ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Konzeption und Realisierung einer Mobile Crowdsensing Anwendung zur Unterstützung von Schwangeren

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Jochen Gindele
jochen.gindele@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Rüdiger Pryss

2018

Fassung 8. Mai 2018

Satz: PDF-LaTeX 2ε

Kurzfassung

In der heutigen Zeit gehören Smartphones und andere mobile Geräte in großen Teilen der Welt zu den Gegenständen, die alltäglich benutzt werden. Sie werden beruflich und privat zur weltweiten Kommunikation verwendet, sind aber durch die unzähligen verfügbaren mobilen Anwendungen auch zu vielen anderen Zwecken nutzbar. Diese Arbeit beschreibt eine Anwendung die Frauen während ihrer Schwangerschaft begleiten und unterstützen soll. Hierfür wird auf den KINDEX Fragebogen zurückgegriffen, der es ermöglicht, psychosoziale Risikofaktoren für Mutter und Kind zu identifizieren. In dieser Arbeit wird beschrieben, wie der papierbasierte KINDEX Fragebogen als Android App umgesetzt werden kann und dabei verschiedene Nutzergruppen berücksichtigt werden. Dadurch wird es möglich, dass Schwangere, orts- und zeitunabhängig, ein qualifiziertes Feedback und Verhaltensempfehlungen von einem begleitenden Arzt oder einer begleitenden Ärztin erhalten, um ihre Schwangerschaft positiv zu beeinflussen. Die Nutzung einer solchen Anwendung kann zu einem effizienteren Ressourceneinsatz bei der Betreuung von Schwangeren führen und dadurch die möglichen Stressfaktoren während einer Schwangerschaft reduzieren.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich unterstützt und zum Gelingen dieser Bachelorarbeit beigetragen haben.

Ich danke Prof. Dr. Manfred Reichert, dem Direktor des Instituts für Datenbanken und Informationssysteme, für die Möglichkeit, an diesem Institut meine Abschlussarbeit anfertigen zu dürfen.

Ein besonders herzlicher Dank gilt meinem Betreuer Dr. Rüdiger Pryss, der mich während der gesamten Dauer, auch in schwierigen Zeiten, mit unterstützenden Worten und hilfreichen Hinweisen begleitet hat.

Für die Bereitstellung der zuverlässigen REST-API und für den nützlichen Input bei der Nutzung der API möchte ich mich ganz besonders bei Johannes Schobel bedanken.

Ich bedanke mich auch bei meinen Kommilitonen für eine gute Zeit und viele fruchtbare Diskussionen, deren Resultate sich sicherlich auch in dieser Arbeit wiederfinden.

Abschließend danke ich meinen Eltern, Alois und Brigitte, und meiner Frau Natallia, die mich während des gesamten Studiums unterstützt und mir stets mit motivierenden Worten zur Seite gestanden haben.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Problemstellung 1

1.2 Zielsetzung 2

1.3 Struktur der Arbeit 2

2 Grundlagen **5**

2.1 Mobile Crowdsensing 5

2.2 KINDEX Fragebogen 7

2.3 Verschiedene Ansätze zur Entwicklung mobiler Anwendungen 9

3 Verwandte Arbeiten **13**

3.1 QuestionSys 13

3.2 TrackYourTinnitus 14

3.3 KINDEX App 15

3.4 myKind 16

3.5 SAP Heidelberg Uniklinik App 16

4 Anforderungsanalyse **17**

4.1 Nutzungsvoraussetzungen 17

4.1.1 Registrierung 17

4.1.2 Rollenkonzept 18

4.1.3 Onlinezwang 18

4.1.4 Tutorialscreen 18

4.2 Rolle: Patientin 18

4.2.1 Studienteilnahme 18

4.2.2 Studie verlassen 19

4.2.3 Startseite 19

4.2.4 Fragebogen 19

4.2.5 Modale Fragen 20

4.2.6 Fortschrittsanzeige 20

4.2.7 Fragebogenabbruch 20
4.2.8 Auf Feedback warten 20
4.2.9 Feedback einsehen 21
4.3 Rolle: Arzt 21
4.3.1 Patientinnenansicht 21
4.3.2 Patientinnendetails 21
4.3.3 Fragebogendetails 22
4.3.4 Feedbackfreigabe 22
4.4 Rollenunabhängig 22
4.4.1 Login 22
4.4.2 Passwort ändern 22
4.4.3 Entwicklerkontakt 23
4.4.4 Mehrsprachigkeit 23
4.4.5 Impressum 23

5 Verwendete Technologien und Architektur 25
5.1 Genereller Ablauf 25
5.2 Android Studio 27
5.2.1 Projektstruktur 27
5.2.2 Gradle 27
5.2.3 Manifest Datei 29
5.3 Activities und Intents 30
5.4 Retrofit 35
5.5 JSON Dateiformat 38
5.6 JSON REST API 39
5.7 Generics 41
5.8 Datenpersistierung 41

6 Implementierung 43
6.1 Implementierung ausgewählter Komponenten 43
6.1.1 Singleton Pattern der User Klasse 43
6.1.2 List Adapter 45

6.1.3 Kommunikation mit dem Backend Server 49

7 Vorstellung der Anwendung **51**

7.1 Tutorialscreen 51
7.2 Onlinezwang 51
7.3 Registrierung 52
7.4 Verifizierungslink 54
7.5 Einloggen 54
7.6 Passwort zurücksetzen 56
7.7 An Studie teilnehmen 56
7.8 Fragebogen starten 58
7.9 Feedback ansehen 59
7.10 Patientinnendetails anzeigen 60
7.11 Feedback freigeben 61
7.12 Passwort ändern 62
7.13 Entwickler kontaktieren 63
7.14 Informationen anzeigen 64
7.15 Sprache einstellen 65

8 Anforderungsabgleich **67**

9 Zusammenfassung **69**

9.1 Ausblick 70

1

Einleitung

In zahlreichen Studien konnte nachgewiesen werden, dass psychosoziale Belastungen einer werdenden Mutter auch schon vor der Geburt einen großen Einfluss auf die psychische und körperliche Gesundheit des ungeborenen Kindes und dessen Entwicklung haben können. Um das Wohlbefinden von Mutter und Kind zu schützen, ist es deshalb notwendig, ergänzend zu bereits bestehenden medizinisch-körperlichen Vorsorgeuntersuchungen, psychosoziale Belastungen einer werdenden Mutter in die Untersuchung einzubeziehen. Ein Instrument, um die psychische Belastung der Mutter zu messen, ist der Konstanzer Index, KINDEX, welcher von Psychologen der Universität Konstanz entwickelt wurde. Der KINDEX wurde ursprünglich als papierbasierter Fragebogen entworfen, der in einer Interviewsituation mit Schwangeren von Frauenärztinnen[1] oder Hebammen durchgeführt werden soll, um psychosoziale Risikofaktoren zu identifizieren [1]. Diese Art von Screening wurde in Gynäkologiepraxen und bei Hebammen unter hohem Einsatz von Ressourcen, Zeit und Personal, durchgeführt und von Psychologinnenen ausgewertet. Um den Ressourceneinsatz bei der Durchführung und Auswertung zu reduzieren wurden verschiedene mobile Applikationen entwickelt, die den KINDEX Fragebogen implementieren.

1.1 Problemstellung

In den vergangenen Jahren wurden bereits zahlreiche mobile Anwendungen zur Unterstützung von Schwangeren mit Hilfe des KINDEX Fragebogens entwickelt. Mit der

[1] Aus Gründen der besseren Lesbarkeit wird auf eine geschlechtsspezifische Berufsbezeichnung verzichtet und stets die weibliche Form verwendet

Umsetzung der KINDEX Mum Screening App konnte nachgewiesen werden, dass die Akzeptanz der Datenerfassung auf Tabletgeräten ausreichend gut ist [1]. Es soll nun eine Mobile Crowdsensing Anwendung entwickelt werden, die durch ein integriertes Rollenkonzept sowohl von Patientinnen als auch von begleitenden Ärztinnen verwendet werden kann. Die Anwendung soll es Benutzerinnen ermöglichen, von jedem beliebigen Ort und zu jeder Zeit, mit ihrem Smartphone den KINDEX Fragebogen auszufüllen und von einer begleitenden Ärztin evaluieren zu lassen.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Erstellung eines Konzepts und die Realisierung einer mobilen Anwendung für die unterstützende Begleitung von Frauen während ihrer Schwangerschaft. Die mobile Anwendung soll dabei den Schwangeren helfen, psychosoziale Risiken zu identifizieren und eine gesunde Schwangerschaft zu unterstützen. Dabei soll die Anwendung von Patientinnen zur Beantwortung von Fragen und von Ärztinnen zum Generieren von Feedback verwendet werden können. Um eine zeitliche wie räumliche Unabhängigkeit zwischen dem Ausfüllen eines Fragebogens durch eine Patientin und der Evaluation durch eine begleitende Ärztin zu ermöglichen muss auf die Kommunikation mit einem Server zurückgegriffen werden, der einen ausgefüllten Fragebogen nach zuvor festgelegten Regeln automatisch auswertet. Es ist allerdings erforderlich, dass das Feedback, das auf die Abgabe der beantworteten Fragen folgt, durch eine begleitende Ärztin abschließend freigegeben werden muss. Dafür ist es notwendig, das automatisch generierte Feedback, zuerst an die mobile Anwendung der begleitenden Ärztin zu schicken.

1.3 Struktur der Arbeit

Zunächst werden Grundlagen für das Verständnis der vorliegenden Arbeit geklärt. Anschließend folgt ein Überblick über Arbeiten, die sich bereits in der Vergangenheit mit ähnlichen Themen beschäftigt haben. In Kapitel 4 folgt eine Anforderungsanalyse, die

nach Anwendungsbereichen aufgeteilt ist. Danach wird die Architektur der mobilen Anwendung diskutiert und verschiedene Teilaspekte der verwendeten Technologien beschrieben. Im Anschluss daran wird die Implementierung vorgestellt und es wird auch auf einzelne Komponenten eingegangen. Es folgt eine Vorstellung der Funktionen der Anwendung. Im letzten Kapitel soll eine Zusammenfassung und ein Ausblick auf zukünftige Entwicklungsmöglichkeiten diese Arbeit abschließen.

2

Grundlagen

In diesem Kapitel werden Grundlagen erläutert, die das Verständnis des vorliegenden Projektes erleichtern sollen. Zunächst wird auf das Thema Mobile Crowdsensing eingegangen und anschließend wird die Entwicklung des KINDEX Fragebogens dargelegt. Außerdem werden verschiedene Ansätze der Entwicklung von mobilen Anwendungen vorgestellt.

2.1 Mobile Crowdsensing

Aufgrund der zunehmenden Verbreitung von Smartphones, Smartwatches, anderen Smart Devices, und dem Internet of Things konnte ein neues Paradigma zur Datenerfassung und -verarbeitung entstehen, Mobile Crowdsensing. Durch die Vielzahl an Sensoren, die in der heutigen Zeit von den meisten Menschen getragen werden, ist es möglich auf eine große Datenmenge zuzugreifen um durch deren Analyse neue Erkenntnisse zu verschiedenen Nutzerverhalten zu gewinnen. Dies ist in vielen Bereichen der Wissenschaft und der Wirtschaft von großer Bedeutung [2]. Zu der großen Menge an Sensoren, die die erwähnten Geräte produktionsbedingt beinhalten zählen Kameras, Mikrofone, GPS Sender und Bewegungssensoren. Man kann sich aber auch vorstellen, Sensoren zur Messung der Luftqualität oder ähnliches in Mobile Crowdsensing Anwendungen zu integrieren. Die Einsatzgebiete für Anwendungen, welche Mobile Crowdsensing verwenden, sind vielfältig - zur Umweltüberwachung, Transport und Verkehrsplanung, Gesundheitskontrolle und um Aussagen über die öffentliche Sicherheit treffen zu können [3]. So können beispielsweise GPS Sensoren von Mobiltelefonen oder im Fahrzeug eingebaute Sensoren verwendet werden, um die Verkehrslage auf

den Straßen zu überwachen und alternative Routen zu berechnen. Außerdem können Mobile Crowdsensing Anwendungen durch ihre variablen Einsatzmöglichkeiten neue Erkenntnisse für die Erforschung von Krankheiten liefern [4]. Dies führt dazu, dass mit Mobile Crowdsensing Anwendungen akkuratere Diagnosen und dadurch bessere Behandlungen von Patienten erreicht werden können [5]. Anwendungen können auf verschiedene Arten klassifiziert werden. Nach der Art der Teilnahme des Nutzers an der Datenerfassung werden Applikationen in Participatory Sensing und Opportunistic Sensing eingeteilt [6].
Beim Participatory Sensing wird eine User Interaktion gefordert, wie ein Bild mit der Kamera zu machen oder einfach Daten durch einen Klick zu senden. Es stellt also eine bewusste Teilnahme des Benutzers an der Datenerfassung dar. Im Gegenteil dazu fordert das Opportunistic Sensing keine explizite Interaktion mit dem Benutzer. Vielmehr werden dabei im Hintergrund, Daten wie eine Standortposition oder Hintergrundgeräusche an den Dienst gesendet, der die Daten zur Weiterverarbeitung benötigt. Eine andere Art der Unterteilung kann anhand vom Umfang der Personen gemacht werden, die sich an der Datenerfassung für die Anwendung beteiligen. Single bzw. Personal Sensing Anwendungen kommen häufig im Gesundheitssektor vor und bezeichnen die Datenerfassung und Auswertung für einen kleinen Benutzerkreis, oft nur den betroffenen einzigen Nutzer und möglicherweise einen begleitenden Experten.

Personal Sensing Anwendungen sind für eine einzelne Person als Nutzer konzipiert [7]. Diese Applikationen setzen meist den Fokus auf die Datenerfassung und die Auswertung der erfassten Daten. Typische Beispiele sind Fitnesstracker, die über die Aufzeichnung von Bewegungsdaten und der Herzfrequenz, die Möglichkeit bieten, den individuellen Fitnesszustand eines Benutzers zu analysieren. Normalerweise erzeugen Personal Sensing Anwendungen nur Daten, die auf Informationsgewinnung und -verarbeitung durch den Nutzer abzielen. Ein gemeinschaftliches Teilen der Daten ist bei Personal Sensing Applikationen nicht vorgesehen. Eine Ausnahme bilden Anwendungen, die ein Überwachen des Gesundheitszustands eines Verbrauchers ermöglichen. Hierbei ist es notwendig, wie auch in der vorliegenden KINDEX Anwendung, dass medizinisches Personal Zugriff auf die gesammelten Daten erhält. In der vorliegenden KINDEX Anwendung trifft dies auf die Ärztin zu, der die Patientin während Ihrer Schwangerschaft und

Studienteilnahme begleitet. Diese Art der Mobile Crowdsensing Anwendung ermöglicht eine sehr individuelle Betreuung der Nutzerin beim Erreichen ihrer gesetzten gesundheitlichen Ziele. Im Gegensatz zum Personal Sensing erfassen Community Sensing Anwendungen Daten, deren Auswertung für eine größeren Kreis von Personen von Relevanz sind und nicht von einem einzelnen Individuum erfasst werden können. Hierzu gehört, dass sich eine Gruppe von Benutzern dazu bereit erklärt, Daten mit ihren eigenen Sensoren zu erfassen und mit der Gemeinschaft an Teilnehmern zu teilen. Das bereits erwähnte Überwachen von Verkehrsflüssen gehört zu einem solchen Phänomen. Die meisten Anwendungen, die auf Daten basieren, die durch Community Sensing erfasst werden, benötigen eine große Anzahl an Teilnehmern um nützliche Rückschlüsse aus der Auswertung zu ermöglichen. Außerdem ist ein gewisses Vertrauen in die geteilten Daten der Teilnehmer notwendig.

