaderItem as React.FC<{ questionItem: ContentDto }>,
  [ElementType.TextQuestion]: TextQuestion as React.FC<{ questionItem: ContentDto }>,
  [ElementType.NumberQuestion]: NumberQuestion as React.FC<{ questionItem: ContentDto }>,
  [ElementType.SliderQuestion]: SliderQuestion as React.FC<{ questionItem: ContentDto }>,
};
```

Abbildung 6.9: Fragebogen-Items Map

```
const PageSegment: React.FC<PageSegment> = ({ pages, pageId, onAnswerChange, onSubmit, answers, viewOnly }) => {
  const { t } = useTranslation();
  let content = pages[pageId].contents;
  const data = content.map(quest => {
    const Component = componentMap[quest.elementType];
    if (Component) {
      return <Component questionItem={quest} key={quest.id} onAnswerChange={onAnswerChange}
        answer={answers[quest.id] || null} viewOnly={viewOnly} />;
    }
    return <p key={quest.id}>Item Type Unknown</p>;
  });

  return (
    <div id="pagecontent">
      {data}
      <div className="center">{ (pageId === pages.length - 1 && !viewOnly) ? (
        <IonButton id="submitB" onClick={onSubmit}>{t("QuestionnaireScreen.Button")}</IonButton>
      ) : null}
      </div>
    </div>
  );
}
```

Abbildung 6.10: Seitenkomponente

7 Anforderungsabgleich

Dieses Kapitel dient zur Überprüfung, inwiefern die entwickelte Anwendung die zuvor in Kapitel 4 definierten Anforderungen erfüllt. Dabei werden die funktionalen und nicht-funktionalen Anforderungen systematisch evaluiert und mögliche Abweichungen oder offene Punkte identifiziert. Durch die Evaluation wird die tatsächliche Qualität und Funktionalität der App geprüft.

7.1 Funktionale Anforderungen

Zunächst sollen die in Kapitel 4.2 spezifizierten funktionalen Anforderungen auf erfolgreiche Umsetzung überprüft werden.

ID	FA01
Titel	Registrierung
Erfüllt	Ja
Begründung	Über den Landing-Screen kann der Nutzer sich mit einem Namen und einem Passwort registrieren. Ist das Passwort zu kurz, wird eine Fehlermeldung angezeigt.

7 Anforderungsabgleich

ID	FA02
Titel	Anmeldung
Erfüllt	Ja
Begründung	Nutzer können sich über den Landing-Screen mit den korrekten Daten einloggen. Bei ungültigen Angaben wird eine Fehlermeldung ausgegeben. Wurde sich auf einem Gerät schon mal angemeldet, so wird dies gespeichert und beim nächsten Öffnen die Anmeldung automatisch durchgeführt.

ID	FA03
Titel	Anonyme Registrierung
Erfüllt	Ja
Begründung	Nutzer haben auf dem Landing-Screen die Option des anonymen Registrierens und müssen dabei keine Daten angeben.

ID	FA04
Titel	Abmeldung
Erfüllt	Ja
Begründung	Der Nutzer kann sich jederzeit von der Anwendung abmelden. Seine Daten gehen dadurch nicht verloren.

ID	FA05
Titel	Navigation mit Tabs
Erfüllt	Ja
Begründung	Die App enthält eine entsprechende Navigation. Home-screen, Fragebogen-Tab, Info-Tab und ein Userscreen wurden implementiert.

7 Anforderungsabgleich

ID	FA06
Titel	Nutzerseite anzeigen
Erfüllt	Ja
Begründung	Der Userscreen stellt dem Nutzer Informationen zu seinem Konto, Einstellungsmöglichkeiten und Informationen zur Anwendung bereit.

ID	FA07
Titel	Nutzerdaten verwalten
Erfüllt	Ja
Begründung	Der Nutzer kann seinen Namen (Vor- und Nachname) in der App ändern. Die Änderungen werden serverseitig gespeichert.

ID	FA08
Titel	Passwort ändern
Erfüllt	Ja
Begründung	Das Passwort kann über den Userscreen geändert werden. Dazu muss das bisherige Passwort korrekt eingegeben werden und das neue Passwort mind. 8 Zeichen lang sein. Ansonsten wird eine entsprechende Fehlermeldung ausgegeben.

