ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Konzeption einer mobilen Anwendung für therapeutische Interventionen unter Verwendung der Google Awareness API

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Simon Merkel
simon.merkel@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Marc Schickler

2018

Fassung 8. April 2018

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Therapeutische Hausaufgaben verbessern das Ergebnis einer Therapie deutlich und kommen daher immer öfters als therapeutische Intervention zum Einsatz. Die Kombination mit mobilen Geräten soll den Patienten bei der Ausführung unterstützen und den Therapeuten wichtige Daten überliefern. So wird die Therapie effizienter und erzielt ein noch besseres Resultat. Ein IT-System zur Planung und Gestaltung von therapeutischen Hausaufgaben wäre sehr nützlich, bisher gibt es jedoch keine zufriedenstellende Lösungen. Das System soll eine schnelle Anpassung von Hausaufgaben und einen verbesserten Informationsaustausch zwischen Therapeut und Patienten schaffen. Um für den Therapeuten wichtige Patientendaten zu sammeln, sollen die Sensoren von intelligenten mobilen Geräten eingesetzt werden.

In dieser Arbeit wird eine Anwendung für Android Smartphones entwickelt, die den Patienten bei seinen Übungsaufgaben unterstützen soll. Im richtigen Kontext wird der Patient an seine Übungen erinnert und wird medial bei der Ausführung angeleitet. Als Kontext wird der Ort, die Zeit, das Wetter und die Aktivität des Patienten unter anderem mittels der Google Awareness API erfasst. Am Ende jeder Übung muss der Benutzer das individuelle Feedback ausfüllen, welches dann an den Therapeuten geschickt werden kann. Die Anwendung steht bislang getrennt von einem zuvor geschilderten IT-System, hat jedoch das Ziel, ein Teil eines solchen Systems zu sein.

Danksagung

An dieser Stelle möchte ich meiner Freundin Leoni danken, die mich während der ganzen Zeit unterstützt, motiviert und hilfreiches Feedback gegeben hat. In stressigen Zeiten hat sie Ruhe und frischen Wind gebracht.

Auch möchte ich meinem Betreuer Marc danken, der sich immer für mich Zeit genommen hat, falls ich Fragen hatte und mir zu jedem Zeitpunkt wichtiges Feedback und neue Anregungen gegeben hat.

Zuletzt möchte ich meiner Familie danken, die sich für mein Thema begeistert hat und immer den aktuellen Fortschritt sehen und testen wollte. Ich danke ihnen für ihre Geduld und die tollen Rückmeldungen zu meiner Arbeit.

Inhaltsverzeichnis

1

Einleitung

Übungsaufgaben für zuhause, die Patienten selbständig ausführen sollen, nehmen immer stärker an Bedeutung in Therapien zu [1]. Einige Studien belegen den Mehrwert von therapeutischen Hausaufgaben und deren positive Auswirkung auf die Therapie [2]. Doch es gibt auch einige Schwierigkeiten, die auftreten können. Patienten müssen sich zwischen den Therapiestunden Zeit nehmen, um die Aufgaben zu erledigen. Zudem müssen sie sich an die korrekte Ausführung erinnern, damit es nicht zu Fehlern kommt. Es ist daher zwingend notwendig, dass sich Therapeut und Patient gut miteinander absprechen und der Patient die richtige Unterstützung bekommt [3].
Da es eine große Vielfalt an Hausaufgaben gibt, ist ein IT-System zur Unterstützung für Therapeuten sinnvoll. Durch ein solches System sollen Therapeuten die Hausaufgaben personalisieren können und so die Bedürfnisse des Patienten besser stillen. Das hat eine effizientere und wirksamere Therapie zur Folge. Schon zwischen den Therapiestunden sollen die Übungen angepasst werden können. Die Anpassungen sollen auf Grund der Daten von intelligenten, mobilen Geräten der Patienten geschehen. Diese Geräte sollen die Patienten nicht nur an die Ausführung von Übungen erinnern und dabei unterstützen, sondern auch noch wichtige Daten mittels den integrierten Sensoren sammeln [4].
Verschiedene Studien unterstützen den Einsatz von mobilen Geräten in Verbindung mit Therapien und betonen deren Vorteil. Sie erhöhen nicht nur den Kontakt zwischen Patient und Therapeut, sondern liefern Erinnerungen und Unterstützung für Hausaufgaben. Insgesamt können mobile Anwendungen dazu beitragen, Fähigkeiten zu erlernen und zu üben, um die Gesundheit von Patienten zu fördern [5].

1.1 Zielsetzung

Das Ziel der Arbeit ist es, einen Prototypen für eine mobile Anwendung zu erstellen, die Patienten bei therapeutischen Hausaufgaben unterstützt. Die Anwendung soll dabei ein Teil eines komplexen IT-Systems für die Gestaltung von therapeutischen Interventionen sein. Patienten können die Anwendung auf ihrem Android Smartphone nutzen und bekommen die benötigte Unterstützungen für ihre therapeutischen Übungsaufgaben zwischen den Therapiestunden. Neben einer Erinnerung an die Hausaufgabe, Anleitung bei der Ausführung der Übung und dem Ausfüllen des Feedbacks durch den Patienten sollen auch Kontext-Daten erfasst werden.
Es soll eine Anwendung geschaffen werden, die neben der Benutzerfreundlichkeit auch das Ziel hat, die Sensoren eines mobilen Gerätes zu nutzen und sinnvoll in die Anwendung zu integrieren. Um den Kontext über verschiedenste Sensoren zu erkennen, sollen die Vorteile der Google Awareness API genutzt werden. Die gesammelten Informationen sollen dazu beitragen, dass der Patient seine Hausaufgaben stets im richtigen Kontext ausführt. So sollen die Aufgaben beispielsweise zum richtigen Zeitpunkt und am richtigen Ort ausgeführt werden. Auch sollen die Wetterverhältnisse und die Aktivität des Patienten als Kontext erfasst werden. Die Ausführung der Übungen im richtigen Kontext führt zu einem besseren Therapieergebnis und kann wertvolle Zusatzinformationen an den Therapeuten liefern. Auch soll das wichtige Feedback zu jeder Übung ausgefüllt werden. Der Patient soll dies direkt nach einer Übung tun, um die direkte Erfahrung aufzunehmen. Therapeuten können durch das Feedback die Effizienz ihrer Therapie weiter steigern und Hausaufgaben gegebenenfalls anpassen. Um dem Patienten eine erleichterte Nutzung der Anwendung anzubieten, sollen körperliche Aktivitäten, die in den Hausaufgaben gefordert sind, automatisch erkannt werden. So soll mittels den verschiedener Sensoren die Aktivität erkannt und mit den offenen Hausaufgaben abgeglichen werden.
Bei bisherigen IT-Systemen zur Verwaltung von therapeutischen Hausaufgaben kommt der Einsatz von mobile Geräten oft zu kurz. Oft werden die integrierten Sensoren nicht mit einbezogen [6]. In dieser Arbeit soll ein mobiler Prototyp entwickelt werden, der die integrierten Sensoren nutzt und Teil eines solchen IT-Systems sein kann.

1.2 Struktur der Arbeit

Nach dieser Einleitung, soll eine Grundlage für die Arbeit geschaffen werden. Daher werden in Kapitel 2 die wichtigsten Begrifflichkeiten geklärt. Es wird genauer erklärt, was therapeutische Interventionen und die damit zusammenhängenden Hausaufgaben sind. Um ein Verständnis für die Funktionen der Anwendung zu bekommen, werden in Kapitel 3 die Anforderungen an den Prototypen definiert. In Kapitel 4 wird der darauf basierende Entwurf vorgestellt. Es wird die Idee der Anwendung erläutert und einige Mockups und das Farbschema zur Oberflächengestaltung vorgestellt. Auch wird in diesem Kapitel das Datenbankmodell für die Umsetzung aufgestellt und erklärt. In Kapitel 5 wird die konkrete Umsetzung vorgestellt. Dabei wird etwas über die verwendete Plattform Android, die eingesetzten Frameworks und insbesondere über die Funktionsweise der Google Awareness API geschrieben. Die konkrete Implementierung wird dann in Oberfläche und Hintergrundaktivität aufgeteilt. Der Abgleich der geplanten und umgesetzten Funktionen wird in Kapitel 6 vorgenommen. Zum Schluss wird in Kapitel 7 eine kurze Zusammenfassung und ein Ausblick gegeben.

2

Grundlagen

In diesem Kapitel werden wichtige Begrifflichkeiten geklärt. So soll eine Grundlage für den Entwurf und die Umsetzung des Prototypen geschaffen werden. Zunächst werden therapeutische Interventionen erklärt, danach werden die damit zusammenhängenden therapeutischen Hausaufgaben und ihre Bestandteile erläutert.

2.1 Therapeutische Interventionen

Therapeutische Interventionen oder auch therapeutische Eingriffe beziehungsweise therapeutische Behandlungen sind ein wichtiger Bestandteil von Therapien. Interventionen können von einfachen Mediakmenteinnahmen bis hin zu einer komplexen Behandlung mit Hausaufgaben variieren [7]. Der für dieses Projekt relevante Teil sind die aufwändigeren Therapien, die Übungsaufgaben für Patienten erfordern. Ein möglicher Ablauf einer Therapie wird im Folgenden anhand der Physiotherapie beschrieben.
Zunächst führt der Therapeut die Anamnese an seinem Patienten durch. Dabei wird neben dem aktuellen Beschwerden auch auf frühere Erkrankungen eingegangen. Mit den gewonnenen Erkenntnissen wird anschließend eine Untersuchung vorgenommen. Mittels spezieller Tests bei der Untersuchung kann der Therapeut eine Diagnose und die damit verbundenen Therapieziele erstellen. Nun besprechen Therapeut und Patient gemeinsam eine daraus folgende Behandlung. Dabei werden klinische Erfahrungen und wissenschaftliche Kenntnisse zu Rate gezogen. Die Behandlung wird individuell auf den Patienten abgestimmt. Auch wird der Patient auf das Problem hin informiert, beraten und geschult. In der Physiotherapie gehören auch meist Übungsaufgaben für Zuhause zur Behandlung [8].

Verschiedene Untersuchungen belegen den positiven Effekt von Hausaufgaben auf das Ergebnis einer Therapie [2]. Allerdings kann es bei der Ausführung durch den Patienten immer wieder zu Schwierigkeiten kommen. Ungenaue Beschreibungen des Therapeuten beziehungsweise eine falsche Erinnerung des Patienten können zu einer veränderten, falschen, unvollständigen oder keiner Ausführung der Übungen kommen. Um einen Erfolg durch die Hausaufgaben zu erzielen, muss ein guter Weg zwischen Einfachheit und Aufwand gefunden werden [3].
Um Therapeuten und Patienten bei der Gestaltung von Therapien und den dazugehörigen Hausaufgaben zu helfen und den Prozess zu optimieren kann ein technisches System unterstützen. Ein IT-System könnte wie in Abbildung 2.1 aussehen.

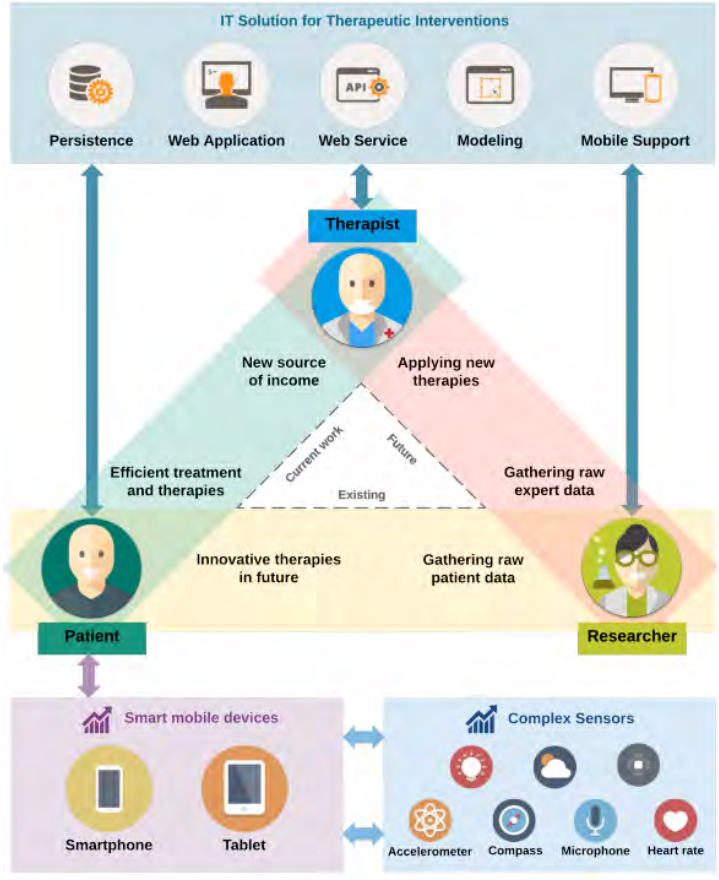

Abbildung 2.1: IT Plattform zur Gestaltung von therapeutischen Interventionen [7]

In dem grafisch dargestellten System fällt auf, dass eine klassische Therapeut-Patient-Beziehung durch einen Wissenschaftler erweitert wird. Dieser soll die verfügbaren Daten der Therapie analysieren und so die Wirksamkeit der Behandlung steigern. Um an die benötigten Daten zu gelangen, sollen intelligente mobile Geräte vom Patienten genutzt werden. Die Geräte sollen den Patienten bei dem Umgang mit Hausaufgaben unterstützen. Neben Erinnerungen und medialen Erklärungen soll die Übung so auch im passenden Kontext (vom Therapeut definiert) ausgeführt werden. Sensordaten jeglicher Art sollen mit einbezogen werden und Wissenschaftler bei der Analyse wichtige Daten liefern. Auch soll der Therapeut zu den Hausaufgaben Feedback bekommen. Das Feedback soll nun nicht mehr erst bei der nächsten Therapiesitzung gegeben werden, sondern sofort nach der Übung. Alle Daten werden in einer gemeinsamen Datenbank gespeichert. So kann der Therapeut dort Übungen anlegen, bei Bedarf Änderungen vornehmen und das gesendete Feedback ablesen. Der Patient bekommt dann die aktuellen Daten auf sein Smartphone, wird bei der Ausführung im passenden Kontext unterstützt und schickt das Feedback zurück. Der Wissenschaftler kann die vorliegenden Daten analysieren und auswerten und somit die Therapie verbessern [7].
In der vorgestellten Arbeit wurde ein Prototyp für den Teil des Patienten entwickelt. Dabei steht das Projekt erst mal unabhängig von Therapeut und Wissenschaftler. Hausaufgaben und Kontextdaten werden daher nicht von einer externen Datenbank gelesen sondern liegen lokal in einer fest definierten JSON-Datei auf dem Smartphone vor. Das Feedback wird nicht an die Datenbank zurück geschickt, sondern wird sofort wieder verworfen.

2.2 Therapeutische Hausaufgaben

Therapeutische Hausaufgaben sind längst keine Neuheit mehr und nehmen laut Umfragen als therapeutische Intervention der Psychotherapie am Stärksten an Bedeutung zu. Der Begriff ist durch die Schulzeit oft negativ geprägt und wird oft durch ’Therapieaufgabe’ oder ’Übungsaufgabe’ ersetzt. Sie werden als Aufgaben definiert, die der Klient außerhalb des Therapiezimmers zwischen Therapiesitzungen durchführt, um das in der Therapie Gelernte einzuüben und zu vertiefen, auf seinen konkreten Lebensbereich zu

übertragen oder Beobachtungsmaterial für die nächste Therapiesitzung zu sammeln [1]. Die Bestandteile von Hausaufgaben in Verbindung mit mobilen Geräten wird in Abbildung 2.2 veranschaulicht. Auf dem ersten Level sieht man eine festgelegte Reihe von Hausaufgaben, die einer therapeutischen Intervention angehören. Das zweite Level zeigt die drei Bestandteile einer Hausaufgabe: Benachrichtigung, Übung und Feedback. Zu jeder Hausaufgabe werden diese drei Bestandteile in der genannten Abfolge ausgeführt. Das dritte Level ist aufgeteilt in zwei Punkte. Zum einen ist das der Teil, der vom Patienten ausgeführt werden muss (zum Beispiel: Übung durchführen, Feedback ausfüllen) und zum anderen ist das ein automatisierter Teil (zum Beispiel: Kontext erfassen) [4].