2.2 KINDEX Fragebogen

Um die Gesundheit von Schwangeren und ihrem ungeborenen Kind vorsorglich sicherzustellen, wurden eine Vielzahl von Untersuchungen entwickelt, die auch in Deutschland angewendet werden. Hierzu zählen unter anderem Ultraschalluntersuchungen, Blutanalysen oder Fruchtwasseruntersuchungen. Allerdings ist einschränkend zu bemerken, dass alle Vorsorgeuntersuchungen darauf abzielen, die Risiken für die körperliche Gesundheit von Schwangeren und Kindern zu identifizieren und zu minimieren. Neuere Studien haben aber gezeigt, dass während einer Schwangerschaft ebenso psychosoziale Faktoren existieren, die die seelische Gesundheit von Schwangeren und Kind potenziell gefährden und einer gesunden Entwicklung von Mutter und Kind entgegenstehen können. Bis zur Entwicklung des Konstanzer Index (KINDEX) war kein wissenschaftlich validiertes Screening Instrument in Deutschland bekannt, mit dem es möglich ist, bereits vor der Geburt psychosoziale Belastungsfaktoren zu erkennen, die zu einer geschädigten Mutter-Kind-Beziehung oder anderen postnatalen emotionalen, sozialen und neuropsychologischen Fehlentwicklungen des Kindes führen können. Der KINDEX wurde auf Basis von Literaturrecherchen und Ergebnissen von bereits in anderen Ländern bestehenden pränatalen Screening Instrumenten entworfen. Ursprünglich wurde für die Ermittlung des KINDEX

ein vollstandardisierter, papierbasierter Fragebogen in Interviewform eingesetzt. Diese Interviews wurden von Gynäkologinnen, Hebammen und weiteren Personen durchgeführt, die die Patientin während der Schwangerschaft begleiten. Der KINDEX erfasst eine große Anzahl an pränatalen Belastungsfaktoren, als Beispiele seien genannt: ein junges Alter der Mutter, familiäre und/oder finanzielle Belastungen oder traumatische Erfahrungen der Mutter. Um den KINDEX empirisch zu validieren wurden Interviews in Deutschland, Griechenland, Spanien und Peru durchgeführt [8]. Dabei wurden auch Einschränkungen und Probleme bei der Durchführung der Interviews und Evaluation des KINDEX beobachtet. Um eine aussagekräftige Anwendung des Screening Instruments zu gewährleisten, ist es notwendig, dass sich das durchführende Personal die Zeit nimmt, um sich das notwendige Wissen zur Durchführung und Auswertung des KINDEX anzueignen. Außerdem muss in der alltäglichen Praxis Zeit aufgebracht werden, die papierbasierten Interviews durchzuführen und anschließend korrekt auszuwerten. Diese Schwierigkeiten führten dazu, dass ein Instrument entwickelt werden sollte, das es Schwangeren ermöglicht selbstbestimmt einen Test durchführen zu können und nur für die Auswertung auf eine begleitende Ärztin oder Hebamme zurückzugreifen. Dafür wurde eine Tablet gestützte Version des KINDEX Fragebogens entwickelt. Auf diese Weise können Schwangere selbstständig, ohne medizinische Begleitung, die standardisierten Fragen des KINDEX beantworten, während des Wartens bei einem Vorsorgetermin in einer Klinik oder gynäkologischen Praxis. Die Tablet gestützte Version ermöglicht das automatische Auswerten und gibt dem begleitenden medizinischen Personal Hinweise auf mögliche psychosoziale Risiken der Mutter, die im persönlichen Gespräch erklärt und durch Verhaltensempfehlungen verringert werden können. Diese Art des KINDEX ermöglichte eine Schonung der Ressourcen des medizinischen Personals, allerdings muss der Fragebogen dennoch örtlich eingeschränkt, in einer gynäkologischen Klinik oder Praxis genutzt werden. Um es einer Schwangeren zu ermöglichen, orts- und zeitunabhängig einen KINDEX Fragebogen auszufüllen sollte eine Weiterentwicklung der Tablet basierten App entwickelt werden. Es soll eine Anwendung entwickelt werden, die auf dem Smartphone der Schwangeren installiert wird und mit der sie zu jedem Zeitpunkt die Fragen des KINDEX ausfüllen und an ihre begleitende Ärztin schicken kann. Diese ist in der Lage durch die automatische Auswertung und Feedbackgenerierung, nach einer

Überprüfung das Feedback freizugeben und an die Schwangere zu schicken. Mit dieser Weiterentwicklung soll die Schwangere Feedback zu den gegebenen Antworten des KINDEX Fragebogens erhalten, auch ohne die Praxis der begleitenden Ärztin aufsuchen zu müssen, was in einer enormen Ressourcenschonung resultiert.

2.3 Verschiedene Ansätze zur Entwicklung mobiler Anwendungen

Dieser Abschnitt behandelt verschiedene Ansätze der Entwicklung von mobilen Anwendungen. Zuerst werden unterschiedliche Vorgehensweisen vorgestellt und deren Vor- und Nachteile herausgearbeitet. Anschließend werden die Entwicklungsansätze miteinander verglichen und die Wahl für einen bestimmten Ansatz für das vorliegende Projekt begründet.

Native Anwendungen

Native Anwendungen werden für eine bestimmte und vorher definierte Plattform entwickelt. Die Plattform muss vorher festgelegt sein, da man bei der nativen App-Entwicklung auf Werkzeuge und Eigenschaften der zugrundeliegenden Plattform zurückgreift, um die bereitgestellten Bibliotheken und Schnittstellen der Plattform bestmöglich ausnutzen zu können. Hierzu gehören auch Zugriffe auf die Hardware. Dies führt dazu, dass bei der nativen Anwendungsentwicklung in einer bestimmten Programmiersprache implementiert werden muss, die von den gewählten Plattformen abhängig ist. Für die Android Entwicklung stehen die Programmiersprachen Java und Kotlin zur Auswahl. Objective-C und Swift können für Apples iOS verwendet werden. Eine native Anwendung erfordert ein umfangreiches plattformspezifisches Wissen des Entwicklers um die Performance der Anwendung zu optimieren und eine zufriedenstellende Nutzererfahrung zu gewährleisten. Allerdings bietet eine native App die größtmögliche Funktionalität unter Ausnutzung der plattformspezifischen Schnittstellen. Ein Ansatz, der auch der nativen Anwendungsentwicklung zugeteilt werden kann, ist das sogenannte „Cross Compiling“. Hierbei wird eine Anwendung auch plattformspezifisch entwickelt, allerdings vor dem Hintergrund, dass die fertige Anwendung auf unterschiedlichen Plattformen

verwendet werden soll. Die Idee basiert darauf, dass sich unterschiedliche Plattformen einen großen Teil des Programmcodes teilen können, während sich bestimmte Bereiche einer App, zum Beispiel die grafische Oberfläche und Bedienelemente, von Plattform zu Plattform zu sehr unterscheiden. Dies wird erreicht, indem der Quellcode in einer Programmiersprache geschrieben wird und anschließend mit Hilfe eines Cross-Compiling-Frameworks und den plattformspezifischen Compilern in den ausführbaren Code der Zielplattform übersetzt wird. Da sich die unterschiedlichen Plattformen allerdings grundlegend unterschieden können, führt dieser Ansatz oft zu umfangreichen Anpassungen, so dass eine vollständige Generalisierung nicht erreicht werden kann.

Webanwendungen

Mobile Web Anwendungen (Web Apps) werden plattformunabhängig implementiert. Web Apps werden im Webbrowser ausgeführt und könne dadurch sehr eingeschränkt auf plattformspezifische Funktionen zugreifen. Als Technologien stehen etablierte Webstandards wie HTML5 oder JavaScript zur Verfügung. Web Apps haben keinen Hardwarezugriff und somit ist es mit diesem Ansatz nicht möglich Anwendungen zu erstellen, die zum Beispiel Sensoren oder andere Daten verwenden sollen. Vorteilhaft ist der, im Vergleich zur nativen Anwendung geringere Implementierungsaufwand, da man nur einen relativ geringen Umfang von Technologien beherrschen muss. Dies führt dazu, das die Funktionalität einer Web App weit von den Möglichkeiten einer nativ programmierten Anwendung entfernt ist.

Hybride Anwendungen

Hybride Anwendungen stellen eine Kombination aus nativer und Web Anwendung dar. Auch hierbei werden die etablierten Webtechnologien zur Implementierung verwendet. Dadurch kann die Benutzeroberfläche in einem Container ausgeführt werden, der Webinhalte darstellen kann, zum Beispiel der native Webbrowser einer Plattform. Allerdings können, unter Verwendung von Cross-Platform-Frameworks, die Bedienelemente des Browsers unterdrückt werden, was zu einer anderen Benutzererfahrung führt, als bei einer Web Anwendung. Cross-Platform-Frameworks bieten durch diverse Bibliotheken einen Funktionsumfang, der weit über die Möglichkeiten einer mobilen Webanwendung hinausgeht, da sie den Zugriff auf die Hardware ermöglichen können und dadurch den

Einsatz von Sensordaten erlauben können. Dennoch ist der Zugriff auf Systemschnittstellen nicht so weitreichend wie bei einer nativen mobilen Anwendung.

Um die bestmögliche Nutzererfahrung zu gewährleisten, basiert das vorliegende Projekt auf dem Ansatz der nativen Entwicklung von mobilen Anwendungen. So können plattformspezifische Bedienelemente verwendet werden und auf Hardwarekomponenten zugegriffen werden. Da es nicht erforderlich ist, dass die Anwendung für unterschiedliche Plattformen entwickelt werden soll, kann auf ein Cross-Compiling-Framework verzichtet werden.

Tabelle 2.1: Verschiedene Ansätze mobiler App Entwicklung nach [9]

	Nativ	**Cross Compiling**	**Hybrid**	**Web**
Beschreibung	Anwendungen werden in der jeweiligen plattformspezifischen Sprache entwickelt. Direkter Zugriff auf die APIs der entsprechenden Plattform	Die Anwendungen werden mittels einer dedizierten Programmiersprache entwickelt und anschließend durch die Compiler in die Zielplattform übersetzt	Die Anwendung wird mit Webtechnologien entwickelt und in einem nativen Container ausgeführt. Dieser erlaubt den eingeschränkten Zugriff auf Gerätehardware	Die Anwendung wird mit etablierten Webtechnologien umgesetzt. Diese wird direkt über den Webbrowser des Geräts aufgerufen und dargestellt
Technologie	Plattformspezifische Programmiersprachen (z.B. Java, ObjectiveC, C#)	Beispielsweise JavaScript oder C# bei Xamarin	JavaScript,HTML5, CSS3	JavaScript,HTML5, CSS3
Vorteile	Uneingeschränkter Zugriff auf die Systm-API & Hardware; beste Performance und User Experience	Gleiche Codebasis trotz unterschiedlicher Zielplattformen; verwendet Bedienelemente der Zielplattform	Gleiche Codebasis trotz unterschiedlicher Zielplattformen; einfache Entwicklung durch Webtechnologien	Einfache Entwicklung durch Webtechnologien; einheitlicher Updateprozess der Anwendung
Nachteile	Aufwendige Entwicklung; jede plattformspezifische Anwendung muss von Experten entwickelt werden	Viele plattformspezifische Anpassungen am Code nötig, da Generalisierung oft nicht möglich	Kein plattformspezifisches User Interface; eingeschränkter Zugriff auf Hardware und relativ schlechte Performance	Kein plattformspezifisches User Interface; sehr eingeschränkter Hardwarezugriff
Frameworks	Android, iOS, Windows Phone	Xamarin	PhoneGap	Bootstrap

3

Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten und Projekte vorgestellt, deren Ergebnisse in bestimmter Weise in die vorliegende Arbeit eingeflossen sind und auf die diese Arbeit aufbaut.

3.1 QuestionSys

Das erste Projekt, das im Zusammenhang mit der vorliegenden Arbeit erwähnt werden muss, ist das von der Universität Ulm initiierte QuestionSys-Projekt. Hierbei handelt es sich um ein Framework, womit Fragebögen auf einfache Weise ohne spezielle Programmierkenntnisse erstellt werden können, die anschließend auf mobilen Endgeräten zur Nutzung zur Verfügung stehen. Das Projekt ist darauf ausgelegt, dass Experten aus dem Themengebiet des gewünschten Fragebogens die Erstellung, Datensammlung und Auswertung hochautomatisiert (prozessgetrieben) durchführen können [10]. Das Projekt besteht aus drei Teilen: dem Fragebogen Konfigurator, der mobilen Anwendung und einer middleware, die für den sicheren Datenaustausch und eine sichere Datenhaltung verantwortlich ist [11].

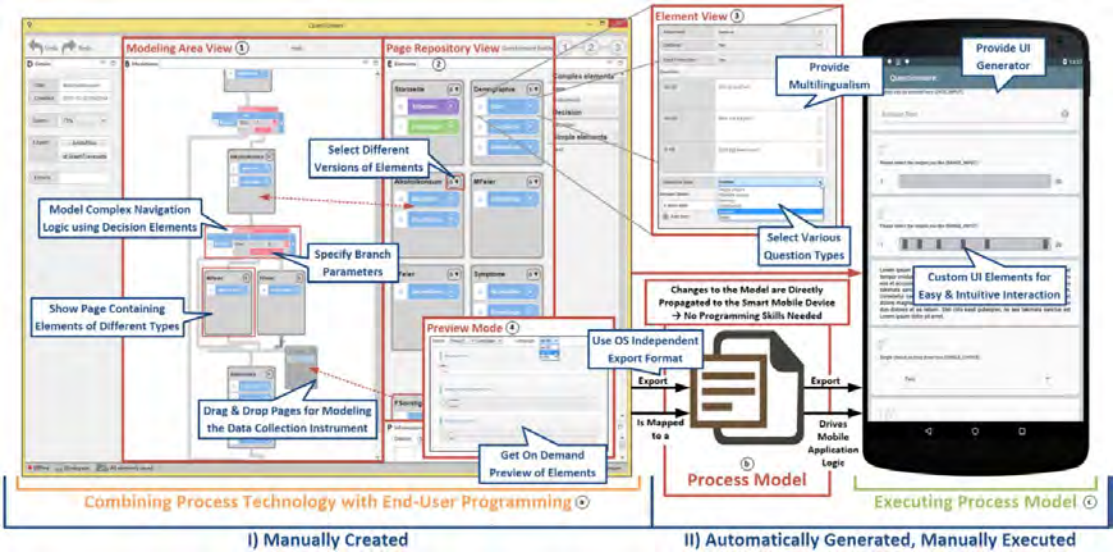

Abbildung 3.1: QuestionSys Übersicht [11]

3.2 TrackYourTinnitus

TrackYourTinnitus ist ein Forschungsprojekt der Universität Ulm [12]. Es hat zum Ziel, Tinnitusgeschädigte im Umgang mit ihrer Krankheit zu unterstützen und gleichzeitig die Tinnitusforschung, durch die Versorgung mit gesammelten Daten, voranzutreiben. Hierfür wird die individuelle Empfindung eines Tinnitusgeschädigten mit Hilfe einer Smartphone Anwendung aufgezeichnet. Ein Betroffener kann so zu verschiedenen Zeitpunkten die Schwankungen seiner Wahrnehmung des Tinnitusgeräusches in einem Fragebogen festhalten und den Verlauf seiner Krankheit über einen längeren Zeitraum überwachen. Die TrackYourTinnitus Anwendung wird seit längerer Zeit in der Praxis erfolgreich angewandt, da sie der Tinnitusforschung eine Messmethodik bietet, die ohne den Einsatz von mobilen Geräten nicht realisiert werden konnte [9]. Auf Basis des TrackYourTinnitus Projekts konnte bereits eine Vielzahl von neuen Erkenntnissen gewonnen und zur Tinnitusforschung beigetragen werden [13][14][15]. Das Crowdsensing Projekt TrackYourTinnitus entstand im Rahmen der Diplomarbeit von Jochen Herrmann [16] und wird kontinuierlich weiterentwickelt.