7 Anforderungsabgleich

ID	FA09
Titel	Sprache ändern
Erfüllt	Ja
Begründung	Die Applikation steht in den drei Sprachen Englisch, Deutsch und Spanisch zur Verfügung. Der Nutzer kann seine gewünschte Sprache auf dem Userscreen über einen Button auswählen.

ID	FA10
Titel	Benachrichtigungen verwalten
Erfüllt	Teilweise
Begründung	Bei Erstanmeldung bzw. -installation wird der Nutzer zum Akzeptieren von Benachrichtigungen aufgefordert. Über den Userscreen kann der Nutzer über die Einstellungselement „Benachrichtigungen“ diese an- und ausschalten, sowie bei angeschalteten Benachrichtigungen die Uhrzeit dieser einstellen. Auf manchen Endgeräten kommt es zu unerwünschtem Verhalten wie beispielsweise sich minütlich wiederholenden Benachrichtigungen.

ID	FA11
Titel	Disclaimer
Erfüllt	Ja
Begründung	Bei Erstanmeldung bzw. -installation muss der Nutzer vor dem Verwenden der App einen entsprechenden Disclaimer akzeptieren. Den Disclaimer kann der Nutzer auch über den Userscreen einsehen.

7 Anforderungsabgleich

ID	FA12
Titel	Impressum, Datenschutzerklärung einsehen
Erfüllt	Teilweise
Begründung	Der Nutzer kann über den Userscreen das Impressum und die Datenschutzerklärung einsehen. Jedoch wurden hier Blindtexte eingesetzt, da keine entsprechende Datenschutzerklärung bzw. kein Impressum aufgesetzt wurden.

ID	FA13
Titel	Fragebögen-Seite
Erfüllt	Ja
Begründung	Es wurde der Fragebögen-Tab implementiert, der eine Übersicht von einerseits ausfüllbaren Fragebogen-Vorlagen und andererseits bereits ausgefüllten Fragebögen gibt. Die Daten dafür werden dabei dynamisch vom Server geladen.

ID	FA14
Titel	Fragebogen ausfüllen
Erfüllt	Ja
Begründung	<p>Durch Auswählen eines Fragebogen-Items in der Übersichtseite kann der Nutzer einen Fragebogen ausfüllen. Die Antworten des Nutzers werden durch das Klicken auf einen Button an den Server zurückgesendet und dort gespeichert. Mit Buttons kann der Nutzer durch die Fragebogenseiten navigieren. Eine Bar zeigt den Fortschritt im Fragebogen an. Es wird erst eine neue Fragebogen-Instanz erstellt, wenn die vorherige vollständig ausgefüllt wurde. Ansonsten wird die vorherige Instanz geladen, und der Nutzer kann diese weiter bearbeiten.</p>
ID	FA15
Titel	Ausgefüllten Fragebogen einsehen
Erfüllt	Ja
Begründung	<p>Durch Auswählen eines Fragebogen-Items in der Verlaufsliste kann der Nutzer seine Antworten zu einem bereits ausgefüllten Fragebogen einsehen. In dieser Ansicht kann der Nutzer den Fragebogen nicht weiter bearbeiten. Oben auf der Seite werden neben dem Namen des Fragebogens auch Startdatum, Anzahl der Seiten und der erreichte Score angezeigt.</p>

ID	FA16
Titel	Knowledge-Base-Seite
Erfüllt	Ja
Begründung	Die Anwendung implementiert den Knowledge-Base-Tab. Dabei werden vorhandene Informationsseiten vom Server geladen und in einer Liste dargestellt. Diese Übersichtsseite zu den verschiedenen Informationen kann durchsucht werden und filtert entsprechend Titel nach der Nutzereingabe.

ID	FA17
Titel	Einzelne Info-Seite ansehen
Erfüllt	Teilweise
Begründung	Die Listenelemente der Knowledge-Base-Seite navigieren zu den jeweiligen Informationsseiten. Die Seiten laden entsprechende Texte und Bilder vom Server. Jedoch handelt es sich dabei um Blindtexte und Platzhalter-Bilder.