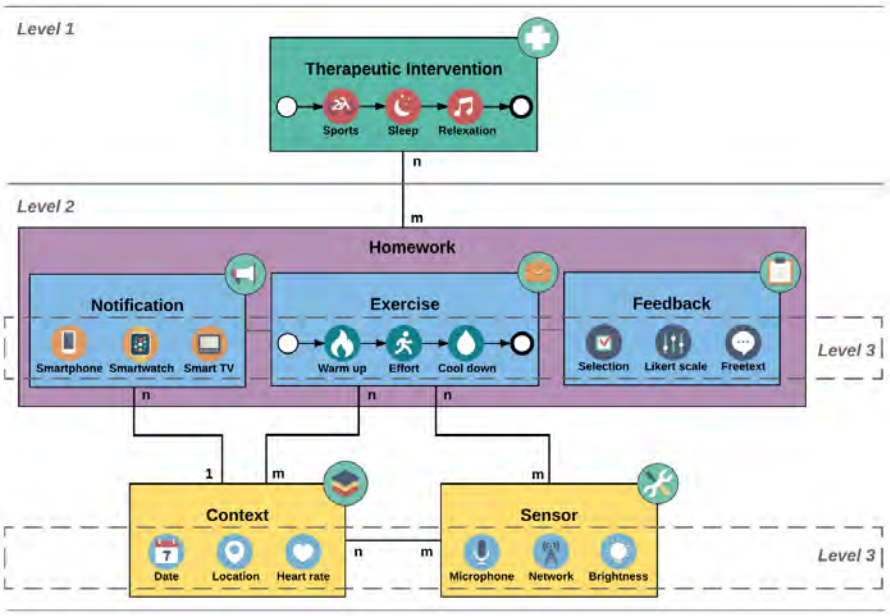

Abbildung 2.2: Schachtelung von Hausaufgaben mit Smartphones [4]

Im nachfolgenden werden die Bestandteile einer Hausaufgabe genauer erklärt.

2.2.1 Benachrichtigung

Die Benachrichtigung ist der erste Schritt einer Hausaufgabe. Ist der definierte Kontext einer Hausaufgabe erfüllt, so bekommt der Patient eine Aufforderung zur Ausführung der Übung auf sein mobiles Gerät. Ein Beispiel: *Der Patient soll Zuhause um 12:00 Uhr eine*

Dehnübung ausführen. In diesem Szenario wird von dem mobilen Gerät der Standort und die Uhrzeit überwacht. So kann beispielsweise immer um 12:00 Uhr der Standort abgefragt werden. Ist der Patient zu diesem Zeitpunk Zuhause, so bekommt er eine Benachrichtigung. Dies leitet die Ausführung der Übung ein.

2.2.2 Übung

Nachdem ein Patient mittels einer Benachrichtigung über eine anstehende Übung im korrekten Kontext informiert wurde, wird nun das Kernstück einer Hausaufgabe ausgeführt. Eine Übung besteht aus mehreren Schritten, die in einer Reihenfolge ausgeführt werden sollen. Die Übung kann dabei in mehrere Phasen aufgeteilt werden. Beispielsweise kann es eine Aufwärmphase, die eigentliche Dehnübung und danach eine Entspannungsphase geben. Während der kompletten Übung können Sensoren eingesetzt werden um Herzschlag oder andere Werte des Patienten zu überwachen. Diese Werte können nach der Ausführung an den Therapeuten geschickt werden, der diese kontrolliert und interpretiert. Die Übung soll passend mit Bildern oder Videos erklärt werden, um die Ausführung für den Patienten zu erleichtern [4].

2.2.3 Feedback

Nach einer Übung folgt der letzte Schritt einer Hausaufgabe: das Feedback. Dies soll direkt nach der Ausführung der Übung ausgefüllt werden, um keine Erinnerungslücken zu bekommen. Für das Feedback werden individuell zu jeder Hausaufgabe vom Therapeuten Fragen gestellt. Antwortmöglichkeiten sollen dem Patienten in verschiedenen Formen präsentiert werden. Texteingaben, Auswahlmöglichkeiten oder Skalen können dabei zum Einsatz kommen.

2.3 Kontext

Der Kontext spielt bei der Ausführung von therapeutischen Hausaufgaben eine große Rolle. Dieser wird an zwei Stellen berücksichtigt, wie schon in Abbildung 2.2 zu sehen

ist. Zum einem wird der Kontext bei den Benachrichtigungen einbezogen. So wird sichergestellt, dass die Übungen immer im vorhergesehenen Kontext ausgeführt werden und nicht ihre Effektivität verlieren [7]. In dieser Hinsicht kann die Zeit, der Ort, Wetterverhältnisse oder gar der Puls des Patienten eine Rolle spielen. Vom Therapeut muss zu jeder Hausaufgabe ein entsprechender Kontext definiert werden. Intelligente mobile Geräte können dann speziell den gewünschten Kontext erfassen und den Patienten benachrichtigen, sobald er sich in der richtigen Umgebung befindet. Ein Szenario könnte wie folgt lauten: *Der Patient soll um 12:00 Uhr bei schönem Wetter, falls er zuhause ist, einen Spaziergang machen*. Das mobile Gerät erfasst bei diesem Beispiel im Hintergrund als Kontextdaten das Wetter, den Ort und die Zeit. Sobald es zwölf Uhr, das Wetter schön und der Patient zuhause ist, wird der Patient an die Ausführung erinnert.

Zum Anderen wird während der Ausführung einer Übung der Kontext erfasst. Hier bedeutet dies, dass wertvolle Patienteninformationen aktiv gesammelt werden. So können beispielsweise Dauer, Puls und Ort während einer Hausaufgabe überwacht werden. Das obige Beispiel kann wie folgt erweitert werden: *Während des Spaziergangs soll der Puls des Patienten aufgezeichnet werden. Die Dauer und die zurückgelegte Strecke sollen zusätzlich erfasst werden*. Nun werden nach dem Start der Übung alle verfügbaren Sensoren dazu genutzt, um die gewünschten Informationen zu sammeln. Dazu können auch Daten von verbundenen Bluetooth Geräten wie beispielsweise einer Smartwatch oder eines Fitnessbandes hilfreich sein. Diese wertvollen Informationen werden dann gemeinsam mit dem Feedback an den Therapeuten geschickt. Dieser kann die erhaltenen Daten analysieren und die Übung gegebenenfalls anpassen. Auch können diese Daten für Wissenschaftler sehr wertvoll sein, um neue Erkenntnisse zu gewinnen. Das Sammeln von Kontextinformation mit mobilen Geräten bringt neben der Zeit- und Kostenersparnis auch eine höhere Qualität und Vollständigkeit der Daten [9]. Bei bisherigen Ansätzen werden die Sensoren der Geräte jedoch nicht ausreichend genutzt [6].

Da es jedoch viele verschiedene Kontexttypen gibt, muss die Anwendung flexibel sein. Dazu zählt auch schon die Erstellung von Umgebungsinformationen für die Hausaufgaben. Ein Therapeut soll auch ohne technischen Hintergrund den passenden Kontext auf

einer grafischen Oberfläche zusammenstellen können. Das daraus entstandene Kontextmodell muss dann auf das Smartphone übertragen werden können. Die einzelnen Teile des Modells müssen unabhängig und nur wenn sie benötigt werden ausgeführt werden können. Dazu muss für die Anwendung eine Anbindung der Sensoren durch passende Frameworks gewährleistet sein [10]. In dem Prototyp wurde dazu die Google Awareness API verwendet.

3

Anforderungsanalyse

In dem folgenden Kapitel wird die Anforderungsanalyse durchgeführt. Die Analyse ist aufgeteilt in funktionale und nicht-funktionale Anforderungen. Somit soll definiert werden, was der Prototyp leisten soll. Basierend auf dieser Analyse wurde ein Entwurf erstellt. Auch sind diese Anforderungen Grundlage für die konkrete Umsetzung.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben, was die Anwendung leisten soll. Dafür wird definiert, was der Prototyp in bestimmten Situation tun soll, was im Hintergrund geschieht und welche Benutzerinteraktionen es geben soll.

Nr.	Anforderung	Beschreibung
1	Benachrichtigung über anstehende Übung	Die Benachrichtigung soll den Benutzer über eine anstehende Übung informieren. Es werden der Titel der Übung und der Ausführungszeitpunkt genannt. Die Benachrichtigung soll entsprechend dem Kontext erscheinen. Der Benutzer wird also nur über Übungen passend zur Uhrzeit, Ort und Wetter informiert.
2	Kontext-Erkennung	Die Anwendung erkennt den jeweiligen Kontext. Es wird die Uhrzeit, der Ort, das Wetter und die Aktivität (Gehen, Rennen, Radfahren und keine Aktivität) erkannt.

3	Anweisungen für Übung mit Video, Bild und Text	Die Übung soll dem Benutzer mit einem Video oder einem Bild erklärt werden. Durch die einzelnen Übungsschritte soll der Benutzer mit Text und einem speziell für diesen Schritt passendem Bild geführt werden.
4	Übungen passend zum Kontext anzeigen	Auf der Startseite soll dem Benutzer nur die Übungen angezeigt werden, die zum Kontext passen. Die aktuelle Aktivität soll dabei nicht berücksichtigt werden.
5	Feedback zu jeder Übung	Jede Übung hat ein individuelles Feedback, bestehend aus einer Frage und einem Bewertungstyp (Text, Ja/Nein, Skala). Der Benutzer soll nach jeder Durchführung das Feedback vollständig ausfüllen, welches an seinen Therapeuten geschickt wird.
6	Übungen stumm schalten	Die Übungen auf der Startseite sollen vom Benutzer stumm geschaltet werden können. Dies bedeutet, dass die Übung aus der Liste der offenen Übungen auf der Startseite verschwindet. Die Übung ist bis zum nächsten Ausführungszeitpunkt stumm geschaltet. Auch über die Benachrichtigung soll der Benutzer die Übung stumm schalten können.

Tabelle 3.1: Funktionale Anforderungen des Prototyps

3.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben im Gegensatz zu den funktionalen Anforderungen nicht die Funktionsweise, sondern die Qualität der Anwendung. Dabei

werden keine neue Funktionalitäten aufgestellt, sondern die Rahmenbedingungen für die eigentlichen Funktionen des Prototyps definiert.

Nr.	Anforderung	Beschreibung
1	Material Design	Die Oberfläche der kompletten Anwendung soll nach den Material Design Styleguides gerichtet sein.
2	Batteriesparend	Die Anwendung soll die Akkulaufzeit nicht sonderlich beeinträchtigen. Besonders die Hintergrundaktivitäten der Anwendungen sollen so gering und so effizient wie möglich gehalten werden.
3	Usability	Die Anwendung soll von Patienten mit verschiedensten Einschränkungen und in allen Situationen genutzt werden können. Die Anweisungen für die Übungen sollen auch während der Ausführung gut erkennbar sein. Die Anwendung gibt die nötige Unterstützung, falls diese benötigt wird, soll aber keine unnötigen Benutzerinteraktionen fordern.

Tabelle 3.2: Nicht-funktionale Anforderungen des Prototyps

4

Entwurf

In diesem Kapitel wird der Entwurf des Prototyps vorgestellt. Dazu werden zuerst die Mockups gezeigt und beschrieben, danach wird kurz das verwendete Farbschema vorgestellt. Und zum Schluss wird noch das Datenmodell definiert, auf dem der Prototyp beruht.

Der Benutzer soll die App unterstützend zu seinen Hausaufgaben benutzen. Dazu ist wichtig, dass er einen guten Überblick über alle noch zu erledigenden Übungen hat und eine detaillierte Ansicht der einzelnen Übungsschritte bekommt. Natürlich sollen die Anweisungen einer Übung auch während der Ausführung gut lesbar und verständlich sein. Um einen guten Überblick über seine Hausaufgaben zu haben, soll der Benutzer eine Liste aller anstehenden Übungen angezeigt bekommen. Die Anzeige soll dabei nur die zur Benutzerumgebung relevanten Übungen enthalten, sodass der Nutzer die Übungen nicht selbständig filtern muss und eventuell die falschen Übungen ausführt. Diese Aufgabenliste soll dem Benutzer direkt beim Start der Anwendung präsentiert werden, um unnötige Klicks zu vermeiden. Der Nutzer soll selbständig entscheiden können, wie viel Unterstützung er bekommen möchte. Bei einiger Routine können die Übungen ohne weitere Anweisungen und unnötige Navigation abgehakt werden, braucht der Nutzer jedoch noch Hilfe bei den Übungen, so werden diese mit verschiedenen Medien dargestellt. Das Feedback muss ganz gleich der Routine immer vollständig ausgefüllt werden. So weiß der Therapeut über den aktuellen Fortschritt des Patienten und den Wirkungsgrad der Übung Bescheid und kann die Therapie optimal anpassen.

4.1 Mockups

Um die Mockups zu entwerfen, wurde die Onlineanwendung NinjaMock genutzt. Mit dieser Anwendung kann man sehr einfach und schnell ein Konzept einer mobilen Anwendung zusammenstellen. Alle typischen Elemente einer Bedienoberfläche von einem mobilen Gerät sind vorhanden und können individuell angepasst werden. Man kann die erstellten Oberflächen durch definierte Klicks miteinander verbinden, um ein echtes Gefühl für die Bedienung zu bekommen. NinjaMock kann auch als Team mit Echtzeit Synchronisation verwendet werden [11].
In Abbildung 4.1 und 4.2 sieht man die Startseite. Neben einer Begrüßung wird eine Liste an offenen Übungen angezeigt. Dabei wird zunächst nur der Titel und die Uhrzeit der Ausführung angezeigt. Mit einem Klick auf die jeweilige Übung, werden weitere Details sichtbar. Unter der Dauer, dem Ort und der nächsten Ausführungszeit befindet sich die Beschreibung der Übung. Zusätzlich wird noch die Möglichkeit angeboten, die Übung direkt zu starten beziehungsweise als erledigt zu markieren.

Abbildung 4.1: Startseite nach dem Start der Anwendung

Abbildung 4.2: Startseite nach dem Klick auf eine Übung

Durch einen Klick auf ’Durchführen’ auf der Startseite gelangt der Benutzer zu der Anleitung einer Übung. In Abbildung 4.3 ist eine mögliche Anleitung dargestellt. Zunächst sieht der Benutzer ein Video, das die Übung im Allgemeinen beschreibt. Darunter sind die einzelnen Übungsschritte, die mittels Text und einem Bild genauer beschrieben werden. Ist die Übung abgeschlossen, so wird der Benutzer durch einen Klick auf ’Fertig’ zum Feedback weitergeleitet. Klickt der Benutzer auf einen Übungsschritt, so gelangt er in eine vergrößerte Ansicht, wie in Abbildung 4.4. Dort kann er mittels des Pfeiles rechts durch die einzelnen Schritte navigieren.

Abbildung 4.3: Anleitung zu einer Übung

Abbildung 4.4: Ein vergrößerter Übungsschritt

Nach dem Beenden einer Übung muss der Benutzer das Feedback ausfüllen und an seinen Therapeuten schicken. Hierfür bekommt er eine Liste an Fragen. Zur Auswahl stehen dabei verschiedene Eingabemöglichkeiten. Ist das Feedback komplett ausgefüllt, so können die Antworten mit einem Klick auf ’Abschicken’ versendet werden. Dadurch gelangt der Benutzer zurück auf die Startseite.

Dort sind nun auch alle erledigten Übungen in einer Liste enthalten wie man in Abbildung 4.6 sieht. Mit einem Klick auf die entsprechende erledigte Übung werden weitere Details angezeigt. Neben der Uhrzeit, wann die Übung erledigt wurde, wird auch eine kurze Zusammenfassung angezeigt. Der Fortschritt der Ausführung von den Übungsaufgaben wird farblich dargestellt. Alle erledigten Aufgaben werden grün markiert, alle längst überfälligen Aufgaben werden rot markiert und alle seit kurzem anstehenden Aufgaben werden gelb markiert.