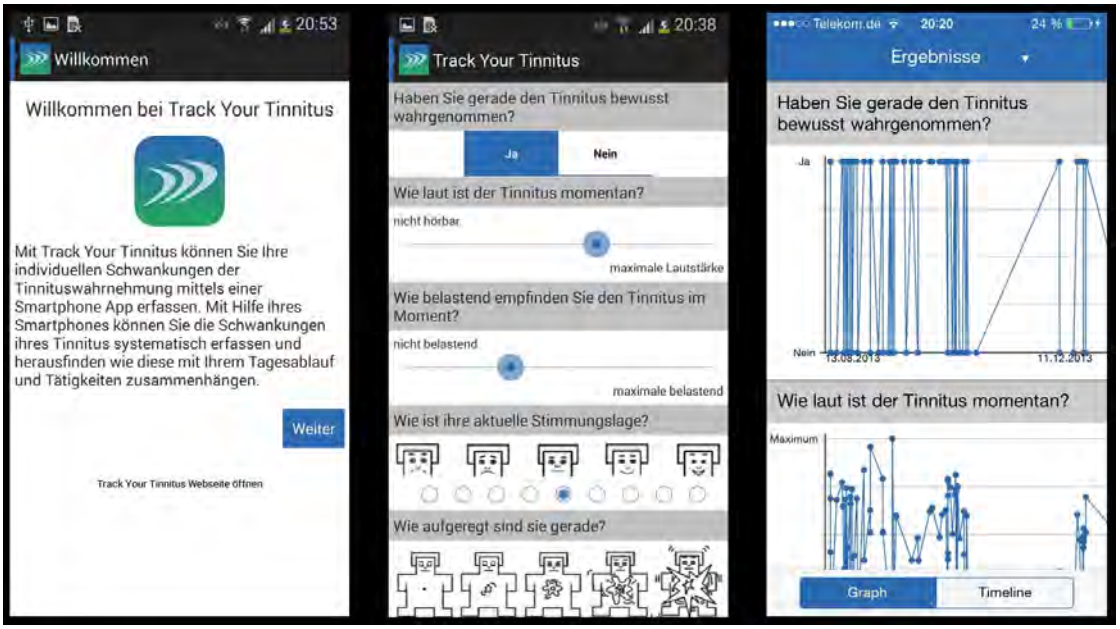

Abbildung 3.2: TrackYourTinnitus Screenshots

3.3 KINDEX App

Die KINDEX App ist eine mobile Anwendung für Tabletcomputer. Sie setzt den papierbasierten KINDEX Fragebogen in digitaler Form um und ermöglicht dadurch eine zeitsparende Auswertung der gegebenen Antworten. Außerdem ermöglicht es die Anwendung, dass Schwangere in einer gynäkologischen Praxis oder bei einer Hebamme selbstständig den Fragebogen ausfüllen können. Durch die unverzügliche automatische Auswertung kann die begleitende Ärztin bzw. Hebamme sofortiges Feedback mit der Schwangeren teilen und gegebenenfalls unterstützende Angebote besprechen. Die KINDEX App wurde auf Validität mit dem papierbasierten Fragebogen getestet und die Akzeptanz der App wurde überprüft und für ausreichend befunden. Die vorliegende Arbeit basiert sehr stark auf dem Projekt der KINDEX App, wobei der Einsatzbereich unterschiedlich ist. Während die KINDEX App für den Einsatz bei einer Ärztin oder Hebamme konzipiert ist, soll die vorliegende Arbeit die Möglichkeit bieten, ortsunabhängig und zeitlich uneingeschränkt, einen Fragebogen auszufüllen [1].

3.4 myKind

Das Projekt "myKind“ mit der dazugehörigen mobilen Anwendung erweitert bzw. verändert das Einsatzgebiet der zuvor genannten KINDEX App. Während es beim Einsatz der KINDEX App notwendig ist, dass sich die Schwangere zur Verwendung der App zu ihrer betreuenden Person begibt, ermöglicht die mykind App den Einsatz auf dem persönlichen Smartphone der Schwangeren und ist somit orts- und zeitunabhängig nutzbar. Allerdings wird bei der "myKind“ App das Ergebnis der Auswertung des Fragebogens, ohne Intervention der betreunden Person, der Schwangeren mitgeteilt. Zusätzlich zur mobilen Anwendung bietet das mykind Projekt die Möglichkeit, über eine Webapplikation den KINDEX Fragebogen auszufüllen und auswerten zu lassen [17].

3.5 SAP Heidelberg Uniklinik App

Die Universitätsklinik in Heidelberg hat in Zusammenarbeit mit SAP und abcmedien eine mobile App entwickelt, die Frauen während ihrer Schwangerschaft begleiten und unterstützen soll. Bei dieser mobilen Anwendung ist das Ziel, Informationen von Schwangeren zu sammeln um gesundheitliche Risiken wie eine Schwangerschaftsdepression frühzeitig zu erkennen und zu verhindern. Die App befragt dabei ihre Nutzerinnen auf eine diskrete Art nach psychologischen Risikofaktoren. Die Antworten werden dem betreuenden Arzt zur Verfügung gestellt. Dadurch können die Antworten analysiert und proaktiv psychologische Risiken identifiziert werden [18].

4

Anforderungsanalyse

Das Ziel dieser Arbeit ist eine funktionsfähige, im Alltag nützliche mobile Applikation zu konzipieren und zu realisieren. Um dies zu erreichen, sind Anforderungen an die Applikation gestellt, die im folgenden Kapitel detailliert besprochen werden. Da es sich bei der Applikation um eine rollenbasierte Anwendung handelt, sind die folgenden Abschnitte zuerst nach Nutzungsvoraussetzungen und anschließend in verschiedene Benutzerrollen gegliedert.

4.1 Nutzungsvoraussetzungen

Vor Nutzung der Funktionen der Applikation sind bestimmte Voraussetzungen zu erfüllen, die im folgenden Abschnitt besprochen werden.

4.1.1 Registrierung

Die Funktionen der Applikation können ohne vorherige Registrierung nicht verwendet werden. Patientinnen müssen sich in der Registrierungsmaske der App registrieren, Ärztinnen können sich nicht selbst registrieren sondern eine Ärztinnenregistrierung wird von der studienbegleitenden Institution durchgeführt. Die nötigen Daten zur Registrierung werden durch das Backendsystem vorgegeben. Eine Verifizierung der E-Mail-Adresse durch eine Verifizierungsmail wird zum Einloggen benötigt. Patientinnen können wiederholt eine Verifizierungsmail anfordern. Für verifizierte Benutzer kann das zugehörige Passwort zurückgesetzt werden.

4.1.2 Rollenkonzept

Die Applikation kann von zwei verschiedenen Benutzerrollen genutzt werden, Ärztinnen und Patientinnen. Patientinnen und Ärztinnen haben getrennte Loginbereiche. Die Funktionen der Applikation sind durch die beiden Rollen komplett getrennt.

4.1.3 Onlinezwang

Die Applikation kann nur mit bestehender Internetverbindung genutzt werden. Da ohne eine aktive Internetverbindung keine Kommunikation zwischen Applikation und Server stattfinden kann, können auch keine Fragebögen zwischen Patientin und Ärztin ausgetauscht werden.

4.1.4 Tutorialscreen

Beim erstmaligen Start der Applikation soll ein Tutorialscreen angezeigt werden, der die Funktionen für die Patientin kurz erklärt. Eine Einführung der Applikation für Ärztinnen ist nicht nötig, da diese von der Universität im persönlichen Gespräch durchgeführt wird.

4.2 Rolle: Patientin

Folgende Funktionen sind nur im eingeloggten Zustand einer Benutzerin mit der Rolle "Patientin“ verfügbar.

4.2.1 Studienteilnahme

Eine eingeloggte Patientin kann ohne vorherige Studienteilnahme keinen Fragebogen ausfüllen. Zuerst muss an einer Studie der behandelnden Ärztin teilgenommen werden. Dies bedeutet, dass das Patientinnen-Konto mit dem Konto ihrer Ärztin verknüpft werden muss. Dafür muss die Patientin die zuvor mit der Ärztin ausgetauschten Teilnahmeinformationen (E-Mail-Adresse der Ärztin und Studienpasswort) eintragen. Die Ärztin

muss dafür schon im System registriert und verifiziert sein, was von der begleitenden Institution übernommen wird.

4.2.2 Studie verlassen

Ist ein Patientinnen-Konto mit dem Konto ihrer behandelnden Ärztin verbunden und nimmt somit an ihrer Studie teil, kann eine Patientin diese Studie auch jederzeit wieder freiwillig verlassen und damit die Studienteilnahme beenden. In diesem Fall werden allerdings keine Daten vom Server gelöscht, sondern nur die temporär gespeicherten Daten auf dem benutzten Endgerät. Als Folge eines Studienabbruchs durch die Patientin kann die behandelnde Ärztin die Fragebögen der Patientin nicht mehr einsehen. Nimmt die Patientin die Teilnahme an einer Studie wieder auf und verbindet ihr Konto mit dem ihrer Ärztin, so können beide, Ärztin und Patientin, vorhandene Fragebögen und Feedbacks wieder einsehen.

4.2.3 Startseite

Auf der Startseite sieht die Patientin eine Übersicht ihrer bereits ausgefüllten Fragebögen. Außerdem sind gesondert auch die Fragebögen sichtbar, die zu einem früheren Zeitpunkt abgebrochen wurden. Feedbacks zu ihren ausgefüllten Fragebögen, die die Patientin noch nicht eingesehen hat, sind entsprechend gekennzeichnet.

4.2.4 Fragebogen

Eine Patientin kann jederzeit und beliebig viele Fragebögen ausfüllen. Nach erfolgtem Ausfüllen von Fragebögen muss aktiv bestätigt werden, dass die Antworten an das Backend übermittelt werden sollen, um dem Arzt die Einsicht des Fragebogens zu ermöglichen.

4.2.5 Modale Fragen

Der KINDEX Fragebogen ist modal aufgebaut. Das bedeutet, dass nicht zwischen den Fragen hin- und hergewechselt werden darf. Es kann immer nur die aktuelle Frage betrachtet und beantwortet werden. Wurde eine Frage beantwortet, kann diese Frage nicht noch einmal aufgerufen werden. Während die Patientin einen Fragebogen ausfüllt, darf keine andere Funktion der Anwendung ausführbar sein.

4.2.6 Fortschrittsanzeige

Eine Fortschrittsanzeige, die einer Patientin anzeigt, welcher Anteil des Fragebogens bereits bearbeitet wurde soll stets sichtbar sein. Während des Ausfüllens eines Fragebogens soll in Form einer Prozentangabe dargestellt werden, wie weit die Patientin fortgeschritten ist.

4.2.7 Fragebogenabbruch

Eine Patientin kann zu jedem Zeitpunkt die Bearbeitung eines Fragebogens abbrechen. Dabei wird der Zustand des Fragebogens lokal gespeichert. Eine Patientin kann dadurch zu einem späteren Zeitpunkt die Bearbeitung eines Fragebogens fortführen. Dies bedeutet, dass bei einer Wiederaufnahme der Bearbeitung, an der letzten nicht beantworteten Frage begonnen wird. Eine Einsicht der bereits beantworteten Fragen ist auch bei Wiederaufnahme nicht möglich. Hat eine Patientin den Fragebogen komplett ausgefüllt, entscheidet sich aber dagegen ihn an den Arzt zu übermitteln, so werden keine Daten gespeichert und der Fragebogen wird komplett verworfen.

4.2.8 Auf Feedback warten

Hat eine Patientin einen Fragebogen an eine Ärztin übermittelt, wird auf dem Startscreen der Patientin angezeigt, dass der übermittelte Fragebogen in Bearbeitung durch die Ärztin ist. Es damit auch nicht möglich, Feedback zu diesem Fragebogen einzusehen. Erst

wenn die Ärztin das Feedback zu einem Fragebogen freigegeben und an die Patientin übermittelt hat, wird dies auf dem Startscreen der Patientin erkenntlich gemacht.

4.2.9 Feedback einsehen

Hat eine Ärztin das Feedback zu einem Fragebogen an die Patientin übermittelt, so kann die Patientin das freigegebene Feedback einsehen. In der Feedbackansicht wird das Ausfülldatum sowie Informationen zur Kritikalität der beantworteten Fragen angezeigt. Für jede als kritisch eingestufte Antwort wird der Patientin das Ausmaß der Kritikalität sowie ein Behandlungsvorschlag angeboten.

4.3 Rolle: Arzt

Folgende Funktionen sind nur im eingeloggten Zustand einer Benutzerin mit der Rolle “Arzt“ verfügbar.

4.3.1 Patientinnenansicht

Auf dem Startscreen der Anwendung erhält die Ärztin eine Übersicht aller Patientinnen, die über eine Studie mit dem Konto der Ärztin verbunden sind. Wenn eine Patientin einen Fragebogen ausgefüllt und an die Ärztin übermittelt hat, so wird dies auf dem Startscreen der Ärztin erkennlich gemacht.

4.3.2 Patientinnendetails

Mit der Auswahl einer Patientin von der Patientinnenübersicht, erhält die Ärztin weitere Details zur Patientin. Der volle Name und die E-Mail-Adresse wird angezeigt, sowie eine Liste von ausgefüllten Fragebögen dieser Patientin.

4.3.3 Fragebogendetails

Durch die Auswahl eines Fragebogens gelangt die Ärztin auf die Detailseite des Fragebogens mit Informationen wie dem Ausfülldatum und der Anzahl kritisch beantworteter Fragen. Für jede kritisch beantwortete Frage wird der Grad der Kritikalität sowie ein Behandlungsvorschlag angeboten.

4.3.4 Feedbackfreigabe

Wenn die Ärztin mit allen Behandlungsvorschlägen einverstanden ist, kann sie das Feedback durch eine aktive Bestätigung an die Patientin übermitteln. Versendete Feedbacks können nicht zurückgenommen oder verändert werden.

4.4 Rollenunabhängig

Die folgenden Funktionen sind unabhängig von der jeweiligen Rolle der Benutzerin nutzbar.

4.4.1 Login

Eine Ärztin oder Patientin kann sich mit der bei der Registrierung angegebenen E-Mail-Adresse und Passwort einloggen. Alle nachfolgenden Funktionen sind nur im eingeloggten Zustand nutzbar.

4.4.2 Passwort ändern

Nachdem eine Benutzerin eingeloggt ist, kann sie ihr Passwort ändern. Hierfür muss ein neues Passwort und dessen Wiederholung eingegeben werden.

4.4.3 Entwicklerkontakt

Die Benutzerin kann über ein Dialogfeld die Entwickler (Universität Ulm oder Universität Konstanz) kontaktieren. Hierfür ist es möglich, Probleme oder Fragen zur Anwendung in eine Textbox einzutragen. Die Anwendung sendet daraufhin eine E-Mail mit dem eingegebenen Text als Inhalt an die Entwickler der Universität Ulm oder Universität Konstanz als Empfänger. Die Anwendung sollte für diesen Vorgang nicht verlassen werden, sondern als E-Mail-Service sollte der Standard der Plattform verwendet werden.

4.4.4 Mehrsprachigkeit

Die Sprache der Anwendung ist über ein Menü einstellbar. Verfügbare Sprachen werden vom Backend geladen. Hierbei ist zu beachten, dass die Fragebögen und die Feedbacks in allen Sprachen angezeigt werden können, die das Backend anbietet. Ist eine vom Backend angebotene Sprache allerdings nicht als Anwendungssprache verfügbar, wird als Standardsprache Englisch verwendet. Zur Verdeutlichung folgt ein Beispiel: Das Backend bietet für die Fragebögen und Feedbacks die Sprachen Deutsch, Englisch, Französisch an. Als Anwendungssprachen stehen aber nur Deutsch und Englisch zur Verfügung. Wählt eine Benutzerin nun Französisch als Sprache aus, werden Fragebogen und Feedbacks auf Französisch angezeigt, Die Texte der Anwendung werden aber in Englisch dargestellt. Die Anwendung wird dann in der gewählten Sprache dargestellt und nicht in der Systemsprache, die zu Beginn als Standardsprache gewählt ist. Die Sprache sollte zur Laufzeit geändert werden, ohne dass ein Neustart der Anwendung nötig ist. Die ausgewählte Sprache bleibt permanent.

4.4.5 Impressum

Das Impressum inklusive Links zu allen verwendeten Ressourcen und Bibliotheken ist einsehbar. Darin sind auch Kontaktinformationen zu den beiden beteiligten Universitäten und den Entwicklern hinterlegt.

Tabelle 4.1: Übersicht der Anforderungen

Label	**Titel**
AF1	Registrierung
AF2	Rollenkonzept
AF3	Onlinezwang
AF4	Tutorialscreen
AF5	Studienteilnahme
AF6	Studie verlassen
AF7	Startseite
AF8	Fragebogen
AF9	Modale Fragen
AF10	Fortschrittsanzeige
AF11	Fragebogenabbruch
AF12	Auf Feedback warten
AF13	Feedback einsehen
AF14	Patientinnenansicht
AF15	Patientinnendetails
AF16	Fragebogendetails
AF17	Feedbackfreigabe
AF18	Login
AF19	Passwort ändern
AF20	Entwicklerkontakt
AF21	Mehrsprachigkeit
AF22	Impressum

5

Verwendete Technologien und Architektur

Dieses Kapitel dient dazu, eine Auswahl an Technologien vorzustellen, die bei der Entwicklung der mobilen Anwendung verwendet wurden. Außerdem werden einzelne Aspekte der Anwendungsarchitektur beschrieben, die bei dem vorliegenden Projekt genutzt wurden. Die Anwendung wurde für Android in der Version 8.0 (API Level 26, siehe Listing 5.1, Zeile 8) als Zielplattform entwickelt. Dennoch ist die App auch auf älteren Versionen bis zur Version 4.0.3 (API Level 15, siehe Listing 5.1, Zeile 7) lauffähig.