ID	FA18
Titel	Schnellzugriffe Home-Seite
Erfüllt	Ja
Begründung	Auf der Homepage kann über unterschiedliche Kacheln zu der Fragebogen-Seite, zur Knowledge-Base und zum Userscreen navigiert werden.

ID	FA19
Titel	Evaluation Home-Seite
Erfüllt	Ja
Begründung	Ein Graph zeigt die Scores der zuletzt ausgefüllten Fragebögen auf dem Homescreen.

7.2 Nicht-funktionale Anforderungen

Abschließend werden in diesem Abschnitt alle zuvor in Kapitel 4.3 definierten Qualitätsanforderungen mit der Implementierung abgeglichen und auf die tatsächliche Erfüllung hinsichtlich der Qualität der Anwendung überprüft.

ID	QA01
Titel	Robustheit
Erfüllt	Ja
Begründung	Die Anwendung ermöglicht eine reibungslose Verwendung aller Funktionen. Ladezustände werden dem Nutzer angezeigt und verhindern für den Moment weitere Nutzerinteraktionen, so dass das System nicht überlastet wird. Bei fehlerhaften Nutzereingaben fängt das System diese ab, bleibt in seinem Zustand und benachrichtigt den Nutzer entsprechend.

ID	QA02
Titel	Usability und Intuitivität
Erfüllt	Ja
Begründung	Das User Interface der App nutzt typische Navigations-elemente wie Tabs, Kacheln für Schnellzugriffe und Buttons zum vor und zurück navigieren innerhalb der Fragebögen. Es werden keine weiteren technischen Kenntnis-se vorausgesetzt.

ID	QA03
Titel	Kompatibilität
Erfüllt	Ja
Begründung	Die Anwendung läuft reibungslos und in vollem Umfang im Browser und auf allen aktuelleren Android- und iOS-Versionen. Dafür wird das plattformübergreifende Frame-work Capacitor von Ionic verwendet.

ID	QA04
Titel	Wartbarkeit
Erfüllt	Ja
Begründung	Der Quellcode der Anwendung ist klar strukturiert und dokumentiert, was anderen Entwicklern das Lesen und Warten der App erleichtert.

7 Anforderungsabgleich

ID	QA05
Titel	Client-Server-Architektur
Erfüllt	Ja
Begründung	Die Anwendung stellt einen Client dar, der Informationen des HSC-Servers erhält und verarbeitet, wie beispielsweise für die Knowledge-Base. Es werden ebenfalls Daten zurücksendet, um sie dort zu speichern. Dabei handelt es sich um die Antworten des Nutzers zu den Fragebögen. Das externe Speichern verhindert den Verlust von Nutzerdaten, auch bei Gerätewechseln.

ID	QA06
Titel	Implementierungssprache
Erfüllt	Ja
Begründung	Die Anwendung ist als Web-App entwickelt und nutzt die herkömmlichen Webtechnologien HTML, TypeScript und CSS. Zudem wird das UI-Toolkit Ionic zusammen mit der Javascript-Bibliothek React verwendet. Die Implementierungssprache ist Englisch.

8 Zusammenfassung und Ausblick

Im abschließenden Kapitel wird die entwickelte Anwendung zusammenfassend betrachtet und die zentralen Ergebnisse der Arbeit reflektiert. Zudem wird ein Ausblick auf mögliche Weiterentwicklungen gegeben, um zukünftige Optimierungspotenziale und Erweiterungsmöglichkeiten aufzuzeigen.

8.1 Zusammenfassung

Ziel dieser Arbeit war es, eine webbasierte Cross-Plattform-Anwendung zu entwickeln, welche zur Präsentation von stressreduzierenden Inhalten bzw. Inhalten zur Ernährungsunterstützung genutzt werden soll. Dadurch richtet sich die Applikation besonders an Nutzer, denen es schwerfällt, in Zeiten von übermäßigem Stress und emotionalen Belastungen an gesunden Ernährungsgewohnheiten festzuhalten. Das Präsentieren von Informationen und das regelmäßige Ausfüllen von Fragebögen unterstützen den Nutzer dahingehend, langfristig ungesunde Angewohnheiten zu erkennen und dagegen anzugehen.