Abbildung 4.5: Feedback

Abbildung 4.6: Startseite mit Verlauf

In Abbildung 4.7 sieht man eine Benachrichtigung über eine anstehende Übung. Der Benutzer wird eine halbe Stunde vor der tatsächlichen Ausführung an die Übung erinnert. Die Benachrichtigung bietet die Möglichkeit, die Übung direkt zu starten oder die Erinnerung auf später zu verschieben. Angezeigt wird die Uhrzeit der Ausführung und der Name der Übung.

Abbildung 4.7: Benachrichtigung

4.2 Farbschema

Um ein harmonierendes Farbschema zu finden, wurde zuerst ein Titelbild ausgesucht. Dazu wurde die Plattform pixabay genutzt. Dort werden mehr als 1,3 Millionen Bilder kostenlos zur Verfügung gestellt [12]. Ausgesucht wurde das Bild aus Abbildung 4.8. Mit diesem Titelbild wurde auf der Plattform coolors ein Farbschema ausgesucht.

Abbildung 4.8: Das Titelbild [13]

Coolors bietet einen extrem schnellen und einfachen Weg an, ein Farbschema zu erstellen. Es werden fünf verschiedene Farben geliefert, die miteinander harmonieren. Nachträglich kann bei den erstellten Farben noch die Sättigung, Helligkeit und weitere Parameter ganz nach eigenem Belieben geändert werden. Außerdem kann das Farbschema anhand eines Bildes gewählt werden. Dafür wird eine perfekte Farbkombination aus einem vom Benutzer bereitgestellten Bild herausgesucht. Somit passen Bild und Farbschema gut zusammen [14]. Das aus dem Titelbild entstandene Farbschema ist in Abbildung 4.9 zu sehen.

Abbildung 4.9: Das benutzte Farbschema [15]

4.3 Datenbankmodell

In diesem Abschnitt wird das Datenbankmodell beschrieben. In dem Entwurf werden die Beziehungen und die Attribute von den benötigten Objekten dargestellt. Ein Objekt ist durch ein Rechteck gekennzeichnet. Die dazugehörigen Attribute sind Ovale, die durch eine Linie mit dem Objekt verbunden sind. Stehen zwei Objekten in einer Beziehung, so wird dies mittels einer Raute, die durch Linien mit den Objekten verbunden ist, veranschaulicht. In der Raute wird die Beziehung durch ein Wort beschrieben. Diese Darstellung nennt man ein ER-Diagramm. Um die Beziehungen näher zu beschreiben wurde die Chen-Notation verwendet. Dadurch kann man die Anzahl an Verbindungen zwischen Objekten ablesen. Die Objekte sind als Tabellen einer Datenbank zu verstehen, die Attribute sind Tabellenspalten und eine Beziehung ist eine Referenzierung von einer Tabelle zur Anderen.

In den Abbildungen 4.10, 4.11 und 4.12 wird das Datenmodell grafisch dargestellt. Jede Tabelle hat dabei einen Primärschlüssel, über den jeder Tabelleneintrag eindeutig identifiziert werden kann. Dieser Schlüssel heißt in allen Tabellen *id*. Fremdschlüssel, die eine Beziehung aufzeigen, wurden nicht gesondert aufgelistet sondern werden durch die Rauten dargestellt.

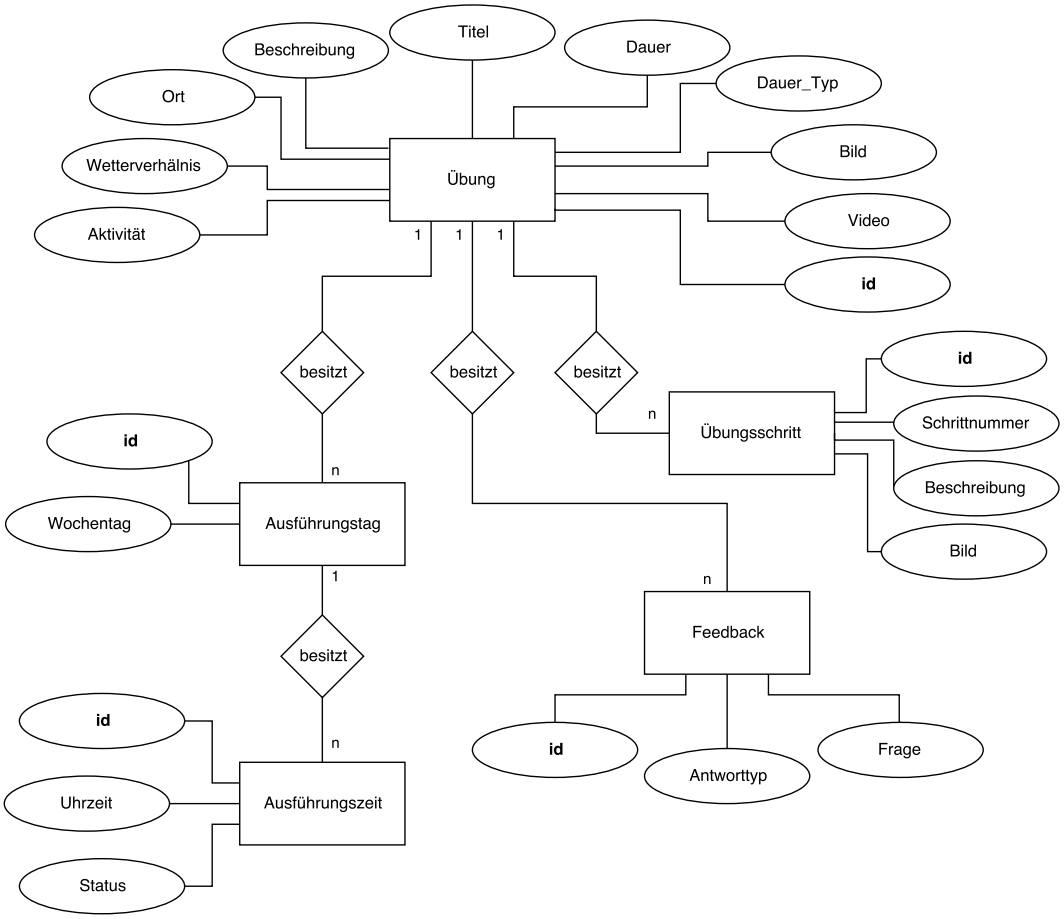

Abbildung 4.10: ER-Diagramm der Übung

Die Tabelle *Übung* besitzt sehr viele Attribute. *Titel* und *Beschreibung* erklären sich selbst. Bei *Ort* soll eingetragen werden, ob die Übung zuhause, bei der Arbeit oder unabhängig vom Ort ausgeführt werden soll. Mit *Wetterverhältnis* ist gemeint, ob die Übung bei schönem, schlechtem oder unabhängig vom Wetter stattfinden soll. Mit *Aktivität* ist

die Benutzeraktivität (Radfahren, Rennen, Gehen, Ohne Aktivität) währen der Übung gemeint, die später vom Smartphone automatisch erkannt werden soll. Bei der *Dauer* wird eine Zahl eingetragen. Mit dem Feld *Dauer_Typ* wird dann unterschieden, ob diese Zahl die Anzahl der Wiederholungen oder die Zeit in Minuten wiedergibt. Bei *Bild* und *Video* kann der jeweilige Speicherort eingetragen werden.
Die Tabelle *Übung* kann beliebig viele Verbindungen zu *Ausführungstag* haben. Dort wird der *Wochentag* gespeichert. An jedem *Ausführungstag* kann es wiederum beliebig viele Verbindungen zu *Ausführungszeit* geben. Dort wird die *Uhrzeit* und der *Status* gespeichert. Im Status wird festgehalten, ob die Übung zu dieser Uhrzeit an diesem Tag noch offen, erledigt oder stumm geschaltet wurde. Durch die Verbindungen von *Übung* über *Ausführungstag* zu *Ausführungszeit* wird realisiert, dass eine Übung an mehreren Tagen und an diesen Tagen zu mehreren Zeiten ausgeführt werden kann. Jede Übung ist somit nicht auf eine festgelegte Anzahl an Tagen oder Zeiten festgelegt.
In der Tabelle *Feedback* wird jeweils die *Frage* und der *Antworttyp* gespeichert. Durch den *Antworttyp* kann unterschieden werden, ob der Benutzer mit Text, Ja/Nein oder einer Skala antworten soll. Die Tabelle *Übung* kann beliebig viele Verknüpfungen zur Tabelle *Feedback* haben.
In der Tabelle *Übungsschritt* sollen alle relevanten Informationen für einen Schritt der Übung gespeichert werden. Wiederum gilt, dass eine *Übung* beliebig viele *Übungsschritte* haben kann.

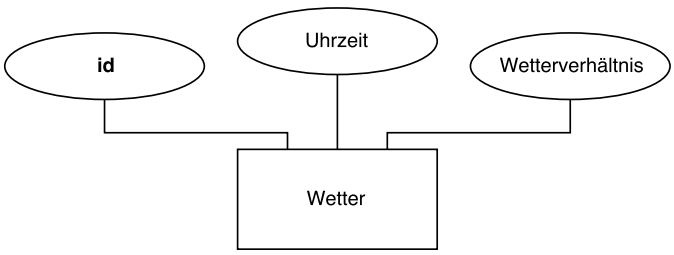

Abbildung 4.11: ER-Diagramm des Wetters

Die Tabelle *Wetter* steht unabhängig von den anderen Tabellen und es wird nur die *Uhrzeit*, zu der das Wetter gilt und das *Wetterverhältnis* gespeichert.

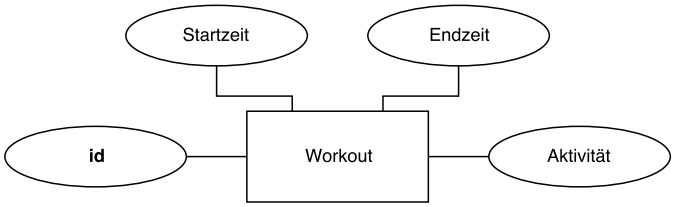

Abbildung 4.12: ER-Diagramm eines Workouts

Auch die Tabelle *Workout* steht unabhängig zu allen anderen Tabellen. Es wird die *Startzeit*, die *Endzeit* und die *Aktivität* (Radfahren, Rennen, Gehen) gespeichert. Bei einem noch nicht beendeten Workout wird die Endzeit mit Null markiert.

4.4 Übungen

Um den Prototyp mit praxisnahen Übungen zu füllen, wurde die Online-Plattform *joggen-online.de* als Basis für die ausgewählten Übungen genutzt. Auf dieser Seite gibt es Tipps und Ausstattung zu jeglichen Sportarten und eine Beratung für gesunde Ernährung [16]. In dem Prototyp werden die Bilder und die Beschreibung von dieser Seite für Dehnübungen eines Lauftrainings genutzt. In den Screenshots dargestellte Bilder zu den einzelnen Übungen sind auf dieser Webseite zu finden. Auch sind die Beschreibungen in gekürzter Form übernommen. Der Kontext für jede Übung wurde nach freiem Belieben und für gute Testbedingungen selbst erfunden. So wurde Ort, Ausführungszeiten, Aktivität und Wetter eigenständig hinzugefügt, um die volle Funktionalität des Prototypen besser testen zu können.
Alle Übungen werden in einer statischen JSON-Datei in dem Smartphone abgelegt. Die dazugehörigen Bilder liegen ebenfalls auf dem Gerät. Einen Ausschnitt der Datei sieht man in dem Codebeispiel A.1.

5

Implementierung

Dieses Kapitel befasst sich mit der Implementierung des Prototypen. Zunächst wird die Plattform Android vorgestellt, danach werden alle verwendeten Frameworks beschrieben, dabei wird besonders auf die Google Awareness API eingegangen. Am Ende wird dann noch die konkrete Umsetzung beschrieben.

5.1 Android

Die Anwendung wurde für die Plattform Android entwickelt. Android ist ein Linux-basiertes Open-Source Betriebssystem [17]. Es kommt auf Smartphones, Smartwatches, Tablets, Fernseher und in Autos zum Einsatz [18]. 2016 war es mit 87,5% des Marktanteils das meistgenutzte Betriebssystem für Smartphones [19]. Android wird von Google entwickelt, kann und wird aber von jedem Smartphonehersteller nach eigenen Vorstellungen verändert. Seit 2008 wurden 24 verschiedene Android-Versionen veröffentlicht. Die meist Verbreitetsten sind dabei die Versionen 6 (Marshmallow) und 7 (Nougat), wie man in Tabelle 5.1 sieht. Die hauseigenen Apps von Google sind meist vorinstalliert, können aber auch jederzeit deaktiviert werden. Weitere Apps kann man sich über den Google Play Store installieren [20]. Zurzeit (März 2018) hat man im Play Store ein Auswahl von über 3,6 Millionen Apps [21]. Die Google-Play-Dienste sind ein in Android enthaltenes Programmiergerüst, das die Arbeit der Entwickler sehr vereinfacht. Mit diesen Diensten lassen sich in Anwendungen ganz leicht Landkarten einfügen oder eine Bestenliste in Spielen bereitstellen [20].
Eine Androidanwendung wird in Java programmiert. Für eine erleichterte Entwicklung gibt es die speziell für Android bereitgestellte Entwicklungsumgebung Android Studio.

Version	Codename	API	Anteil
2.3.3 - 2.3.7	Gingerbread	10	0,3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0,4%
4.1.x	JellyBean	16	1,7%
4.2.x		17	2,6%
4.3		18	0,7%
4.4	KitKat	19	12,0%
5.0	Lollipop	19	12,0%
5.1		21	5,4%
6.0	Marshmallow	23	28,1%
7.0	Nougat	24	22,3%
7.1		25	6,2%
8.0	Oreo	26	0,8%
8.1		27	0,3%

Tabelle 5.1: Android Versionen mit dem aktuellen Marktanteil (05.02.2018). Version mit einem Anteil unter 0.1% wurden weggelassen [22].

Besonders der View-Builder erleichtert die Erstellung und Gestaltung der Oberflächen und erstellt automatisch alle Verknüpfungen zu den dazugehörigen Controllern.
Eine reguläre Anwendung in Android besteht aus verschiedenen Activities. Eine Activity besteht aus einer Oberfläche und läuft im Vordergrund. Die Benutzereingaben werden dort direkt verarbeitet. Ereignisse, die im Hintergrund bei der normalen Benutzung des Smartphones auftreten, können mit einem sogenannte Receiver verarbeitet werden. So kann beispielsweise das Verbinden mit einem WLAN-Netzwerk oder der beendete Bootvorgang des Smartphones verwertet werden. Receiver müssen im System registriert werden, damit sie im richtigen Augenblick aufgerufen werden können und der jeweilige Code ausgeführt wird. Von einem Receiver aus können Vordergrundaktivitäten gestartet werden. Will man im Hintergrund Dinge ausführen ohne dass eine Benutzerinteraktion nötig ist, so kann ein Service benutzt werden. Diese Services laufen im Hintergrund solange genügend Ressourcen zur Verfügung stehen.

Die hier vorgestellte Anwendung beinhaltet sehr viele Ereignisse die im Hintergrund geschehen. Um beispielsweise Updates von der Aktivität oder dem Standort des Nutzers zu bekommen wurden Receiver benutzt. Diese verarbeiten und speichern die empfangenen Umgebungsdaten und benachrichtigen gegebenenfalls den Nutzer.

5.2 Verwendete Frameworks

In diesem Kapitel werden die in der Anwendung verwendeten Frameworks vorgestellt. Da die Google Awareness API eine besondere Rolle gespielt hat, wird diese in einem eigenen Abschnitt genauer beschrieben.