5.1 Genereller Ablauf

Abbildung 5.1 zeigt den generellen Ablauf bei Verwendung der App durch eine Patientin. Nach dem Öffnen der Anwendung kann sich eine registrierte Nutzerin einloggen. Eine Nutzerin, die noch nicht registriert ist, muss zuerst eine Registrierung durchführen. Dafür werden Anmeldedaten eingegeben und an den Server gesendet. Im Anschluss erhält die Nutzerin einen Verifizierungslink, der an ihre E-Mail-Adresse geschickt wird. Eine registrierte Nutzerin loggt sich über den Startbildschirm ein und erhält Zugriff auf die Haupfunktionen der App (siehe Kapitel 7).

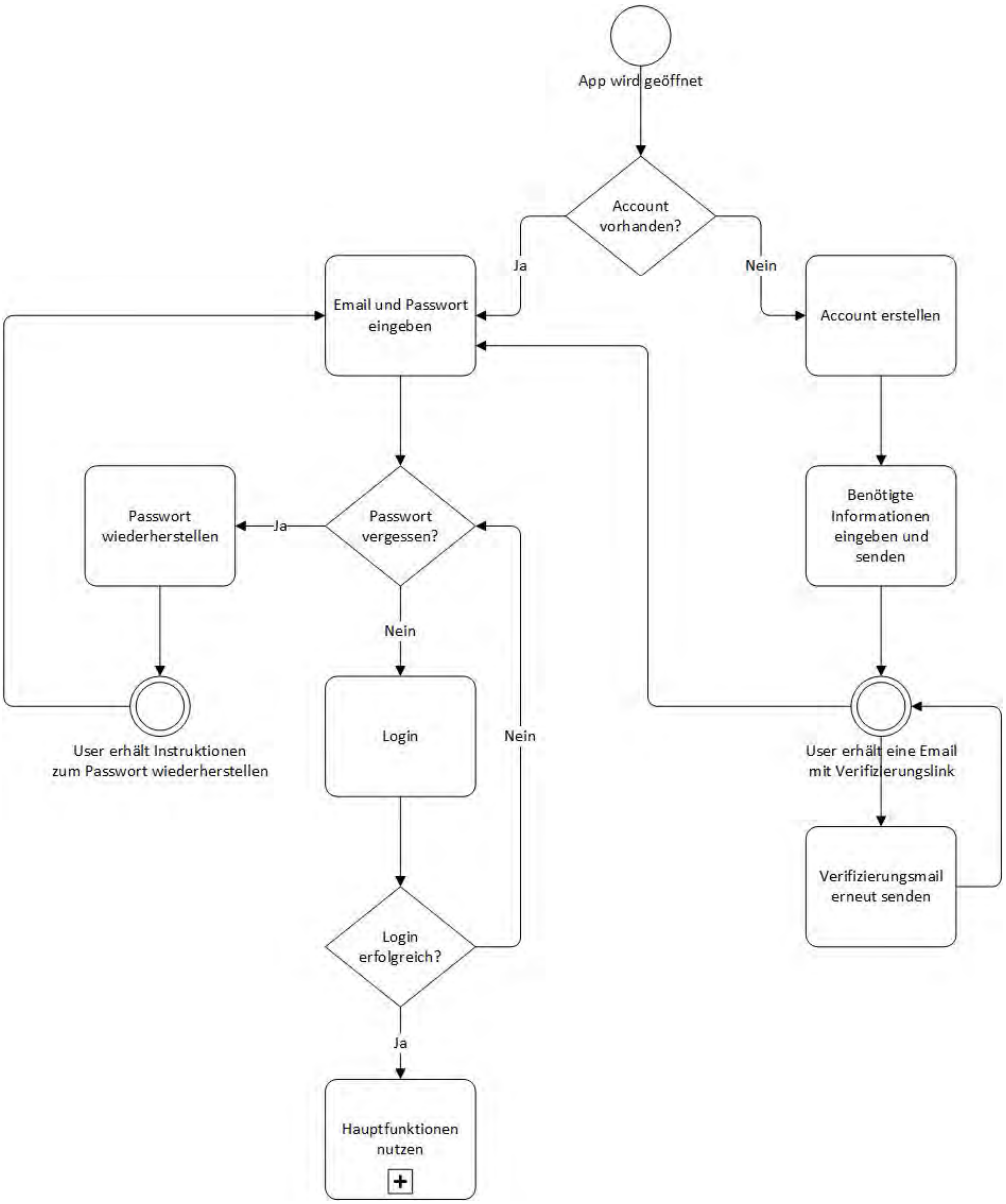

Abbildung 5.1: Genereller Ablauf

5.2 Android Studio

Google bietet für die Anwendungsentwicklung für Android eine eigene integrierte Entwicklungsumgebung, die auf der Intellij IDEA Community Edition von JetBrains basiert [19]. Da Android Studio für die Entwicklung von Android Anwendungen konzipiert wurde, bietet es eine Vielzahl von nützlichen Features, die die Erstellung einer mobilen Anwendung für die Android Plattform vereinfachen.

5.2.1 Projektstruktur

Der Dateiaufbau einer Android App wird in Android Studio übersichtlich in einer Baumstruktur dargestellt. Auf diese Weise lässt sich eine einfache Trennung von Programmlogik und benötigter Ressourcen erreichen, was für eine effiziente Entwicklung sehr vorteilhaft ist. Solche Ressourcen umfassen Texte oder Grafikdateien, die auf der Benutzeroberfläche der App dargestellt werden sollen. Die Auslagerung von Texten in eine einzige separate Datei ermöglicht außerdem eine einfache Umsetzung von Mehrsprachigkeit einer Anwendung, da die übersetzten Texte in der `strings.xml`-Datei aus dem Quellcode referenziert werden können. Auch grafische Elemente werden so übersichtlich an einer Stelle in der Projektstruktur organisiert und können einfach referenziert werden.

5.2.2 Gradle

Das Build System von Android Studio basiert auf dem Build-Management Automatisierungstool Gradle [20]. Damit kann der Build Prozess von zentraler Stelle gesteuert und einfach gestaltet werden. Gradle verwendet `build.gradle`-Dateien um eventuelle Abhängigkeiten zu deklarieren und prüft, ob beispielsweise zusätzliche Bibliotheken eingebunden werden müssen. Mit Gradle lassen sich auch verschiedene Versionen einer einzigen App erstellen, die unterschiedliche Eigenschaften und Funktionen haben, obwohl sie im gleichen Android Projekt erstellt wurden und die gleichen Module nutzen. Die `build.gradle`-Dateien benutzen eine Domänenspezifische Sprache und beste-

hen aus reinen Textdateien mit Groovy-Syntax, welche es dem Entwickler erlaubt den Build Prozess eines Projektes anzupassen.

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 26
5      defaultConfig {
6          applicationId "de.uulm.gindele.akindexapp"
7          minSdkVersion 15
8          targetSdkVersion 26
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner
               "android.support.test.runner.AndroidJUnitRunner"
12         vectorDrawables.useSupportLibrary = true
13     }
14     buildTypes {
15         release {
16             minifyEnabled false
17             proguardFiles
                   getDefaultProguardFile('proguard-android.txt'),
                   'proguard-rules.pro'
18         }
19     }
20 }
21
22 dependencies {
23     implementation fileTree(dir: 'libs', include: ['*.jar'])
24     implementation 'com.android.support:appcompat-v7:26.1.0'
25     implementation 'com.android.support:design:26.1.0'
26     implementation
           'com.android.support.constraint:constraint-layout:1.0.2'
27     implementation
           'com.android.support:support-vector-drawable:26.1.0'
28     testImplementation 'junit:junit:4.12'
```

```
29    androidTestImplementation 'com.android.support.test:runner:1.0.1'
30    androidTestImplementation
          'com.android.support.test.espresso:espresso-core:3.0.1'
31    implementation 'com.google.code.gson:gson:2.8.2'
32    implementation 'com.squareup.retrofit2:retrofit:2.3.0'
33    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
34    //http logging interceptors
35    implementation 'com.squareup.okhttp3:logging-interceptor:3.8.0'
36 }
```

Listing 5.1: `build.gradle`-Datei

5.2.3 Manifest Datei

Die `AndroidManifest.xml` Datei ist die zentrale Beschreibungsdatei einer Android App [21]. Hier werden alle Komponenten eines Programms, zum Beispiel vorhandene Activities, eingetragen und Rechte angefordert, die für die Durchführung der Anwendung notwendig sind. Außerdem wird an dieser Stelle festgelegt, mit welcher Komponente die Anwendung beginnt, wenn sie vom Benutzer gestartet wird [22].

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="de.uulm.gindele.akindexapp">
4
5     <uses-permission android:name="android.permission.INTERNET" />
6     <uses-permission
          android:name="android.permission.ACCESS_NETWORK_STATE" />
7
8     <application
9        android:allowBackup="true"
10       android:icon="@mipmap/ic_launcher"
11       android:label="@string/app_name"
12       android:roundIcon="@mipmap/ic_launcher_round"
13       android:supportsRtl="true"
14       android:theme="@style/AppTheme">
```

```
15        <activity
16            android:name=".MainActivity"
17            android:label="@string/app_name">
18            <intent-filter>
19                <action android:name="android.intent.action.MAIN" />
20
21                <category
                      android:name="android.intent.category.LAUNCHER" />
22            </intent-filter>
23        </activity>
24        <activity android:name=".RegistrationActivity" />
25        <activity
26            android:name=".AttendStudyActivity"
27            android:label="@string/title_activity_attend_study"
28            android:theme="@style/AppTheme.NoActionBar" />
29        ...
30    </application>
31
32 </manifest>
```

Listing 5.2: `AndroidManifest.xml`

5.3 Activities und Intents

Android bietet durch den Einsatz von Activities eine einfache Möglichkeit Anwendungen klar zu strukturieren. Activities bestehen normalerweise aus einer Benutzeroberfläche und repräsentieren eine bestimmte Aktion, die der Benutzer ausführen kann, zum Beispiel, sich für eine Anwendung zu registrieren [23]. Eine Android Anwendung kann aus mehreren Activities bestehen oder auch nur aus einer einzigen. Im Regelfall werden mehrere Activities verwendet um dem Benutzer die Verwendung verschiedener Funktionen einer App zu ermöglichen [24]. Hierzu gehört selbstverständlich auch die Navigation von einer Activity zur nächsten, was entweder durch den Programmablauf vorgegeben ist oder durch eine Benutzerinteraktion gesteuert werden kann.

Bei dem vorliegenden Projekt wurden alle Funktinen in unterschiedlichen Activities aufgeteilt, die von der AppCompatActivity erben. Dadurch soll sichergestellt werden, dass alle verwendeten Komponenten auch abwärtskompatibel sind und die Anwendung auch auf älteren Versionen von Android genutzt werden kann.

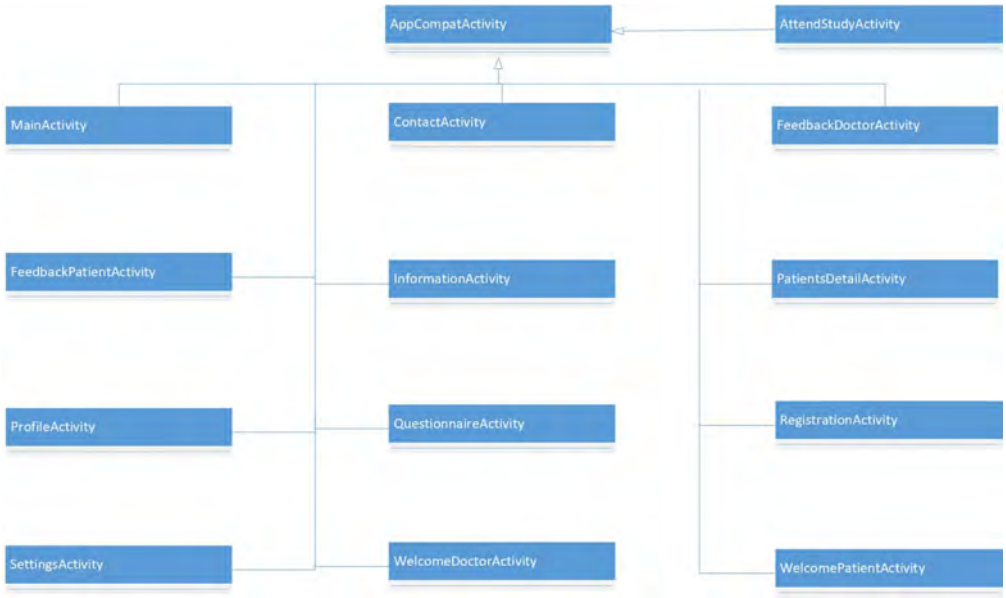

Abbildung 5.2: Vererbung von AppCompatActivity

Activity Lifecycle

Activities nehmen im Laufe ihres Lebenszyklus verschiedene vordefienerte Zustände an [22]. Um mit den Zustandsänderungen umzugehen, bietet Android dem Entwickler verschiedene Callback Methoden zur Implementierung an (siehe Abbildung 5.3), die im Folgenden kurz beschrieben werden [25].

onCreate()

Die onCreate()-Methode muss vom Entwickler implementiert werden, da diese Methode aufgerufen wird, wenn das Android System die Activity erstellt. In der onCreate()-Methode sollten die grundlegenden Komponenten die die Activity benötigt, initialisiert werden. Hierzu gehört beispielsweise das Erstellen der verwendeten View Elemente. Außerdem wird in der onCreate()-Methode das Layout für die Benutzerschnittstelle

geladen, womit der Anwender mit der Activity interagieren kann. Nach Ausführen der onCreate()-Methode wird darauffolgend die onStart()- Methode aufgerufen.

onStart()

Die onStart()-Methode wird aufgerufen, wenn die Activity in den Vordergrund rückt und dem Benutzer sichtbar wird. In dieser Methode werden die letzten Vorbereitungen getroffen, um die Benutzerinteraktion mit der Activity möglich zu machen.

onResume()

Diese Methode wird vom System aufgerufen, wenn die Activity kurz davor ist mit dem Nutzer zu interagieren. Dies kann auch aus dem „Pausiert“-Zustand aufgerufen werden, wenn die Activity zuvor pausiert wurde. Die Activity befindet sich beim Aufruf der onResume()-Methode als oberstes Element auf dem Activity Stack und nimmt die Eingaben des Benutzers auf. Jetzt wird die Activity ausgeführt.

onPause()

Wenn während der Ausführung einer Activity eine andere Activity in den Vordergrund rückt, wird die onPause()-Methode aufgerufen. Dies kann auf verschiedene Arten geschehen. Wenn der Benutzer beispielsweise den „Zurück“-Button betätigt, wird die Activity in den „Pausiert“-Zustand versetzt und verliert den Fokus der Anwendung. Dies gilt als Indikator, dass der Nutzer die aktuelle Activity verlassen möchte und die Activity wahrscheinlich in den „Gestoppt“-Zustand wechselt. Wenn die onPause()-Methode ausgeführt wurde gibt es zwei unterschiedliche Callback Methoden, die ausgeführt werden können, onStop() oder onResume(). Die Ausführung ist davon abhängig, welcher Zustand nach dem „Pausiert“-Zustand erreicht werden soll.

onStop()

Die onStop()-Methode wird vom System aufgerufen, wenn die Activity nicht mehr für den Benutzer sichtbar ist. Es gibt verschiedene Szenarien, die den onStop()-Aufruf auslösen. Der Start einer neuen Activity führt zum Aufruf der onStop()-Methode, ebenso wie die Wiederherstellung einer bereits pausierten Activity, die die gestoppte Activity wieder überlagert. Nach dem Ausführen der onStop()-Methode kann die Activity durch Aufrufen der onRestart()-Methode wieder für den Benutzer aktiv werden um mit der

Activity zu interagieren. Es ist aber ebenso möglich die Activity komplett zu zerstören und die onDestroy()-Methode aufzurufen.

onRestart()

Diese Methode wird aufgerufen, wenn eine Activity vom „Gestoppt“-Zustand wieder gestartet werden soll. Die onRestart()-Methode stellt den Zustand der Activity wieder her, der bestand, bevor die Activity in den „Gestoppt“- Zustand gewechselt ist. Nach der Ausführung der onRestart()-Methode wird immer die onStart()-Methode aufgerufen.

onDestroy()

Bevor eine Activity zerstört wird, ruft das System die onDestroy()-Methode auf. Dies ist die letzte Callback-Methode, die die Activity verwendet. In der onDestroy()-Methode wird der Ablauf implementiert, der sicherstellen soll, dass alle von der Activity verwendeten Ressourcen wieder freigegeben werden, wenn die Activity zerstört wird.