In Kapitel 2 wurden zunächst die Grundlagen für das technologische Verständnis zur Entwicklung der Anwendung gelegt. Dabei wurde auf das Coding-Tool Visual Studio Code, das Framework Ionic, die Programmiersprache TypeScript und die Webtechnologien HTML und CSS eingegangen. Verwandte Arbeiten wurden in Kapitel 3 vorgestellt und dienten der Inspiration sowie der Einordnung der Arbeit in ihren Kontext. Anschließend wurde der Entwicklungsteil in Kapitel 4 durch eine Anforderungsanalyse eingeleitet. Anwendungsfälle wurden identifiziert und daraus funktionale sowie nicht-funktionale Anforderungen abgeleitet. Kapitel 5 beschreibt den Entwurfs- und Konzeptprozess, in welchem Design und Architektur der Anwendung entwickelt wurden. Anschließend wurde in Kapitel 6 die Implementierungslösung

sung vorgestellt. Dabei wurde ein Überblick über die gesamte entstandene Applikation geboten, sowie das Fragebogen-Feature genauer vorgestellt. Um die Arbeit abzuschließen, wurden anschließend die zuvor in Kapitel 4 definierten Anforderungen auf Vollständigkeit und Abweichungen geprüft, um die tatsächliche Qualität und Funktionalität der Anwendung zu evaluieren.

8.2 Mögliche Limitationen und Ausblick

Die Applikation ist bereits nutzbar und bietet einige nützliche Funktionalitäten, um dem Nutzer bei der Steigerung seines Wohlbefindens möglicherweise zu helfen. Kapitel 7, welches den Anforderungsabgleich behandelt, hat gezeigt, welche Funktionalitäten erfolgreich umgesetzt werden konnten, jedoch wurden auch potenzielle Limitationen der Anwendung aufgedeckt. Zum einen konnte die implementierte Benachrichtigungsfunktion nicht erfolgreich getestet werden und wies fehlerhaftes Verhalten durch minütliches Senden von Benachrichtigungen auf. Dies kann zu eingeschränkter Nutzbarkeit der App führen.

Weiterhin ist anzumerken, dass zwar eine Auswertung der Fragebögen implementiert wurde, diese jedoch in ihrer Aussagekraft eingeschränkt ist. Ein bloßer Zahlenwert ist möglicherweise für die Nutzer weniger hilfreich als eine Auswertung mit Skala und Vergleichswerten. Darüber hinaus wäre das Einbringen einer textbasierten Auswertung der Fragebögen, um dem Nutzer Feedback zukommen zu lassen, eine nützliche Erweiterung. Ebenso könnten individuelle Inhaltsvorschläge zu Themen, bei welchen der Nutzer laut den ausgefüllten Fragebögen Defizite aufweist, eine hilfreiche Ergänzung zur aktuellen Version der Applikation darstellen .

Ferner wurde festgehalten, dass einige Seiten der Applikation bisher mit Blindtexten gefüllt sind, beispielsweise sämtliche Informationsseiten der Wissensdatenbank. Eine mögliche Weiterentwicklung wäre demnach, die Applikation mit tatsächlichen Informationen zu füllen.

Darüber hinaus könnten Erweiterungen im Bereich der Personalisierung den Nutzer zu häufigerem Benutzen der App verhelfen. Beispielsweise könnte das Nutzer-Profil um einen Avatar erweitert werden. Auch das individuelle Anpassen der UI-Farbpalette oder die Implementierung eines Dark Modes könnten das Verwenden der Anwendung für den Nutzer angenehmer gestalten.

In Hinblick darauf ist diese Anwendung auch in ihrer Barrierefreiheit limitiert. Wäh-

8 Zusammenfassung und Ausblick

rend drei Sprachen implementiert wurden, steht keine Auswahl für beispielsweise einfache Sprache, große Schrift oder gewünschte Farbkontraste bereit. Dies führt potenziell zu einem ungewollten Einschränken des Nutzerkreises und macht die Applikation unzugänglich für bestimmte Nutzergruppen.