Design

Dieses Framework bietet einige Hilfen an, um das Material Design umsetzen zu können. Das Material Design ist ein Gestaltungsvorschlag für jegliche Arten von Anwendungen, um eine einheitliches Benutzererlebnis zu schaffen, das nicht von der Plattform oder der Bildschirmgröße abhängt. Die Komponenten des Designs leiten sich dabei von dem realen Material ab. So soll der Benutzer die Absichten der Komponente besser erkennen. Licht, Oberfläche und Bewegung von Komponenten stellen die Beziehung zueinander und die Einordnung im Raum dar. Realistische Beleuchtung sorgt für räumliche Trennung und hebt bewegliche Komponenten hervor [23].
In dem *design* Framework werden speziell vom Material Design vorgeschlagene Benutzerinteraktionen zur Verfügung gestellt. Genutzt wurde in der vorgestellten Anwendung unter anderem die Funktion der erweiterten Toolbar. Die Toolbar ist eine Leiste am oberen Rand des Bildschirms in der man normalerweise den Namen der Anwendung und das Menü findet. Da der Inhalt auf der Startseite über den unteren Bildschirmrand hinausragt und somit gescrollt werden muss, bietet es sich an die erweiterte Toolbar zu nutzen. Die Toolbar besteht im erweiterten Zustand aus einem Bild, was in Abbildung 5.1 zu sehen ist. Scrollt der Nutzer nach unten, so wird das Bild zusammengeklappt und die normale Toolbar mit Titel und Menü erscheint, wie man in Abbildung 5.2 sieht.

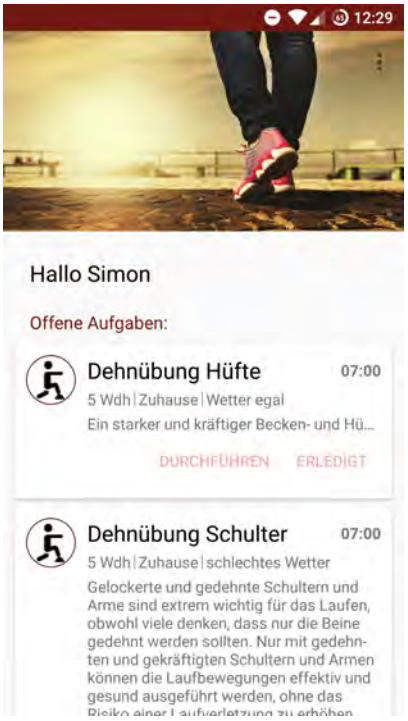

Abbildung 5.1: Eine erweiterte Toolbar auf der Startseite

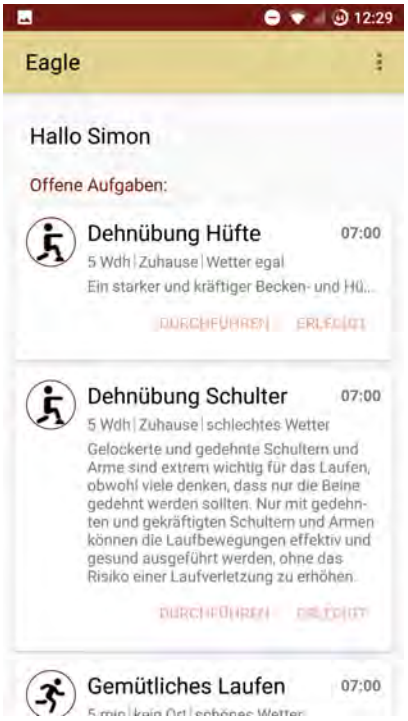

Abbildung 5.2: Eine zusammengeklappte Toolbar nach dem Scrollen

CardView

Die CardView ist eine Komponente aus dem Material Design. Sie wird benutzt, um einen Inhalt, der aus verschiedenen Elementen besteht, getrennt von anderen Inhalten anzuzeigen. So können Bilder, Videos und Text mit einer dazugehörigen Benutzeraktionen als ein einziger zusammengehöriger Inhalt auf einer CardView angezeigt werden. CardViews haben abgerundete Ecken und sind durch einen kleinen Schatten etwas vom Hintergrund abgehoben, wie man in Abbildung 5.3 sieht. Es soll eine Swipegeste für CardViews angeboten werden, um den dargestellten Inhalt zu entfernen. Zu jedem Inhalt soll eine Benutzeraktion angeboten werden. Diese muss nicht aus einem Knopf bestehen, sondern passt sich an den jeweiligen Inhalt an [24].
In dem vorgestellten Prototypen werden die CardViews benutzt, um alle Informationen über eine Übung als ein Ganzes darzustellen und von den anderen Übungen abzugren-

zen. In den Abbildungen 5.1 und 5.2 sieht man die Verwendung auf der Startseite. Als Aktionen werden das Durchführen und das Erledigen einer Übung angeboten. Auch die Swipegeste steht zur Verfügung. Wird eine CardView nach rechts oder nach links gewischt, so wird die Übung zum aktuellen Zeitpunk stumm geschaltet.

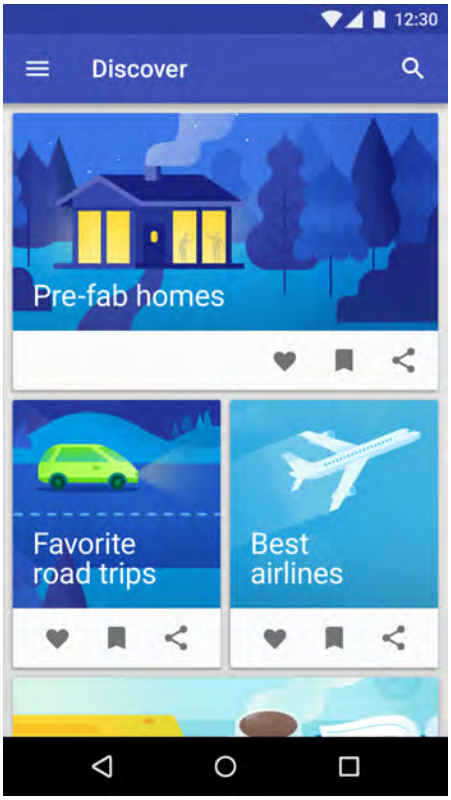

Abbildung 5.3: Beispiele mehrerer CardViews in einer Liste [24]

RecyclerView

Die RecyclerView ist eine grafische Komponente, um Listen anzuzeigen. Sie wird benutzt, um große Datenmengen, die sich oft ändern können, anzuzeigen. Um lange Ladezeiten bei großen Datenmengen zu vermeiden, werden einige Optimierungen implementiert. Nur die wirklich sichtbaren Listenelemente werden erstellt und angezeigt. Alle anderen Elemente, die sich außerhalb des Anzeigebereichs befinden, werden nicht geladen. Scrollt der Benutzer nun nach unten, so werden die neu sichtbaren Listenelemente

erstellt und alle nun nicht mehr sichtbaren Elemente werden zwischengespeichert. So müssen beim Zurückscrollen die vorher angezeigten Elemente nicht neu erstellt werden, sondern können schnell aus dem Speicher geladen werden. Die grafischen Elemente erstellt die RecyclerView dabei nicht jedes Mal neu, sondern verwendet immer die gleichen grafischen Elemente mit dem Inhalt des jeweiligen Listenelement. Bei einer Änderung der Daten werden nur die Elemente neu geladen, die sich auch wirklich geändert haben. Alle gleich gebliebenen Elemente können wiederverwendet werden [25].
Im dem vorgestellten Projekt wurde die RecyclerView für alle verwendeten Listen benutzt. Es wurden zwar keine langen Listen verwendet, jedoch führen die vielen Optimierungen der RecyclerView zu einem guten Benutzererlebnis. Dies ist besonders bei der Darstellung von Bilder zu den Übungen zu spüren.

CircularImageView

Die CircularImageView ist eine grafische Komponente, um Bilder kreisförmig darzustellen. In der nativen Android-Bibliothek gibt es keine einfache Möglichkeit, um ein Bild rund darzustellen. Dieses Framework bietet einen sehr einfachen Weg, ein kreisförmiges Bild zu erstellen. Auch gibt es viele Eigenschaften, mit denen man die Optik verändern kann. Genutzt wurde die CircularImagView mit der CardView auf der Startseite, um ein Icon für jede Übung anzuzeigen. Ein Beispiel kann man in Abbildung 5.4 sehen.

Abbildung 5.4: Ein rundes Bild passend zu jeder Übung

ViewPagerDotsIndicator

Der ViewPagerDotsIndicator ist ebenfalls eine grafische Komponente. Er wird kombiniert mit einem sogenannten ViewPager benutzt. Mit dem ViewPager kann man ganz leicht durch eine Liste von grafischen Komponenten (in diesem Fall ein Übungsschritt) mittels einer Swipegeste wischen. Der ViewPagerDotsIndicator zeigt dabei die Anzahl der Elemente in der Liste als Punkte an. Das aktuelle Element wird als ein längerer Balken angezeigt.
Der ViewPagerDotsIndicator wurde benutzt, um die einzelnen Schritte der Übung zu indizieren. So weiß der Benutzer immer, an welcher Stelle der Übung er sich befindet und wie viele Schritte er noch durchführen muss.

Abbildung 5.5: Am unteren Bildschirmrand wird angezeigt, bei welchem Schritt der Übung sich der Benutzer befindet und wie viele noch erledigt werden müssen.

Sugar

Sugar ORM ist ein Framework, um die Arbeit mit der Datenbank extrem zu vereinfachen. Dabei werden Java-Objekte mittels der API ganz ohne SQL-Abfragen in die Datenbank geschrieben und ausgelesen. Um die Erstellung der Datenbank muss sich der Entwickler nicht kümmern. Sugar basiert auf der SQLite-Datenbank [26]. In Abbildung 5.6 sieht man die Ergebnisse eines Benchmarks mehrerer Datenbank Frameworks. Für den Benchmark wurden insgesamt ca. 25.000 Datenbankeinträge gelesen und dabei die Zeit gemessen [27]. Wie man sieht, ist Sugar ORM im Vergleich zu anderen Frameworks nicht das schnellste Framework was die Lesegeschwindigkeit betrifft. Jedoch ist es extrem einfach zu benutzen und wurde deshalb ausgewählt. Die einfache Verwendung von Sugar sieht man in den Codebeispielen A.2 und A.3.

Abbildung 5.6: Die Lesegeschwindigkeit von Datenbank Frameworks im Vergleich [27].

Benutzt wurde Sugar für das komplette Datenmodel. Lediglich der Ort und die Aktivität des Benutzers wurden als Schlüssel-Wert-Paare gespeichert, um keine unnötigen Datenbankzugriffe bei kleinen Änderungen im Hintergrund zu haben. Beim ersten Anwendungsstart werden alle Übungen aus der JSON-Datei ausgelesen und in der Datenbank gespeichert. Alle Veränderungen an den Übungen (erledigen, stumm schalten) werden dann direkt auf der Datenbank vorgenommen. Die JSON-Datei wurde nur dazu benutzt, um die Übungen auf das Gerät zu bekommen und wird dann von der Datenbank abgelöst. Dadurch lässt sich das ständige Parsen der Datei vermeiden und ist bei Änderungen flexibler.

OpenWeatherMap

OpenWeatherMap ist eine API zur Bereitstellung von Wetterinformationen. Es wird ein einfacher HTTP-Request mit verschiedenen Parameter gesendet, über die man das Format, die Länge und den Aufbau der Antwort ganz nach eigenen Vorstellungen bestimmen kann. Man kann den Ort, für den die Wetterbedingungen geliefert werden sollen, entweder über Koordinaten oder über einen von über 200.000 Städtenamen bestimmen. Die vorhandenen Wetterdaten kommen dabei von über 40.000 Wetterstationen. Wetterinformationen werden immer in drei Stunden Abschnitten gegliedert. Die Anzahl der gelieferten Abschnitte kann man selbst festlegen. Standardmäßig bekommt man die Vorhersage der nächsten fünf Tage. Die Informationen für die Antwort werden dabei in verschiedenen Formaten angeboten. Sie liegen in JSON, XML und HTML bereit. Temperaturen können in Fahrenheit, Celsius und Kelvin ausgewählt werden. In der Antwort bekommt man neben dem Statuscode (der den HTTP-Statuscodes gleicht) auch noch die Bestätigung, für welchen Ort die vorliegenden Informationen gelten. Für den jeweiligen Wetterabschnitt wird neben den Wetterbedingungen auch noch der jeweils geltende Zeitpunkt geliefert. Die Wetterbedingungen bestehen aus einem Hauptteil, in dem unter anderem verschiedene Temperaturangaben, Druck und Luftfeuchtigkeit aufgelistet sind, einem Wetterteil, in dem gegebenenfalls eine Reihe an Wetterverhältnissen (sonnig, regnerisch, ...) zu finden sind und weitere Informationen über Wolken, Wind und Regenvolumen [28].

5.3 Google Awareness API

Die Awareness API ist ein von Google bereitgestelltes Framework, um auf die Umgebung des Nutzers reagieren zu können. Es werden insgesamt sieben verschiedene Typen von Kontext berücksichtigt. Unter anderem kann auf die Zeit, den Ort, das Wetter, die Aktivität und den Kopfhörerstatus zugegriffen werden. Alle Signale können kombiniert benutzt werden, um speziell auf die Situation passenden Inhalt in der Anwendung anzuzeigen. Zum Beispiel kann eine Playlist vorgeschlagen werden, wenn der Benutzer die Kopfhörer eingesteckt hat und anfängt zu joggen. Mit der Awareness API hat man nur noch eine

einzige Schnittstelle, anstelle von sieben Verschiedenen. So kann man ganz einfach die Umgebung des Nutzers berücksichtigen. Die Google Awareness API ist so optimiert, dass die vorhandenen Sensordaten bestmöglich ausgewertet werden und so wenig Strom wie möglich verbraucht wird. Trotz diesen Optimierungen können viele Anfragen an die API (wie zum Beispiel an welchem Ort befindet sich der Benutzer oder welche Aktivität führt der Benutzer aus?) die Batterie stark belasten [29].

Kontexttyp	**Beispiel**
Zeit	Aktuelle Zeit (ortsabhängig)
Position	Längengrad & Breitengrad
Ort	Ort und Orttyp (Restaurant, Supermarkt, ..)
Aktivität	Nutzeraktivität erkennen (Gehen, Rennen, Radfahren, ...)
Beacons	Beacons in der Nähe mit gewissem Namen
Kopfhörer	Sind die Kopfhörer eingesteckt?
Wetter	Das aktuelle Wetter

Tabelle 5.2: Alle sieben Kontexttypen, die in der Awareness API vereint sind [29].

Der Zugriff auf die API erfolgt auf zwei verschiedene Arten. Zum einen gibt es die Fence API mit der man einen sogenannten Fence registrieren kann. Dies bedeutet, dass die Anwendung benachrichtigt wird, sobald ein gewisses Ereignis eintritt. Zum anderen gibt es eine Snapshot API, mittels der man den aktuelle Zustand eines Kontexttyps abfragen kann [29].

Bei dem Zugriff über die Fence API kann die Position, die Aktivität, der Kopfhörerstatus und die Annäherung an ein Beacon (Bluetooth Empfänger) beachtet werden. Diese Kontexttypen können beliebig mittels Und-, Oder- und Nichtverknüpfungen kombiniert oder einzeln benutzt werden. Dabei kann zusätzlich zwischen den verschiedenen Übergängen des Ereignisses gefiltert werden. Es wird zwischen dem Beginn, während und dem Ende des Ereignisses unterschieden. Sobald alle Bedingungen eines Fences erfüllt sind, wird die Anwendung benachrichtigt. Das Wetter und der Typ des Ortes können nicht mittels der Fence API abgefragt werden [29]. Beispielsweise kann ein

Fence registriert werden, der auf Radfahren an einer bestimmten Position in einem Radius achtet. Hierbei wurde die Aktivität und der Standort miteinander verknüpft.

Bei dem Zugriff über die Snapshot API kann auf alle Kontexttypen zugegriffen werden. Die Anfrage wird vom System mit aktuellen Werten, die entweder schon im Cache vorliegen oder neu berechnet beziehungsweise angefragt werden, beantwortet [29].