Intents

Um von einer Activity zu einer anderen zu navigieren werden in Android sogenannte Intents verwendet [26]. Dies sind Anweisungen, denen man zusätzliche Informationen an den Empfänger des Intents mitgeben kann, um von Activity A zu Activity B zu wechseln. Es gibt explizite Intents und implizite Intents.

Explizite Intents

Bei expliziten Intents ist dem Entwickler bereits bewusst, welche Activity der Empfänger des Intents ist, nämlich eine Activity der eigenen Anwendung. Diese Art von Intent wird beispielsweise verwendet, um von einer Activity zu einer anderen Activity zu wechseln. Der Entwickler kann dabei auch über die Verwendung von Extras weitere Informationen an die aufgerufene Activity senden, welche von dieser zur Ausführung von Programmlogik verwendet werden können.

Implizite Intents

Verwendet ein Entwickler einen impliziten Intent, so ist noch nicht geregelt welche Activity der Empfänger des Intents ist. Meist wird es sich bei dem Empfänger eines impliziten Intents um eine andere Anwendung handeln, zum Beispiel ein Webbrowser,

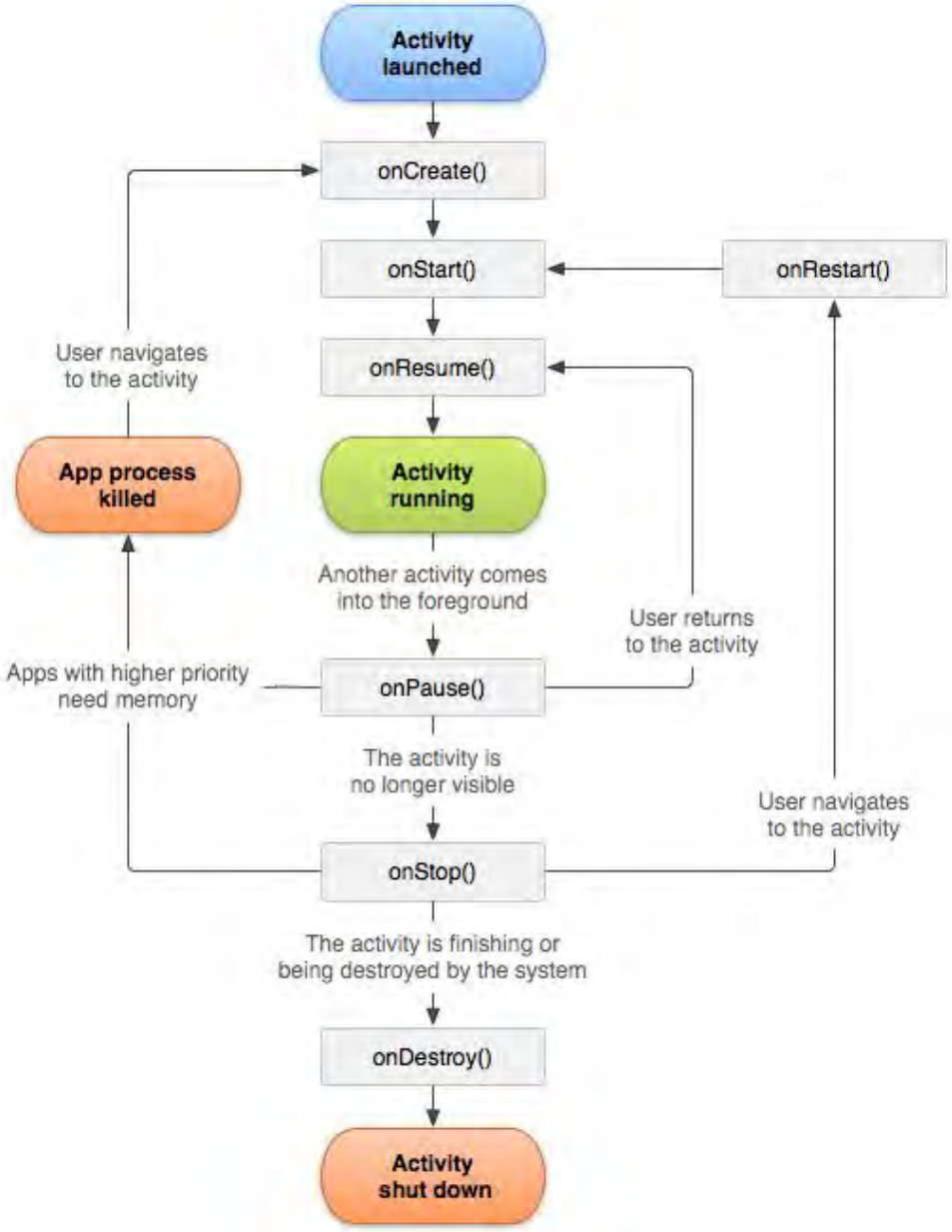

Abbildung 5.3: Activity Lebenszyklus [25]

der eine Website öffnen soll oder ein E-Mail-Client, der eine E-Mail versenden soll. Um als Empfänger eines impliziten Intents reagieren zu können, muss die ausführende Activity in der `AndroidManifest.xml` Datei der zugehörigen Anwendung einen Intent Filter registrieren [27]. Damit wird sichergestellt, dass eine Activity auf vorher definierte implizite Intents reagieren und sie annehmen kann.

5.4 Retrofit

Retrofit ist ein REST Client, der in der Android bzw. Java Programmierung verwendet werden kann. Die Bibiliothek wurde von Square entwickelt und bietet ein mächtiges Framework um mit einem REST Server über eine API zu kommunizieren [28]. Retrofit bietet Typsicherheit und nutzt OkHttp um Requests an einen Server zu schicken. Retrofit bietet sich besonders gut dafür an, JSON oder XML Daten von einer WebAPI herunterzuladen und weiterzuverarbeiten. Hierfür überführt Retrofit die JSON Daten in Java Objekte, welche vorher vom Entwickler definiert werden. Um die Serialisierung und Deserialisierung von Java Objekten zu ermöglichen, können von Retrofit verschiedene Konverter genutzt werden. Das Projekt verwendet hierfür den Gson Converter. Zuerst müssen die benötigten Java Klassen erstellt werden, die die JSON Daten repräsentieren sollen. Außerdem benötigt man eine Instanz des Retrofit Clients (siehe Listing 5.3) um Requests an den Server zu senden. Eine Instanz verwendet die Basis URL des Servers, an die die Anfragen gestellt werden und einen Konverter, mit welchem die JSON Daten der Serverantworten in Java Objekte übersetzt werden und umgekehrt. Diese Instanz von Retrofit muss immer aufgerufen werden, wenn vom Programm eine Anfrage an den Server gesendet werden soll. Wichtig für die Kommunikation mit dem Server sind die vordefinierten Routen der REST API, mit denen auf die jeweiligen Ressourcen zugegriffen werden kann. Die benötigten Routen werden für Retrofit in einem Interface (siehe Listing 5.4) gesammelt. Zu den jeweiligen Ressourcen Endpunkten gehören dabei auch die gewünschten HTTP Methoden und der erwartete Typ der Serverantwort. Dies wird benötigt, um JSON Daten in Java Objekte umwandeln lassen zu können. Retrofit erlaubt es, durch die Nutzung von Variablen, die Routen dynamisch anzupassen. Hierfür werden dem Request einfach Parameter im Methodenaufruf übergeben.

```java
package de.uulm.gindele.akindexapp.backend.retrofit;

import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {
   private static Retrofit retrofit = null;

   public static Retrofit getClient() {
      if(retrofit == null){
         OkHttpClient.Builder okHttpClientBuilder = new
            OkHttpClient.Builder();
         HttpLoggingInterceptor logging = new
            HttpLoggingInterceptor();
         logging.setLevel(HttpLoggingInterceptor.Level.BODY);
         okHttpClientBuilder.addInterceptor(logging);

         retrofit = new Retrofit.Builder()
               .baseUrl("https://tyt.johannesschobel.com/")
               .addConverterFactory(GsonConverterFactory.create())
               .client(okHttpClientBuilder.build())
               .build();
      }
      return retrofit;
   }

}
```

Listing 5.3: `RetrofitClient.java`

```java
...

public interface ApiEndpointInterface {
```

```
4
5
6     @POST("api/v1/studies/{studyID}/subscribe")
7     Call<ResponseBody> subscribeToStudy(@Path("studyID") int studyID,
          @Query("token") String token, @Body Request<Subscribe>
          subscribeStudyRequest);
8
9     @DELETE("api/v1/studies/{studyID}/unsubscribe")
10    Call<ResponseBody> unsubscribeFromStudy(@Path("studyID") int
          studyID, @Query("token") String token);
11
12    @DELETE("api/v1/auth/logout")
13    Call<ResponseBody> logout(@Query("token") String token);
14
15    @POST("api/v1/auth/register")
16    Call<ResponseBody> register(@Body Request<Registration>
          registrationRequestRequest);
17
18    @POST("api/v1/auth/verify/resend")
19    Call<ResponseBody> resendVerificationMail(@Body
          Request<ResendVerification> resendVerificationMailRequest);
20
21    @POST("api/v1/auth/password/reset/instructions")
22    Call<ResponseBody> resetPW(@Body Request<ResetPassword>
          resetPWRequest);
23
24    @POST("api/v1/auth/login")
25    Call<ResponseCustom<LoginRes>> login(@Body Request<Login>
          loginRequest);
26
27    @GET("/api/v1/my/studies")
28    Call<ResponseCustom<List<Study>>> getMyStudies(@Query("token")
          String token);
29
30
```

```
31
32      @PATCH("api/v1/studies/{studyID}/members/{memberID}/answersheets/{answersheet
33      Call<ResponseBody> approveFeedback (@Path("studyID") int studyID,
            @Path("memberID") int memberID, @Path("answersheetID") int
            answersheetID, @Body Request<Flag> patchFlagRequest,
            @Query("token") String token);
34
35      @GET("api/v1/my/roles")
36      Call<ResponseCustom<List<Data>>> getMyRole(@Query("token") String
            token);
37
38      ...
39
40  }
```

Listing 5.4: Auszug aus `ApiEndpointInterface.java`

5.5 JSON Dateiformat

JSON (JavaScript Object Notation) bezeichnet ein einfach zu parsendes Datenaustauschformat, welches durch seine Struktur auch für das menschliche Auge einfach zu lesen und interpretieren ist [29]. JSON kann unabhängig von der verwendeten Programmiersprache zum Datenaustausch zwischen Systemen genutzt werden. JSON basiert grundsätzlich auf zwei verschiedenen Strukturen, Key/Value-Paaren und geordneten Listen. Diese beiden Datenstrukturen können in fast allen modernen Programmiersprachen implementiert werden, was JSON zu einem universellen Datenaustauschformat macht. Objekte können als ungeordnete Menge von Key/Value-Paaren dargestellt werden (siehe Abbildung 5.4). Ein Objekt beginnt immer mit einer geschwungenen Klammer ({) und endet mit der zugehörigen schließenden geschwungenen Klammer (}). Um die Assoziation von Werten (Values) und Schlüsseln (Keys) zu verdeutlichen wird jedem Schlüssel, getrennt durch einen Doppelpunkt (:), ein Wert zugewiesen. Ein Schlüssel ist immer eine Zeichenkette. Ein Wert kann wiederum ein Objekt, ein Array, eine Zeichenkette,

eine Zahl, ein Boolscher Wert oder null sein. Arrays beginnen in JSON mit einer eckigen Klammer ([) und enden auf die schließende eckige Klammer (]). Ein JSON Array kann jede Art von Wert als Elemente aufnehmen. Einzelne Elemente eines JSON Arrays werden durch Komma voneinander getrennt.

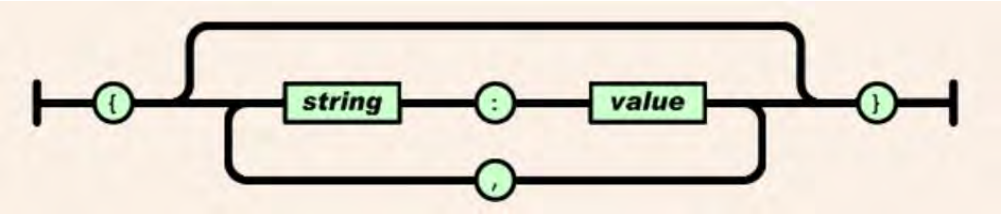

Abbildung 5.4: JSON Objekt [29]

5.6 JSON REST API

Für die Kommunikation zwischen der vorliegenden Anwendung und dem Backend Server wird auf eine REST API zurückgegriffen, die es ermöglicht, Daten im JSON Format auszutauschen. Der Representational State Transfer (REST) ist ein Architekturstil, der eine einheitliche Schnittstelle zum Informationsaustausch fordert. REST wurde von Roy Fielding entwickelt und nutzt die vorhandene Infrastruktur des World Wide Web für die Kommunikation zwischen verteilten Systemen [30]. Fielding entwarf sechs Eigenschaften, die ein Dienst erfüllen muss um als REST-konform zu gelten [31].

Client-Server-Modell: Beim REST-Paradigma wird eine Client-Server-Architektur verwendet. Das bedeutet, dass ein Server einen Dienst bereitstellt, auf den verschiedene, verteilte Clients zugreifen können.

Zustandslosigkeit: REST Clients und REST Server kommunizieren miteinander auf eine zustandslose Art. Das heißt, dass keine Informationen, die den Anwendungszustand betreffen, zwischen zwei aufeinanderfolgenden Nachrichten gespeichert werden. Um eine fehlerfreie Verständigung zwischen Client und Server zu erreichen, werden alle benötigten Informationen in einer REST Nachricht gesendet.

Caching: Clients sollen Nachrichten, die sie von einem Server erhalten haben, speichern können, um die Informationen bei einem späteren gleichartigen Request verwenden zu können ohne eine Verbindung zum Server aufbauen zu müssen. Damit wird die Netzwerkeffizienz verbessert, aber gleichzeitig besteht die Gefahr, dass der Client auf veraltete Daten zugreift.

Einheitliche Schnittstelle: Ein REST-konformer Service bietet den Nutzern eine einheitliche Schnittstelle, die es erlaubt Informationen in einem standardisierten Format, unabhängig von der darauf basierenden Anwendung, auszutauschen.

Mehrschichtiges System: REST soll ein mehrschichtiges, hierarchisches System verwenden. Dabei kann jede Komponente nur mit der direkt angrenzenden Schicht kommunizieren. Mit einer einfachen Schnittstelle kann dem Anwender die Komplexität der dahinter liegenden Schichten verborgen bleiben und somit eine einfachere Architektur erreicht werden.

Code-On-Demand: Dies ist eine optionale Bedingung für einen REST Service. Sie besagt, dass es möglich sein muss, dass die Funktion von Clients durch das Nachladen von ausführbarem Code erweitert werden kann, wenn dies vom Client verlangt wird.

Das REST-Paradigma wird in der vorliegenden Anwendung durch die Verwendung des sicheren Hypertext Übertragungsprotokoll HTTPS realisiert. Die Ressourcen des Servers werden per URI angesprochen. Die HTTP-Methoden (GET, POST, PUT,...) geben an, welche Operation der Backend Server ausführen soll. Die Dokumentation der Schnittstelle, die für die vorliegende Anwendung verwendet wurde, ist unter [32] zu finden. Sie enthält alle verfügbaren Routen die die Schnittstelle anbietet, die für die Kommunikation zwischen Client und Server notwendig sind. Außerdem werden die Daten erläutert, die an die unterschiedlichen Routen gesendet werden können, um eine korrekte Verarbeitung durch den Backend Server zu garantieren. Hierfür ist das JSON Datenformat gewählt worden.