Zusammenfassend läuft die in dieser Arbeit entwickelte Applikation zuverlässig und bietet bereits einige vorteilhafte Features, um potenziellen Nutzern Inhalte zur Stressreduzierung und Ernährungsunterstützung zu präsentieren, sowie sie in der Besserung ihres Wohlbefindens zu unterstützen. Durch das Einbinden tatsächlicher Informationen und Weiterentwicklungen in der Personalisierung der Applikation könnte sich diese zu einer wertvollen Unterstützung für Betroffene erweisen.

Literatur

- [1] Harshil Agrawal und David Fateh. *What is TypeScript and why should you use it?* 2024. URL: <https://www.contentful.com/blog/what-is-typescript-and-why-should-you-use-it/> (besucht am 09.02.2025).
- [2] Laura Basten. *Gamification: Grundbegriffe, Chancen und Risiken.* 2024. URL: <https://www.bpb.de/themen/kultur/digitale-spiele/504558/gamification-grundbegriffe-chancen-und-risiken/> (besucht am 10.02.2025).
- [3] Canida. *Ionic - Einfach erklärt.* 2025. URL: <https://canida.io/wissen/ionic> (besucht am 06.02.2025).
- [4] Valeriu Crudu und MoldStud Research Team. *What are the common challenges faced by Ionic developers?* 2024. URL: <https://moldstud.com/articles/p-what-are-the-common-challenges-faced-by-ionic-developers> (besucht am 06.02.2025).
- [5] Ionic Docs. *Introduction to Ionic.* 2025. URL: <https://ionicframework.com/docs> (besucht am 06.02.2025).
- [6] MDN Web Docs. *CSS: Cascading Style Sheets.* 2025. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (besucht am 10.02.2025).
- [7] MDN Web Docs. *HTML: HyperText Markup Language.* 2025. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (besucht am 10.02.2025).
- [8] MDN Web Docs. *What is CSS?* 2025. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/What_is_CSS (besucht am 10.02.2025).

- [9] Jan Entzminger. *Cross Plattform App Development: Erklärung und Case Study*. 2022. URL: <https://www.blindwerk.de/magazin/cross-plattform-app-development-erklaerung-case-study/> (besucht am 10.02.2024).
- [10] Olha Faryma. *BeeHealthy: Eine ganzheitliche Gesundheits-App für Jugendliche*. 2024. URL: <https://www.bfh.ch/de/aktuell/storys/2024/gesundheits-app-fuer-jugendliche-beehealthy/> (besucht am 10.02.2025).
- [11] Ionic Framework. *Components Documentation*. 2025. URL: <https://ionicframework.com/docs/components> (besucht am 06.02.2025).
- [12] Ionic Framework. *Core Concepts*. 2025. URL: <https://ionicframework.com/docs/core-concepts/fundamentals> (besucht am 06.02.2025).
- [13] Ionic Framework. *Ionic React Documentation*. 2025. URL: <https://ionicframework.com/docs/react> (besucht am 06.02.2025).
- [14] Ionic Framework. *Platform Styles*. 2025. URL: <https://ionicframework.com/docs/theming/platform-styles> (besucht am 06.02.2025).
- [15] Ionic Framework. *WebView - Core Concepts*. 2025. URL: <https://ionicframework.com/docs/core-concepts/webview> (besucht am 06.02.2025).
- [16] Josh Goldberg. *Learning TypeScript*. O'Reilly Media, 2022, S. 45–50. ISBN: 978-1098110338.
- [17] Martin Heller. *Was ist Visual Studio Code?* 2024. URL: <https://www.computerwoche.de/article/3611333/was-ist-visual-studio-code.html> (besucht am 06.02.2025).
- [18] Wilde IT. *Was ist React und wofür wird es eingesetzt?* 2025. URL: <https://www.wilde-it.com/react/> (besucht am 06.02.2025).
- [19] Tobias Kahlert und Kay Giza. *Visual Studio Code - Tipps & Tricks Vol. 1*. 1. Auflage, 1. Aktualisierung im April 2016. Unterschleißheim, Deutschland: Microsoft Deutschland GmbH, 2016. URL: <http://aka.ms/VSCodeTippsTricks>.
- [20] Jerica Koh u.a. „Potential and Pitfalls of Mobile Mental Health Apps in Traditional Treatment: An Umbrella Review“. In: *Journal of Personalized Medicine* 12.9 (2022), S. 1376. DOI: 10.3390/jpm12091376.