Die Daten über die Umgebung des Benutzers werden über verschiedene Sensoren erhoben. Dies gilt insbesondere für die Aktivitätserkennung. Um die Aktivität eines Benutzers zu erkennen, wacht das Gerät immer wieder für einen kurzen Zeitpunkt auf, um eine gewisse Menge an Sensordaten zu lesen [30]. Die rohen Sensordaten werden dann mit Algorithmen aufbereitet und in verschiedene Kategorien interpretiert [29]. Bisher kann die Google Awareness API zwischen Radfahren, in einem Fahrzeug, zu Fuß unterwegs, rennen, gehen und bewegungslos unterscheiden [31]. Um das ständige Aufwachen so energieeffizient wie möglich zu machen, wird nur auf Sensoren zugegriffen, die wenig Energie brauchen (low energy sensors). Auch kann der Zeitabstand zwischen dem Aufwachen reguliert werden, um Energie zu sparen. Ist der Benutzer gerade bewegungslos, so wird das Gerät weniger oft aufgeweckt. Ist der Benutzer gerade aktiv, so werden die Sensordaten wieder häufiger ausgewertet, um die Aktivitätsübergänge besser erkennen zu können. Wenn es auf einem Gerät mehrere Anwendungen gibt, die die Aktivitätserkennung benutzen, so kommt es zu einem Zusammenspiel der Anwendungen. Aktivitätsupdates für die eine Anwendung werden dann, falls passend, auch an die andere Anwendung weitergeleitet, auch wenn gar kein Update verlangt wurde [30].

5.4 Umsetzung

In diesem Kapitel wird die konkrete Umsetzung beschrieben. Zunächst wird der grafische Aspekt im Kapitel 5.4.2 erläutert. Dabei werden Screenshots der Anwendung gezeigt. Da die Umgebung des Nutzers eine große Rolle spielt geschieht viel im Hintergrund der Anwendung. Welche Daten zu welchem Zeitpunkt erfasst werden, wird in Kapitel 5.4.1 erklärt.

5.4.1 Oberfläche

Wie schon in den funktionalen Anforderungen beschrieben wurde, soll die Gestaltung der Oberfläche mit Hilfe des Material Designs geschehen. Dafür wurden einige Frameworks zu Hilfe gezogen, die im Kapitel 5.2 schon aufgezählt und erläutert wurden. Wie die Anwendung aussieht, zeigen die folgenden Bilder.

In Abbildung 5.7 sieht man die Startseite, wie sie nach dem Anwendungsstart vorliegt. Unter der Begrüßung befindet sich eine Liste der offenen Übungen. Zu jeder Übung wird neben einem kleinen Symbol der Titel der Übung und die Uhrzeit der Ausführung angezeigt. Darunter werden dem Nutzer nähere Informationen zur Übung präsentiert. Es wird entweder die Dauer in Minuten oder die Anzahl der Wiederholungen, der Ausführungsort und die Wetterbedingung für jede Übung aufgelistet. Darunter findet der Benutzer eine kurze Beschreibung. Die Beschreibung ist standardmäßig verkürzt. Erst wenn der Nutzer die Übungskachel anklickt, wird die gesamte Beschreibung gezeigt. Auf der Startseite werden nur Übungen angezeigt, die zur Umgebung des Nutzers passen. Es erscheinen die noch nicht erledigten Übungen, die am aktuellen Tag bis spätestens innerhalb der nächsten drei Stunden ausgeführt werden sollten. Auch wird nach dem Standort des Benutzers gefiltert. Ist der Benutzer zuhause oder bei der Arbeit, so werden die jeweiligen Übungen für zuhause oder die Arbeit angezeigt. Ist der Ort unbekannt, so werden alle Übungen angezeigt. Das selbe gilt für das Wetter. Nur die Übungen, die zu den jeweiligen Wetterverhältnissen passen oder bei denen das Wetter keine Rolle spielt werden dem Benutzer angezeigt. Kann aus irgendwelchen Gründen kein Wetter abgefragt werden oder ist keines gespeichert, so werden alle Übungen unabhängig vom Wetter aufgelistet. Alle Übungen können mittels einer Swipegeste nach rechts oder links gewischt werden. Dadurch wird die Übung stumm geschaltet und taucht nicht mehr in der Liste auf. Das letzte Stummschalten kann vom Benutzer auch wieder rückgängig gemacht werden. Durch einen Klick auf 'Durchführen' gelangt der Nutzer direkt zum ersten Übungsschritt, wählt der Benutzer 'Erledigt' aus, so wird er gebeten, das Feedback der Übung einzugeben und abzusenden. Wird kein Feedback eingegeben, so wird der Nutzer bei jedem Start der Anwendung aufgefordert dies nachzuholen.

In Abbildung 5.8 sieht man die Einstellungen. Der Benutzer gelangt über die drei Punkte am oberen rechten Rand der Startseite zu den Einstellungen. Dort kann der Name für die Begrüßung und der Standort für Zuhause und die Arbeit eingegeben werden. Um den Standort besser erkennen zu können, wird dieser auf zwei verschiedene Arten erkannt. Es wird der Name des WLAN Netzwerks oder das GPS genutzt. Um beides einsetzen zu können, kann der Nutzer die jeweiligen Daten in den Einstellungen festlegen. Die Adresse kann dabei über eine integrierte Google Maps Karte ausgewählt werden. Wie die Erkennung des Standortes genau funktioniert, wird in Kapitel 5.4.2 erklärt.

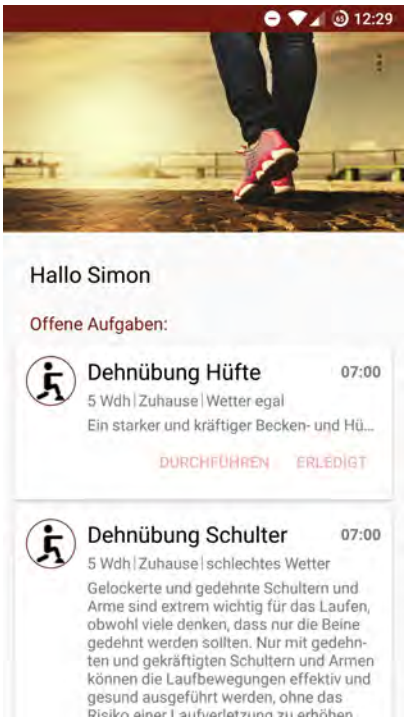

Abbildung 5.7: Startseite mit allen offenen Übungen

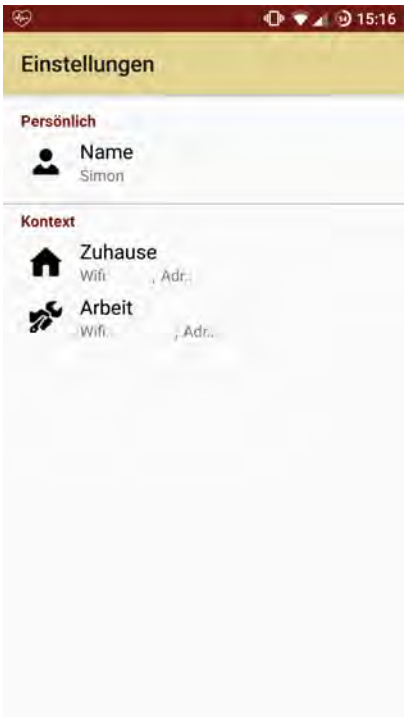

Abbildung 5.8: Alle Einstellungen für den Benutzer

Klickt der Nutzer auf der Startseite bei einer bestimmten Übung auf durchführen, so bekommt er eine Ansicht der einzelnen Übungsschritte wie in Abbildung 5.9. Dort sieht er oben immer das jeweilige Bild der Übung, die Zeit, zu der die Übung ausgeführt werden soll, die Dauer beziehungsweise die Anzahl der Wiederholungen und den Ort der Ausführung. Im unteren Teil bekommt der Nutzer eine genaue Anleitung für die einzelnen

Schritte mit einem dazugehörigen Bild. Der Benutzer kann durch die Schritte mittels einer Wischgeste navigieren. Die aktuelle Position in den Schritten sieht der Benutzer am unteren Bildschirmrand. Der letzte Schritt einer Übung ist immer das Feedback, wie man in Abbildung 5.10 sieht. Der Benutzer kann durch das Feedback scrollen. Erst wenn das Feedback komplett ausgefüllt ist, kann die Übung über den Haken rechts oben abgeschlossen werden.

Abbildung 5.9: Der erste Schritt einer Übung

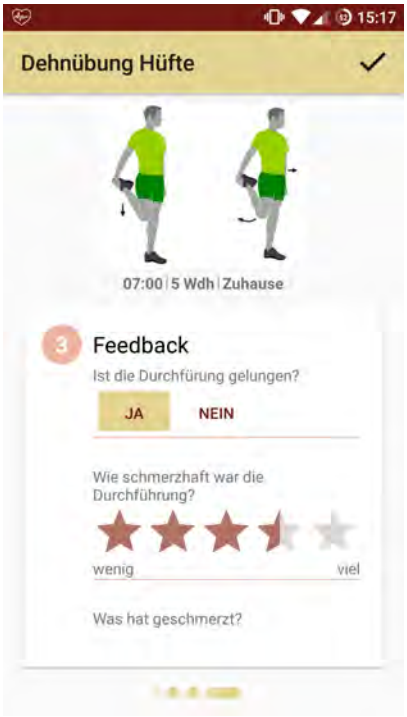

Abbildung 5.10: Feedback am Ende der Übung

Sobald der Benutzer abgeschlossene Übungen hat, werden diese auf der Startseite in einer separaten Liste unter den offenen Übungen, wie in Abbildung 5.11 angezeigt. Unter dem Icon und dem Titel der Übung sieht man wieder die geplante Uhrzeit, die Dauer oder die Anzahl der Wiederholungen und den Ort der Ausführung. Rechts daneben wird die Uhrzeit der tatsächlichen Ausführung angezeigt. Zusätzlich bekommt der Nutzer noch die Information, wann die Übung das nächste Mal ausgeführt werden soll. In der

Liste werden maximal die letzten sieben erledigten Übungen der letzten 36 Stunden angezeigt. Auch die Elemente dieser Liste können nach rechts oder links gewischt werden, was das Element aus der Liste löscht. Der Benutzer kann in diesem Fall jedoch die Aktion nicht wieder rückgängig machen. In Abbildung 5.12 sieht man, wie der Benutzer über anstehende Übungen benachrichtigt wird. Alle Übungen werden gebündelt in einer Benachrichtigung angezeigt. Ab Android 7 kann diese Nachricht ausgeklappt und die Übungen einzeln angeschaut werden. Zu jeder Übung kann der Nutzer dann direkt entscheiden, ob er die Übung ausführen oder stumm schalten möchte. Entscheidet er sich dazu, die Übung auszuführen, so wird er direkt zum ersten Übungsschritt geleitet.

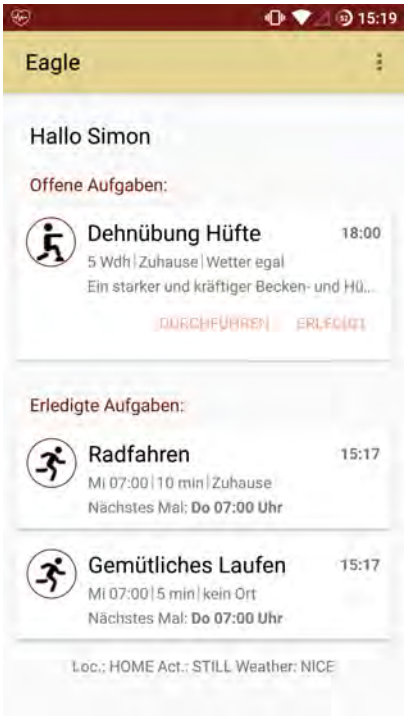

Abbildung 5.11: Die Startseite mit allen erledigten Aufgaben

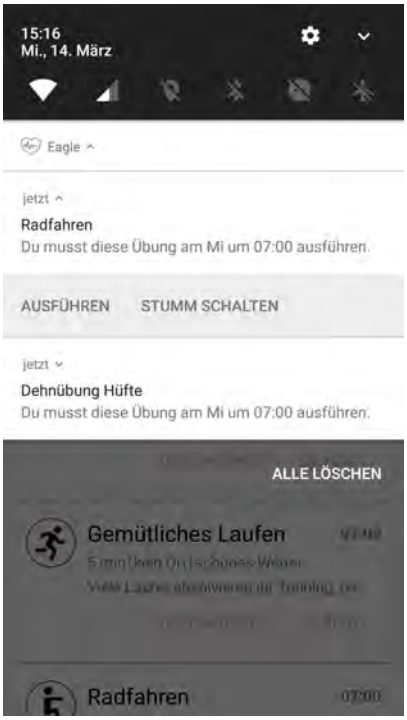

Abbildung 5.12: Benachrichtigungen für anstehende Übungen

Bisher wurde erläutert, was der Benutzer sieht und wie die Informationen durch die Umgebung des Benutzers angepasst werden. Im nächsten Abschnitt wird nun erklärt, wie der Kontext erfasst wird und was noch im Hintergrund passiert.

5.4.2 Hintergrundaktivität

Im Hintergrund wird auf Veränderungen der Nutzerumgebung reagiert und die Benachrichtigungen für Übungen geplant und durchgeführt. Alle Veränderungen werden gespeichert und beim Anwendungsstart beim Erstellen der Oberflächen mit einbezogen. Die Kommunikation der Komponenten sieht man in Abbildung 5.13 und wird näher erklärt.

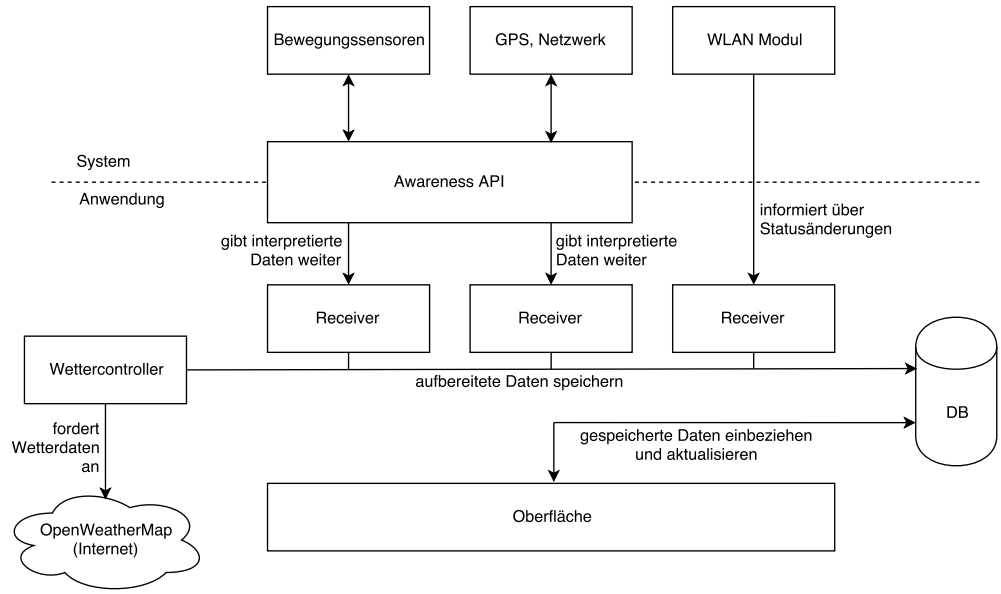

Abbildung 5.13: Das Zusammenspiel aller Komponenten im Hintergrund

Alle Hintergrundaktivitäten sollten durchgängig laufen ohne dass der Benutzer diese starten muss. Nach der Neuinstallation der Anwendung, werden die Hintergrunddienste beim ersten Benutzen gestartet. Auch wird bei jedem Anwendungsstart überprüft, ob noch alle Dienste laufen. Um bei einem Neustart des Smartphones nicht auf die erste Benutzung warten zu müssen, wurde ein Receiver genutzt, der das Booten des Smartphones erkennt. Dazu muss die Berechtigung *android.permission.RECEIVE_BOOT_COMPLETED* und ein Receiver mit der Aktion *android.intent.action.BOOT_COMPLETED* im Manifest der Anwenudng definiert werden. Somit wird nach jedem Boot des Smartphones der Receiver aufgerufen, der dann alle Hintergrunddienste startet. Somit kann die Umgebung des Nutzers immer vollständig erfasst werden.