5.7 Generics

Java erlaubt seit der Version 1.5 den Einsatz von Generics [33]. Generics sind parametrisierte Datentypen, die es dem Programmierer erlauben Klassen und Methoden zu schreiben, die typsicher sind und dennoch eine generische Programmierung ermöglichen. Unter generischer Programmierung versteht man ein Paradigma, bei dem die Funktionen einer Software so allgemein gehalten sind, dass sie auf mehrere unterschiedlichen Datentypen und Datenstrukturen angewandt werden können. Der Vorteil von Generics liegt darin, dass sie in einem gewissen Rahmen Typsicherheit garantieren, da die Klassen und Methoden parametrisierte Typen verwenden. Erst zum Zeitpunkt der Verwendung der generischen Klassen und Methoden werden die parametrisierten Typen durch konkrete Typen ersetzt, die den Anforderungen der parametrisierten Typen genügen müssen. Damit lassen sich `ClassCastExceptions` während der Laufzeit verhindern, da bereits der Compiler eine Typprüfung vornimmt und auf die fehlerhafte Verwendung eines Datentyps aufmerksam macht [34]. Generics wurden im vorliegenden Projekt verwendet um die Klassen übersichtlich zu strukturieren. Die Antworten des Servers im JSON Format werden auf Response Klassen übertragen. Da die Struktur der Serverantworten einem bestimmten Schema folgt, war es durch den Einsatz von Generics möglich, eine übersichtliche Klassenstruktur zu benutzen. Jede Response besteht aus einem Data Objekt und nur die Data Objekte sind unterschiedlich aufgebaut. So konnten für die verschiedenen Serverantworten passende Klassen erzeugt werden.

5.8 Datenpersistierung

Android bietet verschiedene Möglichkeiten zur Speicherung von Informationen. Je nach Einsatzzweck ist es notwendig die interne relationale Datenbank SQLite zu verwenden oder auf das simplere Verfahren, die Nutzung von Shared Preferences, zurückzugreifen. Mit SQLite bietet Android eine bereits integrierte relationale Datenbank an, die besonders gut dafür geeignet ist, strukturierte Daten aufzunehmen und in Tabellen zu speichern [35].Um die SQLite Datenbank zu nutzen muss zuerst ein Datenbank Schema erstellt werden, als eine Basis für die Datenbankoperationen. In diesem Schema werden

die benötigten Tabellen und ihre Beziehungen untereinander definiert. Um Operationen auf der Datenbank durchzuführen, sollte eine DB Helper Klasse genutzt werden, die von SQLiteOpenHelper erbt. Dies vereinfacht die Kommunikation mit der Datenbank und es können relativ einfach Manipulationen an der Datenbank durchgeführt werden. Die Verwendung von SQLite bietet sich in der vorliegenden Applikation an, um abgebrochene Fragebögen zu speichern, welche zu einem späteren Zeitpunkt vervollständigt werden können. Für die Speicherung der gewählten Spracheinstellung der Anwendung können die Shared Preferences von Android genutzt werden. Durch die Verwendung der Shared Preferences Schnittstellen kann eine relativ kleine Menge an Daten in Form von Key/Value Paaren gespeichert werden [36]. Ein Shared Preferences Objekt bietet Methoden, um Informationen in eine Datei zu schreiben und wieder auszulesen.

6

Implementierung

In diesem Kapitel werden ausgewählte Komponenten der Implementierung vorgestellt. Es werden Komponenten behandelt aus den Bereichen der Netzwerkkommunikation sowie der Datenpräsentation für die Benutzerin. Es wird auch auf ein Designmuster für das Datenmodell eingegangen.

6.1 Implementierung ausgewählter Komponenten

6.1.1 Singleton Pattern der User Klasse

Das Model für einen Benutzer der Anwendung wurde in der User Klasse realisiert. Die User Klasse folgt dem Singleton Pattern [37]. Dies bedeutet, dass im gesamten Programm nur eine Instanz der User Klasse vorhanden ist, womit diese Instanz sehr gut dafür geeignet ist, Daten eines Benutzers vorzuhalten, die über den Programmablauf von verschiedenen Activities verwendet werden. Hierzu zählen der Authentifizierungstoken und der Indikator, ob es sich beim aktuell eingeloggten User um einen Arzt oder eine Patientin handelt. Oft wird das Singleton Pattern benutzt, um eine Datenbankverbindung oder eine Netzwerkkommunikation zu realisieren, weil auch in diesen Fällen nur eine Instanz des Objekts erwünscht ist um Dateninkonsistenzen zu vermeiden. Im vorliegenden Projekt wurde das Singleton Pattern mit Lazy Initialization [38] gewählt, weil dadurch erst geprüft wird, ob bereits eine Instanz der Klasse vorhanden ist und nur im Falle des Nichtvorhandenseins eine Instanz der User Klasse erzeugt wird (Listing 6.1, Zeilen 11-16). Erst bei Verwendung der Instanz wird das Objekt angelegt und nicht bei

der Erzeugung der Klasse. Ansonsten beinhaltet die User Klasse private Attribute, die die Informationen halten können, welche für den Programmverlauf benötigt werden, wie den Authentifizierungstoken (Zeile 5) oder die Nutzerrolle (Zeile 7). Außerdem besitzt die Klasse getter- und setter-Methoden, um auf die Attribute zugreifen zu können.

```
1  package de.uulm.gindele.akindexapp.models;
2
3  public class User {
4     private static User myInstance;
5     private String token;
6     private String email;
7     private Boolean doctor;
8     private int studyID;
9
10
11    public static User getInstance() {
12      if(myInstance == null){
13         myInstance = new User();
14      }
15        return myInstance;
16    }
17
18    private User() {
19    }
20
21    public String getToken() {
22       return token;
23    }
24
25    public void setToken(String token) {
26       this.token = token;
27    }
28
29    public String getEmail() {
30       return email;
```

```
31      }
32
33      public void setEmail(String email) {
34          this.email = email;
35      }
36
37      public Boolean getDoctor() {
38          return doctor;
39      }
40
41      public void setDoctor(Boolean doctor) {
42          this.doctor = doctor;
43      }
44
45      public int getStudyID() {
46          return studyID;
47      }
48
49      public void setStudyID(int studyID) {
50          this.studyID = studyID;
51      }
52  }
```

Listing 6.1: `User.java`

6.1.2 List Adapter

Elemente in einer Liste anzuzeigen gehört zu den häufigsten Darstellungsformen einer mobilen Anwendung für Smartphones. Das vorliegende Projekt erfordert häufig die Darstellung von bestimmten Daten in einer Übersicht. Aufgrund der Konstruktion der Daten ist es sinnvoll, diese in einer Liste darzustellen, womit dem Benutzer eine schnell erfassbare Übersicht präsentiert werden kann. Android bietet hierfür mehrere Möglichkeiten an View Elementen. Verwendet wurden ListViews und ExpandableListViews, die bei Anklicken eines Listenelements aufgeklappt werden können um weitere

Informationen anzuzeigen. In einer ListView können beliebige Java Objekte als Datentypen für die Listenelemente verwendet werden. Um die gewünschten Informationen in der ListView anzuzeigen, ist allerdings ein Adapter notwendig, der die erforderlichen Daten aus den Objekten bezieht um sie in der Liste darzustellen [39]. Mit der `MyExpandableListAdapter.java`-Datei (Listing 6.2) wird veranschaulicht, wie der Adapter für eine ExpandableListView realisiert wurde. Der MyExpandableListAdapter erbt von BaseExpandableListAdapter, welcher das ExpandableListAdapter Interface implementiert. Da es sich bei den darzustellenden Daten für diesen Adapter um eine Liste von Feedback Objekten handelt, wird dem Konstruktor der Anwendungskontext und eine Liste von Feedback Objekten übergeben (Zeile 21). Die beiden wichtigsten Methoden, für die Darstellung der Feedbackdaten in einer Liste, sind die getGroupView und die getChildView, die ähnlich aufgebaut sind. Zuerst wird die anzuzeigende Zeichenkette für die oberste Hierarchieebene aus dem jeweiligen Feedback Objekt entnommen (Zeile 63 und Zeile 38). Danach wird eine View mit Hilfe des LayoutInflaters befüllt. Als Layout wird eine zuvor definierte xml-Datei verwendet (Zeile 66). Anschließend wird die TextView, welche in der Liste angezeigt werden soll, mit dem gewünschten Text befüllt (Zeile 71). Am Ende wird die fertige View an den Aufrufenden zurückgegeben. Die getChildView-Methode ist ähnlich aufgebaut, wird aber dafür benötigt, den Text anzuzeigen, der beim Aufklappen eines Listenelements sichtbar wird.

```
1  package de.uulm.gindele.akindexapp.adapters;
2
3  import android.content.Context;
4  import android.graphics.Typeface;
5  import android.view.LayoutInflater;
6  import android.view.View;
7  import android.view.ViewGroup;
8  import android.widget.BaseExpandableListAdapter;
9  import android.widget.TextView;
10
11 import java.util.List;
12
13 import de.uulm.gindele.akindexapp.R;
```

```
14 import de.uulm.gindele.akindexapp.backend.responses.Feedback;
15
16 public class MyExpandableListAdapter extends
       BaseExpandableListAdapter {
17
18    private Context context;
19    private List<Feedback> feedbacks;
20
21    public MyExpandableListAdapter(Context context, List<Feedback>
          feedbacks) {
22       this.context = context;
23       this.feedbacks = feedbacks;
24    }
25
26    @Override
27    public int getGroupCount() {
28       return feedbacks.size();
29    }
30
31    @Override
32    public int getChildrenCount(int i) {
33       return 1;
34    }
35
36    @Override
37    public Object getGroup(int i) {
38       return feedbacks.get(i).getTrueAnswer().get("headline");
39    }
40
41    @Override
42    public Object getChild(int i, int i1) {
43       return feedbacks.get(i).getTrueAnswer().get("text");
44    }
45
46    @Override
```

```
47    public long getGroupId(int i) {
48        return i;
49    }
50
51    @Override
52    public long getChildId(int i, int i1) {
53        return i1;
54    }
55
56    @Override
57    public boolean hasStableIds() {
58        return false;
59    }
60
61    @Override
62    public View getGroupView(int i, boolean b, View view, ViewGroup
          viewGroup) {
63        String headerTitle = (String) getGroup(i);
64        if (view == null) {
65            LayoutInflater inflater = (LayoutInflater)
                  context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
66            view = inflater.inflate(R.layout.list_group, null);
67        }
68
69        TextView tvListHeader = view.findViewById(R.id.lblListHeader);
70        tvListHeader.setTypeface(null, Typeface.BOLD);
71        tvListHeader.setText(headerTitle);
72
73        return view;
74    }
75
76    @Override
77    public View getChildView(int i, int i1, boolean b, View view,
          ViewGroup viewGroup) {
78        String childText = (String) getChild(i,i1);
```

```
79          if (view == null) {
80              LayoutInflater inflater = (LayoutInflater)
                    context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
81              view = inflater.inflate(R.layout.list_child, null);
82          }
83
84          TextView txtChild = view.findViewById(R.id.lblListItem);
85          txtChild.setText(childText);
86
87          return view;
88      }
89
90      @Override
91      public boolean isChildSelectable(int i, int i1) {
92          return false;
93      }
94  }
```

Listing 6.2: `MyExpandableListAdapter.java`

6.1.3 Kommunikation mit dem Backend Server

Am Beispiel der `FeedBackDoctorActivity.java` soll gezeigt werden, wie die Kommunikation zwischen der Anwendung und dem Backend Server implementiert wurde. Da für die Kommunikation das Framework Retrofit zuständig ist, werden im Folgenden der Retrofit Client und das Retrofit Interface verwendet, die unter Kapitel 5 aufgeführt sind. Zunächst wird eine Instanz der User Klasse verlangt, um die benötigten Daten für diesen User zu erhalten (Zeile 1). Da die FeedbackDoctorActivity von einer anderen Activity durch einen expliziten Intent aufgerufen wurde, muss aus diesem Intent die übergebene Information (extras) extrahiert werden (Zeile 5). Da der Request an den Server durch einen Buttonklick ausgelöst werden soll, kommt der OnClickListener zum Einsatz (Zeile 8). In dessen onClick-Methode erzeugt die Retrofit Klasse eine Implementierung des ApiEndpointInterface (Zeile 12). Damit kann nun die approveFeedback-Methode, mit

den benötigten Informationen als Argumente, aufgerufen werden, um ein Call-Objekt zu erzeugen (Zeile 13). Dieses Call-Objekt wird von Retrofit in eine Queue eingereiht (Zeile 14) und bei verfügbarer Kapazität wird der Request an den Server gesendet. Die onResponse-Methode behandelt das Verhalten bei einer Antwort vom Server. Hier wird zuerst anhand des HTTP Statuscodes geprüft, ob die Antwort erfolgreich war (Zeile 18) und im positiven Fall eine Nachricht als Dialog angezeigt (Zeile 19).

```
1  User user = User.getInstance();
2        final String token = user.getToken();
3        final int studyID = user.getStudyID();
4        Intent doctorFeedbackIntent = getIntent();
5        Bundle extras = doctorFeedbackIntent.getExtras();
6        final int answerSheetID = extras.getInt("answerSheetID");
7        final int memberID = extras.getInt("memberID");
8        approveFeedbackBTN.setOnClickListener(new
             View.OnClickListener() {
9           @Override
10          public void onClick(View view) {
11             Retrofit retrofit = RetrofitClient.getClient();
12             ApiEndpointInterface apiService =
                   retrofit.create(ApiEndpointInterface.class);
13             Call<ResponseBody> approveFeedbackCall =
                   apiService.approveFeedback(studyID, memberID,
                   answerSheetID, approveFeedbackRequest, token);
14             approveFeedbackCall.enqueue(new Callback<ResponseBody>()
                   {
15                @Override
16                public void onResponse(Call<ResponseBody> call,
                      Response<ResponseBody> response) {
17                   int statusCode = response.code();
18                   if (statusCode == 202){
19                      Toast.makeText(getApplication(), "Feedback
                            approved", Toast.LENGTH_SHORT).show();
```

Listing 6.3: Auszug aus `FeedbackDoctorActivity.java`

7

Vorstellung der Anwendung

Die vorliegende Android Anwendung basiert auf einem Rollenkonzept, das es Schwangeren sowie Ärzten ermöglicht, diese App auf dem jeweiligen Smartphone zu nutzen. Im folgenden Kapitel werden die Funktionen der KINDEX App vorgestellt und erläutert.

7.1 Tutorialscreen

Die App verwendet zur Einführung neuer Benutzerinnen einen Tutorialscreen. Da Ärztinnen bereits bei der Registrierung durch eine begleitende Institution eine Einführung in die Funktionen der Anwendung erhalten, wird mit dem Tutorialscreen nur die Anwendung durch eine Patientin erklärt. Hierbei wird durch verschiedene Screenshots erklärt, wie sich eine neue Anwenderin registrieren, einloggen und die Hauptfunktionen der App nutzen kann. Der Tutorialscreen wird nur bei der ersten Ausführung der Anwendung angezeigt.

7.2 Onlinezwang

Die Anwendung kann nur mit bestehender Internetverbindung ausgeführt werden. Dafür wird beim Starten der App überprüft, ob eine Verbindung vorhanden ist. Falls keine Netzverbindung besteht, wird die Nutzerin durch einen Dialog darauf hingewiesen.

7.3 Registrierung

Die App kann nur mit einem aktivierten Benutzeraccount und bestehender Internetverbindung genutzt werden. Eine Ärztin wird von der studienbegleitenden Institution mit ihrer E-Mail-Adresse und einem selbstgewählten Passwort registriert, sie kann sich also nicht über die Registrierungsmöglichkeit in der App anmelden. Im Gegensatz dazu, ist es für eine Schwangere Patientin notwendig, sich ein Benutzerkonto über die Anwendung zu erstellen. Nach Starten der App muss auf den Button „Registrieren“ geklickt werden. Dies öffnet einen neuen Bildschirminhalt, die RegisterActivity, welche die Eingabemöglichkeiten für persönliche Informationen beinhaltet. Nach der Eingabe von Vorname, Name, E-Mail, Passwort und einer Passwortbestätigung werden die eingegebenen Daten an den Backend Server geschickt. Die Eingaben werden vom Backend geprüft und bei erfolgreicher Prüfung in einer Datenbank gespeichert, was der Benutzerin über einen kurzen Dialog mitgeteilt wird. Der Benutzerin wird anschließend eine E-Mail an die angegebene Adresse geschickt, die einen Verifizierungslink beinhaltet. Über diesen Link muss die Anmeldung an der KINDEX App bestätigt werden. Sollte es während des Registrierungsprozesses zu Verbindungsproblemen kommen oder fehlerhafte Eingaben gemacht werden, wird dies der Benutzerin ebenfalls über einen kurzen Dialog angezeigt. Nach erfolgter Bestätigung kann man sich in der App einloggen.