Literatur

- [21] Yuanbo Tech LLC. *Moodpress - Stress Monitor & Mood Tracker & Mood Diary*. 2022. URL: <https://yoobool.com/moodpress/> (besucht am 10.02.2025).
- [22] Franzi Lautenbach u. a. „Take it easy“: eine Interventionsstudie zur Wirksamkeit einer Anti-Stress-App auf psychologischen physiologischen akuten sowie chronischen Stress“. In: *Abstractband der 50. Jahrestagung der asp*. Hrsg. von Borges Uirassu u. a. Deutsche Sporthochschule Köln, Mai 2018, S. 23–24.
- [23] Lenovo. *Was ist eine Markup-Sprache und warum ist sie wichtig?* 2025. URL: <https://www.lenovo.com/de/de/glossary/markup-language/> (besucht am 10.02.2025).
- [24] Ben Lutkevich. *HTML (Hypertext Markup Language)*. 2025. URL: <https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language> (besucht am 10.02.2025).
- [25] Microsoft. *Getting started with the terminal*. Microsoft. 2025. URL: <https://code.visualstudio.com/docs/terminal/getting-started> (besucht am 06.02.2025).
- [26] Microsoft. *IntelliSense in Visual Studio Code*. Microsoft. 2025. URL: <https://code.visualstudio.com/docs/editor/intellisense> (besucht am 06.02.2025).
- [27] Microsoft. *Refactoring in Visual Studio Code*. Microsoft. 2025. URL: <https://code.visualstudio.com/docs/editor/refactoring> (besucht am 06.02.2025).
- [28] Microsoft. *Source Control in Visual Studio Code*. Microsoft. 2025. URL: <https://code.visualstudio.com/docs/sourcecontrol/overview> (besucht am 06.02.2025).
- [29] Okta. *Client Credentials Flow*. 2025. URL: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/client-credentials-flow> (besucht am 10.02.2024).
- [30] Okta. *Resource Owner Password Flow*. 2025. URL: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/resource-owner-password-flow> (besucht am 10.02.2024).

Literatur

- [31] Pegotec. *TypeScript in Mobile Application Development*. 2023. URL: <https://pegotec.net/embracing-typescript-how-pegotec-leverages-typescript-to-develop-next-gen-mobile-applications/> (besucht am 09.02.2025).
- [32] Alexandra Schek. „Einfluss der Ernährung auf Depressivität und Stresstoleranz“. In: *Psychotherapeutische Praxis* 3.4 (2003), S. 163–172.
- [33] Capacitor Team. *Capacitor: Cross-platform Native Runtime for Web Apps*. 2025. URL: <https://capacitorjs.com/docs/> (besucht am 06.02.2025).
- [34] Capacitor Team. *Capacitor Plugins - Capacitor Documentation*. 2025. URL: <https://capacitorjs.com/docs/plugins> (besucht am 06.02.2025).
- [35] Ionic Team. *About Ionic*. 2025. URL: <https://ionic.io/about> (besucht am 06.02.2025).
- [36] TypeScript Team. *TypeScript Handbook - Enums*. 2025. URL: <https://www.typescriptlang.org/docs/handbook/enums.html> (besucht am 09.02.2025).
- [37] TypeScript Team. *TypeScript Tooling in 5 Minutes*. 2025. URL: <https://www.typescriptlang.org/docs/handbook/typescript-tooling-in-5-minutes.html> (besucht am 06.02.2025).
- [38] TypeScript Team. *TypeScript for the New Programmer*. 2025. URL: <https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html> (besucht am 06.02.2025).
- [39] TutorialsPoint. *TypeScript Tutorial*. 2025. URL: <https://www.tutorialspoint.com/typescript/index.htm> (besucht am 09.02.2025).
- [40] W3Schools. *JavaScript Number isNaN() Method*. 2025. URL: https://www.w3schools.com/JSREF/jsref_isnan_number.asp (besucht am 06.02.2025).

Name: Anna Marburger

Matrikelnummer: 1085208

Erklärung

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

Ulm, den 26.02.2025



Anna Marburger