Ortserkennung

Das Ziel der Ortserkennung ist es, zwischen ’Zuhause’, ’Arbeit’ und ’Unbekannt zu unterscheiden. Dadurch können die Übungen beim Anwendungsstart nach Ort gefiltert werden. Veränderungen des Ortes werden in der Anwendung auf zwei verschiedene Arten erkannt.

Mittels der Google Awareness API wird ein Geofence eingerichtet. Dafür werden GPS und im Netzwerk hinterlegte Daten genutzt, um das Betreten, Verweilen oder Verlassen eines Ortes zu erkennen. Dies ist in Abbildung 5.14 grafisch dargestellt. Um einen Geofence einrichten zu dürfen, muss der Benutzer nach der Erlaubnis gefragt werden, auf seinen genauen Standort zugreifen zu dürfen. Dafür muss zunächst im Manifest der Anwendung die Berechtigung durch *android.permission.ACCESS_FINE_LOCATION* definiert werden. Die explizite Erlaubnis des Nutzers wird dann bei jedem Anwendungsstart abgefragt, bis der Benutzer entweder zustimmt oder dauerhaft ablehnt. Ist die Berechtigung erteilt, so kann ein Geofence eingerichtet werden und die Ortserkennung arbeitet im Hintergrund. Dabei können beliebig viele Geofences eingerichtet werden. Diese werden durch einen einfachen Namen unterschieden. Um einen Geofence zu aktualisieren wird ein neuer Fence mit dem selben Namen erstellt. Die Awareness API aktualisiert dann automatisch den schon vorhandenen Fence.

Abbildung 5.14: Die grafische Darstellung der Möglichkeiten eines Geofences [32]

In der Anwendung wird ein Fence für Zuhause und ein Fence für die Arbeit eingerichtet. Dazu muss der Benutzer in den Einstellungen die Koordinaten für den jeweiligen Ort auf einer eingebetteten Karten auswählen. Erkennt die Awareness API das Verweilen an den festgelegten Koordinaten innerhalb eines 100 Meter Radius für mindestens zehn Sekunden, so wird die Anwendung benachrichtigt. Gibt die Awareness API das Signal, dass der Benutzer sich an einem der Orte befindet, so wird intern der Ort auf 'Zuhause' beziehungsweise 'Arbeit' gesetzt. Wird das Verlassen des Ortes gemeldet, so wird intern der Ort auf 'Unbekannt' gesetzt. Dies geschieht mittels einem Singleton Objekt[1]. Dadurch haben alle Komponenten der Anwendung, die den Ort benutzen immer den korrekten Standort. Zusätzlich wird der Ort auch noch als Schlüssel-Wert-Paar gespeichert um beim Zerstören des Singletons keine Datenverluste zu haben. Sind keine Koordinaten vorhanden, so kann die Ortserkennung nicht durch die Awareness API durchgeführt werden. In Codebeispiel A.4 wird die konkrete Abfrage nach der Standorterlaubnis durchgeführt. Das Erstellen eines Geofences kann man in Codebeispiel A.5 sehen.

Zum anderen wird der Ort durch den Namen des WLAN Netzwerks erkannt. Dazu muss der Nutzer in den Einstellungen eingeben, wie der Name des Netzwerks Zuhause und bei der Arbeit lautet. Im System wird dann ein Receiver mit der Aktion *android.net.wifi.STATE_CHANGE* registriert, der vom WLAN Modul Statusänderungen mitgeteilt bekommt. Wie die Ortsunterscheidung durch das WLAN abläuft, ist in Abbildung 5.15 dargestellt.

Bei jeglichen Statusänderungen kann der Receiver durch die Statusmeldung herausfinden, ob das Smartphone nun mit einem Netzwerk verbunden ist oder nicht. Bei einer Verbindung wird über den Namen des Netzwerks entschieden, ob der Nutzer nun Zuhause oder bei der Arbeit ist. Ist der Name nicht bekannt, so wird der Ort auf 'Unbekannt' gesetzt. Verliert das Smartphone die Verbindung, so durchläuft das Modul verschiedene Stadien. Um die schnell aufeinander folgenden Meldungen zu umgehen, wird erst bei Ablauf eines 15 Sekunden Timers der Status des WLAN Moduls interpre-

[1] Singleton ist ein Pattern, das sicherstellt, dass nur genau ein Objekt einer Klasse existiert.

tiert. Beim Verlieren der WLAN Verbindung werden Statusmeldung wie zum Beispiel *SCANNING, SUSPENDED* und *DISCONNECTING* durchlaufen. Bei Ablauf des Timers wird abgefragt, ob die WLAN Verbindung nur verloren ging (beispielsweise befindet sich das Smartphone außerhalb der Reichweite) oder ob der Benutzer das WLAN deaktiviert hat. Wurde beispielsweise der Flugmodus eingeschaltet und damit das WLAN Modul deaktiviert, so wird der vorherige Ort beibehalten, da keine weiteren Aussagen über eine Veränderung des Ortes gemacht werden können (Der Benutzer könnte vor dem Schlafengehen den Flugmodus eingeschaltet haben, um nicht gestört zu werden und befindet sich daher immer noch am selben Ort). Hat das Smartphone aber die Verbindung zum WLAN verloren und dieses ist immer noch aktiviert, so wird der Ort auf 'Unbekannt' gestellt. Wurde während des laufenden Timers eine Verbindung zu einem Netzwerk hergestellt, so wird der Timer abgebrochen.

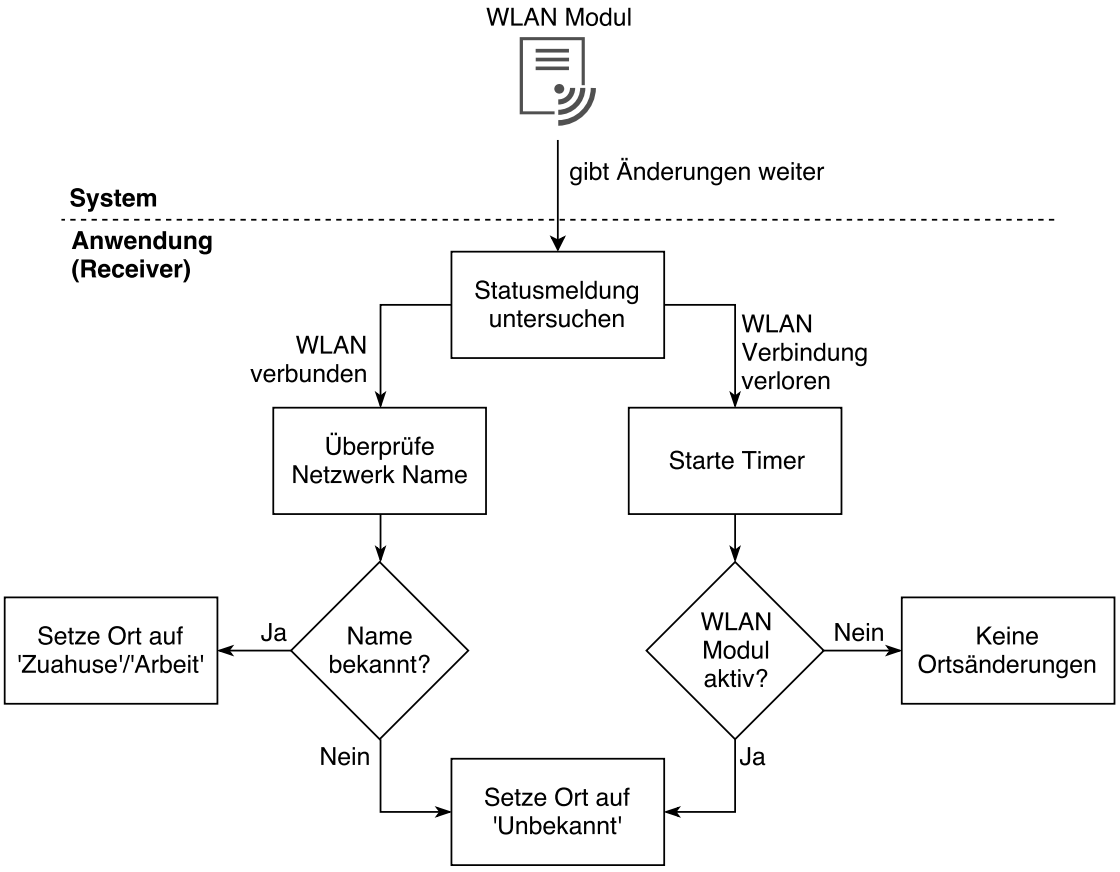

Abbildung 5.15: Ortsunterscheidung durch den WLAN-Name

Gibt der Benutzer keine WLAN Namen ein, so kann der Ort nicht durch das WLAN unterschieden werden. Sind weder Koordinaten noch WLAN Namen bekannt, so werden auf der Startseite alle Übungen unabhängig vom Ort angezeigt. Beide Möglichkeiten, den Ort zu erkennen, arbeiten parallel und es wird immer das neueste Update als das Aktuelle genutzt.

Es wurde beim Ort auf zwei verschiedene Technologien zugegriffen, um ein bestmögliches Ergebnis zu erzielen. Da nicht jeder das GPS immer angeschaltet hat, wurde noch zusätzlich das WLAN für die Erkennung des Ortes genutzt. Auch funktionierte das GPS mit der Google Awareness API bei dem Testen der Anwendung nicht immer zufriedenstellend.

Aktivitätserkennung

Um die Aktivität des Nutzers zu erkennen, wurde die Aktivitätserkennung der Awareness API genutzt. Diese basiert nur auf ('low-energy') Sensoren des Smartphones und nicht auf GPS, wie schon in Kapitel 5.3 erläutert wurde. Ziel der Aktivitätserkennung ist es, die körperlichen Aktivitäten des Benutzers in 'Radfahren', 'Rennen' und 'Gehen' zu erkennen. Die gewonnenen Informationen werden dazu genutzt, um Übungen mit der jeweiligen Aktivität direkt als erledigt markieren zu können. Wie die Awareness API und die Anwendung zusammenspielen, wird in Abbildung 5.16 dargestellt.

Um Updates der Aktivität zu bekommen, wird mittels der Awareness API ein Fence für jede Aktivität registriert. Der Aktivitätsfence ist einem Geofence sehr ähnlich. Fences können für das Starten, Ausführen oder Beenden einer Aktivität hinzugefügt werden. In Codebeispiel A.6 sieht man wie einfach ein Fence für 'Gehen' eingerichtet wird. Um die Aktivitätserkennung benutzen zu dürfen, muss im Manifest der Anwendung die Berechtigung *com.google.android.gms.permission.ACTIVITY_RECOGNITION* definiert werden. Eine explizite Zustimmung des Benutzers ist nicht erforderlich.

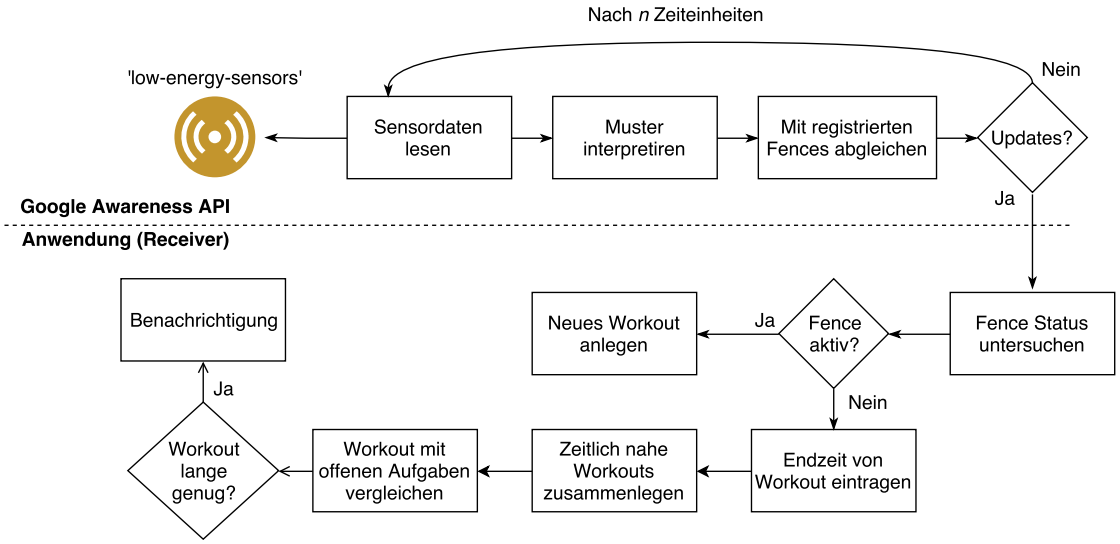

Abbildung 5.16: Aktivitätserkennung

Die Awareness API liest und interpretiert nun in regelmäßigen Abständen Sensordaten. Wird eine Aktivität erkannt und liegt ein entsprechender Fence vor, so wird eine Statusmeldung für den jeweiligen Fence an die Anwendung geschickt. Die Meldung gibt an, ob der Fence gerade aktiv, inaktiv oder unbekannt ist.
In der Anwendung wird jeweils ein Fence für Radfahren, Rennen und Gehen registriert. Dabei wird die Ausführung der Aktivität gefiltert (nicht der Start oder das Ende). Bekommt die Anwendung also die Meldung, dass der Fence aktiv ist, so befindet sich der Benutzer in der Ausführung der jeweiligen Aktivität. Sobald die Meldung, dass der Fence inaktiv ist, vorliegt, so wurde die jeweilige Aktivität beendet.
Wird der Start einer Aktivität erkannt, so wird ein Workout mit einer Startzeit angelegt. Weiter passiert zunächst nichts. Wird das Ende einer Aktivität gemeldet, so wird das zuvor begonnenen Workout beendet. Nun werden alle bisher bekannten Workouts analysiert. Liegen zwei Workouts mit der selben Aktivität zeitlich nahe aneinander, so werden diese beiden Workouts zu einem gemeinsamen Workout zusammengefasst. So können gemachte Pausen des Benutzers während einer Aktivität berücksichtigt werden. Werden beim Gehen maximal fünf Minuten, beim Rennen maximal zehn Minuten und beim Radfahren maximal 15 Minuten Pause gemacht, so werden die beiden getrennten Workouts zu einem einzigen Workout zusammengefügt.

Nach dem Zusammenfügen zweier Workouts werden zusätzlich noch alle offenen Übungen analysiert. Steht am heutigen Tag noch eine Übung mit der Aktivität des abgeschlossenen Workouts an und beträgt die Dauer des Workouts mindestens 75 Prozent der Dauer der Übung, so wird diese Übung automatisch als erledigt markiert. Der Benutzer bekommt eine Benachrichtigung über das erfolgreiche Abschließen der Übung und wird gebeten, das Feedback für die Übung abzugeben.

Die Aktivitätserkennung funktioniert schon kurze Augenblicke nach dem Start der Aktivität sehr zuverlässig. Auch wird die Aktivitätsart sehr zuverlässig unterschieden, solange sich das Smartphone in der Hosentasche des Nutzers befindet.

Wetter

Das Wetter wird mittels der OpenWeatherMap API geladen. Dazu wird ein HTTP-Requests mit den gewünschten Werten als Parameter übertragen. Als Antwort werden die Wetterverhältnisse in drei-Stunden-Abschnitten überliefert. Das Ziel der Wetterabfrage ist es, Wetterinformationen, die in ’Schön’ und ’Schlecht’ gegliedert sind, vorliegen zu haben. Damit werden auf der Startseite die Übungen passend gefiltert.