Abbildung 7.1: Registrierung

7.4 Verifizierungslink

Nach erfolgreicher Registrierung wird der Benutzerin ein Verifizierungslink per E-Mail zugesandt. Sollte es bei der Zustellung der E-Mail zu Unregelmäßigkeiten oder Problemen gekommen sein, besteht die Möglichkeit, erneut einen Verifizierungslink vom Backend anzufordern. Hierfür ist es notwendig, die E-Mail-Adresse einzutragen, die auch bei der Registrierung verwendet wurde. Durch das Betätigen des „Verifizierungsmail erneut senden“-Buttons wird vom Backend Server wiederholt eine E-Mail an die Adresse der Benutzerin verschickt, die einen neuen Verifizierungslink beinhaltet. Nach erfolgter Bestätigung durch Anklicken des Verifizierungslinks kann man sich mit seinem registrierten Benutzerkonto in der App einloggen.

7.5 Einloggen

Um sich einloggen zu können, muss eine Benutzerin bereits registriert sein. Dies geschieht im Falle einer Ärztin durch die studienbegleitende Institution und im Falle einer Patientin durch die oben beschriebene Registrierungsfunktion. Zum Einloggen muss auf dem Startbildschirm der App zuerst die E-Mail-Adresse der Benutzerin, welche bei der Registrierung angegeben wurde, eingetragen werden. Außerdem ist es erforderlich, das Passwort, welches dem Benutzerkonto zugeordnet ist, einzugeben. Dieses Passwort muss nicht mit dem initialen Passwort übereinstimmen, welches die Benutzerin bei der Registrierung angegeben hat, da es nach einer Registrierung mehrere Möglichkeiten gibt, das Passwort zu ändern, worauf in diesem Kapitel näher eingegangen wird. Nach der Eingabe von korrekten Zugangsdaten und Betätigung des „Einloggen“-Buttons wird der Benutzerin über einen kurzen Dialog mitgeteilt, dass die Aktion erfolgreich war und die Benutzerin wird auf den Hauptbildschirm geleitet. Sind die eingegebenen Daten nicht korrekt oder sollten während des Vorgangs Verbindungsprobleme auftreten, wird dies ebenfalls in einem kurzen Dialog angezeigt.

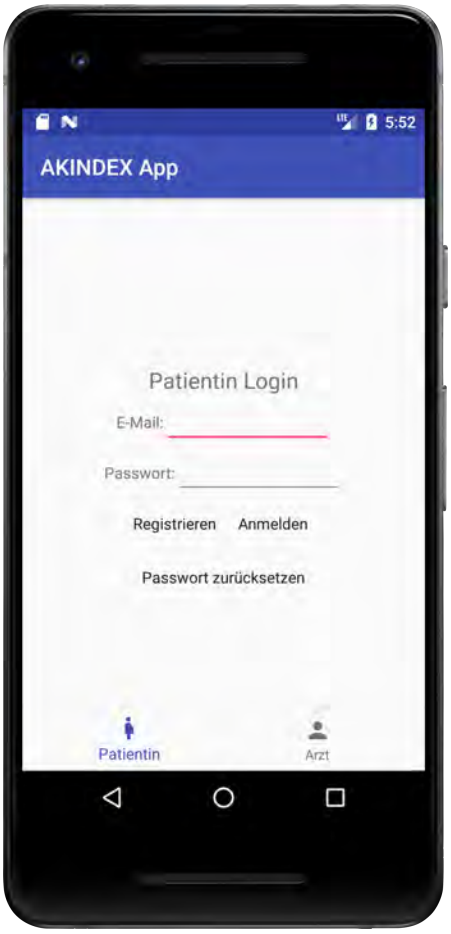

Abbildung 7.2: Login Screen

7.6 Passwort zurücksetzen

Nach erfolgreicher Registrierung ist es möglich, das Passwort des Benutzerkontos zurücksetzen zu lassen. Hierfür muss im Startbildschirm der App die E-Mail-Adresse des Benutzerkontos eingegeben werden und auf „Passwort zurücksetzen“ geklickt werden. Wenn die eingegebene E-Mail-Adresse in der Datenbank des Backend Servers vorhanden ist, wird an diese Adresse eine E-Mail verschickt, die weitere Instruktionen enthält um das Passwort des Benutzerkontos zurückzusetzen. Eine erfolgreiche Versendung der Nachricht wird durch einen kurzen Dialog angezeigt. Sollte es während der Übertragung zu Verbindungsproblemen gekommen sein oder ist die angegeben E-Mail Adresse nicht in der Datenbank des Backend Servers vorhanden, wird dies ebenfalls in einem kurzen Dialog angezeigt.

7.7 An Studie teilnehmen

Beim ersten erfolgreichem Login einer Patientin wird zunächst die Möglichkeit geboten, sich an einer Studie anzumelden. Dies ist nötig, um die Hauptfunktion der Anwendung, das Ausfüllen von KINDEX Fragebögen, zu nutzen. Um sich an einer Studie anzumelden muss vorab ein persönlicher Kontakt zu der begleitenden Ärztin bestehen. Die Ärztin muss ihrer Patientin die Anmeldeinformationen zu ihrer Studie mitteilen. Dazu gehört die E-Mail-Adresse der Ärztin, die für die Studie verwendet wird und das von der Ärztin vergebene Studienpasswort. Das Studienpasswort ist vergleichbar mit einem Einschreibeschlüssel, der nur Studienteilnehmerinnen und studienbegleitendem Personal bekannt ist. Die Anmeldung zu einer Studie wird dann über die App durchgeführt. Hierfür gibt die Patientin die Anmeldeinformationen ein und klickt anschließend den „Anmelden“-Button. War die Anmeldung zu einer Studie erfolgreich, kann ab diesem Zeitpunkt auf weitere Funktionen der App zugegriffen werden. Erst bei bestehender Anmeldung an einer Studie ist es möglich, einen KINDEX Fragebogen auszufüllen. Nach erfolgter Anmeldung kann die Patientin über den Menüpunkt „Studie verlassen“ die Studie auch wieder verlassen. Danach sind das Ausfüllen eines Fragebogens und auch das Betrachten von früherem Feedbacks nicht mehr möglich.

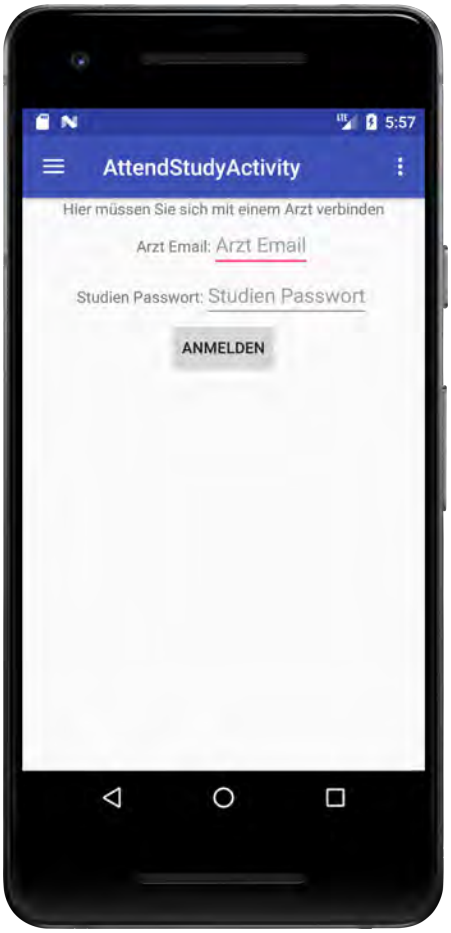

Abbildung 7.3: An Studie teilnehmen

7.8 Fragebogen starten

Nach erfolgreicher Anmeldung an einer KINDEX Studie erhält die Schwangere Zugang zu dem KINDEX Fragebogen. Auf dem Willkommensbildschirm der App, der nach erfolgreichem Login angezeigt wird, kann durch Aktivierung des „Neuer Fragebogen“-Buttons ein neuer Fragebogen vom Backend Server geladen werden. Nach erfolgreicher Übertragung des Fragebogens werden der Nutzerin die Fragen des Fragebogens einzeln angezeigt. Die Patientin kann nun den Fragebogen komplett ausfüllen und an den Backend Server schicken.

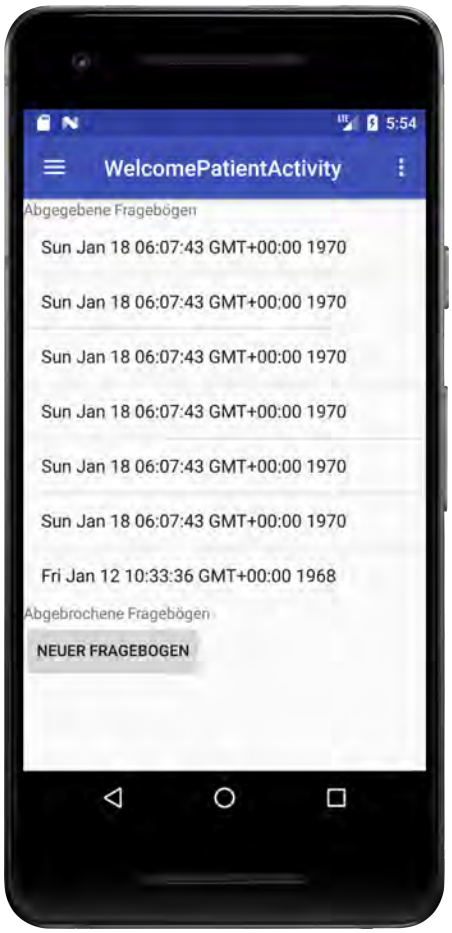

Abbildung 7.4: Fragebogen starten

Die Nutzerin sieht immer eine Frage und je nach Fragetyp verschiedene Antwort- bzw. Eingabemöglichkeiten. Außerdem ist durch eine Prozentangabe der momentane Fortschritt zu erkennen. Während der Beantwortung der Fragen sind alle Menüeinträge deaktiviert, wodurch keine anderen Funktionen nutzbar sind.

Abbildung 7.5: Fragebogenansicht

7.9 Feedback ansehen

Nach dem Absenden eines ausgefüllten Fragebogens an den Backend Server werden die gegebenen Antworten evaluiert. Sobald eine Prüfung und Freigabe durch eine begleitende Ärztin stattgefunden hat, kann eine Patientin das Feedback für ihren Fragebogen

ansehen. Das bedeutet, dass kritisch beantwortete Fragen in einer Liste dargestellt werden. Durch einen Klick auf den jeweiligen Eintrag wird die Anzeige erweitert und es werden Verhaltensempfehlungen für die Patientin angeboten.

Abbildung 7.6: Feedbackansicht Patientin

7.10 Patientinnendetails anzeigen

Eine eingeloggte Ärztin sieht auf ihrem Startbildschirm zuerst eine Liste von Patientinnen, die an der Studie der eingeloggten Ärztin teilnehmen. Mit einem Klick auf eine Patientin wird die Patientinnendetailansicht geöffnet. Hier werden Name und Vorname der Patientin angezeigt, sowie die E-Mail-Adresse. Außerdem wird eine Liste von ausgefüllten

Fragebögen der Patientin dargestellt. Diese Liste ist ebenfalls anklickbar, um der Ärztin die Auswertung eines Fragebogens zu ermöglichen.

Abbildung 7.7: Patientinnendetails

7.11 Feedback freigeben

Nachdem eine Patientin und ein zugehöriger ausgefüllter Fragebogen ausgewählt wurde, sieht die Ärztin eine Übersicht der kritisch beantworteten Fragen. Die Ärztin kann nun durch einen Klick auf einen Listeneintrag die Verhaltensempfehlungen aufklappen. Hat die Ärztin alle Empfehlungen begutachtet, kann sie durch einen Klick auf den „Feedback

absenden“-Button das Feedback freigeben. Ab jetzt ist das Feedback auch für die Patientin sichtbar.

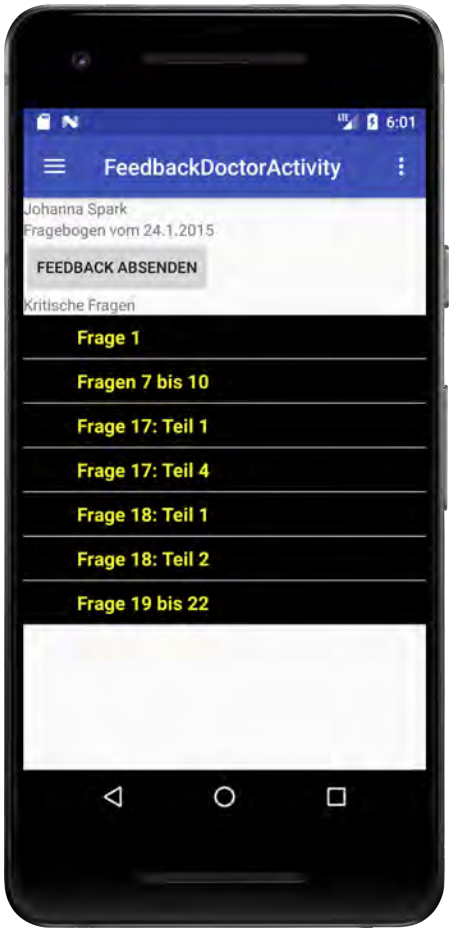

Abbildung 7.8: Feedback freigeben

7.12 Passwort ändern

Im eingeloggten Zustand ist es einem Nutzer möglich, das Passwort für das aktuelle Benutzerkonto zu ändern. Dazu muss im Hauptmenü der Eintrag „Profil“ gewählt werden. In der folgenden Eingabemaske muss zweimal das neu gewählte Passwort eingegeben werden. Mit der Betätigung des „Passwort ändern“-Buttons wird das Passwort für das

Benutzerkonto geändert. Der Nutzer erfährt durch einen kurzen Dialog, ob die Änderung erfolgreich durchgeführt werden konnte.

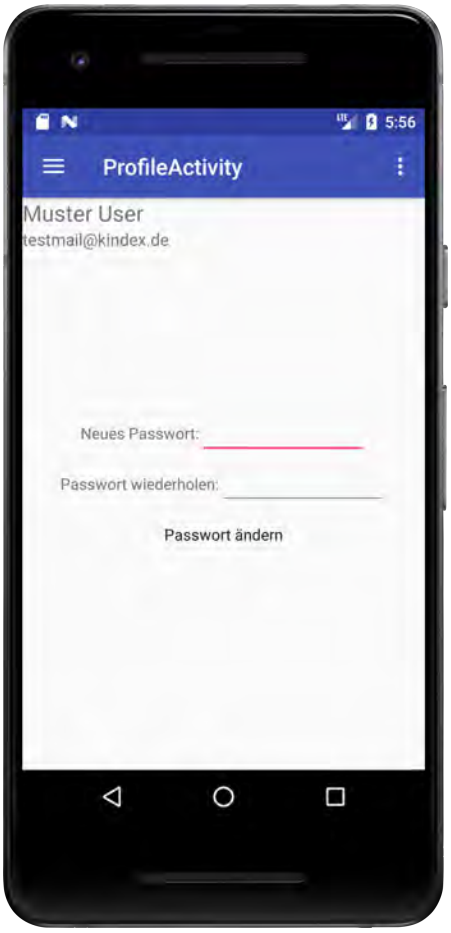

Abbildung 7.9: Passwort ändern

7.13 Entwickler kontaktieren

Die Nutzerinnen der KINDEX App haben die Möglichkeit, mit den Entwicklern der Anwendung in Kontakt zu treten, um ihnen Feedback, Kritik und Verbesserungsvorschläge zukommen zu lassen. Dafür wurde eigens der Menüpunkt „Entwickler kontaktieren“ eingerichtet. Über diesen Eintrag gelangt der Benutzer zu einem Textfeld, das mit einer Nachricht an die Entwickler befüllt werden kann. Mit einem Klick auf den „Senden“-Button

wird die Nachricht an den Standard Mail Client des Smartphones mittels implizitem Intent übermittelt. Der Standard-Mail-Client wird geöffnet und ist bereits mit einem Betreff und der Standard-E-Mail-Adresse der Entwickler versehen. Nun kann der Benutzer seine Nachricht als E-Mail von seinem gewohnten Standard-Mail-Client versenden.