Das Wetter wird morgens, mittags und abends im Hintergrund abgefragt. Dafür ist eine Internetverbindung notwendig. Es werden die Wetterverhältnisse der nächsten 15 Stunden geladen und gespeichert. Wird Wetter abgefragt, so werden die schon vorliegenden Daten aktualisiert und veraltete Daten gelöscht. Kommt es bei der aktiven Nutzung der Anwendung durch den Benutzer dazu, dass keine oder nur noch wenig aktuelle Wetterinformationen vorliegen, so werden sofort neue Wetterinformationen geladen. Das Wetter wird über die Koordinaten des aktuellen Ortes in Celsius angefordert. Die überlieferten Daten werden im JSON-Format dargestellt. Um die Wetterverhältnisse in ’Schön’ und ’Schlecht’ interpretieren zu können, werden die überlieferten Wetterabschnitte genauer analysiert. Für jedes Wetterverhältnis in den Abschnitten wird eine ID, ein Überbegriff, eine Beschreibung und ein empfohlenes Icon übertragen. Allein durch die ID wird das Wetter bewertet. In Tabelle 5.3 sieht man die empfangenen IDs und deren

Beschreibung. Jede dieser IDs wird als 'Schön' aufgefasst. Werden mehrere kombinierte IDs empfangen (beispielsweise Windstill und Regnerisch), so wird dieser Wetterabschnitt nur als 'Schön' markiert, wenn alle einzelnen IDs auch als 'Schön' eingestuft sind.

ID	Beschreibung
800	Klarer Himmel
801 / 802	Einzelne Wolken
951	Windstill
952 / 953	Leichte Brise

Tabelle 5.3: Empfangene Wetter-IDs und deren Beschreibung

Um die Wetterinformationen für den Standort des Nutzers zu bekommen, wird zuerst abgefragt, ob der Benutzer sich zuhause oder bei der Arbeit befindet. Ist der Ort bekannt, so können die in den Einstellungen festgelegten Koordinaten benutzt werden. Sind keine Koordinaten hinterlegt oder ist der Standort des Nutzers nicht bekannt, so wird der Standort über GPS bestimmt und damit die Wetterinformationen abgerufen. Ist jedoch das GPS nicht aktiviert, so werden die Wetterdaten nicht aktualisiert. Der genaue Ablauf kann auch in Abbildung 5.17 betrachtet werden.

Mit der Awareness API können zwar auch Wetterinformationen abgefragt werden, jedoch ist der Prozess sehr zeitaufwändig und liefert die Wetterdaten nur für den aktuellen Zeitpunkt. Zuerst wird mittels GPS der Standort abgefragt und mit diesen Koordinaten die Wetterbedingungen geladen. Bei deaktiviertem GPS kommt es zu einem Fehler und es können keine Wetterinformationen geladen werden. Da das Wetter beim Anzeigen der Übungen auf der Startseite benutzt wird und es durch die aufwändige Abfrage durch die Awareness API zu zeitlichen Verzögerungen kommt, sollten die Wetterinformationen schon im Voraus vorliegen. Um eine Wettervorhersage von mehreren Stunden zu bekommen wurde die *OpenWeatherMap* API verwendet.

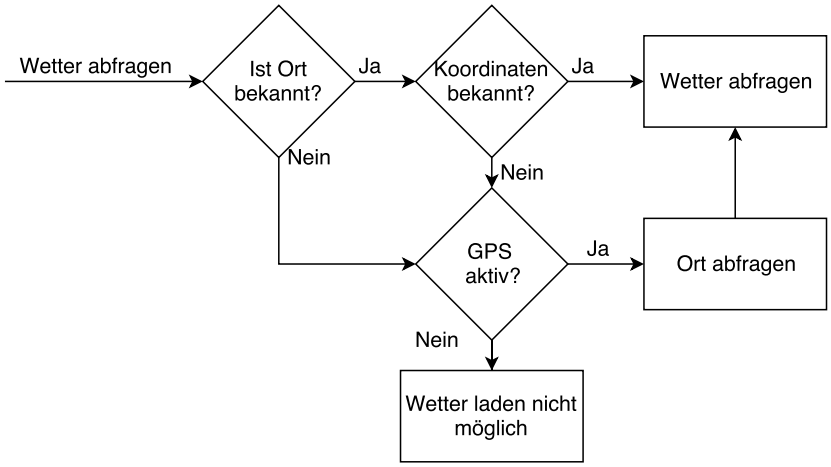

Abbildung 5.17: Ablauf für die Standortbestimmung bei einer Wetterabfrage

Benachrichtigungen

Im Hintergrund geschieht auch die Organisation der Benachrichtigungen über anstehende Übungen. Der Benutzer soll morgens, mittags und abends an die Übungen erinnert werden. Dafür werden zwei verschiedene Receiver genutzt. Zum einen wird ein AlarmReceiver eingesetzt, der bei Ablauf eines Alarms benachrichtigt wird. Zum anderen wird ein ScreenOnReceiver genutzt, der das Einschalten des Bildschirms erkennt. Wie die beiden Receiver miteinander in Kontakt stehen ist in Abbildung 5.18 zu sehen und wird im Folgenden genauer erklärt.

Um die Benachrichtigungen anzuzeigen, werden drei Alarme gestellt. Läuft ein Alarm ab, so wird ein Receiver aufgerufen. In diesem Receiver wird dann für alle anstehenden Übungen, die zum Kontext passen, eine Benachrichtigung erstellt. Im Folgenden wird dieser Receiver als AlarmReceiver bezeichnet. Die erste Benachrichtigung soll der Benutzer nach dem Aufwachen beim ersten Benutzen des Smartphones angezeigt bekommen. Da es keine Lösung für das Erkennen der ersten Benutzung am Morgen gibt, wurde folgendes getan: Es wird ein Alarm auf fünf Uhr morgens gestellt. Läuft der Alarm ab, so wird der AlarmReceiver aufgerufen. Dort werden nun keine Benachrichtigungen erstellt, sondern die Maßnahme ergriffen, um das erste Einschalten des Bildschirms zu

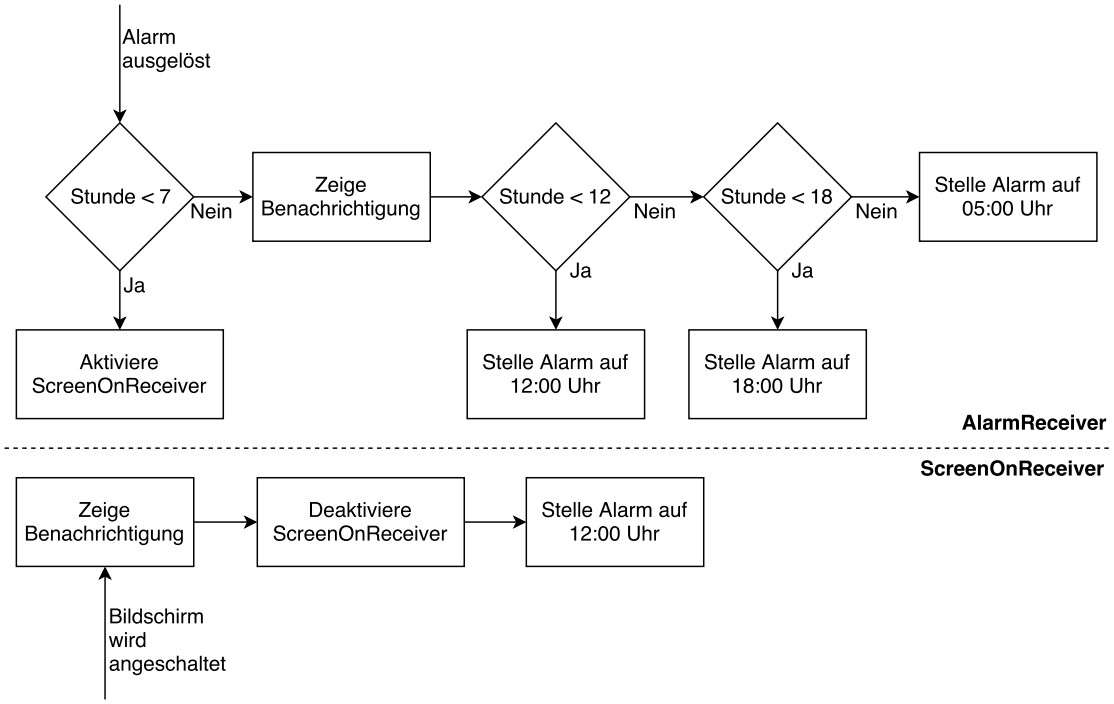

Abbildung 5.18: Ablauf der Planung der Benachrichtigungen

erkennen. Dafür wird ein weiterer Receiver aktiviert. Diesem Receiver wird die Aktion *ACTION_SCREEN_ON* übergeben und reagiert somit auf das Einschalten des Displays. Im Folgenden wird er ScreenOnReceiver genannt. Schaltet der Benutzer nun seinen Bildschirm an, so zeigt der ScreenOnReceiver die entsprechenden Benachrichtigungen an. Danach deaktiviert er sich selbst, da er nicht mehr benötigt wird. Außerdem stellt er den nächsten Alarm auf 12:00 Uhr. Zu dieser Uhrzeit wird wieder der AlarmReceiver aufgerufen, welcher direkt die Benachrichtigungen anzeigt. Das Selbe geschieht um 18:00 Uhr. Dort wiederum wird der nächste Alarm auf fünf Uhr morgens gestellt, womit sich das Ganze wiederholt.

6

Anforderungsabgleich

In diesem Kapitel sollen die Funktionen des entstandenen Prototypen mit den zuvor definierten funktionalen und nicht-funktionalen Anforderungen abgeglichen werden. Der entstandene Prototyp wird mit den in Kapitel 3 aufgelisteten Anforderungen verglichen.

6.1 Funktionale Anforderungen

In diesem Abschnitt soll die Funktionsweise des Prototyps abgeglichen werden. Es wird näher beschrieben, wie die einzelne Funktionalität im Prototyp konkret umgesetzt wurde.

Nr.	Anforderung	Status	Beschreibung
1	Benachrichtigung über anstehende Übung	*erfüllt*	Morgens nach dem Aufstehen, mittags und abends wird eine Benachrichtigung über im Kontext anstehende Übungen angezeigt. Die Benachrichtigungen enthalten Titel und Ausführungszeitpunkt.
2	Kontext-Erkennung	*erfüllt*	Die Anwendung erkennt den Kontext des Benutzers. Um den Ort zu erkennen, müssen entweder die WLAN Namen von Zuhause und der Arbeit oder die Koordinaten in den Einstellungen festlegen werden. Eine Internetverbindung wird für Wetterabfragen benötigt. Die Zeit und die Aktivität wird offline ohne jegliche Nutzereingaben erkannt.

3	Anweisungen für Übung mit Video, Bild und Text	*erfüllt*	Eine Übung besitzt ein generelles Bild und zu jedem Übungsschritt eine Beschreibung mit einem unterstützenden Bild. Bisher gibt es keine Übungen, die mit einem Video erklärt wurden.
4	Übungen passend zum Kontext anzeigen	*erfüllt*	Die Startseite wird passend zum erfassten Kontext gestaltet. Die angezeigten Übungen werden mit dem aktuellen Zeitpunkt, dem erkannten Ort und dem geladenen Wetter gefiltert. Liegen bestimmte Kontextinformationen nicht vor, so werden die Übungen ohne den entsprechenden Kontext gefiltert und versucht die entsprechenden Informationen zu beschaffen. Der Benutzer wird über eine erkannte Aktivität, die zu einer noch auszuführenden Übung passt benachrichtigt.
5	Feedback zu jeder Übung	*erfüllt*	Nach jeder Übung muss der Benutzer ein individuelles Feedback ausfüllen. Dieses besteht aus Text, Ja/Nein-Fragen und einer Bewertungsskala von null bis fünf. Wird zu einer abgeschlossenen Übung kein Feedback geschickt, so wird der Nutzer bei jedem Start der Anwendung gebeten, das fehlende Feedback auszufüllen.
6	Übungen stumm schalten	*erfüllt*	Der Benutzer kann sowohl auf der Startseite als auch direkt bei den Benachrichtigungen (ab Android 7) die Übung stumm schalten. Dies geschieht auf der Startseite durch eine einfache Swipegeste, bei der Benachrichtigung durch einen Klick.

Tabelle 6.1: Anforderungsabgleich der funktionalen Anforderungen

6.2 Nicht-funktionale Anforderungen

In diesem Abschnitt werden die Rahmenbedingungen für die Funktionalitäten bewertet. Es wird wieder zu jeder Anforderung näher beschrieben, wie diese umgesetzt wurde.

Nr.	Anforderung	Status	Beschreibung
1	Material Design	*erfüllt*	Es wurden viele Frameworks benutzt, um das Material Design umzusetzen. Auch wurden die von Google bereitgestellten Styleguides berücksichtigt. Dadurch sind die vorhandenen grafischen Komponenten wie in den Styleguides beschrieben gestaltet.
2	Batteriesparend	*erfüllt*	Da bei der Anwendung sehr viel im Hintergrund geschieht, wurde darauf geachtet, keine zu aufwändigen Dinge unnötig oft im Hintergrund durchzuführen. Die Google Awareness API nimmt davon schon einiges selbst in die Hand und hat verschiedene Optimierungen zur Verminderung des Batterieverbrauchs [29]. Wetteraktualisierungen wurden auf das Nötigste reduziert und zwischengespeichert. Das ständige Nutzen des GPS kann vermieden werden, da der Ort auch durch WLAN Namen unterschieden wird.
3	Usability	*teils erfüllt*	Ein Test von Patienten mit verschiedenen Einschränkungen wurde nicht durchgeführt. Daher kann die gute Bedienbarkeit von jeglichen Personen nicht beurteilt werden. Dem Benutzer werden verschiedene Möglichkeiten geboten, die Übungen direkt als erledigt zu markieren, ohne unnötigen Aktionsschritte. Somit wurde dieser Teil der Anforderung erfüllt.

Tabelle 6.2: Anforderungsabgleich der nicht-funktionalen Anforderungen

7

Zusammenfassung und Ausblick

Therapeutische Hausaufgaben nehmen immer mehr an Bedeutung für therapeutische Interventionen zu [1]. Besonders auch die Verbesserung des Therapieergebnisses durch den Einsatz mobiler Geräte [5] befürworten das Entwickeln einer Anwendung zur Unterstützung des Patienten bei Übungsaufgaben. Dabei sind besonders die Sensoren der intelligenten mobilen Geräte und das direkte Feedback für Therapeuten und Wissenschaftler hilfreich [7].
Im Rahmen dieser Arbeit wurde ein Prototyp entwickelt, der Patienten auf Android Smartphones bei der Durchführung ihrer therapeutischen Hausaufgaben unterstützt. Dazu erkennt das Gerät automatischen den jeweiligen Kontext. Es wird der Ort, die Zeit und das Wetter erkannt, um dem Benutzer die Übungen gefiltert zu präsentieren. Morgens nach dem Aufstehen, mittags und abends wird der Benutzer zu der Ausführung noch offener Übungen aufgefordert. Dabei werden nur Benachrichtigungen für die zum Kontext passenden Übungen angezeigt. Wird bei einer Übung eine körperliche Aktivität des Nutzers gefordert, so erkennt das Smartphone diese automatisch und markiert die Übung im Hintergrund als erledigt. So wird Radfahren, Gehen und Laufen automatisch erkannt. Nach dem Abhaken einer Übung (sowohl aktiv durch den Benutzer als auch passiv durch die Aktivitätserkennung) muss der Benutzer das individuell zur Übung passende Feedback ausfüllen.
Bei der Entwicklung wurde besonders auf eine energiesparende Hintergrundaktivität sowie eine ansprechende Oberfläche im Material Design geachtet. Insbesondere die einfache Benutzung der Google Awareness API war von großem Vorteil für die Erkennung des Kontextes. Speziell die Aktivitätserkennung funktioniert zuverlässig und genau. Um die Ortserkennung noch genauer zu gestalten, wird der Ort zusätzlich über den

WLAN Namen unterschieden. Da die Awareness API das Wetter nur zum aktuellen Zeitpunkt liefern kann, wurde auf die OpenWeatherMap API zugegriffen. Damit lassen sich Wetterverhältnisse für mehrere Stunden im Voraus abfragen und speichern.