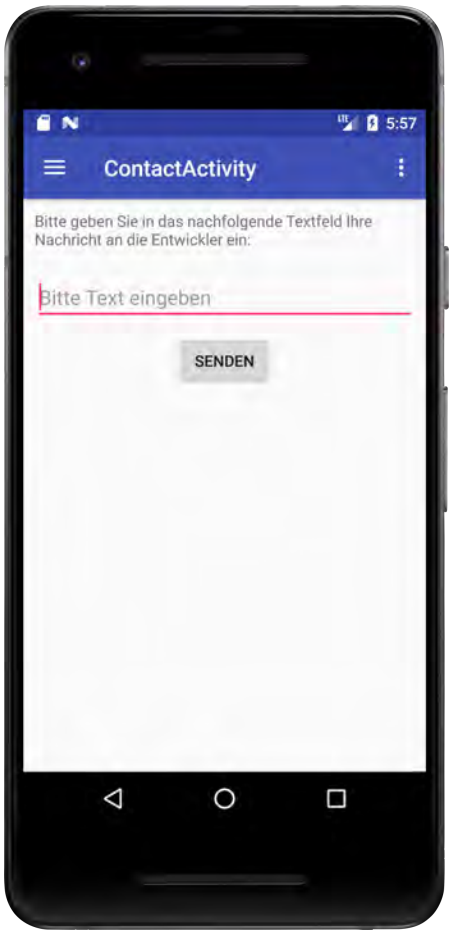

Abbildung 7.10: Entwickler kontaktieren

7.14 Informationen anzeigen

Über den Menüpunkt „Informationen“ gelangt der Nutzer zur Übersicht von Hintergrundinformationen der Anwendung. Hier findet man Hinweise zum Projekt und Datenschutz-

bestimmungen. Außerdem findet man eine Liste von Kontakten, die mit diesem Projekt verbunden sind.

Abbildung 7.11: Informationen anzeigen

7.15 Sprache einstellen

Da es sich bei dem vorliegenden Projekt um eine Android Anwendung handelt, die Mehrsprachigkeit unterstützen soll, enthält das Hauptmenü auch einen Eintrag zum Ändern der Spracheinstellung. Unter diesem Menüpunkt werden die Sprachen angezeigt, in denen der KINDEX Fragebogen auf dem Backend Server vorhanden ist.

Abbildung 7.12: Sprache einstellen

8

Anforderungsabgleich

In diesem Kapitel werden die Anforderungen aus Kapitel 4 noch einmal dargestellt um mit Hilfe der Tabelle 8.1 abzugleichen, welche Anforderungen in dem vorliegenden Projekt umgesetzt werden konnten.

Tabelle 8.1: Anforderungsabgleich

Label	**Titel**	**Status**
AF1	Registrierung	Anforderung erfüllt (siehe 7.3)
AF2	Rollenkonzept	Anforderung erfüllt (siehe 7)
AF3	Onlinezwang	Anforderung erfüllt (siehe 7.2)
AF4	Tutorialscreen	Anforderung erfüllt (siehe 7.1)
AF5	Studienteilnahme	Anforderung erfüllt (siehe 7.7)
AF6	Studie verlassen	Anforderung erfüllt (siehe 7.7)
AF7	Startseite	Anforderung erfüllt (siehe 7.8)
AF8	Fragebogen	Anforderung erfüllt (siehe 7.8)
AF9	Modale Fragen	Anforderung erfüllt (siehe 7.8)
AF10	Fortschrittsanzeige	Anforderung erfüllt (siehe 7.8)
AF11	Fragebogenabbruch	Anforderung nicht erfüllt
AF12	Auf Feedback warten	Anforderung erfüllt (siehe 7.8)
AF13	Feedback einsehen	Anforderung erfüllt (siehe 7.9)
AF14	Patientinnenansicht	Anforderung erfüllt (siehe 7.10)
AF15	Patientinnendetails	Anforderung erfüllt (siehe 7.10)
AF16	Fragebogendetails	Anforderung erfüllt (siehe 7.10)
AF17	Feedbackfreigabe	Anforderung erfüllt (siehe 7.11)
AF18	Login	Anforderung erfüllt (siehe 7.5)
AF19	Passwort ändern	Anforderung erfüllt (siehe 7.12)
AF20	Entwicklerkontakt	Anforderung erfüllt (siehe 7.13)
AF21	Mehrsprachigkeit	Anforderung nicht erfüllt
AF22	Impressum	Anforderung erfüllt (siehe 7.14)

9

Zusammenfassung

Das Ziel dieser Arbeit war es, eine Android Anwendung zu entwickeln, die mit Hilfe des Mobile Crowdsensing Paradigmas ermöglicht, dass Frauen während ihrer Schwangerschaft begleitet und unterstützt werden können. Dabei basiert die Anwendung auf dem KINDEX Fragebogen, welcher zur Erkennung von Risikofaktoren während der Schwangerschaft entwickelt wurde.

Zu Beginn der Arbeit wurden Grundlagen vermittelt, die zum Verständnis der Ausarbeitung beitragen sollen. Es wurde das Thema "Mobile Crowdsensing"generell beschrieben und auf verschiedene Aspekte des Paradigmas eingegangen. Als wichtigste Grundlage dieser Arbeit wurde der KINDEX Fragebogen detailliert beschrieben und auf Komplikationen mit der aktuellen Situation der KINDEX Interviews hingewiesen, um die Notwendigkeit einer Mobile Crowdsensing Anwendung nachvollziehbar zu machen.

Im Anschluss wurde ein kurzer Überblick über bereits entwickelte Projekte gegeben, die Grundlagen für die vorliegende Android Anwendung lieferten und die Möglichkeiten von mobilen Anwendungen aufzeigen.

Darauffolgend wurden die Anforderungen an die mobile Anwendung erarbeitet. Da es sich um eine rollenbasierte Anwendung handelt, wurden die Anforderungen nach den jeweiligen Rollen gegliedert und detailliert beschrieben. Um einen einfachen Abgleich mit den implementierten Funktionen zu erreichen, wurden die Anforderungen in einer Tabelle zusammengefasst.

Anschließend wurden Technologien und Architekturaspekte der App erklärt, die bei der Implementierung von Bedeutung waren. Da die Anwendung aus vielen Activities besteht,

wurde unter anderem darauf näher eingegangen. Außerdem wurden die Komponenten der Netzwerkkommunikation präsentiert.

Bei der Implementierung wurde der Fokus auf drei verschiedene Aspekte der Anwendung gelegt. Als Beispiel für ein Design Pattern konnte das Singleton Pattern verwendet werden, welches bei der Implementierung der User Klasse zum Einsatz kam. Für die Darstellung von Listen wurde auf einen eigenen Listenadapter zurückgegriffen und zur Kommunikation mit dem Backend Server wird die Vorgehensweise mittels Retrofit dargestellt.

Obwohl im Rahmen dieser Arbeit nicht alle geforderten Funktionen erfüllt werden konnten, bietet die mobile Anwendung, die aus diesem Projekt resultiert, eine gute Basis für weitere Entwicklungsmöglichkeiten. Die Umsetzung der verschiedenen Rollenprofile ermöglicht es zukünftig, Anwendungen zu entwickeln, die von unterschiedlichen Nutzergruppen verwendet werden können.

9.1 Ausblick

Die entstandene Mobile Crowdsensing Anwendung bietet bereits einen Funktionsumfang, der es erlaubt, von Patientinnen und Ärztinnen in sinnvoller Weise verwendet zu werden. Dennoch sollte zukünftig eine mehrsprachige Anwendung umgesetzt werden, um den Einsatz für Benutzerinnen zu ermöglichen, die eine andere Muttersprache haben. Sicherlich wäre es auch sinnvoll, eine Anwendung für unterschiedliche Plattformen und Endgeräte zu entwickeln um eine möglichst große Abdeckung von genutzten Apparaten zu erreichen. Außerdem könnte man bei einer rollenbasierten App eine Funktion zum Erstellen von Fragebögen integrieren, wodurch es einer bestimmten Rolle möglich ist, neue Fragebögen anzulegen oder bestehende zu verändern. Da die Kommunikation mit dem Backend Server über eine einheitliche Schnittstelle realisiert wurde, sollte eine solche Funktionserweiterung möglich sein und in einer großen Nutzensteigerung resultieren.

Literaturverzeichnis

[1] Ruf-Leuschner, M., Brunnemann, N., Schauer, M., Pryss, R., Barnewitz, E., Liebrecht, M., Reichert, M., Elbert, T.: Die KINDEX-App - ein Instrument zur Erfassung und unmittelbaren Auswertung von psychosozialen Belastungen bei Schwangeren in der täglichen Praxis bei Gynäkologinnen, Hebammen und in Frauenkliniken. Verhaltenstherapie (2016)

[2] Liu, J., Shen, H., Zhang, X.: A Survey of Mobile Crowdsensing Techniques: A Critical Component for the Internet of Things. In: 2016 25th International Conference on Computer Communication and Networks (ICCCN). (2016) 1–6

[3] Guo, B., Wang, Z., Yu, Z., Wang, Y., Yen, N.Y., Huang, R., Zhou, X.: Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm. ACM Comput. Surv. **48** (2015) 7:1–7:31

[4] Pryss, R., Probst, T., Schlee, W., Schobel, J., Langguth, B., Neff, P., Spiliopoulou, M., Reichert, M.: Mobile Crowdsensing for the Juxtaposition of Realtime Assessments and Retrospective Reporting for Neuropsychiatric Symptoms. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017), IEEE Computer Society Press (2017)

[5] Pryss, R., Probst, T., Schlee, W., Schobel, J., Langguth, B., Neff, P., Spiliopoulou, M., Reichert, M.: 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 (2018)

[6] Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Communications Magazine **49** (2011) 32–39

[7] Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T.: A survey of mobile phone sensing. IEEE Communications Magazine **48** (2010) 140–150

[8] Ruf-Leuschner, M., Pryss, R., Liebrecht, M., Schobel, J., Spyridou, A., Reichert, M., Schauer, M.: Preventing further trauma: KINDEX mum screen - assessing and reacting towards psychosocial risk factors in pregnant women with the help of smartphone technologies. In: XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference. (2013) 70–70

[9] Schickler, M., Reichert, M., Pryss, R., Schobel, J., Schlee, W., Langguth, B.: Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health. eXamen.press. Springer Vieweg, Berlin (2015)

[10] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: A Configurator Component for End-User Defined Mobile Data Collection Processes. In: Demo Track of the 14th International Conference on Service Oriented Computing (ICSOC 2016). (2016)

[11] QuestionSys - Universität Ulm. `https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys/` (2018) [Online; abgerufen am 28.4.2018].

[12] TrackYourTinnitus - Universität Ulm. `https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/trackyourtinnitus/` (2018) [Online; abgerufen am 28.4.2018].

[13] Probst, T., Pryss, R., Langguth, B., Schlee, W.: Emotional states as mediators between tinnitus loudness and tinnitus distress in daily life: Results from the "TrackYourTinnitus" application. Scientific Reports **6** (2016)

[14] Schlee, W., Pryss, R., Probst, T., Schobel, J., Bachmeier, A., Reichert, M., Langguth, B.: Measuring the Moment-to-Moment Variability of Tinnitus: The TrackYourTinnitus Smart Phone App. Frontiers in Aging Neuroscience **8** (2016) 294–294

[15] Probst, T., Pryss, R., Langguth, B., Rauschecker, J., Schobel, J., Reichert, M., Spiliopoulou, M., Schlee, W., Zimmermann, J.: Does Tinnitus Depend on Time-of-Day? An Ecological Momentary Assessment Study with the "TrackYourTinnitus" Application. Frontiers in Aging Neuroscience **9** (2017) 253–253

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

[17] myKind - Universität Ulm. https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/mykind/ (2018) [Online; abgerufen am 28.4.2018].

[18] Mobile Health Monitoring Assists Healthy Pregnancy | SAP Blogs. https://blogs.sap.com/2015/03/05/mobile-health-monitoring-assists-healthy-pregnancy/ (2015) [Online; abgerufen am 28.4.2018].

[19] Meet Android Studio | Android Developers. https://developer.android.com/studio/intro/ (2018) [Online; abgerufen am 28.4.2018].

[20] Configure Your Build | Android Developers. https://developer.android.com/studio/build/ (2018) [Online; abgerufen am 28.4.2018].

[21] App Manifest Overview | Android Developers. https://developer.android.com/guide/topics/manifest/manifest-intro (2018) [Online; abgerufen am 28.4.2018].

[22] Künneth, T.: Android 7: Das Praxisbuch für Entwickler. Rheinwerk Verlag (2017)

[23] Activity | Android Developers. https://developer.android.com/reference/android/app/Activity (2018) [Online; abgerufen am 28.4.2018].

[24] Introduction to Activities | Android Developers. https://developer.android.com/guide/components/activities/intro-activities (2018) [Online; abgerufen am 28.4.2018].

[25] Understand the Activity Lifecycle | Android Developers. https://developer.android.com/guide/components/activities/activity-lifecycle (2018) [Online; abgerufen am 28.4.2018].

[26] Intent | Android Developers. https://developer.android.com/reference/android/content/Intent (2018) [Online; abgerufen am 28.4.2018].

[27] Intents and Intent Filters | Android Developers. https://developer.android.com/guide/components/intents-filters (2018) [Online; abgerufen am 28.4.2018].

[28] Retrofit. http://square.github.io/retrofit/ (2018) [Online; abgerufen am 28.4.2018].

[29] JSON. https://www.json.org/json-de.html (2018) [Online; abgerufen am 28.4.2018].

[30] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis (2000) AAI9980887.

[31] Was ist eine REST API? https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/ (2017) [Online; abgerufen am 28.4.2018].

[32] API Documentation. https://tyt.johannesschobel.com/dingodocs/v1.html (2018) [Online; abgerufen am 28.4.2018].

[33] Ullenboom, C.: Java ist auch eine Insel: Einführung, Ausbildung, Praxis. Galileo Press (2014)

[34] Langer, A., Kreft, K.: Java Generics - Einführung. http://www.angelikalanger.com/Articles/JavaMagazin/Generics/GenericsPart1.html (2004) [Online; abgerufen am 28.4.2018].

[35] Save data using SQLite. https://developer.android.com/training/data-storage/sqlite (2018) [Online; abgerufen am 28.4.2018].

[36] Save key-value data. https://developer.android.com/training/data-storage/shared-preferences (2018) [Online; abgerufen am 28.4.2018].

[37] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software. Addison-Wesley (2007)

[38] Patel, K.: Singleton Pattern. https://medium.com/@kevalpatel2106/digesting-singleton-design-pattern-in-java-5d434f4f322 (2018) [Online; abgerufen am 28.4.2018].

[39] ListAdapter. https://developer.android.com/reference/android/widget/ListAdapter (2018) [Online; abgerufen am 28.4.2018].

Abbildungsverzeichnis

3.1 QuestionSys Übersicht [11] 14
3.2 TrackYourTinnitus Screenshots 15

5.1 Genereller Ablauf 26
5.2 Vererbung von AppCompatActivity 31
5.3 Activity Lebenszyklus [25] 34
5.4 JSON Objekt [29] 39

7.1 Registrierung 53
7.2 Login Screen 55
7.3 An Studie teilnehmen 57
7.4 Fragebogen starten 58
7.5 Fragebogenansicht 59
7.6 Feedbackansicht Patientin 60
7.7 Patientinnendetails 61
7.8 Feedback freigeben 62
7.9 Passwort ändern 63
7.10 Entwickler kontaktieren 64
7.11 Informationen anzeigen 65
7.12 Sprache einstellen 66

Tabellenverzeichnis

2.1 Verschiedene Ansätze mobiler App Entwicklung nach [9] 12

4.1 Übersicht der Anforderungen 24

8.1 Anforderungsabgleich 68

Name: Jochen Gindele Matrikelnummer: 888102

Erklärung

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

Ulm, den .....

Jochen Gindele