Um die Integration der Anwendung in ein komplexes IT-System zur Handhabung von therapeutischen Hausaufgaben zu schaffen, muss noch geklärt werden, wie und wann die Daten vom Therapeuten zum Patienten und anders herum übertragen werden. Dazu muss zunächst einmal eine Verbindung zwischen diesen beiden Parteien zustande kommen. Möglich wäre eine Art Einladungssystem, bei dem der Therapeut den Patienten zu der Behandlung mit dem mobilen Gerät während einer der ersten Therapiesitzungen einlädt. Danach können die Fragen geklärt werden, in welchem Format, wie oft und wann die Hausaufgaben auf das Gerät des Patienten übertragen und aktualisiert werden. Typische Offline/Online-Szenarien können bei Smartphones jederzeit auftreten und sollten der Therapie nicht im Wege stehen. Bei einer Datenübertragung muss natürlich auch die Sicherheit der Daten im Vordergrund stehen.
Bei der Ausführung einer Hausaufgabe sind oft auch die Vitalwerte von Bedeutung. Beispielsweise ist bei bestimmten Übungen die Messung des Puls interessant und liefert dem Therapeuten wichtige Informationen. Dazu könnte mittels der Google Fit API jegliche Bluetooth-Fitness-Geräte verwaltet werden [33]. So könnten Patienten schon vorhandene Geräte regulär weiter nutzen und liefern zu der Hausaufgabe noch wertvolle Zusatzinformationen. Neben der Erfassung des Puls, können insbesondere Smartwatches auch dafür genutzt werden, um Feedback auszufüllen. Die TrackYourTinnitus Plattform kam zu dem Entschluss, dass Smwartwatches eine echte Alternative für Smartphones beim Ausfüllen von Fragebögen sind [34]. So könnte dem Patienten nach der erkannten körperlichen Aktivitäten das Feedback direkt auf der Smartwatch angezeigt werden, um so ein noch direkteres Feedback zu bekommen und eine weitere Benutzung des Smartphones zu vermeiden.
Um dem Therapeuten die bestmöglichen Kontextdaten zu liefern, muss die Anwendung sehr flexiblen sein. Dafür müssen dem Therapeuten zunächst entsprechende Werkzeuge zur Verfügung gestellt werden, damit dieser den Kontext individuell zu jeder Übung erstellen kann. Die Anwendung muss dann flexibel auf die verschiedensten Kontexte regieren

können. Das QuestionSys Framework beschäftigt sich mit solchen flexiblen Erstellungen für die mobile Datenerfassung und präsentiert einige Patterns [35]. Dabei wird darauf geachtet, dass die Erstellung auch von Personen ohne technischen Hintergrund über eine grafische Oberfläche durchgeführt werden kann [36][37]. Dieser Prozess könnte auf therapeutische Hausaufgaben mit mobilen Geräten übertragen werden, um ein flexibleres System zur Kontexterfassung zu schaffen.
Zuletzt sollte auch noch durchdacht werden, was passiert, wenn der Patient von einem Kontext in einen anderen Kontext übergeht. Ein Beispiel soll dies deutlich machen und könnte wie folgt aussehen: *Ein Patient soll Zuhause Übung A und B morgens nach dem Aufstehen und Übung C mittags erledigen. Ist er bei der Arbeit, so soll er zur Mittagszeit Übung D machen.* Nun tritt folgender Fall ein: Der Patient findet morgens nur die Zeit für Übung A und geht danach zur Arbeit, wo er Übung D regulär ausführt. Nun ist die Frage, ob der Patient bei der Arbeit immer noch an Übung B erinnert werden soll, obwohl er nicht mehr Zuhause ist, die Übung jedoch noch erledigt werden muss. Kommt der Patient abends nach Hause, so muss nun entschieden werden, welche verbleibenden Aufgaben er noch nachholen soll. Soll er nun nur die noch offene Übung B nachholen, da er morgens Zuhause war und muss er zusätzlich auch Übung C machen, da er nun Zuhause ist und diese Übung ebenfalls unerledigt ist? Oder vielleicht muss er keine der beiden Aufgaben erledigen, da zwar der Ort stimmt aber die Ausführungszeit nicht abends lautet. Diese Fragen konnten im Prototyp nicht geklärt werden, da keine ausreichenden realen Daten vorlagen. Außerdem sollten solche Fragen, die durchaus Auswirkungen auf die Therapie haben, mit professionellen Fachkräften geklärt werden.

Literaturverzeichnis

[1] Fehm, L., Fehm-Wolfsdorf, G. In: Therapeutische Hausaufgaben. Springer Berlin Heidelberg, Berlin, Heidelberg (2009) 709–719

[2] Kazantzis, N., Deane, F.P., Ronan, K.R.: Homework Assignments in Cognitive and Behavioral Therapy: A Meta-Analysis. Clinical Psychology: Science and Practice (**7**) 189–202

[3] Deutscher Ärzteverlag GmbH, R.D.r.: Hausaufgaben in der Psychotherapie: Noch unentdecktes Potenzial. Website (2018) Online erhältlich unter `https://www.aerzteblatt.de/archiv/67358/Hausaufgaben-in-der-Psychotherapie-Noch-unentdecktes-Potenzial` - aufgerufen am 24.03.2018.

[4] Schickler, M., Pryss, R., Schobel, J., Reichert, M.: Supporting Remote Therapeutic Interventions with Mobile Processes. In: 6th IEEE International Conference on AI & Mobile Services (IEEE AIMS 2017), IEEE Computer Society Press (2017)

[5] Lindhiem, O., Bennett, C.B., Rosen, D., Silk, J.: Mobile Technology Boosts the Effectiveness of Psychotherapy and Behavioral Interventions: A Meta-Analysis. Behavior Modification **39** (2015) 785–804 PMID: 26187164.

[6] Schickler, M., Pryss, R., Schobel, J., Schlee, W., Probst, T., Reichert, M.: Towards Flexible Remote Therapeutic Interventions. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017), IEEE Computer Society Press (2017)

[7] Schickler, M., Pryss, R., Stach, M., Schobel, J., Schlee, W., Probst, T., Langguth, B., Reichert, M.: An IT Platform Enabling Remote Therapeutic Interventions. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017), IEEE Computer Society Press (2017)

[8] Mediscope: Physiotherapie, Krankengymnastik - Abklärung, Behandlung, Prävention. Website (2018) Online erhältlich unter https://www.sprechzimmer.ch/sprechzimmer/Physiotherapie/Krankengymnastik_Abklaerung_Behandlung_Praevention.php - aufgerufen am 24.03.2018.

[9] Schobel, J., Pryss, R., Schlee, W., Probst, T., Gebhardt, D., Schickler, M., Reichert, M.: Development of Mobile Data Collection Applications by Domain Experts: Experimental Results from a Usability Study. In: 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017). Number 10253 in LNCS, Springer (2017) 60–75

[10] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: Towards Flexible Mobile Data Collection in Healthcare. In: 29th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2016). (2016) 181–182

[11] NinjaMock.com: NinjaMock. Website (2018) Online erhältlich unter https://ninjamock.com/ - aufgerufem am 20.03.2018.

[12] Pixabay: Kostenlose Bilder. Website (2018) Online erhältlich unter https://pixabay.com/ -aufgerufen am 20.03.2018.

[13] Pixabay: Kostenloses Foto: Spaziergang, Pfad, Füße, Trail. Website (2018) Online erhältlich unter https://pixabay.com/de/spaziergang-pfad-f%C3%BC%C3%9Fe-trail-tennis-2635038/ - aufgerufen am 20.03.2018.

[14] Bianchi, F.: Coolors.co - The super fast color schemes generator. Website (2018) Online erhältlich unter https://coolors.co/ - aufgerufen am 20.03.2018.

[15] coolors: Generate - Coolors.co. Website (2018) Online erhältlich unter https://coolors.co/701913-e3d493-af7367-b99561-edb6a3 - aufgerufen am 18.03.2018.

[16] JoggenOnline: Joggen Online: Dein Sportmagazin | Rund um Laufen und Fitness. Website (2018) Online erhältlich unter https://www.joggen-online.de/ - aufgerufen am 27.03.2018.

[17] Developers, A.: Android, the world's most popular mobile platform. Website (2018) Online erhältlich unter https://developer.android.com/about/index.html - aufgerufen am 08.03.2018.

[18] Google: Android. Website (2018) Online erhältlich unter https://www.android.com/ - aufgerufen am 08.03.2018.

[19] Weck, A.: 9 von 10 Smartphones: Android-Marktanteil erreicht neuen Rekord. Website (2018) Online erhältlich unter https://t3n.de/news/android-marktanteil-weltweit-2016-763193/ - aufgerufen am 08.03.2018.

[20] Google: Android ist für alle da. Website (2018) Online erhältlich unter https://www.android.com/everyone/facts/ - aufgerufen am 08.03.2018.

[21] Brain, A.: Number of Android applications. Website (2018) Online erhältlich unter https://www.appbrain.com/stats/number-of-android-apps - aufgerufen am 08.03.2018.

[22] Developers, A.: Dashboards. Website (2018) Online erhältlich unter https://developer.android.com/about/dashboards/index.html - aufgerufen am 08.03.2018.

[23] Google: Introduction - Material Design. Website (2018) Online erhältlich unter https://material.io/guidelines/# - aufgerufen am 06.04.2018.

[24] Google: Cards - Components - Material Design. Website (2018) Online erhältlich unter https://material.io/guidelines/components/cards.html#cards-usage - aufgerufen am 09.03.2018.

[25] Developers, G.: Create a List with RecyclerView. Website (2018) Online erhältlich unter https://developer.android.com/guide/topics/ui/layout/recyclerview.html - aufgerufen am 09.03.2018.

[26] Satyan: Sugar ORM - Insanely easy way to work with Android Databases. Website (2018) Online erhältlich unter http://satyan.github.io/sugar/index.html - aufgerufen am 11.03.2018.

[27] AlexeyZatsepin: GitHub - AlexeyZatsepin/Android-ORM-benchmark: Performance comparison of Android ORM Frameworks. Website (2018) Online erhältlich unter `https://github.com/AlexeyZatsepin/Android-ORM-benchmark` - aufgerufen am 11.03.2018.

[28] OpenWeatherMap.org: Current weather data - OpenWeatherMap. Website (2018) Online erhältlich unter `https://openweathermap.org/current` - aufgerufen am 15.03.2018.

[29] Developers, G.: Google Awareness API | Google Developers. Website (2018) Online erhältlich unter `https://developers.google.com/awareness/` - aufgerufen am 06.04.2018.

[30] Developers, G.: ActivityRecognitionClient. Website (2018) Online erhältlich unter `https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionClient` - aufgerufen am 08.03.2018.

[31] Developers, G.: DetectedActivity. Website (2018) Online erhältlich unter `https://developers.google.com/android/reference/com/google/android/gms/location/DetectedActivity` - aufgerufen am 08.03.2018.

[32] Developers, A.: Creating and Monitoring Geofences. Website (2018) Online erhältlich unter `https://developer.android.com/training/location/geofencing.html` - aufgerufen am 28.03.2018.

[33] Developers, G.: Android APIs | Google Fit | Google Developers. Website (2018) Online erhältlich unter `https://developers.google.com/fit/android/` - aufgerufen am 27.03.2018.

[34] Schickler, M., Pryss, R., Reichert, M., Heinzelmann, M., Schobel, J., Langguth, B., Probst, T., Schlee, W.: Using Wearables in the Context of Chronic Disorders - Results of a Pre-Study. In: 29th IEEE Int'l Symposium on Computer-Based Medical Systems. (2016) 68–69

[35] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: Towards Patterns for Defining and Changing Data Collection Instruments in Mobile Healthcare Scenarios. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017). (2017)

[36] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: Process-Driven Mobile Data Collection (Extended Abstract). In: 8th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2017). (2017)

[37] 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)

A

Quelltexte

In diesem Anhang sind einige wichtige Quelltexte aufgeführt.

```
[
  {
    "title":"Dehnuebung Huefte",
        "description":"Ein starker und kraeftiger ...",
        "duration":"5 REPETITION",
        "location":"HOME",
        "weather":"UNKNOWN",
        "activity":"STILL",
        "image":"dehnuebung_huefte.jpg",
        "video":"-",
        "steps":[
                { "stepid":0,
                  "desc":"Winkeln Sie Ihr rechtes ...",
                  "img":"dehnuebung_huefte_0.jpg"
                }, ...
        ],
        "days":[
                {  "day":"monday",
                   "times":["WAKEUP","EVENING"]
                }, ...
        ],

```

```
        "feedback":[
                {  "question":"Ist die Durchfuerung gelungen?",
                   "type":"y/n"
                }, ...
        ]
  },
  ...
]
```

Listing A.1: JSON Datei mit den Übungen

```
public class Book extends SugarRecord<Book> {
        String title;
        String edition;

        public Book(){}

        public Book(String title, String edition){
                this.title = title;
                this.edition = edition;
        }
}
```

Listing A.2: Das erstellen eines Datenbankobjekts mit dem Sugar Framework [26]

```
//Objekt erstellen
Book book = new Book(ctx, "Title here", "2nd edition");

//Objekt in Datenbank speichern
book.save();

//Objekt aus Datenbank loeschen
book.delete();
```

```
//Alle Objekte einer Klassen laden
List<Book> books = Book.listAll(Book.class);

//Alle Objekte einer Klasse loeschen
Book.deleteAll(Book.class);
```

Listing A.3: Operationen auf Datenbankobjekten [26]

```
//Berechtigung fuer Standortzugriff ueberpruefen
if (ActivityCompat.checkSelfPermission(context,
        Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {

        //Keine Berechtigung -> Benutzer um Erlaubnis Fragen
        ActivityCompat.requestPermissions(MainActivity.this,
                new String[]{
                  Manifest.permission.ACCESS_FINE_LOCATION
                },
                0);
        return;
}
```

Listing A.4: Berechtigung für Standortzugriff abfragen

```
//Berechtigung fuer Standortzugriff noch abfragen
//Google API Client fuer Zugriff auf Awareness API
GoogleApiClient client = ...
client.connect();

//Callback fuer Anwendung festlegen
//(UserLocationReceiver wird bei Updates aufgerufen)
Intent i = new Intent(context, UserLocationReceiver.class);
PendingIntent pI = PendingIntent.getBroadcast(context, 1, i, 0);
```

```
10
11 //Geofence fuer Koordinaten in 100 Meter Radius
12 //bei einem Aufenthalt von 10 s
13 AwarenessFence geoFence = LocationFence
14                 .in(latitude, longitude, 100, 10 * 1000);
15                 //Lat, Lng, Radius, Zeit (Millisekunden)
16
17 //Geofence bei API im System registrieren
18 Awareness.FenceApi.updateFences(
19         client,
20         new FenceUpdateRequest.Builder()
21                 .addFence("fenceName", geoFence, pI)
22                 .build())
23         .setResultCallback(new ResultCallback<Status>() {
24                 ...
25         });
```

Listing A.5: Erstellen eines Geofences

```
1  //Google API Client fuer Zugriff auf Awareness API
2  GoogleApiClient client = ...
3  client.connect();
4
5  //Callback fuer Anwendung festlegen
6  //(UserActivityReceiver wird bei Updates aufgerufen)
7  Intent i = new Intent(context, UserActivityReceiver.class);
8  PendingIntent pI = PendingIntent.getBroadcast(context, 1, i, 0);
9
10 //Fence der aktiv ist solange Benuutzer laeuft
11 AwarenessFence walkingFence = DetectedActivityFence
12                         .during(DetectedActivityFence.WALKING);
13
```

```
14 //Fence bei API im System registrieren
15 Awareness.FenceApi.updateFences(
16         client,
17         new FenceUpdateRequest.Builder()
18                 .addFence("walkingFence", walkingFence, pI)
19                 .build())
20         .setResultCallback(new ResultCallback<Status>() {
21                 @Override
22                 public void onResult(@NonNull Status status) {
23                         ...
24                 });
```

Listing A.6: Erstellen eines Fences zur Erkennung von einer Aktivität

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Simon Merkel Matrikelnummer: 889850

Erklärung

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

Ulm, den .....

Simon Merkel